entropy

MDPI

Article

Generative Bayesian Computation for Maximum Expected Utility

Nick Polson [1], Fabrizio Ruggeri [2] and Vadim Sokolov [3,*]

[1] Booth School of Business, University of Chicago, Chicago, IL 60637, USA; ngp@chicagobooth.edu
[2] Institute of Applied Mathematics and Information Technologies, Italian National Research Council, 20133 Milan, Italy; fabrizio@mi.imati.cnr.it
[3] Volgenau School of Engineering, George Mason University, 4400 University Drive, MSN 5D3, Fairfax, VA 22030, USA
* Correspondence: vsokolov@gmu.edu

Abstract: Generative Bayesian Computation (GBC) methods are developed to provide an efficient computational solution for maximum expected utility (MEU). We propose a density-free generative method based on quantiles that naturally calculates expected utility as a marginal of posterior quantiles. Our approach uses a deep quantile neural estimator to directly simulate distributional utilities. Generative methods only assume the ability to simulate from the model and parameters and as such are likelihood-free. A large training dataset is generated from parameters, data and a base distribution. Then, a supervised learning problem is solved as a non-parametric regression of generative utilities on outputs and base distribution. We propose the use of deep quantile neural networks. Our method has a number of computational advantages, primarily being density-free and an efficient estimator of expected utility. A link with the dual theory of expected utility and risk taking is also described. To illustrate our methodology, we solve an optimal portfolio allocation problem with Bayesian learning and power utility (also known as the fractional Kelly criterion). Finally, we conclude with directions for future research.

Keywords: generative methods; quantile networks; decision theory; Bayesian computations

Citation: Polson, N.; Ruggeri, F.; Sokolov, V. Generative Bayesian Computation for Maximum Expected Utility. *Entropy* **2024**, *26*, 1076. https://doi.org/10.3390/e26121076

Academic Editor: Éloi Bossé

Received: 23 September 2024
Revised: 28 November 2024
Accepted: 6 December 2024
Published: 10 December 2024

1. Introduction

Generative Bayesian Computation (GBC) is a statistical method that estimates the posterior distribution of model parameters when the likelihood function is intractable or hard to calculate. Similar to Approximate Bayesian Computation (ABC), GBC methods are likelihood-free and use simulation methods to estimate the posterior distribution. GBC constructs a probabilistic map to represent a posterior distribution and to calculate functionals of interest. Our goal here is to extend GBC methods to solve maximum expected utility (MEU) problems. We propose a density-free generative method that has the advantage of being able to compute and optimize expected utility as a by-product. To perform this, we find a deep quantile neural map to represent the distributional utility. Then, we provide a key identity which represents the expected utility as a marginal of quantiles.

Although deep learning (DL) has been widely used in business, engineering [1] and finance [2], DL was shown to outperform classical methods for prediction in [3]. Solving optimal decision problems, however, has received less attention. Our work fills this gap and builds on the reinforcement learning literature [4,5], where it is not necessary to know the utilities but, rather, one needs a panel of known rewards and input parameters. The main difference then is our assumption of a utility function [6] and its use in architecture design at the first level of the hierarchy. Recent work on generative methods includes [7,8] in spatial settings, Ref. [9] for causal modeling and [10] for engineering applications.

Our work also builds on [11], whose authors used curve fitting techniques to solve MEU problems. Our work is also related to the reinforcement learning literature of [4]. It differs in that we assume a given utility function, and we directly simulate and model the

random utilities implicit in the statistical model. We also focus on density-free generative AI methods. There is a large body of literature on density-based generative methods such as normalized flows or diffusion-based methods. Refs. [12,13] used ABC methods and classification to solve posterior inference problem.

The idea of generative methods is straightforward. Let Y denote data and θ a vector of parameters, including any hidden states (also known as latent variables) z. Typically, $Y = (y_1, \ldots, y_n)$ is a vector of observations. First, we generate a "look-up" table of "fake" data $\{Y^{(i)}, \theta^{(i)}\}_{i=1}^{N}$ by simulating a training dataset of parameters and data. This allows us to use deep learning to solve for the inverse map via a supervised learning problem. Generative methods have the advantage of being likelihood-free. For example, our model might be specified by a forward map $Y^{(i)} = f(\theta^{(i)})$ rather than a traditional random draw from a likelihood function $Y^{(i)} \sim p(Y^{(i)} \mid \theta^{(i)})$. Our method works for traditional likelihood-based models but avoids the use of MCMC.

Posterior uncertainty is solved via the inverse non-parametric regression problem, where we predict $\theta^{(i)}$ from $Y^{(i)}$. The uncertainty is modeled using $Z^{(i)}$, which is a random variable that follows some base distribution and is independent of θ. The base distribution is typically uniform, although this does not have to be the case; for example, one could use a very large dimensional Gaussian vector. This approach is justified by the so-called noise outsourcing theorem [14].

Theorem 1 (Noise Outsourcing Theorem)**.** *If* (Y, Θ) *are random variables in a Borel space* $(\mathcal{Y}, \Theta)$*, then there exists a r.v.* $z \sim U(0,1)$*, which is independent of* Y *and a function* $H : [0,1] \times \mathcal{Y} \to \Theta$*, such that*

$$(Y, \Theta) \overset{a.s.}{=} (Y, H(Y, Z))$$

Moreover, if there is a statistic $S(Y)$ with $Y \perp\!\!\!\perp \Theta | S(Y)$, then

$$\Theta \mid Y \overset{a.s.}{=} H(S(Y), Z).$$

The role of $S(Y)$ is equivalent to the one used in ABC literature. It performs dimension reduction in n, the dimensionality of the signal. Our approach is then to first use a deep neural network to calculate the inverse probability map (also known as posterior):

$$\theta \overset{D}{=} F^{-1}_{\theta|y}(Z),$$

where Z is uniform. In the multi-parameter case, we use an RNN or autoregressive structure, where we model a vector via a sequence $(F^{-1}_{\theta_1|Y}(z_1), F^{-1}_{\theta_2|\theta_1,Y}(z_2) \ldots)$. A remarkable result from [15] shows that we can learn S independent of H simply via OLS.

Then, we need to train a deep neural network, H, on

$$\theta^{(i)} = H(S(Y^{(i)}), Z^{(i)}),$$

Here, $S(y)$ is a statistic to perform dimension reduction with respect to the signal distribution. The existence of H follows from the noise outsourcing theorem. Specifying H is the key to the efficiency of the approach. Ref. [10] proposes the use of quantile neural networks implemented with ReLU activation functions. Quantile neural network is a deep neural network (DNN) that approximates the quantile function. The training dataset acts as a supervised learning problem and allows us to represent the posterior as a map from input $Y^{(i)}$ and output $\theta^{(i)}$. A deep neural network is an interpolator and provides an optimal transport map from base distribution to output. The parameters of the neural network do not need to be identified. The map will provide a probabilistic representation of the posterior for any data vector as we can evaluate the map at any y. The question is whether our quantile neural network will generalize well. This is an active area of research and there is a double descent phenomenon that has been found for the generalization risk (see [16,17]).

