Some Results Regarding the Estimation of Densities and Random Variate Generation Using Neural Networks

Malik Magdon-Ismail
Dept. of Computer Science
Rensselaer Polytechnic Institute, Lally 207
110 8th Street
Troy, NY 12180
magdon@cs.rpi.edu

Amir Atiya
Learning Systems Group
Dept Electrical Engineering
California Institute of Technology, MS 136-93
Pasadena, CA 91125
amir@work.caltech.edu

CaltechCSTR/2000.005
Submission Date: September 8, 2000.

Abstract

In this paper we consider two important topics: density estimation and random variate generation. We will present a framework that is easily implemented using the familiar multilayer neural network. First, we develop two new methods for density estimation, a stochastic method and a related deterministic method. Both methods are based on approximating the distribution function, the density being obtained by differentiation.

In the second part of the paper, we develop new random number generation methods. Our methods do not suffer from some of the restrictions of existing methods in that they can be used to generate numbers from an arbitrary density provided that certain smoothness conditions are satisfied. One of the methods is based on an observed inverse relationship between the density estimation process and the random number generation process. We present two variants of this method – a stochastic and a deterministic version. We propose a second method that is based on formulating the task as a control problem, where a "controller network" is trained to shape a given density into the desired density.

We justify the use of all the methods that we propose by providing theoretical convergence results. In particular, we prove that the L_∞ convergence to the true density for both the density estimation and random variate generation techniques occurs at a rate $O((\log\log N/N)^{(1-\epsilon)/2})$ where N is the number of data points and ϵ can be made arbitrarily small for sufficiently smooth target densities. This bound is very close to the optimally achievable convergence rate under similar smoothness conditions. Also, for comparison, the L_2 (RMS) convergence rate of a *positive* kernel density estimator is $O(N^{-2/5})$ when the optimal kernel width is used.

We present numerical simulations to illustrate the performance of the proposed density estimation and random variate generation methods. In addition, we present an extended introduction and bibliography that serves as an overview and reference for the practitioner.

Keywords: density estimation, random number generation, distribution function, multilayer network, neural network, estimation error, convergence rate, stochastic algorithms.

1 Introduction

A majority of problems in science and engineering have to be modeled in a probabilistic manner. Even if the underlying phenomena are inherently deterministic, the complexity of these phenomena often makes a probabilistic formulation the only feasible approach from the computational point of view. Consequently, tools from probability theory have become very valuable in the characterization, analysis, and solution of many such problems. Although quantities such as the mean, the variance, and possibly higher order moments of a random variable have often been sufficient to characterize a particular problem, the quest for higher modeling accuracy, and for more realistic assumptions drives us towards modeling the available random variables using their probability density. This of course leads us to the problem of density estimation (see [56]). Density estimation has been a very active research topic in the past three decades. There has always been competition among researchers to invent more accurate density estimation techniques, since this will significantly impact the applications that rely on density estimation as an essential component. Examples of applications that need a density estimation step include the following: the application of the optimal pattern classification procedure, the Bayes procedure (see [14], and [25]), the determination of the optimal detection threshold for the signal detection problem [72], time series prediction applications where the density for a future data point is required rather than a single forecast for the value (see [78], [69]), clustering for unsupervised classifier design [25], the design of optimal scalar quantizers for the signal encoding problem [31], and finding estimates of confidence intervals and quantile levels for the parameter estimation and regression problems [40].

1.1 Existing Density Estimation Techniques

Traditional density estimation methods can be grouped into two broad categories (see [56], [14], and [25]). The first of these is the parametric approach. It assumes that the density has a specific functional form, such as a Gaussian or a mixture of Gaussians. The unknown density is estimated by using the data to obtain estimates for the parameters of the functional form. Typically, the parameters are estimated using maximum likelihood or Bayesian techniques. The drawback of the parametric approach is that the functional form of the density is rarely known beforehand, and the commonly assumed Gaussian or mixture of Gaussian models rarely fit densities that are encountered in practice.

The more common approach for density estimation is the nonparametric approach, where no functional form for the underlying density is assumed. Rather than expressing the density as a specific parametric form, it is determined according to a formula involving the data points available. The most common nonparametric method is the kernel density estimator, also known as the Parzen window estimator [48]. The density is estimated by summing "kernel functions" centered at each data point. A typical kernel function is the "Gaussian bump." The problem with the kernel method is its extreme sensitivity to the choice of the kernel width, which acts as regularization parameter. A wrong choice can lead to either under-smoothing or over-smoothing. Methods to estimate the kernel width are either asymptotic and given in terms of the unknown density, thus impractical, or are based on cross-validation, thus prone to statistical error. Another drawback of the kernel method is that it has the tendency to exhibit bumpy behavior at the tails [56]. If the bumps are smoothed out by increasing the kernel width, then essential detail in the main part of the density is masked out or the width of the main part of the density will be exaggerated.

The other common nonparametric approach is the k-nearest neighbor technique [30]. In this approach, the density at a particular point x is estimated as inversely proportional to the volume of the hypersphere centered at x, with radius equal to the Euclidean distance to the k^{th} nearest neighbor. The k-nearest neighbor approach shares some of the drawbacks of the kernel density estimator, such as the difficulty of obtaining the best smoothing parameter k. In addition, the density estimate is not a smooth function, has a very heavy tail, and integrates to infinity on non-compact sets. One of the advantages of the kernel and k-nearest neighbor methods is their ease of implementation.

A number of other nonparametric or semiparametric methods have been proposed in the literature. For example, penalized likelihood methods (see for example [57], [22]) penalize the likelihood by some regularizing functional. Orthogonal expansions [19], [76], and wavelet based methods [43], [16] obtain density estimates by expanding in some suitable basis. Complexity based approaches [8], [50] are similar to penalized likelihood methods in that they try to achieve a compromise between likelihood and simplicity. Methods using families of exponentials represent the density in terms of some exponential family (see for example [10]). Histospline approaches use splines to fit the distribution function, and obtain the density by differentiation, [53], [77], [75], [76].

There have been a number of methods on using neural networks for density estimation. The majority of the approaches tend to be parametric in nature, and therefore share many of the limitations of the parametric approach discussed above. For example, Traven's approach [70] and Cwik and Koronacki's method [21] are based on mixture of Gaussian density estimation method. Bishop and Legleye [15], Bishop [13], and Husmeier and Taylor [35], [36], consider a mixture of Gaussians model where the mean and variance are estimated using a multilayer network. Williams [79], proposes a similar approach for the multidimensional case. Schioler and Kulczyki's method [52] is based on the kernel estimation method. Roth and Baram [51], and Miller and Horn [44] developed elegant methods by training networks to maximize the entropy of the outputs. Van Hulle [71] developed an interesting technique based on a self–organizing approach, whereby the algorithm converges to a solution, such that at any point, the density of the weight vectors is an estimate of the unknown density. Martinez [42] used a competitive learning tree to create equiprobable quantizations of the input space. Modha and Fainman [45] proposed a multilayer network, that is trained by maximizing the log-likelihood of the data. Weigend and Srivastava [78] developed a fractional binning approach, developed in the context of time series prediction that is based on partitioning the input space into bins, assigning a softmax output neuron for every bin, and training the network on the fraction of data falling in each respective bin. Smyth and Wolpert [59] suggested a method for combining the estimates of several density estimators (such as kernel estimators or mixture of Gaussian estimators). Zeevi and Meir [81] proposed the use of convex combinations of density estimators and analyzed their estimation error. Neural network based methods have also been developed for estimating discrete distributions, see for example Thathachar and Arvind [66], and Adali et al. [3].

1.2 Overview of the New Density Estimation Techniques

First we will propose two new methods for density estimation using multilayer networks. The approaches can be considered as semi-parametric models. The reason is that the model is described in terms of a number of parameters (the weights of the network), rather than the actual data points, but at the same time the number of parameters can be increased in a way that would ultimately achieve an all-powerful model capable of approximating any (continuous) density function. One important advantage of multilayer networks over local methods such as the kernel estimator is that they have been shown in theory and in practical problems to be superior for high-dimensional problems. For example Barron et al. [7] and Hornik et al. [33], [34] show that a neural networks ability to simultaneously approximate a function and its derivatives does not decay with increasing input dimension. Modha and Masry [46] obtain a similar result for the case of neural networks approximating density functions. (in both cases, the coefficient

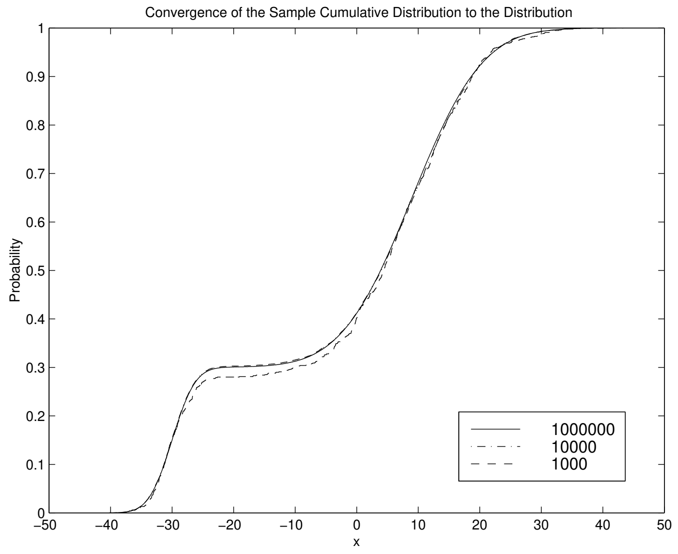


Figure 1: The convergence of the sample distribution function. The sample distribution function for a mixture of two Gaussians is shown for 1000, 10000 and 1000000 data points. Notice that the curve for 10000 data points is essentially overlapping the 1000000 data points curve.

of convergence may depend on the dimension). Further, multilayer networks give us the flexibility to choose an error function to suit our application. The methods developed here are based on approximating the distribution function, in contrast to most previous works which focus on approximating the density itself. Straightforward differentiation then gives us the estimate of the density function. The distribution function is often useful in its own right - one can directly evaluate quantiles or the probability that the random variable occurs in a particular interval. Although the methods by Wahba [77], [75] are based on approximating the distribution function, our approach is quite different and obtains a better convergence rate for the error than the cubic spline techniques presented there.

Figures 1 and 2 indicate the essential intuition behind why one might focus attention towards learning the distribution function rather than the density. One can see the higher sensitivity of the density estimator to finite sample size and the bin width (smoothness parameter). Too small a bin width increases the sensitivity to the 'noise' due to the small sample size. A larger bin width will smooth out this noise at the expense of a loss of information. The distribution estimator, however, does not appear as sensitive to the small sample size, and no smoothness parameter needs to be chosen. The regularization that is controlled by the bin width (or kernel width) is embedded in the integral operator that is used in passing from the histogram to the distribution.

We develop two methods for the estimation of densities by first learning the cumulative distribution function and then taking the derivative of the resulting function. The first method is based on a stochastic algorithm (SLC), and the second method is a deterministic technique based on learning the cumulative (SIC). We show that these two techniques yield equivalent results. The stochastic technique will generally be smoother on smaller numbers of data points, however, the deterministic technique is faster and applies to the multi-dimensional case.

We analyze the consistency and the convergence rate of the estimation error for our methods in the univariate case. When the unknown distribution is bounded and has bounded derivatives up to order K, we prove that the L_∞ estimation error is $O((\log\log N/N)^{\frac{K-1}{2K}})$, where N is the number of data points. As a comparison, for the kernel density estimator (with non-negative kernels) and the k^{th} nearest neighbor density estimator, the L_2 (RMS)

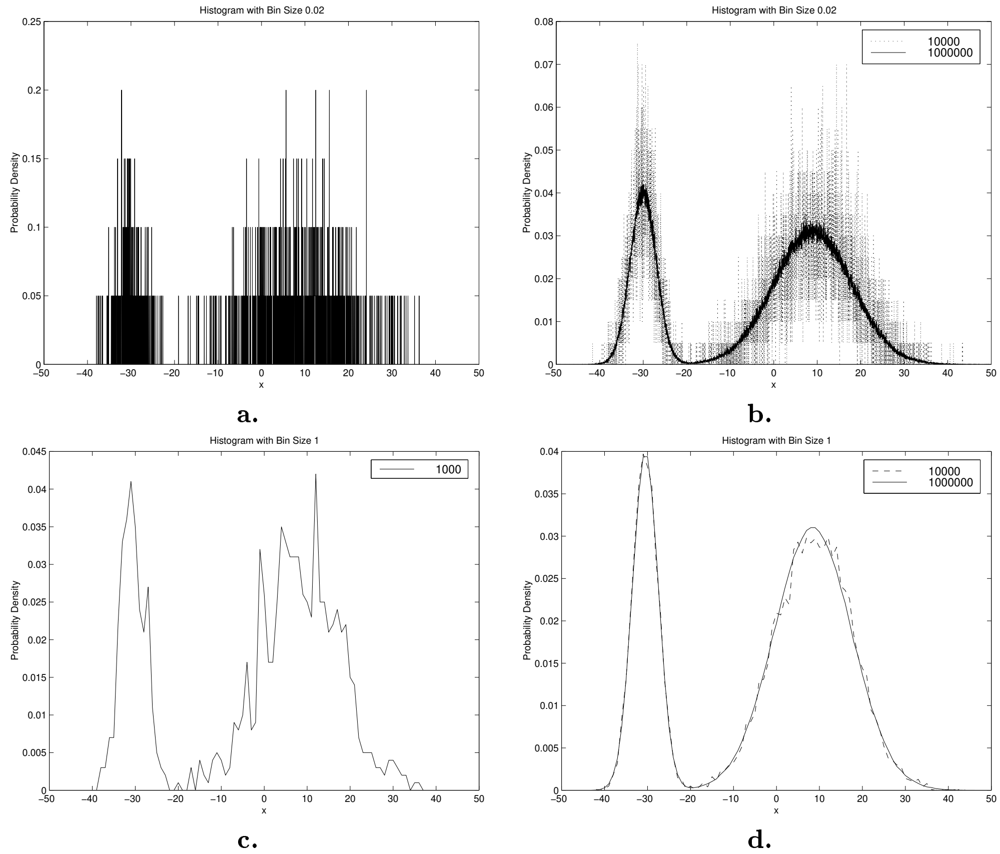


Figure 2: Convergence of the histogram estimator. The histograms of data drawn from data drawn from a mixture of two Gaussians. (a) and (b) are for a bin width of 0.02, and (c) and (d) are for a bin width of 1. The histograms for 1000, 10000, 1000000 points are shown. Notice the extreme sensitivity to the bin width.

estimation error is $O(N^{-2/5})$, under the assumptions that the unknown density has a square integrable second derivative (see [56]), and that the *optimal* kernel width is used, which is not possible in practice because computing the optimal kernel width requires knowledge of the true density. One can see that for smooth density functions with bounded derivatives, our methods achieve an L_∞ error rate that is better than $O((1/N)^{(1-\epsilon)/2})$ for any $\epsilon > 0$.

We have found in the literature only the following methods that achieve comparable error rates. The first is a kernel estimator with the specific choice of kernel that possesses zero l^{th} order moments for $l = 1, ..., L$, L being even, [56]. For such kernels, the L_2 (RMS) convergence rate is $O(N^{-L/(2L+1)})$. The problem, however, with this method is that the kernel has negative parts, and hence the resulting density estimate can be negative at some points and could therefore be an illegitimate density function. Because of this problem, this method is impractical and is not commonly used. The other method is the adaptive kernel estimator, [56]. The L_2 (RMS) error rate is theoretically $O(N^{-4/9})$ for this method. However, the error estimate for this method is based on using the optimum smoothing function, which is given in terms of the unknown density and its derivatives. Even if one uses pilot estimates of these unknown characteristics of the true density to determine the smoothing function, the resulting statistical error will leave a residual $O(N^{-2/5})$ term that cannot entirely cancel out. In short, no *positive* kernel estimator can have a faster L_2 (RMS) rate of convergence than $O(N^{-2/5})$ [10]. The error estimate of our method, in contrast to these methods, does not rely on optimizing parameters based on the unknown density (other than knowing K). A class of semi-parametric methods that achieve an L_∞ error rate of $O((\log N/N)^{\frac{K-1}{2K-1}})$ and L_2 error rate of $O((1/N)^{\frac{K-1}{2K-1}})$ have also been proposed (see [64], [10], [17], [28], [53]), where K is the maximum k for which $\int |F^k| < \infty$ where F is the unknown distribution function. . These methods have been shown to achieve the optimal convergence rates on *compact* sets (see [28], [27], [26], [60], [61], [62], [63]). Thus, our proposed methods achieve a convergence rate very close to the optimally achievable rate.

1.3 The Random Variate Generation Problem

In the second part of the paper, we consider the problem of generating a random variable according to a given density function. The topic of random variate generation has been a vast research topic in the past three decades. In many scientific problems in areas such as physics, biology, engineering, and economics it is now more the rule than the exception to be confronted with problems where analytic solutions are not possible and simulating the given problem would be the only feasible approach. Problems which possess some random element will necessitate an accurate and fast random variable generation procedure for the Monte Carlo simulation to be efficiently performed. Examples of applications include the simulation of a biological system, such as how nerve cells interact together, the simulation of a communication network where the packets are assumed to arrive according to a particular density function, the simulation of a control system where the plant disturbance is a random variable, and the simulation of the stock market in an attempt to estimate a valuation for some derivative instrument. In the past, the assumption of standard density functions such as Gaussian or exponential densities for these random variables often allowed analytical solutions for these problems. But now, with more accuracy needed, the next step has been to assume more realistic density functions for the random variables (possibly estimated from the data).

1.3.1 Existing Random Variate Generation Techniques

Unfortunately, there are very few techniques that can generate a random variable possessing an arbitrary density function (see [49] and [23] for reviews of the different approaches). One well known method is the transformation method: a uniformly distributed random number x is generated, and this number is passed through a nonlinear transformation $y = G^{-1}(x)$, where $G(x)$ is the given (cumulative) distribution function. The problem with this method is that in the majority of cases $G^{-1}(x)$ (or even $G(x)$) is not known analytically. Solving for $G^{-1}(x)$ numerically is not practical since speed is of the essence for the random number generation problem. In Monte Carlo simulations, random variables are typically generated millions of times, in order to obtain statistically reliable results. There are many variations of the transformation method, such as obtaining transformations of several random variables (see [38] and [37]). These methods are largely ad hoc, and can be used to generate only some of the well-known density functions such as Gaussian, Gamma, Beta, etc.

The other method for random variable generation is the rejection method. It is a more general method than the transformation method. To use the rejection method, a comparison function $h(x)$ that dominates $g(x)$ everywhere is required. A point ($p = [x, y]$) is generated uniformly in the two-dimensional region bounded by the comparison function and the x-axis. For this purpose, H^{-1} is needed where $H(z) = \int_{-\infty}^{z} h(t)\, dt$. The point p will be accepted or rejected based on the realization of a second uniform in $[0, 1]$ random variable (u) as follows: u is compared to $g(x)/h(x)$ and the point p is accepted if $u < g(x)/h(x)$ and rejected otherwise. The problem with the rejection method is that for some large-tail densities it is not possible to find a comparison function with an analytically invertible distribution function (a necessary condition for the method) that is everywhere larger than the given density. Therefore, it cannot be guaranteed that this method is capable of generating random variates from an arbitrary density. The acceptance rate is $1/A$ where A is the area under $h(x)$. The acceptance rate can be small (~ 0.01), thus slowing down the random variate generation process. Further, if the functional form of the given density is not known and it is only characterized by a finite set of data points, then it becomes a computationally extensive procedure and one has to resort to other techniques. One such technique would be to form a kernel density estimate and generate the points from that estimate. This approach, however, is prone to statistical error in the estimate of the density, which is of order $N^{-2/5}$ for the L_2 (RMS) error and therefore fairly high.

1.3.2 New Random Variate Generation Techniques

Our goal is to develop several new random variate generating techniques using multilayer networks. The methods we propose can be used to generate points from a distribution given by a functional form or given by a set of data points drawn from that distribution. The first method (for which we develop stochastic and deterministic versions) is based on viewing random number generation as the inverse process of the density estimation (this inverse or dual nature of density estimation and random variate generation has also been pointed out by [51], [44]). One of the advantages of our method is that a density estimation step need not be performed. We also propose a method that formulates the problem within a "control" framework. A cascade structure is proposed, that consists of a density shaping network (acting as the "controller"), and a density estimation network (acting as the "plant"). For the methods that we propose, the bulk of the time is spent in "learning" to generate the random variates. Once this learning is done, the actual generation of these random numbers is fast and can thus be used for efficient Monte Carlo simulations.

In the remainder of the paper we develop and theoretically justify the new methods we propose: In section 2, we develop the two new density estimation techniques in detail. In section 3, we present the convergence results of the proposed density estimators. In section 4, we describe the new random number generating techniques followed by their convergence properties in section 5. In section 6, we present the simulation results for both the density estimation techniques and the random variate generation techniques. We conclude with a brief discussion in section 7. We present the proofs of the convergence theorems in the appendix.

2 New Density Estimation Techniques

In this section we will propose two new methods for density estimation. For both methods, one can use a standard multilayer network, such as a one-hidden-layer network. We will use neural networks throughout for illustrative purposes, but stress that any sufficiently general class of functions will do just as well. The network's output will represent an estimate of the distribution function, and its derivative will be an estimate of the density.

2.1 SLC (Stochastic Learning of the Cumulative)

Let $x_n \in \mathbf{R},\ n = 1, ..., N$ be the given data points. Let the underlying density be $g(x)$ and its distribution function $G(x) = \int_{-\infty}^{x} g(t)dt$. We assume the density to be continuous and have continuous derivatives of all orders. Let the neural network output be $H(x, w)$, where w represents the set of weights of the network. Ideally, after training the neural network, we would like to have

$$H(x, w) = G(x). \tag{1}$$

Learning requires a set of targets. To determine a target for the network training, we make the following observation. The density of the variable $y \equiv G(x)$ (x being generated according to $g(x)$) is uniform in $[0, 1]$. The reason is as follows. Let $g_Y(y)$ represent the density of y. Using the well known formula for transformation of random variables,

$$g_Y(y) = \frac{g(x)}{|\frac{dy}{dx}|} = \frac{g(x)}{|\frac{dG(x)}{dx}|} = 1 \qquad \text{for } 0 \le y \le 1, \tag{2}$$

and $g_Y(y) = 0$ for $y < 0$ or $y > 1$. We used the fact that $g(x) = dG(x)/dx$. Thus, if $H(x, w)$ is to be as close as possible to $G(x)$, then the network output should have a density as close as possible to uniform in $[0, 1]$. This is what our goal will be. We will attempt to train the network such that its output density is uniform. Having achieved this goal, the network mapping should represent the distribution function $G(x)$. A somewhat similar philosophy is used in [51] and [44], where they show that their entropy maximization method implies transformation to a uniform density (and also implies the necessity to maximize the expected logarithm of the determinant of the transformation Jacobian). We present here (and fully analyze) a different and more direct method for obtaining a mapping to a uniform distribution than the methods proposed in [51], [44].

The basic idea behind the proposed algorithm is to use the N data points drawn from the unknown density as inputs to the network. For every training cycle we generate a different set of N network targets randomly from a uniform distribution in $[0, 1]$, and adjust the weights to map the data points (sorted in ascending order) to these generated targets (also sorted in ascending order). Thus we are training the network to map the data to a uniform distribution.

Before describing the steps of the algorithm, we note that the resulting network has to represent a monotonically nondecreasing mapping, otherwise it will not represent a legitimate distribution function. Let $M = \{w | \partial H(x, w)/\partial x \ge 0, \ \forall x\}$. Then we wish to pick $w \in M$, such that $H(x, w)$ is as close as possible to $G(x)$. In practice, enforcing monotonicity could be done by using a class of "monotonic networks" [55], or else by using *hint* penalty terms [2], [1]. Hints are auxiliary information or constraints on the target function, that are known *a priori* (independent of the training examples). By using hints in the form of penalty terms, we can guide the learning process, and obtain a network that satisfies the hints. In our simulations, we used the latter approach of adding a term that penalizes non-monotone mappings to the error function. The proposed algorithm is as follows.

1. Let $x_1, x_2, ..., x_N$ be the points drawn from the unknown density. Without loss of generality assume the points are sorted in ascending order: $x_1 \le x_2 \le \cdots \le x_N$.

2. Set $t = 1$, where t is the training cycle number. Initialize the weights randomly to $w(1)$.

3. Generate randomly from a uniform distribution in $[0, 1]$ another N points $u'_1, u'_2, ..., u'_N$. These are the network targets for this cycle.

4. Sort the targets in ascending order, i.e. $u_1 \le u_2 \le ... \le u_N$ (where we have renamed the ordered targets u_n). Then, the point u_n is the target output for x_n.

5. Adjust the network weights according to the backpropagation scheme:

$$w(t+1) = w(t) - \eta(t) \frac{\partial \mathcal{E}(w)}{\partial w} \tag{3}$$

   where $\mathcal{E}$ is the objective function that includes the error term and the monotonicity hint penalty term:

$$\mathcal{E}(w) = \sum_{n=1}^{N} \Big[H(x_n, w) - u_n\Big]^2 + \lambda \sum_{k=1}^{N_h} \Theta\Big(H(y_k, w) - H(y_k + \Delta, w)\Big) \Big[H(y_k, w) - H(y_k + \Delta, w)\Big]^2 \tag{4}$$

The second term is the monotonicity penalty term, λ is a positive weighting constant, Δ is a small positive number, $\Theta(x)$ is the familiar unit step function, and the y_k's are any set of points where we wish to enforce the monotonicity. Because the hint is known to be true, λ can be chosen very large.

6. Set $t = t + 1$, and go to step 3 to perform another cycle until the error is small enough.

7. Upon convergence, the density estimate becomes

$$\hat{g}(x) = \frac{\partial H(x, w)}{\partial x} \tag{5}$$

Note that as presented, the randomly generated targets are different for every cycle, which will have a smoothing effect that will allow convergence to a truly uniform distribution. One other version, that we have implemented in our simulation studies, is to generate new targets after every fixed number L of cycles, rather than every cycle. This generally improves the speed of convergence as there is more "continuity" in the learning process. Also note that it is preferable to choose the activation function for the output node to be in the range of 0 to 1, to ensure that the estimate of the distribution function is in this range. A sample run of how SLC performs on 100 data points drawn from a mixture of two Gaussians is shown in figure 6 of section 6.1.

SLC is only applicable to estimating univariate densities. The reason is that for the multivariate case, the nonlinear mapping $y = G(x)$ will not necessarily result in a uniformly distributed output y. Fortunately, many, if not the majority of problems encountered in practice are univariate. This is because multivariate problems, with even a modest number of dimensions, need a huge amount of data to obtain statistically accurate results. Our second method, described next, is applicable to the multivariate case as well.

2.2 SIC (Smooth Interpolation of the Cumulative)

Again, we have a multilayer network, to which we input the point $\mathbf{x}$, and the network outputs the estimate of the distribution function. Let $g(\mathbf{x})$ be the true density function, and let $G(\mathbf{x})$ be the corresponding distribution function. Let $\mathbf{x} = (x^1, ..., x^d)^T$. The distribution function is given by

$$G(\mathbf{x}) = \int_{-\infty}^{x^1} \cdots \int_{-\infty}^{x^d} g(\mathbf{x}) dx^1 \cdots x^d \tag{6}$$

A straightforward estimate of $G(\mathbf{x})$ could be the fraction of data points falling in the area of integration:

$$\hat{G}(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \Theta(\mathbf{x} - \mathbf{x}_n), \tag{7}$$

where Θ is defined as

$$\Theta(\mathbf{x}) = \begin{cases} 1 & \text{if } x^i \geq 0 \text{ for all } i = 1, \ldots, d, \\ 0 & \text{otherwise.} \end{cases}$$

The statistical properties of the estimate (7) have been widely studied ([54], [29]). For the method we propose, we will use such an estimate for the target outputs of the neural network.

The estimate given by (7) has a staircase-like shape if plotted against $\mathbf{x}$, and thus is discontinuous. The neural network method developed here provides a smooth, and hence more realistic estimate of the distribution function. Further, the density can be obtained by differentiating the output of the network with respect to its inputs.

For the low-dimensional case, we can uniformly sample (7) using a grid, to obtain the examples for the network. Beyond two or three dimensions, this is not feasible of course because of computational considerations. One idea is to sample the input space randomly (using say a uniform distribution over the approximate range of $\mathbf{x}_n$'s), and for every point determine the network target according to (7). Another option is to use the data points themselves as examples. The target for a point $\mathbf{x}_m$ would then be

$$\hat{G}(\mathbf{x}_m) = \frac{1}{N-1} \sum_{n=1,\ n\neq m}^{N} \Theta(\mathbf{x}_m - \mathbf{x}_n). \tag{8}$$

This target is unbiased, i.e., $E\left[\hat{G}(\mathbf{x}_m)\right]_{\mathbf{x}_m} = G(\mathbf{x}_m)$. Another alternative would be to use $E\left[G(\mathbf{x}_m)\right]$. This expected value can be calculated only for the one dimensional case. This is because for the one dimensional case, the data points can be naturally ordered. Let x_m be the m^{th} order statistic of the data points. Then, one can show that

$$E\left[G(x_m)\right] = \frac{1}{N+1} \sum_{n=1}^{N} \Theta(x_m - x_n) = \frac{m}{N+1} \tag{9}$$

Thus for one dimension, we can use the targets represented in (9), however for more than one dimension we resort to (8).

For the one dimensional case, the algorithm is exactly as for SLC except for steps 3 and 4. Instead, the targets u_i are given by $\{u_i = i/(N+1)\}_{i=1}^{N}$. It is easy to see how the algorithm generalizes to the multi-dimensional case. Once again, we use monotonicity as a hint to guide the training. Once training is performed, and $H(\mathbf{x}, w)$ approximates $G(\mathbf{x})$, the density estimate can be obtained as

$$\hat{g}(\mathbf{x}) = \frac{\partial^d H(\mathbf{x}, w)}{\partial x^1 \cdots \partial x^d}. \tag{10}$$

We note that for a few dimensions, a derivation of the derivatives in (10) will be straightforward. For larger dimensions a numerical differentiation scheme such as the simple differencing method is more feasible. A sample run of how SIC performs on the same 100 data points used to run SLC is shown in figure 7 in section 6.1.

3 Convergence of the Density Estimation Techniques

In this section we derive the convergence properties of the proposed density estimation techniques. Our goal is to justify the methods introduced in the previous section by showing that convergence to the true density does occur. Further, we analyze the rate of convergence and compare with various other estimators. Due to the technical nature of such convergence issues, we will take this opportunity to summarize the essential details of the section.

First, we consider the stochastic method (SLC). The targets are uniform in $[0, 1]$. The expected value of the target u_i, the i^{th} order statistic of a uniform distribution, is $i/(N+1)$. Therefore we should expect the learned function to be approximately performing the mapping

$$x_i \to \frac{i}{N+1} \tag{11}$$

where x_i represents the i^{th} element of the ordered data set. But this is exactly the mapping we are trying to learn in SIC (smooth interpolation of $G(x)$). Thus we expect SLC to converge to a solution that is also a solution of SIC.

The formal statement and proof of this claim are the contents of section 3.1. For the proof, we will need some results from recursive stochastic approximation theory, which we will review briefly in the appendix.

Having shown that SLC$\rightarrow$SIC, we will restrict our analysis to SIC. First we define a set of functions which we call generalized sample distribution functions. These functions are exactly that set of functions that approximately interpolate the sample distribution function given by (9). We prove that this set of functions converges uniformly to the true cumulative distribution function in the L_1 and L_2 (RMS) sense. In fact, we will show that the rate of convergence is $O(N^{-1/2})$ for the L_1 and L_2 errors, and $O(\sqrt{\log\log(N)/N})$ for the L_∞ error. Further, because the neural network is an arbitrarily powerful class of functions, these generalized distribution functions can be implemented, therefore the neural network implementations we have discussed in the previous section also converge.