To extend our generative method to MEU problems, we assume that the utility function U is given. Then, we simply draw, from above, the additional associated utilities $U_d^{(i)} := U(d, \theta^{(i)})$ for a given decision d and $\theta^{(i)}$. Then, we append the utilities to our training dataset, including the baseline distribution $z^{(i)}$, to yield a new training dataset:

$$\{U_d^{(i)}, Y^{(i)}, \theta^{(i)}, Z^{(i)}\}_{i=1}^N.$$

Now, we construct a non-parametric estimator of the form

$$U_d^{(i)} = H(S(Y^{(i)}), \theta^{(i)}, Z^{(i)}, d),$$

where H is a quantile deep learner and z is uniform. Again, $S(\cdot)$ is a summary statistic which allows for dimension reduction in the signal space. A number of authors have discussed the optimal choice of summary statistics, S. For example, the authors of [18,19] used deep learning to learn the optimal summary statistics. We add another layer H to learn the full posterior distribution map (see also [20–24]).

Given that the posterior quantiles of the distributional utility, denoted by $F^{-1}_{U|d,y}(z)$, are represented as a quantile neural network, we then use a key identity, which shows how to represent any expectation as a marginal over quantiles, namely

$$E_{\theta|y}[U(d, \theta)] = \int_0^1 F^{-1}_{U|d,y}(z)dz$$

This is derived in Section 2.1. The optimal decision function, $d^\star(y) := \arg max_d E_{\theta|y}[U(d, \theta)]$, simply maximizes the expected utility. This can then be approximated via Monte Carlo and optimized over any decision variables. We show that quantiles update as composite functions (also known as deep learners) and that the map can be viewed as a concentration function. The Lorenz curve of the utility function can be used to prove the key identity above where expectations are written as marginals of quantiles. There is a similarity with nested sampling [25] and vertical-likelihood Monte Carlo [26].

Our approach focuses on generative density-free quantile methods. Quantile neural networks (QNNs) implemented via deep ReLU networks have good theoretical [27,28] and practical properties. Ref. [29] provides standard non-parametric asymptotic bounds in N for the approximation of conditional quantile functions. Ref. [10] proposes the use of quantile posterior representations and the use of ReLU neural networks to perform this task. Rather than dealing directly with densities and the myriad of potential objective functions, we directly model any random variables of interest via a quantile map to a baseline uniform measure. Our neural estimator network directly approximates the posterior CDF and any functions of interest. To solve maximum expected utility problems, we simply add a given utility function as the first layer of the network architecture.

The interpolation property of deep learners is a key feature of our generative AI method as opposed to kernel-based generative methods such as Approximate Bayesian Computation (ABC), which uses accept–reject methods to calculate the posterior at a given output. The authors of [16] pointed out a fascinating empirical property of deep learners in terms of their interpolation approximation properties (see also [17]). There is a second bias–variance trade off in the out-of-sample prediction problem. The interpolation for deep learners suggests that our generative method provides good predictive rules.

Another class of estimators are those based on kernel methods, such as Approximate Bayesian Computations (ABCs). ABC methods differ in the way that they generate their "fake" look-up table. Rather than providing a neural network estimator for any output y. ABC methods approximate the likelihood function by locally smoothing using a circle of radius ϵ around the observed data. This can be interpreted as a nearest neighbor model; see [10] for a discussion. The advantage of ABC is that the training dataset is "tilted" towards the observed y; the disadvantage is that it uses accept–reject sampling that fails

in high-dimensions. Ref. [23] provides theoretical bounds for the generalizability of non-parametric kernel methods.

The rest of this paper is outlined as follows. Section 1.1 first compares GBC methods to ABC posterior simulation. Section 1.1 then provides a description of the generative AI model for learning the utility function. Section 2 defines the distributional utility function and its expectation. Section 3 provides a link with the dual theory of expected utility due to [30]. We introduce the Lorenz curve of the utility function and quantile methods as a way of estimating the posterior expected utility. Section 4 provides an application to portfolio learning. We show how to use generative methods for the normal-normal learning model and to find an optimal portfolio allocation problem based on the Kelly criterion [31]. Section 5 concludes with directions for future research.

1.1. Generative Bayesian Computation (GBC)

To fix the notation in the setting of Bayesian parameter inference, let $\mathcal{Y}$ denote a locally compact metric space of signals, denoted by y, and $\mathcal{B}(\mathcal{Y})$ the Borel σ-algebra of $\mathcal{Y}$. Let λ be a measure on the measurable space of signals $(\mathcal{Y}, \mathcal{B}(\mathcal{Y}))$. Let $P(dy|\theta)$ denote the conditional distribution of signals given the parameters. Let Θ denote a locally compact metric space of admissible parameters (also known as hidden states and latent variables $z \in \mathcal{Z}$) and $\mathcal{B}(\Theta)$ the Borel σ-algebra of Θ. Let μ be a measure on the measurable space of parameters $(\Theta, \mathcal{B}(\Theta))$. Let $\Pi(d\theta|y)$ denote the conditional distribution of the parameters given the observed signal y (also known as the posterior distribution). In many cases, Π is absolutely continuous with density π such that

$$\Pi(d\theta|y) = \pi(\theta|y)\mu(d\theta).$$

Moreover, we will write $\Pi(d\theta) = \pi(\theta)\mu(d\theta)$ for prior density π when available.

Our framework allows for likelihood and density-free models. In the case of likelihood-free models, the output is simply specified by a map (also known as forward equation):

$$y = f(\theta)$$

When a likelihood $p(y|\theta)$ is available with respect to the measure λ, we write

$$P(dy|\theta) = p(y|\theta)\lambda(dy).$$

There are a number of advantages of such an approach, primarily the fact that they are density-free. They use simulation methods and deep neural networks to invert the prior to posterior map. We build on this framework and show how to incorporate utilities into the generative procedure.

As a default choice of network architecture, we will use a ReLU network for the posterior quantile map. The first layer of the network is given by the utility function, and hence, this is what makes the method different from learning the posterior and then directly using naive Monte Carlo to estimate the expected utility. This would be inefficient, as quite often, the utility function places high weight on the region of low-posterior probability, representing a tail risk.

Maximum Expected Utility

Decision problems are characterized by a utility function $U(\theta, d)$ defined over parameters, θ, and decisions, $d \in \mathcal{D}$. We will find it useful to define the family of utility random variables indexed by decisions as

$$U_d := U(\theta, d) \text{ where } \theta \sim \Pi(d\theta).$$

Optimal Bayesian decisions [32] are then defined by the solution to the prior expected utility:

$$U(d) = E_\theta(U(d, \theta)) = \int U(d, \theta)p(\theta)d\theta,$$

$$d^\star = \arg\max_d U(d)$$

When information in the form of signals y is available, we need to calculate the posterior distribution $p(\theta|y) = f(y|\theta)p(\theta)/p(y)$. Then, we have to solve for the optimal *a posteriori* decision rule $d^\star(y)$, defined by

$$d^\star(y) = \arg\max_d \int U(\theta, d)p(\theta|y)d\theta$$

where expectations are now taken with respect to $p(\theta|y)$, the posterior distribution.

1.2. GBC vs. ABC

GBC works by generating a synthetic dataset to train a quantile neural network, which yields a posterior map (also known as optimal transport) from a base distribution. ABC works by generating a synthetic dataset (the so-called reference table) in the neighborhood of the observed output and uses summary statistics and kernel methods to provide a posterior distribution. As with GBC, this bypasses the need to evaluate a likelihood function or to known densities. Both methods provide a natural alternative to MCMC simulation methods.

Approximate Bayesian Computation (ABC) is a generative method for obtaining samples from the posterior distribution. ABC relies on comparing a summary statistic $S(y)$ to that of the observed output. Denote the training sample by $(\theta^{(i)}, Y^{(i)})$. ABC requires a dimensionality-reducing summary statistic $S(y)$, kernel $K(\cdot)$ and a tolerance level ϵ. The tilted-posterior is defined by

$$\pi^\epsilon_{ABC}(\theta|Y = y_{obs}) = \frac{1}{m_\epsilon(y_{obs})} \int K_\epsilon(S(y) - y_{obs})\lambda_\theta(dy)\pi(d\theta).$$

Here, $y = (y_1, \ldots, y_n)$ is high-dimensional. Hence, the need for a k-dimensional summary statistic $S : \Re^n \to \Re^k$, where k is fixed.

This ensures that the mean of the ABC posterior matches that of the posterior of interest. Furthermore, under a uniform kernel $K = I(|S(y) - s_{obs}| < eps)$, convergence

$$\pi^\epsilon_{ABC}(\theta \mid Y = y_{obs}) \to \pi(\theta \mid Y = y_{obs})$$

is guaranteed, namely

$$\begin{aligned}\pi^\epsilon_{ABC} &= \frac{1}{m_\epsilon(y_{obs})} \int_\Theta I(|S(y) - s_{obs}| < \epsilon)\delta(y - f(\theta))\pi(\theta)dyd\theta = \pi(\theta \mid |S(y) - s_{obs}| < \epsilon) \\ &\to \pi(\theta \mid Y = y_{obs}), \text{ where } S(y) = \hat{\theta}(y) = E_\pi(\theta \mid y).\end{aligned}$$

An estimator $\hat{\theta}(y) = S_{\hat{\psi}}(y)$ can be found using a deep NN, effectively learn a good approximation to the posterior mean from a large dataset $(Y^{(i)}, \theta^{(i)})_{i=1}^N \sim \pi \times \mathcal{M}$ and solve the ℓ_2—minimization problem:

$$\arg\min_\psi \frac{1}{N} \sum_{i=1}^N \|S_\psi(Y^{(i)}) - \theta^{(i)}\|^2.$$

This is equivalent to the high-dimensional non-parametric regression $\Theta = S(Y) + \epsilon$ and provides methods for estimating the conditional mean. Typically, estimators, $\hat{S}$, include KNN and kernel methods. Recently, deep learners have been proposed and the theoretical properties of superpositions of affine functions (also known as ridge functions) have been provided [23,33].

Generative Bayesian Computation (GBC), on the other hand, takes this approach one step further. Let $Z \sim P_Z$ be a base measure for a latent variable, Z, typically a standard multivariate normal or vector of uniforms. The goal of generative methods is to characterize

the posterior measure $P_{\Theta|Y}$ from the training data $(\Theta_i, Y_i)_{i=1}^N \sim P_{\Theta,Y}$, where N is chosen to be suitably large. A deep learner is used to estimate $\hat{f}$ via the non-parametric regression $\Theta = H(Y, Z)$. In this case, Z is a base distribution. In the univariate case, where Z is uniform, this amounts to inverse cdf sampling, namely $\Theta = F^{-1}_{\Theta|Y}(U)$.

An important feature of GBC methods is that we have a transport map that holds for *any* output Y. We simply evaluate the network at any given Y. Moreover, our neural architecture usually includes dimension reduction for the data Y via a summary statistic $S(Y)$ and a kernel embedding, typically cosine transform, ψ, for Z. Our map is then

$$\Theta = H(S(Y), \psi(Z))$$

where Z is a new base draw. Here, ψ denotes the cosine embedding so that the architecture for the latent variable corresponds to a Fourier approximation with rates of convergence given by $O(N^{-\frac{1}{2}})$ (see Barron (1993)). The deep learner, H, is estimated via a quantile NN from the triples $(\Theta_i, Y_i, Z_i)_{i=1}^N \sim P_{\Theta,Y} \times P_Z$.

Double Descent

There is still the question of approximation and the interpolation properties of a DNN. Recent research on the interpolation properties of quantile neural networks was recently conducted by [23,27,34]. See also [16,17]. The folklore theorem of deep learning is that shallow deep learners provide good representations of multivariate functions with minimal parameters in the network and are good interpolators.

2. Generative Expected Utility

Decision problems under uncertainty are characterized by a utility function $U(d, y, \theta)$ defined over decisions, $d \in \mathcal{D}$, signals $y \in \mathcal{Y}$ and parameters, $\theta \in \Theta$. The *a priori* expected utility is defined by [32] as

$$u(d) = E_{y,\theta}(U(d, y, \theta)) = \int U(d, y, \theta) d\Pi(y, \theta).$$

The *a posteriori* expected utility for decision function, $d(y)$, is given by

$$u(d, y) = E_{\theta|y}(U(d, y, \theta)) = \int U(d, y, \theta) dF_{\theta|y}(\theta)$$

with expectation taken with respect to posterior cdf.

The distributional form is found by defining the family of utility random variables indexed by decisions defined by

$$U_{d,y} \stackrel{D}{=} U(d, y, \theta) \text{ where } \theta \sim \Pi(d\theta \mid dy)$$

Then, we write

$$u(d, y) = E_{U \sim U_{d,y}}(U)$$

This makes clear the fact that we can view the utility as a random variable defined as a mapping (also known as optimal transport) of (y, θ) evaluated at d. Now, we need

$$d^\star(y) = \arg\max_d u(d, y).$$

Our deep neural estimator then takes the form

$$U_{d,y} \stackrel{D}{=} U(d, H(S(y), z)).$$

Generative AI will model θ as a mapping from the data y and the quantile z as a deep learner. The nonlinear map is then estimated using a simulated training dataset of utilities,

signals and parameters, denoted by the set $\{U^{(i)}, Y^{(i)}, \theta^{(i)}\}$, $i = 1, \ldots, N$. We augment this training dataset with a set of independent baseline variables $z^{(i)}, 1 \leq i \leq N$.