How about the convergence to the true density? Some assumptions have to be made about the true density. It is well known that the density estimation problem is an ill-posed problem[1]. Without any *a priori* assumptions on the true density function, it will be hard to judge the suitability of an estimator. Even the trivial estimator consisting of the summation of delta functions centered at the data points could be as valid as other estimators. In fact, [73], [9] and [24] show that no density estimator is consistent for certain types of error measure unless one makes some *a priori* assumptions about the distribution function. Typical *a priori* assumptions used in the density estimation literature are smoothness constraints on the class of considered densities. These are realistic assumptions that are usually obeyed by densities that are typically encountered in real world applications. We will consider such constraints in our analysis. We assume boundedness of the derivatives. We prove that when the true distribution function has K bounded derivatives, the L_∞ convergence rate is $O((\log\log(N)/N)^{(K-1)/2K})$ which as $K \rightarrow \infty$ is faster than $1/N^{(1-\epsilon)/2}$, $\forall\epsilon > 0$. For comparison, any positive kernel estimate has an L_2 (RMS) convergence rate of $N^{-2/5}$ using the *optimal* smoothing parameter, which is inaccessible in practice as it depends on some detailed properties of the true density. We then show that neural networks can achieve these same rates on compact sets by using the universal approximation results in [33], [34]. We will illustrate our convergence rate with simulations using neural networks.

3.1 Convergence of SLC to SIC

In this section we will analyze the stochastic method (SLC). Let $D = \{x_1, x_2, \ldots, x_N\}$ be the data points sorted in ascending order, and let $u_1(t), u_2(t), ..., u_N(t)$ be the output targets for training cycle t, where the $u_i(t)$'s are generated from a uniform density and then sorted in an ascending order. Ignore for the time being the effect of the hint penalty term in (4), as this term can only help convergence, and, if monotonicity is satisfied (as in the case of convergence to the true density), then the hint term will equal zero. The convergence behavior is given in the following theorem:

Theorem 3.1 *Let the data $x_1 \ldots x_N$ be generated according to the true distribution G. Then, SLC converges with probability 1 to a local minimum of the error function*

$$\mathcal{E} = \sum_{n=1}^{N} \left[H(x_n, w) - \frac{n}{N+1} \right]^2 \tag{12}$$

provided that the learning rate $\eta(t)$ is a decreasing sequence, that satisfies the following conditions

[1] A more detailed discussion of ill-posed problems can be found in [68], [67], [74]

a. $\sum_{t=1}^{\infty} \eta(t) = \infty$,

b. $\sum_{t=1}^{\infty} \eta^p(t) < \infty$, *for some* $p > 1$,

c. $\lim_{t \to \infty} \sup \left[1/\eta(t) - 1/\eta(t-1) \right] < \infty$.

PROOF: See theorem B.1 in appendix B.1. ∎

We note that the conditions on $\eta(t)$ guarantee that the learning rate is decreasing in a way that will dampen the random fluctuations around the minimum, but at the same time, not decreasing too fast to prevent reaching the minimum. A possible choice could be

$$\eta(t) = \frac{\eta'}{t^p} \tag{13}$$

where η' is a constant and $0 \leq p \leq 1$. $\eta(t) = \eta'/\log(t)$, however, would not work. This theorem shows that SLC trains the network to map point x_n to $n/(N+1)$ as $t \to \infty$. By comparing with SIC, described in the previous section, we see that both methods possess similarities for the univariate case. In fact, as $t \to \infty$, both methods are equivalent. Therefore, we will restrict the remaining analysis to SIC – smooth interpolation of the sample cumulative $G(x)$. We will first look at the convergence to the true distribution function and then the convergence to the density.

3.2 Convergence to the True Distribution Function

SIC gives an estimate of the distribution function, from which we get the density by differentiation. The distribution function is useful in its own right, so we will first look at the consistency and convergence rate of SIC as an estimator of the distribution function in the limit of large N.

Two well studied statistics of the sample distribution function are the Kolmogorov-Smirnov statistic ($\mathcal{D}_N$) and the Cramér–von Mises statistic ($\mathcal{C}_N$), defined as follows

$$\begin{aligned} \mathcal{D}_N &= \sup_x |G_N(x) - G(x)| & (14) \\ \mathcal{C}_N &= \int_{-\infty}^{\infty} (G_N(x) - G(x))^2 dG(x) & (15) \end{aligned}$$

where G_N is the sample distribution function given in (7). The following theorems are valid [54, pg 61–64]

Theorem 3.2 *With probability 1*

$$\limsup_{N \to \infty} \mathcal{D}_N \sqrt{\frac{N}{2 \log \log(N)}} = \frac{1}{2} \tag{16}$$

PROOF: See [58], [18] or [20]. ∎

Theorem 3.3 *With probability 1*

$$\limsup_{N \to \infty} \mathcal{C}_N \frac{N}{2 \log \log(N)} = \frac{1}{\pi^2} \tag{17}$$

PROOF: See [29]. ∎

It is not hard to extend these theorems to apply to the generalized sample distribution functions which we define shortly. These theorems give the probability 1 convergence behavior. We will look at $E\left[\mathcal{C}_N\right]_D$, the expected performance in the L_2 error criterion (RMS). The expectations are with respect to the data set. Let us introduce the following definition.

Let $\mathcal{G}$ be the space of functions such that for every $X \in \mathcal{G}$, the following holds

- $X : \mathbf{R} \to [0,1]$
- $X(t)$ is continuously differentiable and $X'(t) \geq 0$.
- $X(-\infty) = 0$ and $X(\infty) = 1$.

Thus, $\mathcal{G}$ is the space of (monotonic) distribution functions on the real line that possess continuous density functions, which is the class of functions that we will be interested in. We define a metric, the L_p X-norm of f, as follows

$$\| f \|_{X,p} = \left(\int_{-\infty}^{\infty} |f(t)|^p dX(t) \right)^{1/p} \tag{18}$$

$\| f \|_{X,p}^p$ is the expectation of $|f|^p$ with respect to the distribution $X \in \mathcal{G}$. The L_∞ X-norm is defined as

$$\| f \|_{X,\infty} = \sup_{x \in supp(X)} |f(x)| \tag{19}$$

Let the data set (D) be $\{x_1 \leq x_2 \leq \ldots \leq x_N\}$, and corresponding to each x_i, let $y_i = i/(N+1)$. As mentioned in the previous section, SIC attempts to map the order statistics x_i to $i/(N+1)$. In general this is possible given a large enough neural network. However, we will allow the mapping to be approximate because under some smoothness assumptions on the true distribution, a smoother fit would warrant a small sacrifice in the fit accuracy. With this in mind, we define the set of approximate sample distribution functions as follows

Definition 3.4 *A ν-approximate generalized sample distribution function, H, satisfies the following two conditions*

- $H \in \mathcal{G}$
- $\left| H(x_i) - \frac{i}{N+1} \right| \leq \nu(N) \sqrt{\frac{\log\log(N)}{2N}}, \ \forall i$

Where $\nu(N)$ is some function of N. We will denote the set of all ν-approximate sample distribution functions for a data set, D, and a given $\nu(N)$ by $\mathcal{H}_{D_N}^{\nu}$.

The set $\mathcal{H}_{D_N}^{\nu}$ contains those continuously differentiable distribution functions that approximately interpolate the data set $\{(x_i, y_i)\}_{i=1}^N$. For $\nu(N) = 0$ we have a generalized sample distribution function which is a smooth generalization of the conventional sample distribution function. Note that the conventional distribution function estimator (7) is not in this class of generalized sample distribution functions, but it is the limit of functions that are in this class.

Let us now derive the estimation error for the distribution function estimator. Let $H \in \mathcal{H}_{D_N}^{\nu}$, and let $G \in \mathcal{G}$ be the true distribution function. As we mentioned, H approximates the distribution function by approximate interpolation. Write $H(x_i) = y_i - \eta_i$, where $|\eta_i| \leq \nu(N)\sqrt{\log\log N/2N}$. The true distribution function evaluated at x_i is generally not equal to y_i due to statistical error. Therefore we will write $G(x_i) = y_i + \epsilon_i$. There are two sources of estimation error (see figure 3): The error at each data point, $\epsilon_i + \eta_i$, and the interpolation error between the data points – an error that would persist even if the $\epsilon_i + \eta_i$'s were all zero. As a first step, let us analyze the statistics of the ϵ_i's. It is well known that the random variable $G(x)$, x being generated according to $g(x)$, has a uniform distribution in [0,1] (see (2) in section 2.1). Therefore, $u_i = G(x_i)$ is the i^{th} order statistic of the uniform distribution, which is distributed according to the well known Beta distribution. The joint density of (u_i, u_j) can be obtained as (see [12, pg 200])

$$g(u_i, u_j) = \frac{N!}{(i-1)!(j-i-1)!(N-j)!} u_i^{i-1}(1-u_j)^{N-j}(u_j - u_i)^{j-i-1} \quad j \geq i \tag{20}$$

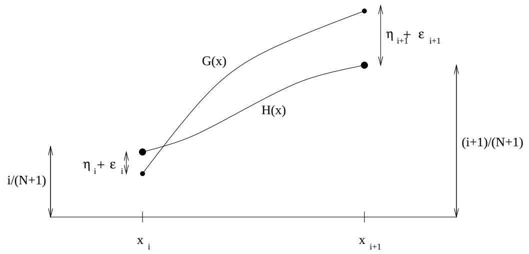


Figure 3: Interpolation error when interpolating the distribution function.

Noticing that $\epsilon_i = u_i - i/(N+1)$, we can calculate the first two moments of the ϵ_i's as

$$E\left[\epsilon_i\right] = 0 \qquad E\left[\epsilon_i \epsilon_j\right] = \frac{i(N+1-j)}{(N+1)^2(N+1)} \tag{21}$$

The following theorem provides a bound for the estimation error, which will prove consistency of the distribution estimator.

Theorem 3.5 (L_2 convergence to the true distribution) *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. For every $H \in \mathcal{H}^{\nu}_{D_N}$ and every $F \in \mathcal{G}$, the inequality*

$$E\left[\| H - G \|^2_{F,2}\right]_D = E\left[\int_{-\infty}^{\infty} (H(x) - G(x))^2 \, dF(x)\right]_D \leq \mathcal{Q}_2(N) \tag{22}$$

holds, where

$$\mathcal{Q}_2(N) = 2\left[\frac{1}{2(N+1)} + \nu(N)^2 \frac{\log\log N}{N} + \frac{3}{2(N+1)^2} + \frac{2\nu(N)}{N+1}\sqrt{\frac{\log\log N}{2N}}\right] \tag{23}$$

Proof: See theorem B.2 in appendix B.1.

Note 1: We stress that the theorem applies to *any* $H \in \mathcal{H}^{\nu}_{D_N}$ and *any* $F \in \mathcal{G}$, in particular to $F(x) = G(x)$, the true distribution function. Thus the expected mean integrated squared error approaches zero asymptotically at a rate $\mathcal{Q}_2(N)$.

Note 2: The same proof can be modified for the case of the integrated squared error over some bounded interval.

Note 3: Without much extra effort, this result can be extended to a larger class ${\mathcal{H}^{\nu}_{D_N}}^*$ of interpolators where the monotonicity condition is replaced by the less restrictive condition $H \in {\mathcal{H}^{\nu}_{D_N}}^*$ if

$$x \in [x_i, x_{i+1}] \quad \Rightarrow \quad H(x) \in [H(x_i), H(x_{i+1})] \tag{24}$$

with no other constraints being made on continuity or the existence of derivatives. Note that the conventional distribution function (7) belongs to this extended class ${\mathcal{H}^{\nu}_{D_N}}^*$.

Note 4: If $\nu(N) \leq 1/\sqrt{\log\log N}$, convergence to the true distribution function with respect to the L_2 (RMS) error is obtained at a rate $O(N^{-1/2})$.

In the appendix, we also give the L_1 and L_∞ rates, which are $O(N^{-1/2})$ and $O(\sqrt{\log\log N/N})$ respectively (theorems B.3 and B.4).

3.3 Convergence to the True Density Function

The density estimate is the derivative of the distribution function estimate and thus we need to consider $|H' - G'|$. Obtaining a tight convergence rate for the density estimation error is a tougher job. The reason is that the derivative operation accentuates the noise. In the next theorem, we present a bound on the estimation error for the density estimate obtained by using SIC. Its essential content is that if the true distribution function has bounded derivatives to order K, then by picking the approximate distribution function with "minimum" value for B_K, where B_K is a bound on the magnitude of the K^{th} derivative, we obtain convergence in probability at a rate $O((\log\log(N)/N)^{(K-1)/2K})$ in the L_∞ norm.

Theorem 3.6 (Convergence in probability to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0 \ldots K$, where $K \geq 2$. Let $\nu(N) > 2$ and fix $\epsilon > 0$. Let $B_K^\nu(D) = \inf_{H\in\mathcal{H}^\nu_{D_N}} \sup_x \left|H^{(K)}\right|$. Let $H \in \mathcal{H}^\nu_{D_N}$ be a ν-approximate distribution function with $B_K = \sup_x \left|H^{(K)}\right| \leq B_K^\nu + \epsilon$ (by the definition of B_K^ν, such a ν-approximate distribution function must exist). Then, the inequality*

$$\sup_x |H'(x) - G'(x)| \leq \mathcal{F}(N) \tag{25}$$

where

$$\mathcal{F}(N) = 2^{(K-1)}(2A_K+\epsilon)^{\frac{1}{K}}\left[(1+\nu(N))\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1}\right]^{1-\frac{1}{K}} \tag{26}$$

holds with probability 1, as $N \to \infty$. By this is meant

$$\lim_{N\to\infty} P\left[\sup_x |H'(x) - G'(x)| \leq \mathcal{F}(N)\right] = 1 \tag{27}$$

PROOF: See theorem B.11 in appendix B.1.

Note 5: This theorem holds for *any* $\epsilon > 0$ and *any* $\nu(N) > 2$ and to *any* H satisfying the conditions of the theorem. Therefore we see that for smooth density functions, with bounded higher derivatives, the convergence rate approaches $O((\log\log(N)/N)^{1/2})$.

Note 6: No smoothing parameter needs to be determined unlike the other traditional techniques. The regularization is accomplished by minimizing the norm of the K^{th} derivative.

Note 7: The convergence rate is very close to the rate proven to be optimal under similar smoothness assumptions (see [28], [27], [26], [60], [61], [62], [63]).

Note 8: From the theorem, it is clear that one should try to find a generalized ν-approximate distribution function with the smallest possible derivatives. Specifically, of all the sample distribution functions, pick the one that "minimizes" B_K, the bound on the K^{th} derivative. Thus, when using optimization techniques to find the generalized distribution function, one would be justified in introducing penalty terms, penalizing the magnitudes of the derivatives (for example Tikhonov type regularizers [68]).

Note 9: For compact support, choosing $F(x)$ to be the uniform measure on that support, one obtains a result for the integrated squared measure.

Note 10: We see that consistent density estimation involves solving a constrained optimization problem in order to *guarantee* convergence at the prescribed rate. The objective function would be the bound on the K^{th} derivative and the constraints are that you fit the density sufficiently closely. The constraints can be enforced softly by penalizing any violation of the constraints. Practically speaking, it is often more convenient to approximately impose the smoothness constraints (for example by starting at small weights or using weight decay [14], [5]) while attempting to fit the distribution function. Our simulations (see section 6) indicate that this works quite well.

3.4 Implementation by Neural Networks

In the previous section, we have proved the L_2 convergence to the distribution and the L_∞ convergence to the density. We have shown that any functions satisfying the conditions of the theorems will converge at the given rates. In particular, if neural networks can be found that satisfy the required conditions, then these convergence rates will also apply to implementations with neural networks. The goal here is to show that one can find neural networks that satisfy the required conditions. In fact, letting the size of the neural networks increase at a rate $O(N \log N)$ would suffice.

In order to show that neural networks can be chosen to satisfy the conditions of the theorems, it suffices to show that neural networks are dense in a certain sense in the space $\mathcal{H}^{\nu}_{D_N}$. In other words, let a sequence of functions $H_N \in \mathcal{H}^{\nu}_{D_N}$ be given where the N indexes the number of data points. Suppose that H_N satisfies the conditions of theorem 3.5 or 3.6 (whichever we are interested in). Then we know that the error for H_N's converges to zero. If we show that there is also a sequence of neural networks g_N such that the following conditions hold simultaneously:

1. $sup_x |g_N(x) - H_N(x)| < O(1/N)$
2. $|g'_N - H'_N| \leq O(1/N)$

then the error for the g_N's converges to zero at the same rate as the H_N's. To show the existence of such a sequence of neural networks, it suffices to show that given an arbitrary degree of accuracy ϵ, there is a sequence of g_N's that approximate the H_N's to within ϵ and simultaneously the g'_N's also approximate the H'_N's to within ϵ. To this end, we will use the approximation theorems proved in [33], [34]. First we set up the notation to state the theorem, following very closely the setup in [33], [34].