Latent States

We allow for the possibility of further hidden states z in the parameter. Our method clearly extends the models that also have hidden states (deterministic or stochastic); for example, many econometric models have deterministic models for the states (e.g., dynamic general stochastic equilibrium (DGSE) models). Our methods are particularly useful for dynamic learning in economics and finance, where MCMC can be computationally prohibitive. We illustrate our method with a portfolio allocation problem and use normal-normal learning. Another class of models where our methods are particularly efficient are the structured sufficient statistics that can depend on hidden latent states and that naturally perform dimensionality reduction for posterior parameter learning (see [35,36]).

2.1. Calculating Expected Utility

Expected utility is estimated using a quantile re-ordering trick, and then the optimal decision function maximizes the resulting quantity. We propose using a quantile neural network as the nonlinear map. Notice that we assume that the training data are simulated by the model and are easy to sample, and the simulation costs are low. We can make N as large as we want. The key to generative methods is that we directly model the random variable θ as a nonlinear map (deep learner) from the data y and the quantile z. This is a generalization of the quantile regression to the Bayesian setting.

Quantile Re-ordering

Ref. [4] uses quantile neural networks for decision-making and applies quantile neural networks to the problem of reinforcement learning. Specifically, the authors relied on the fact that expectations are quantile integrals. Let $F_U(u)$ be the CDF of the distributed utility. The key identity follows from the Lorenz curve:

$$E_{U \sim U(d,\theta)}(U) = \int_0^1 F_U^{-1}(z)dz.$$

This key identity follows from the identity

$$\int_0^\infty u dF_U(u) = \int_0^1 F_U^{-1}(z)dz$$

which holds true under the simple transformation $z = F_U(u)$, with Jacobian $dz = f_U(u)du$.

Utility Lorenz Curve

The quantile identity also follows from the Lorenz curve of the utility r.v. as follows. We can compute $\mathbb{E}(U)$ using the mean identity for a positive random variable and its CDF or equivalently, via the Lorenz curve. Given the survival function $S_U(u) = 1 - F_U(u)$, we have

$$E(U) = \int_0^\infty (1 - F_U(u))du = \int_0^\infty S_U(u)du$$
$$E(U) = \int_0^1 F_U^{-1}(s)ds = \int_0^1 \Lambda(1-s)ds = \int_0^1 \Lambda(s)ds$$

We do not have to assume that $F_U^{-1}(s)$ or, equivalently $\Lambda(s)$, is available in closed form, rather we can find an unbiased estimate of this by simulating the Lorenz curve.

The Lorenz curve $\mathcal{L}$ of U is defined in terms of its CDF, $F_U(u)$, as

$$\mathcal{L}(u) = \frac{1}{E(U)} \int_0^u F_U^{-1}(s)ds \quad \text{where} \quad u \in [0,1]$$
$$\mathbb{E}_{U \sim U(d,\theta)}(U) = \int_\Theta U(d,\theta)\Pi(d\theta) \, .$$

One feature of a Lorenz curve is that is provides a way to evaluate

$$\mathbb{E}(U) = \int_0^1 F_U^{-1}(s)ds$$

Hence, we only need to approximate the quantile function $F_U^{-1}(s)$ with a deep Bayes neural estimator.

2.2. GenBayes-MEU Algorithm

The method will generalize to the problems of the form

$$\arg\max_d u(d,y) = \int U(\theta,d)p(\theta \mid d,y)d\theta$$

First, rewrite the expected utility in terms of posterior CDF of a random variable $U_d = U(d,\theta)$, where $\theta \sim p(\theta \mid d,y)$ and U_d is simply a transformation of $F_{U_{d,y}}^{-1}(z)$, approximated by a quantile neural network (QNN). We will further approximate the approximate CDF with a quantile neural network. This is a function approximation, which can be achieved using deep learning.

Given the deep learner for the stochastic utilities, namely

$$U_d = U(H(S(y),z),d),$$

which is a function of base distribution z and the data y, we use y_{obs} to draw a value of U from z. Then, we can use Monte Carlo to estimate the expected utility

$$\hat{U}^* = \frac{1}{N}\sum_{i=1}^N U_d^{(i)}.$$

Our algorithm starts by simulating forward $\{y_i,\theta_i\}_{i=1}^N$ and proceeds as Algorithm 1:

Algorithm 1 GBC for MEU

Simulate $(Y^{(i)},\theta^{(i)})_{1\le i\le N} \sim p(y \mid \theta)$ or $Y^{(i)} = f(\theta^{(i)})$ and $\theta^{(i)} \sim \pi(\theta)$.
Simulate the utility $u^{(i)} = U(d^{(i)},Y^{(i)},\theta^{(i)})$
Train H using the simulated dataset for $i = 1,\dots N$, via $\hat{\theta}^{(i)} = H(Y^{(i)},z^{(i)})$
Train U using the simulated dataset $U_d = U(H(S(Y^{(i)}),z)^{(i)},d)$ for $i = 1,\dots N$
Pick a decision d that maximizes the expected utility. We use Monte Carlo to estimate the expected utility.

$$E(U_d) = \sum_{i=1}^N F_{U_d}^{-1}(u_i) \to \underset{d}{\text{maximize}}$$

To find the arg max, we can use several approaches, including Robbins–Monro [37] or temporal differencing (TD) learning [38].

A related problem is that of reinforcement learning and the invariance of the contraction property of the Bellman operator under quantile projections [5].

3. Dual Theory of Expected Utility

Similar approaches that rely on the dual theory of expected utility can be found in [30]. How does one evaluate the risky gamble? One way to introduce a utility function on payouts and not change the probabilities and calculate $E(u(X))$ is to have a distortion measure on the probabilities, also known as the survival function, and leave the payouts alone, and then calculate the expectation of the distorted survival function. A distortion measure is a monotonic function $g : [0,1] \to [0,1]$ such that $g(0) = 0$ and $g(1) = 1$. Yaari showed that one can pick distortion g to be u^{-1}.

Risky prospects are evaluated by a cardinal numerical scale, which resembles an expected utility, except that the roles of payments and probabilities are reversed. Under expected utility, we assess gambles according to

$$E(u(X)) = \int_0^\infty u(x)p_X(x)dx = \int_0^\infty u(x)dF_X(x)$$

The dual theory then will order gambles according to

$$\tilde{E}(u(X)) = \int_0^1 g(1 - F_X(z))dz = \int_0^1 g(S_X(z))dz$$

Ref. [30] shows that one can take $g = u^{-1}$ and still obtain the same stochastic ordering of gambles.

$$\int_0^1 u^{-1}(1 - F_x(x))dx.$$

Specifically, let $Y = u(X)$, and then, picking $g(x) = S_x\left(u^{-1}\left(S_X^{-1}(x)\right)\right)$ yields $S_Y(t) = g(S_X(t))$ as required. Hence, the expected utility decomposes as

$$\tilde{E}(u(X)) = \int_0^\infty S_y(t)dt = \int_0^\infty g(S_x(z))dz$$

The function g, being monotonic, has similar properties to a distortion function. This provides a class of functions that can be used as a distortion measure. The notion of a concentration function is explored in [39–41]. Another key insight is that g can be estimated using a deep quantile NN, given a large training sample cam be generated from the direct model.

Distortion (also known as Transformation) Duality

The dual theory has the property that utility is linear in wealth (in the usual framework, the agent would be risk-neutral). To compensate, the agent has to apply a nonlinear transformation known as a distortion measure to the probabilities of payouts. This "tilting" of probabilities is also applied in derivative pricing [42] using a change in measure. In the dual theory, we are interested in the inverses of the distribution function.

The dual theory is motivated by the two representations of the expected value of a random variable, namely

$$E(X) = \int_0^\infty (1 - F_X(x))dx = \int_0^1 F_X^{-1}(x)dx.$$

We will show that the latter is more useful from a computational perspective. Adding risky choice then transforms the inner payouts (standard expected utility) or the probabilities (dual theory).

This also leads to the question of how to calculate and optimize expected utility efficiently using generative methods. We propose the use of a deep neural Bayes estimator.

Let the random utility $U \stackrel{D}{=} u(X)$, where $X \sim F_X$. Let $F_U(u)$ be the corresponding cdf. Then, we can write the expected utility as

$$E(U) = \int_0^1 u dF_U(u) = \int_0^1 (1 - F_U(u))du = \int_0^1 S_U(t)dt$$

where the de-cumulative distribution (also known as survival) function $S_U(\cdot)$ is defined as

$$S_U(t) = \mathbb{P}(U > t).$$

The survival function is a non-increasing function of t and $S_U(0) = 1$.

We obtain the dual theory by transforming these survival probabilities. Notice that the dual theory is linear in payouts. The distortion comes from this "risk-neural" probability. Specifically,

$$EU(X) = \int_0^1 g(S_X(t))dt.$$

If g is differentiable, then we obtain the so-called Silver formula:

$$EU(X) = \int_0^1 tg'(S_X(t))dF_x(t) \text{ with } \int_0^1 g'(S_X(t))dF_X(t) = 1.$$

Hence, the weights can be interpreted as a tilted probability measure in the dual sense.

If g is convex, then g' is non-decreasing and

$$EU(X) = \int_0^1 tg'(S_X(t))dF_X(t) = \int_0^1 \phi(t)dF_X(t) = \int_0^1 \phi(F_X^{-1}(z))dz.$$

This is a linear utility function and $g'(S_X(t))$ is a distortion. We can write

$$E(U) = \int_0^1 \phi(t)d(f \circ F_X)(t) = \int_0^1 f(F_X(t))d\phi(t).$$

Appendix A provides a $O(N^{-4})$-error bound for the quantile re-ordering estimator.

4. Application

4.1. Bayes Rule for Quantiles

Ref. [43] shows that quantile methods are direct alternatives to density computations. Specifically, given $F_{\theta|y}(u)$, a non-decreasing and continuous from right function, we define

$$Q_{\theta|y}(u) := F^{-1}_{\theta|y}(u) = \inf\Big(\theta : F_{\theta|y}(\theta) \geq u\Big),$$

which is non-decreasing, continuous from the left. Ref. [43] shows the important probabilistic property of quantiles

$$\theta \stackrel{P}{=} Q_\theta(F_\theta(\theta))$$

Hence, we can increase the efficiency by ordering the samples of θ and the baseline distribution and use the monotonicity of the inverse CDF map.

Let $g(y)$ be non-decreasing and continuous from left with $g^{-1}(z) = \sup(y : g(y) \leq z)$. Then, the transformed quantile has a compositional nature, namely

$$Q_{g(Y)}(u) = g(Q(u))$$

Hence, quantiles act as superposition (also known as deep Learner).

This is best illustrated in the Bayesian learning model. We have the following result updating prior to posterior quantiles known as the conditional quantile representation:

$$Q_{\theta|Y=y}(u) = Q_\theta(s) \text{ where } s = Q_{F(\theta)|Y=y}(u)$$

To compute s by definition,

$$u = F_{F(\theta)|Y=y}(s) = P(F(\theta) \leq s|Y = y) = P(\theta \leq Q_\theta(s)|Y = y) = F_{\theta|Y=y}(Q_\theta(s)).$$

We now provide an application of deep generative quantile networks to Bayesian learning.

4.2. Normal-Normal Bayes Learning: Wang Distortion

For the purpose of illustration, we consider the normal-normal learning model. We will develop the necessary quantile theory to show how to calculate posteriors and expected utility without resorting to densities. Also, we show a relationship with Wang's risk distortion measure as the deep learning that needs to be learned.

Specifically, we observe the data $y = (y_1, \ldots, y_n)$ from the following model:

$$\begin{aligned} y_1, \ldots, y_n \mid \theta &\sim N(\theta, \sigma^2) \\ \theta &\sim N(\mu, \alpha^2) \end{aligned}$$

Hence, the summary (sufficient) statistic is $S(y) = \bar{y} = \frac{1}{n}\sum_{i=1}^n y_i$.

Given the observed samples $y = (y_1, \ldots, y_n)$, the posterior is then $\theta \mid y \sim N(\mu_*, \sigma_*^2)$ with

$$\mu_* = (\sigma^2\mu + \alpha^2 s)/t, \quad \sigma_*^2 = \alpha^2\sigma^2/t,$$

where

$$t = \sigma^2 + n\alpha^2 \text{ and } s(y) = \sum_{i=1}^n y_i.$$

The posterior and prior CDFs are then related via the

$$1 - \Phi(\theta, \mu_*, \sigma_*) = g(1 - \Phi(\theta, \mu, \alpha^2)),$$

where Φ is the normal distribution function. Here, the Wang distortion function is defined by

$$g(p) = \Phi\Big(\lambda_1\Phi^{-1}(p) + \lambda\Big),$$

where

$$\lambda_1 = \frac{\alpha}{\sigma_*} \text{ and } \lambda = \alpha\lambda_1(s - n\mu)/t.$$

The proof is relatively simple and is as follows:

$$\begin{aligned} g(1 - \Phi(\theta, \mu, \alpha^2)) = g(\Phi(-\theta, \mu, \alpha^2)) &= g\left(\Phi\left(-\frac{\theta - \mu}{\alpha}\right)\right) \\ &= \Phi\left(\lambda_1\left(-\frac{\theta - \mu}{\alpha}\right) + \lambda\right) = 1 - \Phi\left(\frac{\theta - (\mu + \alpha\lambda/\lambda_1)}{\alpha/\lambda_1}\right) \end{aligned}$$

Thus, the corresponding posterior updated parameters are

$$\sigma_* = \alpha/\lambda_1, \quad \lambda_1 = \frac{\alpha}{\sigma_*}$$

and

$$\mu_* = \mu + \alpha\lambda/\lambda_1, \quad \lambda = \frac{\lambda_1(\mu_* - \mu)}{\alpha} = \alpha\lambda_1(s - n\mu)/t.$$

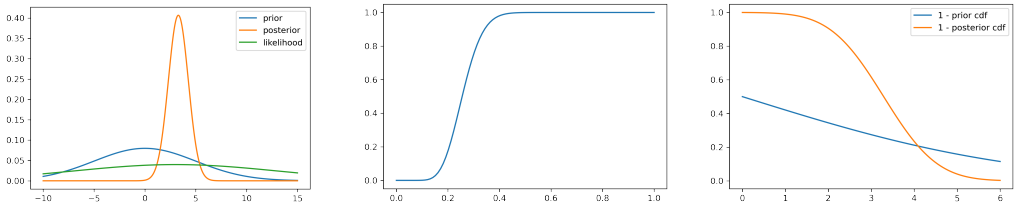
We now provide an empirical example.