Define the class of neural networks we are interested in as the functions defined by the set

$$\Sigma(\Phi) \equiv \left\{ g : \mathbf{R} \to \mathbf{R} | g(x) = \sum_{i=1}^{N_H} v_i \Phi(w_i x + b_i), \ N_H \in \mathbf{N} \quad v_i, w_i, b_i \in \mathbf{R} \right\} \tag{28}$$

where Φ is conventionally called the activation function. We make some assumptions on Φ. Though some of these assumptions could be dropped (see [33], [34]), there is negligible practical gain in maintaining more generality.

1. $\Phi \in C^\infty$ is a symmetric sigmoidal non-linearity with exponentially decaying derivatives of all orders.
2. $\int_{-\infty}^{\infty} |\Phi^{(i)}(x)| dx < \infty$ for all $i = 1 \ldots m$. i.e., $\Phi \in S_1^m(\mathbf{R})$ where $S_p^m(U)$ is a Sobolev space[2] in the L_p metric on

[2] for more details on Sobolev spaces, see [4]

the open set U for functions in C^m.

We make one further restriction which is once again of no practical consequence. We suppose that we are interested in approximating the distribution on some arbitrary compact set K. Let U be an open bounded set containing K. Then, the restriction of $\mathcal{H}^{\nu}_{D_N}$ to U is a subset of S^1_1. The following theorem is valid.

Theorem 3.7 (Universal Approximation) *Let $K \subset \mathbf{R}$ be a compact set. Let $f \in S^m_p(U)$ where $K \subset U$ and let $\Phi \in S^m_1(\mathbf{R})$. Given $\epsilon > 0$, there $\exists g \in \Sigma(\Phi)$ such that simultaneously for all $i = 0 \ldots m$*

$$\sup_{x \in K} |f^{(i)}(x) - g^{(i)}(x)| < \epsilon \tag{29}$$

PROOF: See Hornik et al. [33], theorem 3.1 and corollary 3.5. ■

This is a very powerful theorem. It states that neural networks can simultaneously approximate a function and its derivatives provided that certain conditions are met. The ability to approximate a function and its derivatives up to order m is termed *m-uniform denseness* in [33]. Thus theorem 3.7 can be summarized by saying that neural networks are *m-uniformly dense*[3] in $S^m_p(U)$.

Corollary 3.8 *$\Sigma(\Phi)$ is 1-uniformly dense in $\mathcal{H}^{\nu}_{D_N}$ restricted to U.*

PROOF: Since Φ and $\mathcal{H}^{\nu}_{D_N}$ restricted to U satisfy the conditions of theorem 3.7, the corollary is immediate. ■

The following theorem is now immediate.

Theorem 3.9 (Consistent distribution and density estimation using neural networks)

1. *Consistent distribution estimation with an L_2 error rate $O(N^{-1/2})$ can be obtained on any given compact set K using neural networks.*

2. *Consistent density estimation with an L_∞ error rate $O((\log\log N/N)^{(K-1)/2K})$ can be obtained on any given compact set K using neural networks, where we assume the true distribution has bounded derivatives up to order K.*

PROOF: Let H_N be a sequence that has the desired convergence for the distribution (theorem 3.5) or density (theorem 3.6). Then there exists a sequence of neural networks that simultaneously approximates H_N and H'_N with a maximum error less than $1/N$ (corollary 3.8). Thus the added approximation error between H_N and g_N does not affect the order of convergence. ■

Further, by the results given in [34], corollary 2.5, we see that in order to guarantee that a neural network will be able to approximate to within $O(1/N)$, the number of hidden units, N_H, must be $O(N \log N)$. Thus we see that by increasing the size of the neural network at a rate $O(N \log N)$, one can obtain consistent distribution/density estimation using neural networks on any given compact set. As a practical note, it is clear that the theoretical results give justifications for SIC and provide definite guidelines as to how to choose the size of the network as a function of N to guarantee convergence. In practice, one knows a lot more about the distribution and usually a small number of hidden units will suffice.

[3] see [33] for the precise definition.

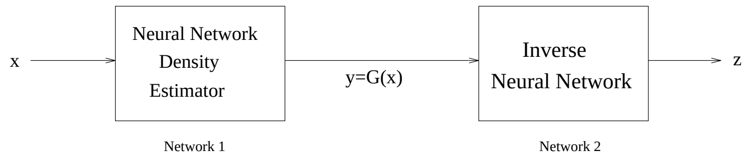


Figure 4: Training a network to implement $G^{-1}(x)$. The first network learns $G(x)$ using SLC or SIC. This is done using the data available. Now, for an arbitrary set of inputs, we take Network 1's outputs as the inputs to Network 2 and the target are the inputs that went into Network 1. Thus we train Network 2 to implement the inverse of Network 1 which should be G^{-1}.

4 New Random Variate Generation Techniques

4.1 Learning the Inverse Distribution Function from a Finite Sample (SLCI and SICI)

Suppose that we wish to generate random variates from a specified univariate density $g(x)$. It could also be that $g(x)$ is represented only by a number of data points drawn from it, rather than a functional form. Typical approaches to such a problem would estimate the density from the given data, and apply some of the well-known methods such as the transformation method or the rejection method (which would be computationally extensive, since a pass through all the given data points has to be performed for each evaluation of the density).

We propose a method using multilayer networks, that is inspired by the transformation method. But, unlike the transformation method, we do not assume that $G^{-1}(x)$ is known analytically. The network learns to implement the function $G^{-1}(x)$. The basic structure of the method is illustrated in figure 4. It consists of two cascaded multilayer networks. The first network is trained to estimate the distribution function $G(x)$ from the data using one of the techniques described in the previous sections (SLC or SIC). Once the first network is trained, the density of its output is uniform in $[0, 1]$ as discussed in sections 2.1 and 2.2. Then, we train the second network to invert the mapping produced by the first network. This inversion can be accomplished to an arbitrary precision by using as large a network and as many data points as we want. Learning the inverse proceeds as follows. Let $H_1(x, w)$ and $H_2(x, v)$ be the functions implemented by Network 1 and Network 2 respectively. Let $\{x_i\}$ be an arbitrary set of inputs. Then, the input/output examples for Network 2 will be $\{H_1(x_i, w), x_i\}$. Once the entire training process is complete, random variate generation according to $g(x)$ is a done by passing y, a uniform deviate in $[0, 1]$, through Network 2, i.e., $H_2(y, v) \sim g(x)$. To see why this is so, observe that for the cascade structure of the two networks in figure 4, the density of the output z is equal to density of the input x because Network 2 is the inverse of Network 1. Since the density of the output y of Network 1 is uniform, inputing a uniform deviate (y) into Network 2 should produce a variable (z) having a density equal to that of x. More formally, assume that Network 1 was implementing $G(x)$. Then, Network 2 is implementing $G^{-1}(x)$. Therefore, $z = G^{-1}(y)$ where y has a uniform distribution. Using the formula for transformations of random variables, the density g_Z of the output z is evaluated as

$$g_Z(z) = \frac{g_Y(y)}{|\frac{dG^{-1}(y)}{dy}|} = \frac{1}{|1/g(z)|} = g(z). \tag{30}$$

This method applies only for the one-dimensional case because the mapping $G(x)$ transforms x into a uniform density only for the univariate case. A second method based on a control formulation of the problem, described later, is more general and applies to the multi-dimensional case as well.

Methods similar to the above method have been independently proposed in [51], [44]. A drawback of this approach is that it is wasteful to first learn $G(x)$,and then invert it. We propose here two new methods that are based on directly learning $G^{-1}(x)$. These two methods are inspired by the density estimation techniques that we have developed in the previous sections. They are variants of the method described above in that they build upon the idea that a network mapping a uniform distribution to the true distribution must be implementing $G^{-1}(x)$. The two methods described below can be applied either when the true distribution function $G(x)$ is given or when a finite data set drawn from $G(x)$ is given.

4.1.1 SLCI (Stochastic Learning of the Cumulative Inverse)

This technique is very similar to the stochastic method for estimating $G(x)$ and can be viewed as the inverse of SLC. Once again, G^{-1} is a monotonic function so a monotonicity hint should be used here as well. The algorithm is as follows.

1. Sort the data points $x_1 \leq x_2 \leq \cdots \leq x_N$.
2. Set $t = 1$ and initialize the weights of the network to $w(1)$.
3. Generate N numbers ($\{u_i\}_{i=1}^{N}$) from a uniform density in $[0, 1]$ and sort them so that $u_1 \leq u_2 \leq \cdots \leq u_N$.
4. Train the network to map input u_n to output target x_n exactly as in steps 5 and 6 of the stochastic density estimation technique. Every cycle or every L cycles, generate new u_n's.
5. After training is complete, input to the network a uniformly generated number. The output is distributed according to $G(x)$.

4.1.2 SICI (Smooth Interpolation of the Cumulative Inverse)

This approach is analogous to the smooth interpolation approach for estimating the density (SIC). It is identical to SLCI, except that the input examples are $i/(N + 1)$ (corresponding to output example x_i), instead of the uniform deviates. The algorithm is as follows.

1. Sort the data points $x_1 \leq x_2 \leq \cdots \leq x_N$.
2. Set $t = 1$ and initialize the weights of the network to $w(1)$.
3. Let $u_i = i/(N + 1)$ for $i = 1 \ldots N$.
4. Train the network to map input u_n to output target x_n as in steps 5 and 6 of the stochastic density estimation technique (SLC).
5. After training is complete, input to the network a uniformly generated number. The output is distributed according to $G(x)$.

Another version of these methods that we have implemented is to determine the input/output examples by gridding the space $[x_1, x_N]$ into M points $\{z_i\}_{i=1}^{M}$. One then computes $u_i = \hat{G}(z_i) = 1/(N + 1) \sum_{n=1}^{N} \Theta(z_i - x_n)$. The input/output examples are then $\{u_i, z_i\}_{i=1}^{M}$. The advantage of this version is that when only a small number of sample points x_i are available, one can generate many more examples that help to learn G^{-1}.

The two methods presented above are applicable to generating random variates given a sample of points. It is often the case that one has a functional form for the density from which one wishes to generate. The next method uses a neural network to learn G^{-1} directly.

4.2 Learning the Inverse Distribution Function given the Density

Suppose that one is given the functional form $g(x)$ for the density. The data for training the neural network are generated as follows. The $x-$space is gridded into (say) M points, $\{y_i\}_{i=1}^{M}$. Assume that the y_i have been ordered. The input examples to the network are given by $x_i = G(y_i)$, i.e., one computes $x_i = \int_{-\infty}^{y_i} g(t)dt$, where $g(t)$ is given. As the y_i's form an increasing sequence, we do not have to perform M numerical integrations. A single numerical integration from $-\infty$ to ∞ will suffice if one outputs the value of the integral each time a y_i is crossed. One now trains a neural network using (say) the squared error criterion to implement the mapping $x_i \mapsto y_i$. Known facts about G^{-1} should be used in this training, such as monotonicity. The resulting neural network now implements an estimate of G^{-1}. Random variates are obtained by passing a uniform deviate through the resulting network.

The universal approximation theorems in section 3.4 guarantee that by choosing M large enough and a large enough neural network, one can generate random variates possessing a density arbitrarily close to $g(x)$.

4.3 Control Formulation of Random Variate Generation

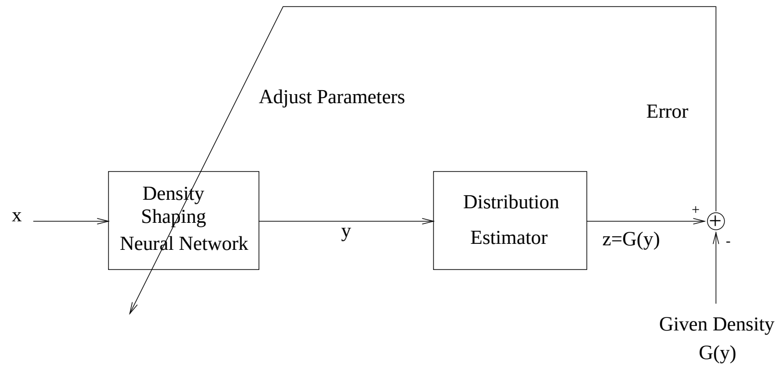


Figure 5: Control formulation for random variate generation. The first network shapes the density of x. The transformed density is estimated, and then compared with the true density. The parameters of the first network are altered until the transformed density matches the required density.

The main contribution of this section is a novel formulation of the random number generation problem as a control problem. Figure 5 illustrates the basic idea. The structure consists of two cascaded units. A random variable x is generated according to any standard density function, for example Gaussian or uniform. In principle, an unlimited number of x's can be generated. This random variable is the input to the composite structure. The first network acts as a nonlinear transformation whose goal is to shape the density of x into the target density. Let the distribution function of the transformed variable be $Z(y)$, where y is the transformed variable. The second unit serves to estimate this distribution function, for example SLC or SIC (sections 2.1 and 2.2) or a kernel estimator could be used. The output of the second unit, is an estimate of the distribution function $Z(y)$ of Network 1's output. An error signal,

$\mathcal{E}(G(y), Z(y))$ [4] is generated where $G(y)$ is the target distribution function. This error signal is backpropagated through the networks and is used to adjust the weights of Network 1 so as to minimize $\mathcal{E}(G(y), Z(y))$. Such a training process would change Network 1's mapping in such a way that would shape the distribution function of y to get it as close as possible to the desired distribution function. Once training is complete, we can generate numbers according to the density $G(x)$, by generating x according to the standard density, and then passing it through Network 1.

The analogy with the control problem is as follows. The second unit represents the "plant," and Network 1 is the "controller." The controller is trained, such that the plant produces the desired output. This technique has a similar form to a neural network control structure (see [47]) where, typically, a neural network is trained to identify the plant (estimate the plant output), analogous in our set up to the density estimation unit. Then, a neural network controller is trained to control the plant – i.e., the identification network estimates the plant output and the controller network is altered until this output is as desired. The details of the algorithm are as follows:

I: Initialization:

1. Let the desired density be $g(y)$ and the corresponding distribution function $G(y)$. Generate N samples $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbf{R}^d$ according to any standard density such as a Gaussian density. The larger N is, the more accurate the final result will be.

2. Let $H_1(\mathbf{x}, w)$ and $H_2(\mathbf{y}, v)$ be the functions implemented by Network 1 and Network 2 respectively (we have assumed that the density estimator is a neural network such as those developed in the previous sections). Network 1 has d outputs, and Network 2 has one output. Initialize the weights (w and v) of the networks.

3. Calculate Network 1's output: $\mathbf{y}_n(w) = H_1(\mathbf{x}_n, w),\ n = 1, \cdots, N$.

4. Train Network 2 using SLC or SIC in order to estimate the distribution of the $\mathbf{y}_n$'s. Thus, $H_2(\mathbf{y}, v(w))$ is implementing the distribution function of y. Note that H_2 implicitly depends on w.

II: Training Iterations:

5. Compute the error function $(\mathcal{E}(H_2(v(w)), G))$ given by the squared error

$$\mathcal{E} = \sum_{n=1}^{M} \Big[H_2(\mathbf{z}_n, v(w)) - G(z_n) \Big]^2 \tag{31}$$

where $\{\mathbf{z}_n\}_{n=1}^M$ be any set of M points. One could even choose z_n to be y_n, however, choosing $\mathbf{z}_n$ to be on a finer grid usually works better.

6. Adjust the weights (w) of the first network in the direction of the negative gradient:

$$w(t+1) = w(t) - \eta \frac{\partial \mathcal{E}}{\partial w} \tag{32}$$

For the case of the squared error above, we have,

$$\frac{\partial \mathcal{E}}{\partial w} = 2 \sum_{n=1}^{M} \Big[H_2(z_n, v(w)) - G(z_n) \Big] \frac{\partial H_2(z_n, v(w))}{\partial w} \tag{33}$$

[4]The error signal could also penalize deviations of G' from Z' thus ensuring that one learns the density.