Numerical Example

Consider the normal-normal model with prior $\theta \sim N(0, 5)$ and likelihood $y \sim N(3, 10)$. We generate $n = 100$ samples from the likelihood and calculate the posterior distribution.

The posterior distribution calculated from the sample is then $\theta \mid y \sim N(3.28, 0.98)$.

Figure 1 shows the Wang distortion function for the normal-normal model. The left panel shows the model for the simulated data, while the middle panel shows the distortion function, the right panel shows the $1 - \Phi$ for the prior and posterior of the normal-normal model.



(**a**) Model for simulated data (**b**) Distortion function g (**c**) $1 - \Phi$

Figure 1. Density for prior, likelihood and posterior, distortion function, and $1 - \Phi$ for the prior and posterior of the normal-normal model.

4.3. Portfolio Learning

Consider power utility and log-normal returns (without leverage). We assume that a portfolio value $X = e^W$ follows a log-normal distribution:

$$W(\omega) = (1 - \omega)r_f + \omega R, \quad R \sim \mathcal{N}(\mu, \sigma^2)$$

Here, $\omega \in (0, 1)$ is the portfolio weight, r_f is the risk-free rate, μ is the mean return and σ^2 is the variance of the return. The utility function is then given by

$$U(W) = -e^{-\gamma W}.$$

Here, U^{-1} exists, and the expected utility is

$$U(\omega) = E(-e^{\gamma W}) = \exp\left\{\gamma \mathrm{E}(W) + \frac{1}{2}\omega^2 \mathrm{Var}(W)\right\}.$$

In this case, we have a closed-form solution for the expected utility, as a function of the decision variable ω (portfolio weight). It is the moment-generating function of the log-normal. We can plug-in the mean and variance of W to obtain the expected utility:

$$U(\omega) = \exp\left\{\gamma\left\{(1 - \omega)r_f + \omega\mu\right\}\right\} \exp\left\{\frac{1}{2}\gamma^2\omega^2\sigma^2\right\}.$$

The optimal Kelly–Brieman–Thorpe–Merton value of ω is given by

$$\omega^* = (\mu - r_f)/(\sigma^2\gamma).$$

Within the GBC framework, it is easy to add learning or uncertainty on top of σ^2 and have a joint posterior distribution $p(\mu, \sigma^2 \mid R)$.

Now, we re-order the integral in terms of quantiles of the utility function. We assume utility is the random variable and re-order the sum as the expected value of U:

$$E(U(W)) = \int_0^1 F^{-1}_{U(W)}(z)dz$$

Hence, if we can approximate the inverse of the CDF of $U(W)$ with a quantile NN, we can approximate the expected utility and optimize over ω.

The stochastic utility is modeled with a deep neural network, and we write

$$Z = U(W) \approx F, \; W = U^{-1}(F)$$

We can perform optimization by carrying out the grid search for ω.

The decision variable ω affects the distribution of the returns. The utility only depends on the returns W. Our GenAI solution is given by Algorithm 2.

Algorithm 2 GBC for Portfolio Learning

Simulate log-returns $W^{(i)} \mid \omega^{(i)} \sim N((1-\omega^{(i)})r_f + \omega^{(i)}\mu, \sigma^2\omega^{(i)}2)$
Calculate corresponding utilities $Z^{(i)} = U(W^{(i)})$
Learn $F^{-1}_{Z_\omega}$ with a quantile NN
Find the optimal portfolio weight $\omega^\star$ via

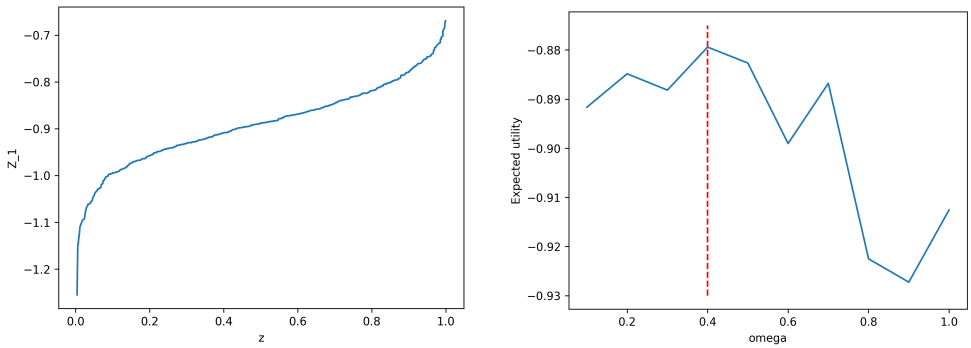
$$E(Z_\omega) = \sum_{i=1}^{N} F^{-1}_{Z_\omega}(u_i) \rightarrow \underset{\omega}{\text{maximize}}$$

Empirical Example

Consider $\omega \in (0,1)$, $r_f = 0.05$, $\mu = 0.1$, $\sigma = 0.25$, and $\gamma = 2$. We have the closed-form fractional Kelly criterion solution:

$$\omega^* = \frac{1}{\gamma}\frac{\mu - r_f}{\sigma^2} = \frac{1}{2}\frac{0.1 - 0.05}{0.25^2} = 0.40$$

We can simulate the expected utility and compare with the closed-form solution. Figure 2 shows the results of the simulation. The left panel shows the sorted values of the random draws from $-\exp(-0.1W)$ vs. the sorted values of the posterior quantiles. The right panel shows the integral of Z with respect to ω vs. the corresponding values of ω. The red vertical line corresponds to $\omega = 0.4$, which is the analytical optimum.



(**a**) $Z_1 = -\exp(-0.1 * W)$ vs. z (**b**) Integral of Z vs. ω

Figure 2. Left panel (**a**) shows plot of sorted values of z vs. sorted values of random draws from $-\exp(-\omega * W)$ for $\omega = 0.1$. Right panel (**b**) shows values of integral of Z with respect to z vs. the corresponding values of ω. The integral was calculated using the trapezoid rule. The red vertical line corresponds to $\omega = 0.4$, which is the analytical optimum.

5. Discussion

Generative Bayesian Computation (GBC) is a simulation-based approach to statistical and machine learning. Finding optimal decisions via maximum expected utility is challenging for a number of reasons: first, we need to calculate the posterior distribution over uncertain parameters and hidden states; second, we need to perform integration to find the expected utility; and third, we need to optimize the expected utility. We show how to use deep learning to solve these problems.

We propose a density-free generative method that finds posterior quantiles (and hence, the posterior distribution) via a deep learning estimator. Quantiles are shown to be particularly useful in solving for expected utility densities. Optimization is then performed

via a Monte Carlo approximation of the expected utility. We show how to apply this method to the normal-normal model and the portfolio learning problem.

Our goal then was to show their use in solving expected utility problems. It can be viewed as a direct implementation of Yaari's dual theory of expected utility and to risk distortion measures that are commonplace in risk analysis. There are many avenues for further work; for example, the multi-parameter case and sequential decision problems are two rich areas of future research [44].

Future extensions of this work could include the development of GBC for variable selection, sequential decision-making and general optimization problems. However, we need to note that each application requires a different NN architecture to model the quantiles, and the algorithm may need to be adapted to a specific problem.

Author Contributions: All authors have equal contribution to the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study.

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

Appendix A. Calculating Functionals of Interest

Another advantage of quantile methods is that we can find tight error bounds of $O(N^{-4})$ for estimating functionals of interest. Another important property of quantiles is that they can be used to calculate functionals of interest:

$$E_{\theta|y}(\theta) = \int_{-\infty}^{\infty} \theta \pi(\theta|y) d\theta = \int_0^1 Q_{\theta|y}(z) dz$$

We can then use the trapezoidal rule and obtain an efficient approximation due to the lemma of [45].

Proposition A1. *Assume $Q(x)$ is a function with a continuous second derivative on $[0,1]$ and*

$$\theta \equiv \int_0^1 Q(x) dx.$$

If $Y_0 \equiv 0, Y_{n+1} \equiv 1$, and $\{Y_i\}_{i=1}^n$ is the ordered sample associated with n independent uniform observations on [0, 1], then for the estimator

$$\theta_n = \frac{1}{2}\left[\sum_{i=0}^{n} (Q(Y_i) + Q(Y_{i+1}))(Y_{i+1} - Y_i)\right],$$

we have that for some constant M,

$$E[(\theta_n - \theta)^2] \leq M/n^4, \qquad \text{for all } n \geq 1.$$

Proof. For any particular sample $Y_1, Y_2, ..., Y_n$, we have

$$\theta - \theta_n = \sum_{j=1}^{n} \left(\int_{Y_j}^{Y_{j+1}} Q(t) dt - \frac{1}{2}[(Q(Y_j) + Q(Y_{j+1}))(Y_{j+1} - Y_j)] \right).$$

A well-known error bound for the trapezoidal rule is that for any number a, b, such that $0 \leq a < b \leq 1$,

$$\int_a^b Q(t) dt - \frac{b-1}{2}[Q(b) + Q(a)] = -\frac{(b-a)^3}{12} Q''(\xi),$$

where $a \leq \xi \leq b$. Using this result, we have that if $C \geq |Q''(x)|$, $0 \leq x \leq 1$, then

$$\begin{aligned} |\theta - \theta_n| &\leq \left| \int_0^{Y_1} Q(t)dt - \frac{1}{2} Y_1 (Q(0) + Q(1)) \right| \\ &+ \left| \left[\sum_{i=1}^{n-1} \int_{Y_i}^{Y_{i+1}} Q(t)dt - \frac{1}{2}[(Q(Y_i) + Q(Y_{i+1}))(Y_{i+1} - Y_i)] \right] \right| \\ &+ \left| \int_{Y_n}^{1} Q(t)dt - \frac{1}{2}[(Q(Y_n) + Q(Y_1))(1 - Y_n) \right| \\ &\leq \frac{C}{12} \sum_{i=1}^{n+1} Z_i^3, \end{aligned}$$

where Z_i are as defined below.

Let $\{Y_i\}_{i=1}^n$ be the ordered sample constructed from n independent, uniform observations on the unit interval. Define

$$Z_1 = Y_1, \qquad Z_j = Y_j - Y_{j-1}, \qquad 2 \leq j \leq n, \text{ and } Z_{n+1} = 1 - Y_n.$$

Ref. [45] provides the following result:

Let $\{Y_i\}_{i=1}^n$ be the ordered sample constructed from n independent, uniform observations on the unit interval, then

$$E[Z_i^6] = 6!n!/(n+6)!, \qquad i = 1, 2, ..., n+1,$$

$$E[Z_i^3 Z_j^3] = (3!)^2 n!/(n+1)!, \qquad i, j = 1, 2, ..., n+1, \qquad i \neq j.$$

Consequently, letting i and j range from 1 to $n+1$, we have

$$E[(\theta_n - \theta)^2] \leq E\left[\left(\frac{C}{12} \sum_i Z_i^3 \right)^2 \right] = \frac{C^2}{144} \left[\sum_i E[Z_i^6] + \sum_{i \neq j} E[Z_i^3 Z_j^3] \right] = O\left(n^{-4}\right).$$

□

References

1. Dixon, M.F.; Polson, N.G.; Sokolov, V.O. Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. *Appl. Stoch. Models Bus. Ind.* **2019**, *35*, 788–807. [CrossRef]
2. Heaton, J.B.; Polson, N.G.; Witte, J.H. Deep learning for finance: Deep portfolios. *Appl. Stoch. Models Bus. Ind.* **2017**, *33*, 3–12. [CrossRef]
3. Sokolov, V. Discussion of 'deep learning for finance: Deep portfolios'. *Appl. Stoch. Models Bus. Ind.* **2017**, *33*, 16–18. [CrossRef]
4. Dabney, W.; Rowland, M.; Bellemare, M.G.; Munos, R. Distributional Reinforcement Learning with Quantile Regression. *arXiv* **2017**, arXiv:1710.10044. [CrossRef]
5. Dabney, W.; Ostrovski, G.; Silver, D.; Munos, R. Implicit Quantile Networks for Distributional Reinforcement Learning. In Proceedings of the 35th International Conference on Machine Learning (PMLR), Stockholm, Sweden, 10–15 July 2018; pp. 1096–1105.
6. Lindley, D.V. A Class of Utility Functions. *Ann. Stat.* **1976**, *4*, 1–10. [CrossRef]
7. Zammit-Mangion, A.; Sainsbury-Dale, M.; Huser, R. Neural Methods for Amortised Inference. *arXiv* **2024**, arXiv:2404.12484.
8. Sainsbury-Dale, M.; Zammit-Mangion, A.; Huser, R. Likelihood-Free Parameter Estimation with Neural Bayes Estimators. *Am. Stat.* **2024**, *78*, 1–14. [CrossRef]
9. Nareklishvili, M.; Polson, N.; Sokolov, V. Generative Causal Inference. *arXiv* **2023**, arXiv:2306.16096.
10. Polson, N.G.; Sokolov, V. Generative AI for Bayesian Computation. *arXiv* **2024**, arXiv:2305.14972.
11. Müller, P.; Parmigiani, G. Optimal Design via Curve Fitting of Monte Carlo Experiments. *J. Am. Stat. Assoc.* **1995**, *90*, 1322–1330.
12. Wang, Y.; Kaji, T.; Ročková, V. Approximate Bayesian Computation via Classification. *J. Mach. Learn. Res.* **2022**, *23*, 1–49.
13. Wang, Y.; Ročková, V. Adversarial Bayesian Simulation. *arXiv* **2022**, arXiv:2208.12113.
14. Kallenberg, O. *Foundations of Modern Probability*; Springer: Berlin/Heidelberg, Germany, 1997; Volume 2.
15. Brillinger, D.R. A Generalized Linear Model with "Gaussian" Regressor Variables. In *Selected Works of David Brillinger*; Guttorp, P., Brillinger, D., Eds.; Selected Works in Probability and Statistics; Springer: New York, NY, USA, 2012; pp. 589–606.