For clarity, we restrict the remainder of the algorithm to the case where the set $\{\mathbf{z}_n\}$ is the same as the set $\{\mathbf{y}_n\}$. Leave $v(w)$ fixed for a certain number (L) of training cycles. Then the change in $\mathcal{E}$, as w changes can be computed by the chain rule. We get

$$\frac{\partial \mathcal{E}}{\partial w_i} = 2\sum_{n=1}^{N}\Big[H_2(\mathbf{y}_n, v) - G(\mathbf{y}_n)\Big]\frac{\partial H_2(\mathbf{y}_n, v)}{\partial w_i} \tag{34}$$

$$= 2\sum_{n=1}^{N}\Big[H_2(\mathbf{y}_n, v) - G(\mathbf{y}_n)\Big]\nabla_{\mathbf{y}_n} H_2(\mathbf{y}_n, v)\cdot\frac{\partial H_1(\mathbf{x}_n, w)}{\partial w_i} \tag{35}$$

The weights in the first network can now be updated using a backpropagation-type scheme [32, pg 115], where one backpropagates the δ's first through Network 2, then through Network 1 and one only adjusts the weights of Network 1.

7. After L cycles of weight update for Network 1 in step 6, train Network 2 for a few cycles on the new $\mathbf{y}_n's$, to allow it to track the small change in the distribution of $\mathbf{y}$. Thus we now change v which was kept constant in step 6.

8. Go to step 6 to perform another iteration of training, and continue doing so until the error becomes small enough.

9. After training is complete, the random numbers can be generated by generating $\mathbf{x}$ according to the standard density that was initially used (e.g. Gaussian), and computing $y = H(\mathbf{x}, w^*)$ (where w^* represents the final weights after the entire learning process). Now, $\mathbf{y}$ should be the distributed according to $g(\mathbf{y})$.

Note 1: This algorithm is not restricted to the case of scalar random variable, and so it can be used to generate multi-dimensional variables.

Note 2: Although we used the squared error function (step 6), the algorithm can be used with other error functions such as the cross entropy error function (Kullback-Liebler distance). In addition, the error function need not only penalize differences in the distributions, but could also penalize differences in the derivatives of the distributions, ensuring that the correct density and higher order smoothness characteristics are learned.

Note 3: For clarity, in step six we considered the case where the $\mathbf{z}_n$'s were the same as the $\mathbf{y}_n$'s, and we did not update the second network weights for some fixed number of iterations. The general case is treated in appendix A. The exact update rule is given by (33) and is computed for two cases: the case where the density estimator (H_2) is a neural network trained according to SIC (see equations (54), (55), (56)), and the case where the density estimator is a Gaussian kernel estimate (see equations (59), (60)). The general case as described in appendix A generally produces better results.

Note 4: To generate from a density represented only by a set of data points, we use the same method above but replace G by an estimate of G using the data points. This estimate can be obtained using one of the methods described in sections 2.1 and 2.2.

Note 5: The entire learning process can be an involved process. However, once the learning has been done, the generation of the random variates is fast since it only involves the evaluation of a multilayer network function.

5 Convergence of the Random Variate Generation Techniques

In this section we will analyze the random variate generation methods to show that they generate the true distribution as $N \to \infty$. The random number generation techniques (SLCI and SICI) are essentially techniques for estimating G^{-1}. We restrict the analysis to the case where G^{-1} is continuous on $[0,1]$. First we will show the convergence of SLCI to SICI with probability 1 as $N \to \infty$. Then we obtain an L_2 convergence rate for SICI that is $O(N^{-1/2})$. Finally we will show that the random variate generation techniques converge to generating from the true density with an L_∞ error rate approaching $O(\sqrt{\log\log N/N})$ for smooth densities. These theorems are very similar in flavor to those in section 3, so we do not present as much detail here.

5.1 Convergence of SLCI to SICI

The following theorem shows that SLCI converges to a solution of SICI (in the limit $N \to \infty$) with probability 1. With this in mind, we restrict the further analysis to SICI.

Theorem 5.1 *Let the data $x_1 \dots x_N$ be generated according to the true distribution G. Then, in the limit $N \to \infty$, SLCI converges with probability 1 to a minimum of the error function*

$$\mathcal{E} = \sum_{n=1}^{N} \left[H\left(\frac{n}{N+1}, w \right) - x_n \right]^2, \tag{36}$$

provided that the learning rate $\eta(t)$ is a decreasing sequence, that satisfies the following conditions

a. $\sum_{t=1}^{\infty} \eta(t) = \infty$,

b. $\sum_{t=1}^{\infty} \eta^p(t) < \infty$, *for some* $p > 1$,

c. $\lim_{t\to\infty} \sup \left[1/\eta(t) - 1/\eta(t-1) \right] < \infty$.

Proof: See theorem B.14 in appendix B.2. ■

Thus, we see that in the limit $N \to \infty$, H is performing the mapping $n/(N+1) \mapsto x_n$, which is exactly what SICI trains the network to do.

5.2 Convergence of SICI

We will make the simplifying assumption that $G^{-1}(u)$ is continuously differentiable on the compact interval $[0,1]$. Therefore the input support is compact and, further, we have that $\left| dG^{-1}(u)/du \right|$ is bounded. Let the bound on the derivative be S. Then, the input distribution is bounded in a range of size $2S$.

Let $u_i = i/(N+1)$ and let $H^{-1}(u)$ be the a generalized inverse distribution function that approximately interpolates the points $\{u_i, x_i\}$. i.e.,

$$H^{-1}(u_i) = x_i - \kappa_i \qquad |\kappa_i| \leq \nu(N) \sqrt{\frac{\log\log N}{2N}} \tag{37}$$

We would like to analyze the rate of convergence of the integrated squared error $\mathcal{E} = E\left[\int_0^1 \left(H^{-1}(u) - G^{-1}(u) \right)^2 d\mu(u) \right]$, where μ is any continuous measure on $[0,1]$. Write

$$G^{-1}(u_i) = x_i + \xi_i \tag{38}$$

Then, the following theorem is valid.

Theorem 5.2 (L_2 convergence to the true inverse distribution function) *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. Assume that G^{-1} is continuous on the closed interval $[0,1]$ and let its derivative be bounded by S. For every generalized inverse distribution function that is zero outside the range of G^{-1} and for any continuous measure $d\mu(u)$ on $[0,1]$ (with bound B on $[0,1]$), the inequality*

$$E\left[\| H^{-1} - G^{-1} \|_{\mu}^{2}\right]_D \leq \mathcal{Q}_3(N) \tag{39}$$

holds, where

$$\mathcal{Q}_3(N) = \frac{1}{N}\left(\frac{S^2}{2} + \nu(N)^2 \log\log N + \sqrt{\frac{S^2\nu(N)^2\log\log N}{N}} + \frac{2S^2}{N} + 8BS^2\right) \tag{40}$$

PROOF: See theorem B.15 in appendix B.2. ∎

Thus, we have convergence to G^{-1} provided the conditions of theorem 5.2 hold, and further that the rate of convergence is $O(N^{-1/2})$ if $\nu(N) < 1/\sqrt{\log\log N}$. In a similar manner, one could obtain L_1 convergence. We omit the details here.

We now look at the convergence to the density. We impose one additional restriction, that on the compact support of interest, $g(x) = G^{(1)}(x) > \delta$ (in other words, the true density is bounded away from zero). Further, we already assume that $G^{(i)} \leq A_i$ for $i = 1 \ldots K$. The following theorem gives the convergence to the true density.

Theorem 5.3 (Convergence in probability to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0 \ldots K$, where $K \geq 2$, and let $G^{(1)} \geq \delta$ Fix $\epsilon > 0$ and let $\nu(N) > 2/(\delta - \epsilon)$. Let $B_K^{\nu}(D) = \inf_{Q^{-1}} \sup_x \left|Q^{(K)}\right|$ where Q^{-1} represents a generalized ν-approximate inverse distribution function and Q the corresponding distribution function. Let H^{-1} be a generalized ν-approximate inverse distribution function with $B_K = \sup_x \left|H^{(K)}\right| \leq B_K^{\nu} + \epsilon$ (by the definition of B_K^{ν}, such a ν-approximate inverse distribution function must exist). Then, the inequality*

$$\sup_x |H'(x) - G'(x)| \leq \mathcal{F}(N) \tag{41}$$

where

$$\mathcal{F}(N) = 2^{(K-1)}(2A_K + \epsilon)^{\frac{1}{K}} \left[(1+\nu(N))\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1}\right]^{1-\frac{1}{K}} \tag{42}$$

holds with probability 1, as $N \to \infty$. By this is meant

$$\lim_{N\to\infty} P\left[\sup_x |H'(x) - G'(x)| \leq \mathcal{F}(N)\right] = 1 \tag{43}$$

PROOF: See theorem B.16 in appendix B.2. ∎

The theorems in this section show that the generalized inverse distribution functions converge to the true inverse distribution function. Further, the resulting density also converges. The discussion on implementing the density estimation techniques with neural networks in section 3.4 also applies here. We do not go into the details as they are similar to the analysis in section 3.4. Essentially, the inverse distribution functions that achieve the desired rates of convergence to the true inverse distribution function can be implemented by neural networks. Hence, consistent random variate generation can be obtained at the same rates using neural networks if the size of the neural networks increases as $O(N \log N)$.

The theorems in this section are not only relevant to generating given a finite amount of data, but also to the control formulation where one learns by generating an arbitrary amount of data to learn from. These theorems show that one can guarantee convergence to generating from the true density with enough data.

6 Simulation Results

6.1 Density Estimation Techniques

To test the proposed density estimation techniques, we considered the following mixture-of-Gausians density:

$$g(x) = \frac{3}{10}\frac{1}{\sqrt{18\pi}}e^{-\frac{(x+30)^2}{18}} + \frac{7}{10}\frac{1}{\sqrt{162\pi}}e^{-\frac{(x-9)^2}{162}} \tag{44}$$

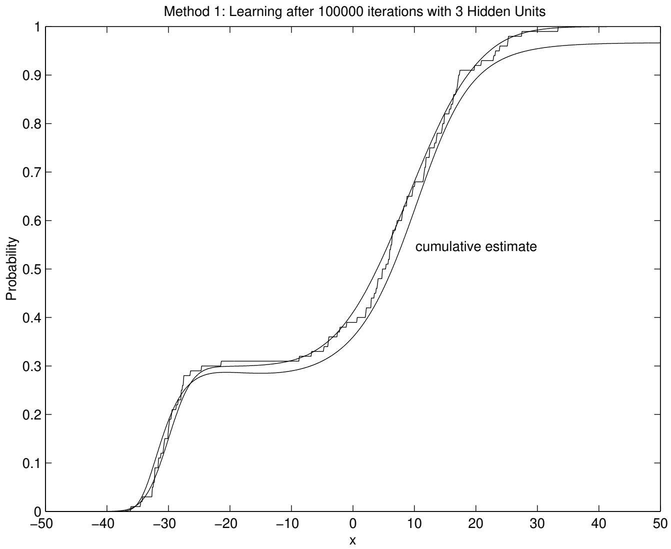
We generated 100 *i.i.d.* data points from this density, and have used these data points to design the density estimators according to SLC and SIC (sections 2.1 and 2.2). The results of SLC and SIC are shown in figures 6 and 7 respectively. One can see that even with 100 data points, the density estimates are quite reasonable.

To compare the proposed methods with the kernel estimation method, we implemented the three methods (SLC, SIC, and the kernel estimation method) on two samples, of sizes 100 and 200 respectively, drawn from the above density (44). The optimal kernel width h has been calculated according to the formula $h \approx 5.12/n^{1/5}$ [56, pg 40]. Since this is not possible in practice, the results of the kernel estimation method tend to be optimistic. The results are shown in figure 8. As can be seen, the kernel estimator displays bumpy behavior, even with the optimal choice of the kernel width. The neural network implementations, on the other hand, are relatively smooth. Of course, performance on a single test case might not be conclusive, but backed by the theoretical results of section 3 they are certainly compelling.

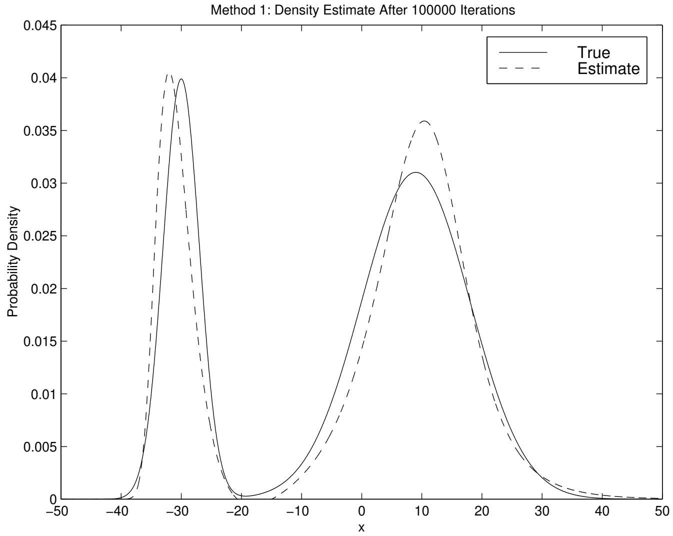
The details of the neural network implementation for both the SLC and SIC method are as follows. We used a 1 hidden layer network with 3 hidden units. The activation function of the hidden units is *tanh*, and that of the output unit is *erf* [5]. We trained the network for 10000 iterations using the conjugate gradient algorithm. We enforced the monotonicity hint by a penalty term with weight 10000. As we mentioned in the derivation of the convergence rates, we wish to pick the "smoothest" interpolator. We softly enforced this by starting the learning with small weights [6].

We also implemented the proposed approaches on a real-world example, namely estimating the density of stock price changes (in the log space, i.e. $log(S_{t+1}/S_t)$). Theoretical studies suggest that the logarithm of a stock price follows a Brownian motion [80], and hence the log price difference is Gaussian. However, practical observation has shown that stock prices appear to have densities with fatter-than-Gaussian tails. The accurate measurement of the density of stock price changes is very important, since it can significantly impact the pricing of stock derivative instruments such as options. We implemented our proposed methods on such a problem, by applying them to three representative Dow-component stocks (IBM, GE, and JP Morgan). Figure 9 shows the estimated densities for the SIC method along with the true histograms and the Gaussian with the same mean and variance. Notice how the true distribution significantly deviates from Gaussian, displaying the well-established fat-tail behavior.

[5] $erf(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}dt$.

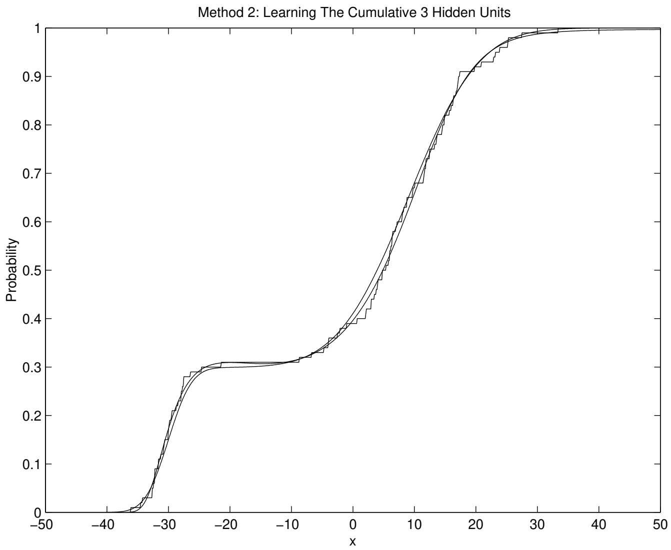


(a)

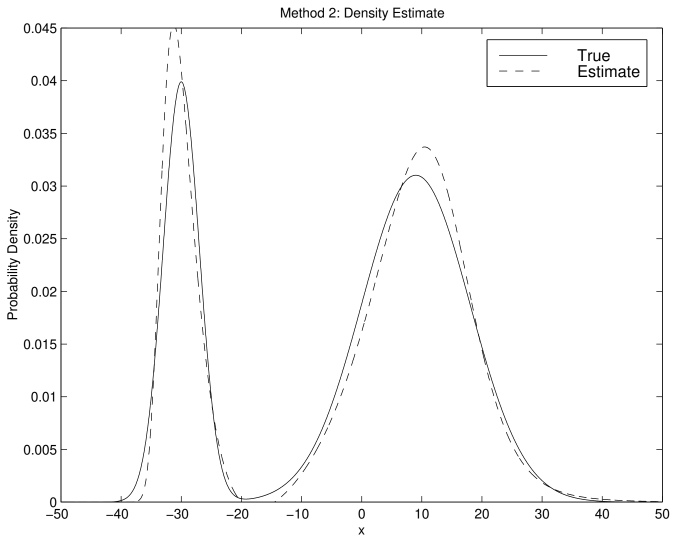


(b)

Figure 6: Stochastic learning of the distribution function (SLC). The cumulative estimate with 100 data points is shown in (a) along with the sample and true distribution functions. The lower curve is the neural network estimate, the upper curve is the true distribution function and the stair-case like curve is the sample distribution function. The density estimate is shown in (b).