16. Belkin, M.; Rakhlin, A.; Tsybakov, A.B. Does Data Interpolation Contradict Statistical Optimality? In Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, PMLR, Okinawa, Japan, 16–18 April 2019; pp. 1611–1619.
17. Bach, F. High-Dimensional Analysis of Double Descent for Linear Regression with Random Projections. *SIAM J. Math. Data Sci.* **2024**, *6*, 26–50. [CrossRef]
18. Jiang, B.; Wu, T.Y.; Zheng, C.; Wong, W.H. Learning Summary Statistic For Approximate Bayesian Computation via Deep Neural Network. *Stat. Sin.* **2017**, *27*, 1595–1618.
19. Albert, C.; Ulzega, S.; Ozdemir, F.; Perez-Cruz, F.; Mira, A. Learning Summary Statistics for Bayesian Inference with Autoencoders. *SciPost Phys. Core* **2022**, *5*, 043. [CrossRef]
20. Beaumont, M.A.; Zhang, W.; Balding, D.J. Approximate Bayesian Computation in Population Genetics. *Genetics* **2002**, *162*, 2025–2035. [CrossRef]
21. Papamakarios, G.; Murray, I. Fast ϵ-Free Inference of Simulation Models with Bayesian Conditional Density Estimation. *Adv. Neural Inf. Process. Syst.* **2018**, *29*. Available online: https://proceedings.neurips.cc/paper/2016/hash/6aca97005c68f1206823815f66102863-Abstract.html (accessed on 28 November 2024).
22. 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 (PMLR), Okinawa, Japan, 16–18 April 2019; pp. 837–848.
23. Schmidt-Hieber, J. Nonparametric Regression Using Deep Neural Networks with ReLU Activation Function. *Ann. Stat.* **2020**, *48*, 1875–1897.
24. Gutmann, M.U.; Cor, J. Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models. *J. Mach. Learn. Res.* **2016**, *17*, 1–47.
25. Skilling, J. Nested Sampling for General Bayesian Computation. *Bayesian Anal.* **2006**, *1*, 833–859. [CrossRef]
26. Polson, N.G.; Scott, J.G. Vertical-Likelihood Monte Carlo. *arXiv* **2015**, arXiv:1409.3601.
27. Padilla, O.H.M.; Tansey, W.; Chen, Y. Quantile Regression with ReLU Networks: Estimators and Minimax Rates. *J. Mach. Learn. Res.* **2022**, *23*, 247:11251–247:11292.
28. Bos, T.; Schmidt-Hieber, J. A Supervised Deep Learning Method for Nonparametric Density Estimation. *arXiv* **2024**, arXiv:2306.10471.
29. White, H. Nonparametric Estimation of Conditional Quantiles Using Neural Networks. In Proceedings of the Computing Science and Statistics, Pittsburgh, PA, USA, 24–27 October 1992; Page, C., LePage, R., Eds.; Springer: New York, NY, USA, 1992; pp. 190–199.
30. Yaari, M.E. The Dual Theory of Choice under Risk. *Econometrica* **1987**, *55*, 95–115. [CrossRef]
31. Jacquier, E.; Polson, N.G. Asset Allocation in Finance: A Bayesian Perspective. In *Hierarchinal Models and MCMC: A Tribute to Adrian Smith*; Oxford Academic: Oxford, UK, 2012; pp. 56–59.
32. DeGroot, M.H. *Optimal Statistical Decisions*; John Wiley & Sons: Hoboken, NJ, USA, 2005.
33. Montanelli, H.; Yang, H.; Du, Q. Deep ReLU Networks Overcome the Curse of Dimensionality for Bandlimited Functions. *arXiv* **2020**, arXiv:1903.00735.
34. Shen, G.; Jiao, Y.; Lin, Y.; Horowitz, J.L.; Huang, J. Deep Quantile Regression: Mitigating the Curse of Dimensionality Through Composition. *arXiv* **2021**, arXiv:2107.04907.
35. Smith, A.F.M.; Gelfand, A.E. Bayesian Statistics without Tears: A Sampling-Resampling Perspective. *Am. Stat.* **1992**, *46*, 84–88.
36. Lopes, H.F.; Polson, N.G.; Carvalho, C.M. Bayesian Statistics with a Smile: A Resampling-Sampling Perspective. *Braz. J. Probab. Stat.* **2012**, *26*, 358–371. [CrossRef]
37. Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [CrossRef]
38. Sutton, R.S. Reinforcement learning: An introduction. In *A Bradford Book*; MIT Press: Cambridge, MA, USA, 2018.
39. Fortini, S.; Ruggeri, F. Concentration Functions and Bayesian Robustness. *J. Stat. Plan. Inference* **1994**, *40*, 205–220. [CrossRef]
40. Kruglov, V.M. Concentration Functions. In *Selected Works of A. N. Kolmogorov: Volume II Probability Theory and Mathematical Statistics*; Shiryayev, A.N., Ed.; Mathematics and Its Applications (Soviet Series); Springer: Dordrecht, The Netherlands, 1992; pp. 571–574.
41. Fortini, S.; Ruggeri, F. Concentration Function and Sensitivity to the Prior. *J. Ital. Stat. Soc.* **1995**, *4*, 283–297. [CrossRef]
42. Girsanov, I.V. On Transforming a Certain Class of Stochastic Processes by Absolutely Continuous Substitution of Measures. *Theory Probab. Its Appl.* **1960**, *5*, 285–301. [CrossRef]
43. Parzen, E. Quantile Probability and Statistical Data Modeling. *Stat. Sci.* **2004**, *19*, 652–662. [CrossRef]
44. Soyer, R.; Tanyeri, K. Bayesian portfolio selection with multi-variate random variance models. *Eur. J. Oper. Res.* **2006**, *171*, 977–990. [CrossRef]
45. Yakowitz, S.; Krimmel, J.E.; Szidarovszky, F. Weighted Monte Carlo Integration. *SIAM J. Numer. Anal.* **1978**, *15*, 1289–1300. [CrossRef]