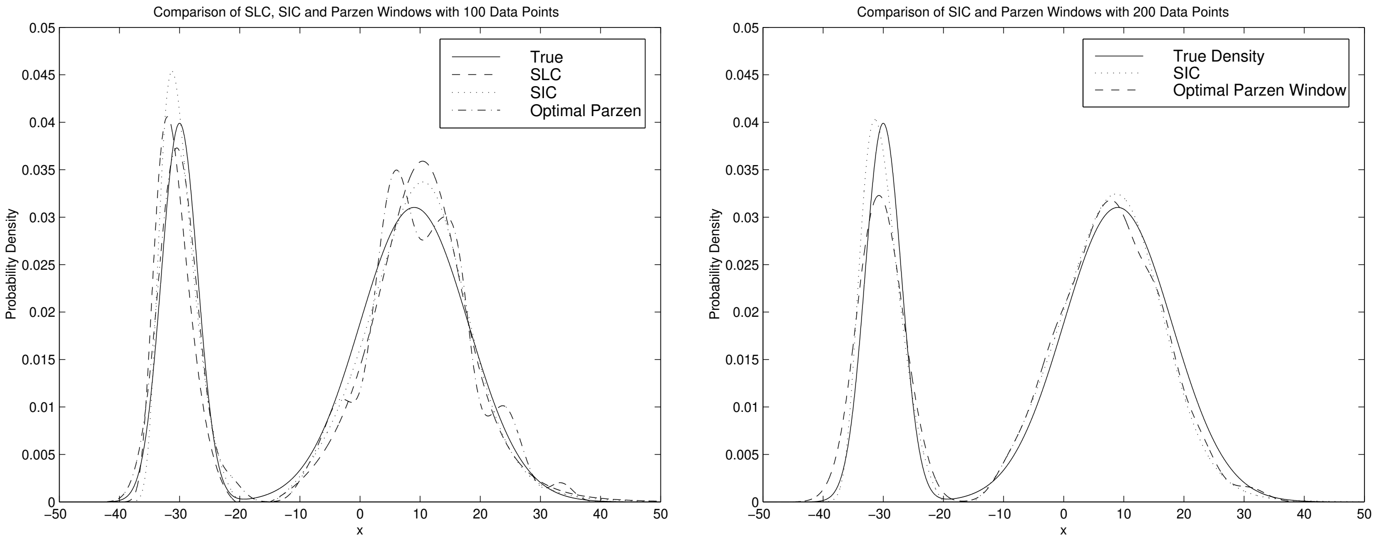


(a)



(b)

Figure 7: Smooth interpolation of the distribution function (SIC). The cumulative estimate is shown in (a) along with the sample and true distribution functions. The lower curve is the neural network estimate, the upper curve is the true distribution function and the stair-case like curve is the sample distribution function. The density estimate is shown in (b).



(a) **(b)**

Figure 8: Comparison of the optimal kernel estimate, with the neural network estimators. Plotted are the true density and the estimates (SLC, SIC, Kernel): (a) Density estimation with 100 data points and (b) Density estimation with 200 data points.

6.2 Random Variate Generation Techniques

6.2.1 Learning the Inverse Distribution Function

We generated 5000 data points according to the mixture-of-Gaussians density given by equation (44). Using these data points, we implemented SICI (section 4.1) in order to generate data according to the specified density. We used a single hidden layer network with 50 *tanh* hidden nodes, and a linear output node. In order to create singularities at the end points $y = 0$ and $y = 1$, we added a tangent function to the output. We enforced the monotonicity of the mapping by using a hint penalty term with a hint weight of 10000. The training algorithm we used was 10000 iterations of a conjugate gradient descent algorithm on the squared error, and the data for learning was generated as follows: the $x-$space was gridded into 10000 points and these grid points y_i served as the outputs. The inputs were computed as $x_i = \hat{G}(y_i)$ as described in section 4.1.2. To test how well G^{-1} was learned, we generated two million points from a uniform distribution and passed them through the network. The output of the network should then be distributed according to the density in (44). Figure 10 shows the sample distribution and histogram of these two million network outputs as compared to the desired distribution and density. For comparison, figure 11 shows the resulting distribution and density that were obtained by first forming a kernel estimate using the optimal kernel width given by $h \approx 5.12/5000^{1/5}$ [56, pg 40] and then generating according to this kernel estimate. In practice, the optimal kernel width is not available, but even using the optimal kernel width, we see that SICI appears to outperform the kernel method. Once again, we note that performance on a single test case might not be conclusive, but the theoretical results of section 5 provide additional support for SICI.

We also implemented the method described in section 4.2 for learning the inverse distribution function given the functional form in equation (44). The data was generated as follows. 5000 uniformly spaced points on the interval $(-50, 50)$ served as the target outputs y_i (outside the range (-50,50) the density in (44) is essentially zero). The

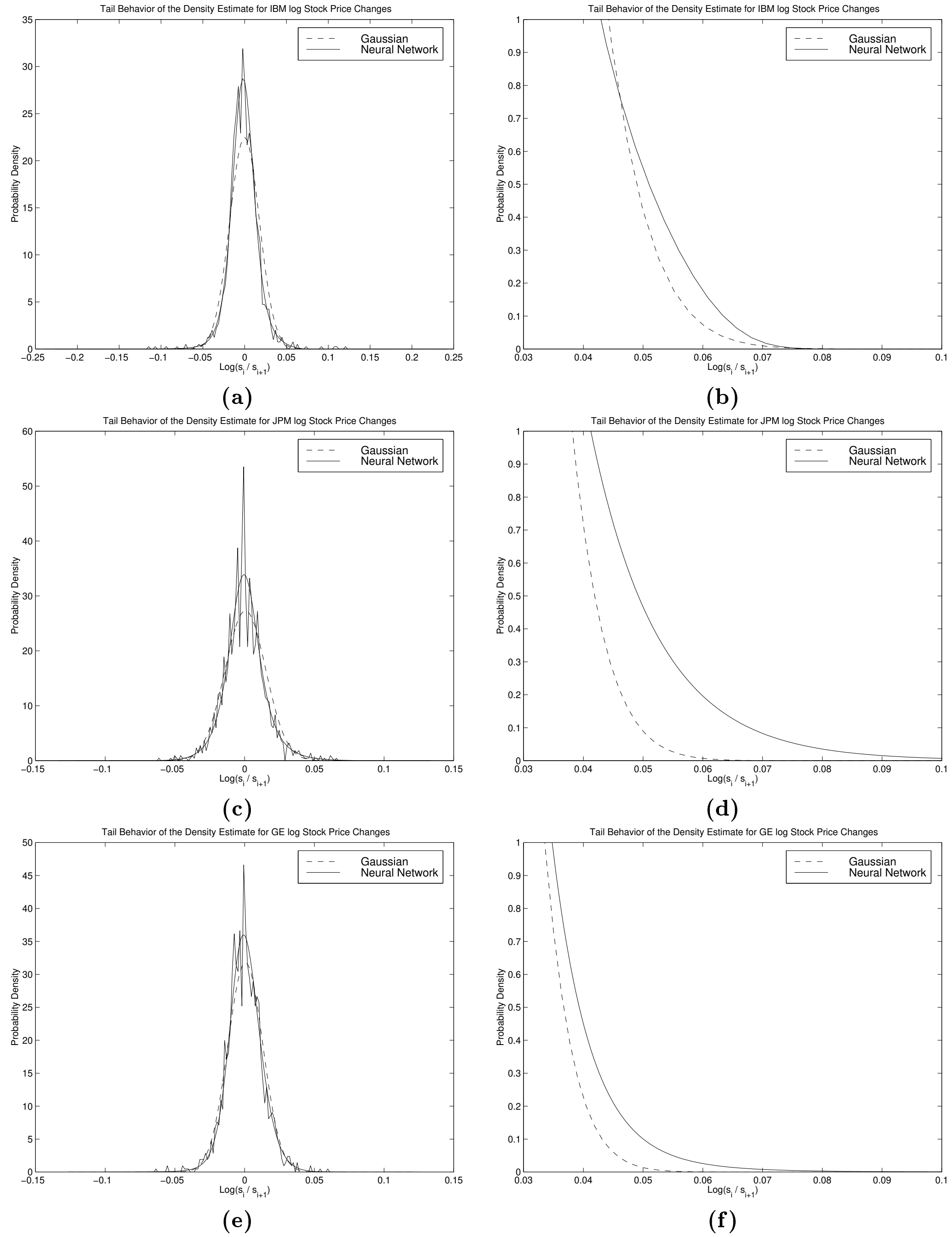
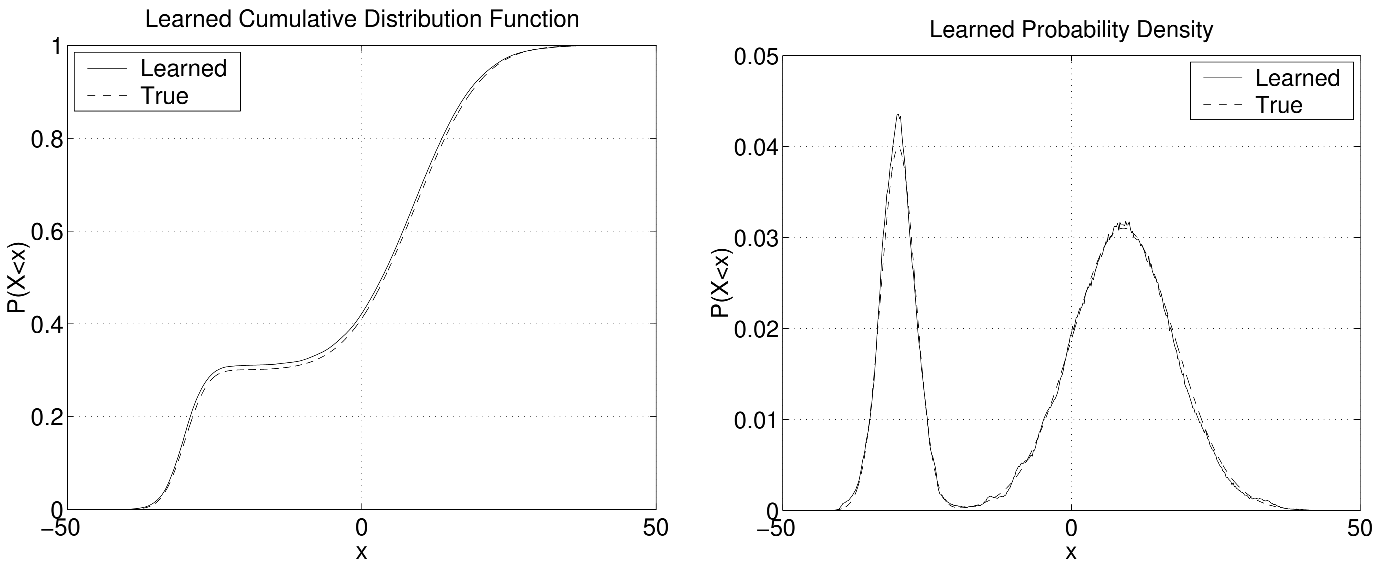
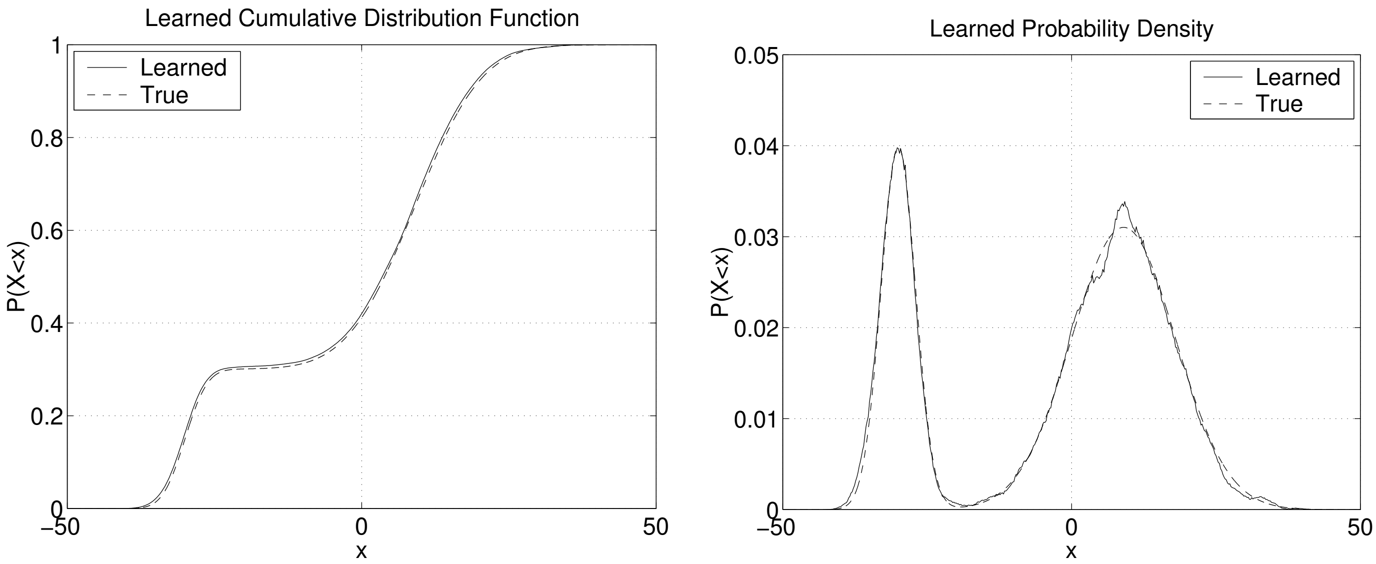


Figure 9: Density estimates for the log stock price changes of a number of U.S. stocks. Shown are the density estimates of the changes in log stock price for IBM, JP Morgan (JPM) and General Electric (GE) using SIC. For each company, about 1530 data points were available. Also shown for comparison are the histogram of the data and a Gaussian with the same mean and variance as the data . The estimated distribution is significantly non-Gaussian. We magnify the tail behavior in (b), (d) and (f). Notice that the true density has a considerably fatter tail.



(a) **(b)**

Figure 10: Using SICI for random variate generation from the density in (44). In (a) is shown the learned cumulative compared to the true cumulative, and in (b) is shown the density resulting from the random variate generation compared to the desired density.



(a) **(b)**

Figure 11: Using a Gaussian kernel technique for random variate generation from the density in (44). In (a) is shown the learned cumulative compared to the true cumulative, and in (b) is shown the density resulting from the random variate generation compared to the desired density.

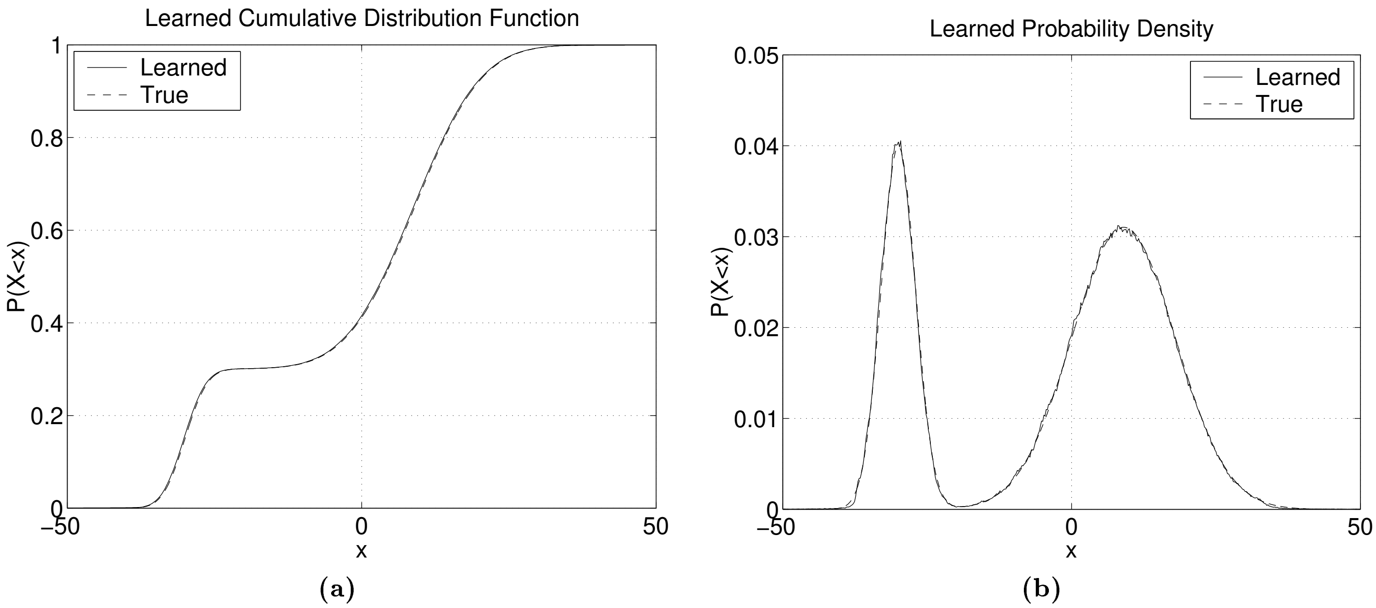


Figure 12: Learning G^{-1} for random variate generation from the density in (44) as described in section 4.2. In (a) is shown the learned cumulative compared to the true cumulative, and in (b) is shown the density resulting from the random variate generation compared to the desired density.

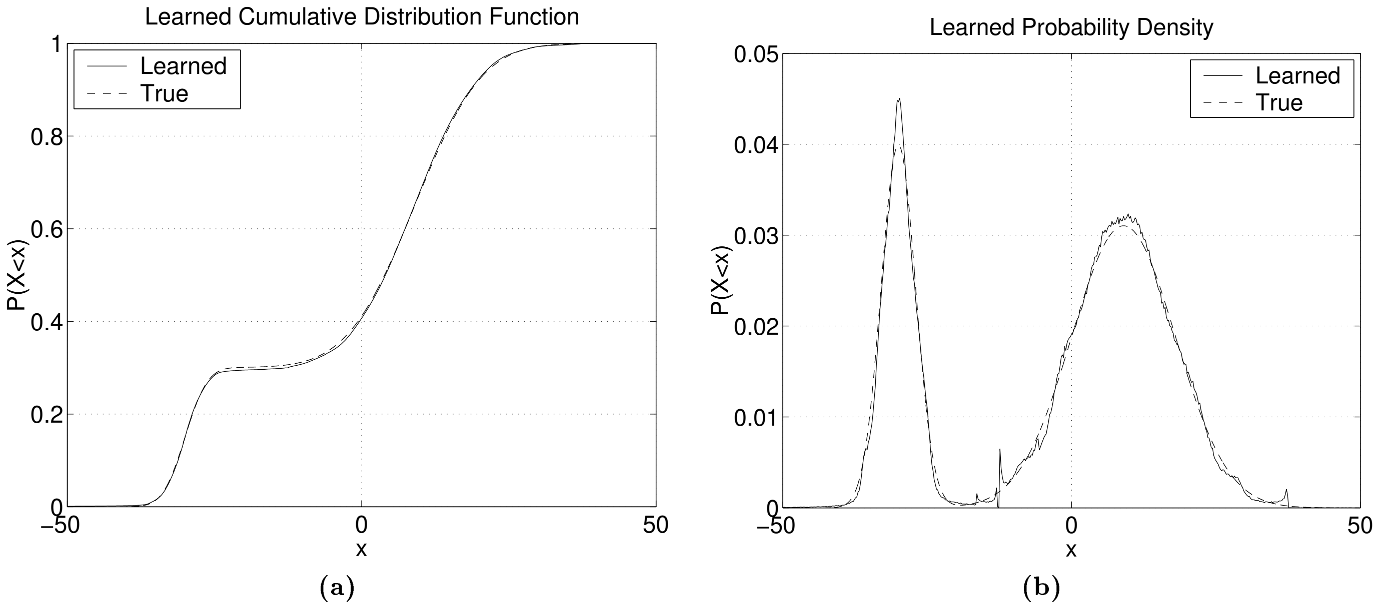


Figure 13: Control formulation for random variate generation from the density in (44). In (a) is shown the learned cumulative compared to the true cumulative, and in (b) is shown the density resulting from the random variate generation compared to the desired density.

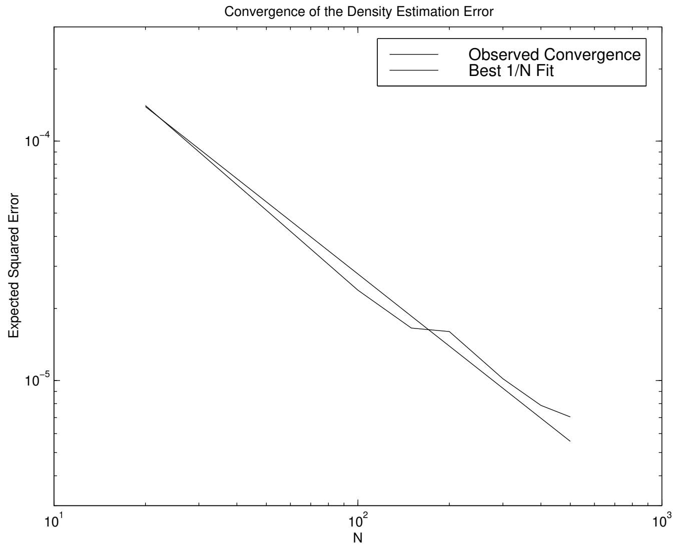


Figure 14: Convergence of the density estimation error. Plotted in the estimation error versus N on a Log-Log scale. For comparison, also shown is the best $1/N$ fit.

inputs are given by $x_i = G(y_i)$ which are obtained by numerical integration as described in section 4.2. Once again, to test how well G^{-1} was learned, we generated two million points from a uniform distribution and passed them through the network. Figure 12 shows the sample distribution and histogram of these two million network outputs. The results indicate that the network learned G^{-1} almost perfectly. In addition, the resulting density of the network output matched the true density extremely well. This implies that dG^{-1}/dx has also been learned well, which might come as a surprise as this derivative was not incorporated as part of the error function. The fact that the neural network could implement the distribution inverse and its derivative simultaneously are consequences of theorems 5.2 and 5.3 and corollary 3.8.

6.2.2 Control Formulation for Random Variate Generation

We implemented the control formulation method described in section 4.3 to generate data according to the mixture-of-Gaussians density in equation (44). We used the kernel estimator as the density estimation network H_2 and a single hidden layer network with 50 *tanh* hidden nodes and a linear output node as the density shaping network H_1. To create singularities at the end points $y = 0$ and $y = 1$, we added a tangent function to the output. Training was performed with 4000 data points using the algorithm proposed in section 4.3. The inputs to the first network were generated according to a uniform distribution. To test how well the control method performed, two million random variates were generated and their distribution was compared to the true distribution. Figure 13 shows the results. The results indicate that the generated points obey the target density fairly accurately. The difference between the true and generated densities can be made even smaller by using more training data (increasing M and N). In fact, by increasing the number of training data one can get arbitrarily close to the true distribution. This is a consequence of the theoretical results in section 5. Again, constraints on G^{-1} such as monotonicity could be used to improve the learning.

6.2.3 Convergence of Density Estimation

To test the theoretical bounds derived for the density estimator, we have performed a Monte Carlo experiment to measure the L_2 (RMS) estimation error as a function of the number of data points N. We used a five hidden unit neural network and trained it according to the SIC method. For various numbers of data points, N, the resulting density estimation error, E, was computed by averaging over 100 runs for each N. Shown in figure 14 is the behavior of $\log E$ versus $\log N$, and for comparison, we show the best $1/N$ fit. The optimal linear fit had slope -0.97.

7 Discussion

We have developed two techniques for density estimation based on the idea of learning the cumulative by mapping the data points to a uniform density. In doing so, we placed the density estimation problem, traditionally an unsupervised learning problem, within the supervised learning framework. We focused on implementations using multilayer networks, because multilayer networks have certain desirable properties such as universal approximation theorems. However, it should be noted that any sufficiently general function class could be used in conjunction with our methods.

Two techniques were presented, a stochastic technique (SLC), which is expected to inherit the characteristics of most stochastic iterative algorithms, and a deterministic technique (SIC). SLC tends to be slow in practice, however, because each set of targets is drawn from the uniform distribution, this is anticipated to have a smoothing/regularizing effect – this can be seen by comparing SLC and SIC in figure 8 (a). A similar outcome is obtained by adding small random perturbations to the targets of SIC.

Our simulations demonstrated that the our techniques performed well in comparison to the kernel estimator using the optimal kernel width. Further, there is no smoothing parameter that needs to be chosen. Smoothing occurs naturally by picking the interpolator with the lowest bound for a certain derivative. In our simulations, we enforced smoothing by starting the network at small weights, which for practical purposes seemed sufficient. For our methods, the majority of time is spent in the learning phase, but once learning is done, evaluating the density is fast. We used our methods to demonstrate the fat-tailed behavior in the stock markets – price changes in the stock markets obey a distribution that has a fatter-than-Gaussian tail.

We then developed techniques for non-uniform random variate generation that are applicable to the generation of random variates from a density represented by a set of data points (SLCI or SICI) or from a density whose functional form is given (learning of the inverse distribution). We presented a second method based on a control formulation. Our methods are applicable to any density, and the generation of random variates is fast, once the learning is performed. The learning phase is the rate limiting step, but this will become insignificant when one needs to generate millions of points, many times.

We have also provided convergence results applicable to the use of generalized distribution functions. Thus, we have laid a theoretical foundation upon which our methods stand. The conditions that we required of the true distribution function are not unreasonable from the practical point of view. We see that the L_∞ convergence rate for the density estimation approaches $O((\log\log N/N)^{1/2})$ for smooth densities. Our theoretical convergence results are near optimal and were obtained without any restrictions being placed on the support of the true distribution. However, for practical purposes, one is usually interested in some compact set.

Extensions along the following lines are possible: why restrict oneself to a uniform density? One could map to any standard density (say) $q(x)$. For example, one can replace the uniform targets for SLC by targets drawn from (say) a Gaussian density. Let the network function being implemented be $H(x, w)$, and let the mapping that converts a Gaussian random variable to a uniform be $Q(x)$. This is just the error function $erf(x)$. Then the composite mapping

$$Q(H(w, x))$$

is the required estimate $G(x)$, from which the density can be obtained by differentiation with respect to x. For SIC, the targets would be $E[u_i]$, the i^{th} order statistic of q. Thus our methods could be extended by mapping to any standard density for which $Q(x)$ is known analytically. The same comments apply to the random variate generation process. The advantage of doing this can be seen by supposing that the true density is close to a Gaussian or some other standard density. Then, we let the mapping $Q(x)$ "do the bulk of the work" and the neural network has to learn only the deviation of the true density from the standard density. This mapping could be considerably simpler to learn as it will by assumption be close to the identity mapping[6].

8 ACKNOWLEDGMENTS:

The authors would like to acknowledge the helpful comments of Dr. Kurt Hornik, Yaser Abu-Mostafa and the Caltech Learning Systems Group. The authors would like to acknowledge the support of NSF's Engineering Research Center at Caltech.

Appendix: Mathematical Derivations

In this appendix we present all the technical details for results that are used in the main part of the paper.

A Weight Updates for Control Formulation

In this part of the appendix we derive the exact weight update rule for the control algorithm in section 4.3 for the case where the density estimator (H_2) is either a neural network trained according to SIC or a kernel density estimator. The steps of the algorithm are given in section 4.3. The only change would be the weight update step as will be described next.

A.1 Neural Network Density Estimator

The error gradient can be written as

$$\frac{\partial \mathcal{E}}{\partial w} = 2\sum_{n=1}^{M}\Big[H_2(z_n, v(w)) - \mathcal{G}(z_n)\Big]\frac{\partial H_2(z_n, v(w))}{\partial w} \tag{45}$$

We need to compute $\partial H_2(z_n, v(w))/\partial w$, which by the chain rule we can write as

$$\frac{\partial H_2((z_i), v(w))}{\partial w} = \sum_k \frac{\partial H_2((z_i), v)}{\partial v_k}\frac{\partial v_k}{\partial w} \tag{46}$$

[6] A technicality in the proof of convergence with probability 1 requires that the random targets be bounded with probability 1. However we can ignore this technical requirement without significant practical loss.

where the summation is over the weights of H_2. When using SIC as the density estimator, v is a minimizer of the error function

$$\hat{\mathcal{E}} = \sum_{i=1}^{N} \left(H_2(y_i, v) - \frac{i}{N+1} \right)^2 \tag{47}$$

Define the matrix of second derivatives (the Hessian) of $\hat{\mathcal{E}}$ with respect to v by

$$\mathbf{H}_{ij} = \frac{\partial^2 \hat{\mathcal{E}}}{\partial v_i \partial v_j} \tag{48}$$

which is a function of v (and implicitly of w). Then, the lowest order term in the Taylor expansion of $\hat{\mathcal{E}}$ around v is a quadratic:

$$\hat{\mathcal{E}}(v + \Delta v) = \hat{\mathcal{E}}(v) + \Delta v^T \mathbf{H} \Delta v \tag{49}$$

A change in w_j produces a change in the y_i's. This in turn produces a change in $\hat{\mathcal{E}}$:

$$\hat{\mathcal{E}} \to \hat{\mathcal{E}} + 2\Delta w_j \underbrace{\left(\sum_{i=1}^{N} \left(H_2(y_i, v) - \frac{i}{N+1} \right) \frac{\partial H_2(y_i, v)}{\partial y_i} \frac{\partial H_1(x_i, w)}{\partial w_j} \right)}_{\Omega_j(v,w)} \tag{50}$$

Therefore, if $w_j \to w_j + \Delta w_j$, in the limit of small Δw_j, $\hat{\mathcal{E}}(v + \Delta v)$ is given by (in the limit of small Δv)

$$\hat{\mathcal{E}}(v + \Delta v) = \hat{\mathcal{E}}(v) + \Delta v^T \mathbf{H}(v, w) \Delta v + 2(\Delta v \cdot \nabla_v \Omega_j(v, w)) \Delta w_j \tag{51}$$

Setting the gradient of this expression (with respect to Δv) to zero, one obtains for the change in v

$$\Delta v = -\Delta w_j \mathbf{H}^{-1}(v, w) \nabla_v \Omega_j(v, w) \tag{52}$$

therefore we find that

$$\frac{\partial v}{\partial w_j} = -\mathbf{H}^{-1}(v, w) \nabla_v \Omega_j(v, w) \tag{53}$$

and we finally get

$$\frac{\partial \mathcal{E}}{\partial w_j} = 2 \sum_{n=1}^{M} [H_2(z_n, v) - \mathcal{G}(z_n)] (\nabla_v H_2(z_n, v)) \cdot (-\mathbf{H}^{-1}(v, w) \nabla_v \Omega_j(v, w)) \tag{54}$$

and the weight update is given by

$$w_j(t+1) = w_j(t) - \eta \frac{\partial \mathcal{E}}{\partial w_j} \tag{55}$$

(55) can be used in (52) to obtain the weight updates (Δv) for the density estimating network:

$$\Delta v = -\sum_j \Delta w_j \mathbf{H}^{-1}(v, w) \nabla_v \Omega_j(v, w) \tag{56}$$

where the summation is over the weights in H_1.

A.2 Gaussian Kernel Density Estimator

In this case let $\mathcal{G}(x) = g(x)$, the desired density. Let $g(z_n)$ be the density on the "test points". The error function once again is

$$\mathcal{E} = \sum_{n=1}^{M} \Big[H_2(z_n) - g(z_n) \Big]^2 \tag{57}$$

where now $H_2(z)$ is given by

$$H_2(z) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-y_i)^2}{2\sigma^2}} \tag{58}$$

where σ is appropriately chosen to give a good density estimator on the N points. If w are the weights of H_1 then $y_i = H_1(x_i, w)$. A straightforward computation shows that

$$\frac{\partial \mathcal{E}}{\partial w_j} = -\frac{1}{MN\sigma^3\sqrt{2\pi}} \sum_{n=1}^{M} \sum_{i=1}^{N} (H_2(z_n) - g(z_n))(H_1(x_i, w) - z_n) e^{-\frac{(y_i - z_n)^2}{2\sigma^2}} \frac{\partial H_1(x_i, w)}{\partial w_j} \tag{59}$$

and the weight update is given by

$$w_j(t+1) = w_j(t) - \eta \frac{\partial \mathcal{E}}{\partial w_j} \tag{60}$$

B Proofs for Convergence Theorems

B.1 Convergence of the Density Estimation Techniques

Theorem B.1 *Let the data $x_1 \ldots x_N$ be generated according to the true distribution G. Then, SLC converges with probability 1 to a local minimum of the error function*

$$\mathcal{E} = \sum_{n=1}^{N} \Big[H(x_n, w) - \frac{n}{N+1} \Big]^2 \tag{61}$$

provided that the learning rate $\eta(t)$ is a decreasing sequence, that satisfies the following conditions

a. $\sum_{t=1}^{\infty} \eta(t) = \infty$,

b. $\sum_{t=1}^{\infty} \eta^p(t) < \infty$, *for some* $p > 1$,

c. $\lim_{t\to\infty} \sup \big[1/\eta(t) - 1/\eta(t-1) \big] < \infty$.

PROOF: We first summarize some aspects of the theory of stochastic approximation, that will be the main tool in proving the theorem[7]. A general stochastic iterative algorithm is given by

$$w(t+1) = w(t) + \eta(t) Q\big(t, w(t), u(t)\big) \tag{62}$$

where $u(t)$ is a sequence of independent random variables (Ljung considers a more general form, where $u(t)$ is given by an iteration in terms of the random input and w). The algorithm converges with probability 1 (w.p.1) *only* to a

[7]Stochastic approximation deals with the convergence analysis of iterative algorithms where the inputs are random variables. We follow the analysis of Ljung [41], though other approaches that have different sets of assumptions, such as Kushner and Clark [39] could also be used. For more details on stochastic algorithms in general, see [11].

stable stationary point of the ordinary differential equation.

$$\frac{dw}{dt} = \lim_{t \to \infty} E\Big[Q\big(t, w, u(t)\big)\Big] \tag{63}$$

if the following assumptions are satisfied (see [41]):

A1: $u(t)$ is bounded w.p.1.

A2: The function $Q(t, w, u)$ is continuously differentiable w.r.t. w and u, and the derivatives, for fixed w are bounded in t.

A3: $\eta(t)$ is a decreasing sequence, and

- $\sum_{t=1}^{\infty} \eta(t) = \infty$,
- $\sum_{t=1}^{\infty} \eta^p(t) < \infty$ for some $p > 1$,
- $\lim_{t \to \infty} \sup\big[1/\eta(t) - 1/\eta(t-1)\big] < \infty$.

A4: $\lim_{t \to \infty} E\Big[Q\big(t, \bar{w}, u(t)\big)\Big]$ exists for all $\bar{w}$.

By observing the proof of convergence in [41], it can be shown that condition A2 can be replaced by the following condition:

A2′: Q is locally Lipschitz continuous in w and u, meaning that $|Q(t, w_1, u_1) - Q(t, w_2, u_2)| < R(w, u, \rho, v)\big(|w_1 - w_2| + |u_1 - u_2|\big)$, for $w_1,\ w_2 \in B(w, \rho)$ for some $\rho = \rho(w)$, and $u_1,\ u_2 \in B(u, v)$ for $v \geq 0$, where $B(x, y)$ is the ball centered at x with radius y. Moreover, the Lipschitz constant $R(w, u(t), 0, v(t))$ is a bounded function of t for fixed w and bounded $v(t)$.

To see why this is the case, we note that condition A2 is used in the proof in [41] only in Lemma 2, pg. 569. There, the equivalent condition A2′ can be used in its place.

The update equation for SLC is given by

$$w(t+1) = w(t) + \eta(t) Q\big(w(t), u(t)\big) \tag{64}$$

where $u(t)$ is the vector of u_n's that are presented for training cycle t, and

$$Q\big(w(t), u(t)\big) = -\sum_{n=1}^{N} \Big[H(x_n, w(t)) - u_n(t)\Big] \frac{\partial H(x_n, w(t))}{\partial w} \tag{65}$$

Note that x_n is fixed and considered constant in the proof, because it does not change from one cycle to the next. The first condition, the independence of successive $u(t)$ vectors, is satisfied, by the construction of the algorithm. We now verify that conditions A1, A2′, A3 and A4 are satisfied.

- Condition A1 is satisfied, because $u_n(t)$ is generated from a uniform density, and hence it is bounded.

- Q is a piecewise (with respect to t) differentiable function of u and w (assuming standard sigmoidal neural networks). Further, the derivatives are bounded in t for fixed w because $u_n(t)$ is bounded in t. Therefore, Q is locally Lipschitz. In addition, the local Lipschitz constant $R(w, u(t), 0, v(t))$ is bounded in time (for fixed w) as the derivatives are bounded in time for fixed w. Hence condition A2′ is satisfied.

- Condition A3 is equivalent to the conditions of the theorem.
- Let us now consider Condition A4:

$$\lim_{t\to\infty} E\Big[Q\big(w,u(t)\big)\Big] = -\lim_{t\to\infty}\sum_{n=1}^{N}\Big(H(x_n,w) - E\big[u_n(t)\big]\Big)\frac{\partial H(x_n,w)}{\partial w} \tag{66}$$

The variables $u_n(t)$ are the order statistics of a sample of size N drawn from a distribution that is uniform in $[0,1]$. The density of the order statistic can be found in most texts on probability theory, e.g. [12]. We get

$$f(u_n) = \frac{N!}{(n-1)!(N-n)!}u_n^{n-1}(1-u_n)^{N-n} \tag{67}$$

which is a beta distribution. The first moment can be obtained in a straightforward manner, as follows:

$$E(u_n) = \frac{n}{N+1} \tag{68}$$

Substituting in (66), we see that the limit exists and is independent of t for all w.

Thus, all conditions of [41] are satisfied, therefore we conclude that the algorithm converges w.p.1 to the stable stationary points of the following ordinary differential equation

$$\frac{dw}{dt} = -\sum_{n=1}^{N}\Big[H(x_n,w) - \frac{n}{N+1}\Big]\frac{\partial H(x_n,w)}{\partial w} \tag{69}$$

which are precisely the local minima of the error function

$$\mathcal{E} = \sum_{n=1}^{N}\Big[H(x_n,w) - \frac{n}{N+1}\Big]^2 \tag{70}$$

because the right-hand side of (69) is the negative gradient of the error function in (70)[8]. ■

Theorem B.2 (L_2 convergence to the true distribution) *Let the data set D consist of N data points, x_i drawn i.i.d. from the true distribution $G \in \mathcal{G}$. For every $H \in \mathcal{H}^{\nu}_{D_N}$ and every $F \in \mathcal{G}$, the inequality*

$$E\Big[\| H-G\|^2_{F,2}\Big]_D = E\Big[\int_{-\infty}^{\infty}(H(x)-G(x))^2dF(x)\Big]_D \leq \mathcal{Q}_2(N) \tag{71}$$

holds, where

$$\mathcal{Q}_2(N) = 2\left[\frac{1}{2(N+1)} + \nu(N)^2\frac{\log\log N}{N} + \frac{3}{2(N+1)^2} + \frac{2\nu(N)}{N+1}\sqrt{\frac{\log\log N}{2N}}\,\right] \tag{72}$$

PROOF: Let $x_0 = -\infty$, $x_{N+1} = \infty$.

$$\| H-G\|^2_F = \int_{-\infty}^{\infty}(H(x)-G(x))^2dF(x) = \sum_{i=0}^{N}\int_{x_i}^{x_{i+1}}(H(x)-G(x))^2dF(x) \tag{73}$$

[8]Observing this proof, it should be clear that if the hint term had been kept in (4) the proof would have proceeded in exactly the same manner resulting in convergence to the error function in (4). If we require that the solution be monotonic, then the hint term must be zero, therefore, the function converged to must be a minimum of the remaining term which is precisely the error function (70). Ultimately, the same final result is obtained.

Now by definition, $G(x_i) = i/N+1+\epsilon_i$ and $H(x_i) = i/N+1-\eta_i$. Let $x \in [x_i, x_{i+1}]$. Because H and G are monotonically increasing, we have the inequalities $H(x_i) \leq H(x) \leq H(x_{i+1})$ and $G(x_i) \leq G(x) \leq G(x_{i+1})$ therefore,

$$\epsilon_i \leq G(x) - \frac{i}{N+1} \leq \epsilon_{i+1} + \frac{1}{N+1}$$
$$-\eta_i \leq H(x) - \frac{i}{N+1} \leq -\eta_{i+1} + \frac{1}{N+1} \tag{74}$$

Thus, using the well known inequality $(a-b)^2 \leq 2(a^2+b^2)$ with $a = G(x) - i/N+1$ and $b = H(x) - i/N+1$ we have:

$$\begin{aligned}
(G(x)-H(x))^2 &\leq 2\left(\left(G(x) - \frac{i}{N+1}\right)^2 + \left(H(x) - \frac{i}{N+1}\right)^2\right) && (75)\\
&\leq 2\left(\epsilon_i^2 + \left(\epsilon_{i+1} + \frac{1}{N+1}\right)^2 + \right. && (76)\\
&\qquad \left. \eta_i^2 + \left(\eta_{i+1} - \frac{1}{N+1}\right)^2\right) && (77)
\end{aligned}$$

Combining this inequality with (21), we get

$$\begin{aligned}
E\left[(G(x)-H(x))^2\right]_D &\leq 2\left(\frac{2}{(N+1)^2} + \frac{i(N+1-i)+(i+1)(N-i)}{(N+1)^2(N+2)} + \right.\\
&\qquad \left. \eta_i^2 + \eta_{i+1}^2 - \frac{2}{N+1}\eta_{i+1}\right)\\
&\leq \mathcal{Q}_2(N), && (78)
\end{aligned}$$

where the last inequality follows by maximizing over i and using the bound for the η_i's. Thus we find

$$\begin{aligned}
E\left[\| H - G \|_F^2\right]_D &= \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} E\left[(H(x)-G(x))^2\right]_{x_i} dF(x), && (79)\\
&\leq \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} \mathcal{Q}_2(N) dF(x), && (80)\\
&= \mathcal{Q}_2(N) && (81)
\end{aligned}$$

The last line follows because $\int_{-\infty}^{\infty} dF(x) = 1$ since F is a distribution function. ■

The previous theorem can easily be extended to obtain the L_1 convergence.

Theorem B.3 (**L_1 convergence to the true distribution**) *Let the data set D consist of N data points, x_i drawn i.i.d. from the true distribution $G \in \mathcal{G}$. For every $H \in \mathcal{H}_{D_N}^{\nu}$ and every $F \in \mathcal{G}$, the inequality*

$$E\left[|H-G|_F\right]_D = E\left[\int_{-\infty}^{\infty} |H(x) - G(x)| dF(x)\right]_D \leq \mathcal{Q}_1(N) \tag{82}$$

holds, where

$$\mathcal{Q}_1(N) = \frac{1}{\sqrt{N+2}} + 2\nu(N)\sqrt{\frac{\log\log N}{2N}} + \frac{2}{N+1} \tag{83}$$

Proof: Using similar reasoning that led up to (74), we find for $x \in [x_i, x_{i+1}]$

$$|H(x) - G(x)| \leq |\epsilon_i| + |\epsilon_{i+1}| + \frac{2}{N+1} + |\eta_i| + |\eta_{i+1}| \tag{84}$$

By Jensen's inequality, $E[|\epsilon_i|] \leq \sqrt{E[\epsilon_i^2]}$ therefore we find that $E[|\epsilon_i|] \leq \frac{1}{2}\sqrt{1/(N+2)}$. The rest of the analysis is exactly analogous to the L_2 convergence proof, the final result being (82). ■

Thus we have L_1 convergence at a rate $O(1/\sqrt{N})$ for $\nu(N) \leq 1/\sqrt{\log\log N}$. Let H_N denote a sequence of functions in $\mathcal{H}^{\nu}_{D_N}$ for increasing data set sizes N. Define the statistics

$$\mathcal{D}_{H_N} = \sup_x |H_N(x) - G(x)| \tag{85}$$

$$\mathcal{C}_{H_N} = \int_{-\infty}^{\infty} (H_N(x) - G(x))^2 dG(x) \tag{86}$$

which are analogous to (14), (15). The following theorems are valid.

Theorem B.4 (L_∞ convergence to the true distribution)

$$\limsup_{N\to\infty} D_N^H \sqrt{\frac{N}{2\log\log N}} \leq \lim_{N\to\infty} \nu(N) + \frac{1}{2} \tag{87}$$

where we assume that $\lim_{N\to\infty} \nu(N)$ *exists.*

Proof: From the proof of theorem 3.5, it is easily seen that $|H - G_N| \leq |H - i/(N+1)| + |G_N - i/(N+1)| \leq 2(\nu(N)\sqrt{\log\log N/2N} + 1/(N+1))$, hence

$$\begin{aligned}
\limsup_{N\to\infty} D_N^H \sqrt{\frac{N}{2\log\log N}} &\leq \limsup_{N\to\infty} \left(\sup_x |H - G_N| + \sup_x |G_N - G|\right) \sqrt{\frac{N}{2\log\log N}} \\
&\leq \lim_{N\to\infty} \nu(N) + \frac{1}{2}
\end{aligned} \tag{88}$$

where the last equation follows by an application of theorem 3.2. ■

In an exactly analogous way, the following theorem can be established.

Theorem B.5

$$\limsup_{N\to\infty} C_N^H \frac{N}{2\log\log N} \leq \lim_{N\to\infty} \frac{\nu(N)}{2} + \frac{2}{\pi^2} \tag{89}$$

where we assume that $\lim_{N\to\infty} \nu(N)$ *exists.*

The proof follows in an analogous way to the previous theorem and an application of theorem 3.3. ■

To prove convergence to the true density, we will need the following results relating the higher derivatives of a function to lower order derivatives. In the following, we use the notation $f^{(i)}$ to denote the i^{th} derivative of f.

Lemma B.6 *Let a twice differentiable function $f(x)$ be defined on the interval (a, ∞) where the lower limit, a, could be $-\infty$. Suppose $\sup_x |f| = M_0$, $\sup_x |f'| = M_1$, and $\sup_x |f''| = M_2$, where the $\sup_x$ is taken over $x \in (a, \infty)$. Then,*

$$M_1^2 \leq 4M_0M_2 \tag{90}$$

PROOF: Let $h > 0$. Then for $x \in (a, \infty)$, by Taylor's theorem, we have

$$f(x+2h) = f(x) + 2hf'(x) + (2h)^2 \frac{f''(\zeta)}{2} \tag{91}$$

for some $\zeta \in (x, x+2h)$. Therefore,

$$|f'(x)| \leq \frac{M_0}{h} + hM_2 \tag{92}$$

This inequality holds for every $h > 0$. In particular for the h that minimizes the right hand side. This minimum is attained at $h = \sqrt{M_0/M_2}$ and so we have the bound

$$|f'(x)| \leq 2\sqrt{M_0 M_2} \tag{93}$$

Thus $\sup_x |f'(x)| \leq 2\sqrt{M_0 M_2}$. ■

Corollary B.7 *Let $G(x)$ be differentiable m times on the interval (a, ∞). Let $\sup_x \left|G^{(i)}\right| = M_i$ for $i = 1 \ldots m$. Then*

$$M_i^2 \leq 4M_{i-1}M_{i+1} \tag{94}$$

for $i = 1 \ldots (m-1)$.

PROOF: Let $f(x)$ in lemma B.6 be $G^{(i-1)}(x)$. ■

Lemma B.8 *Let $G(x)$ be differentiable m times on the interval (a, ∞). Let $\sup_x \left|G^{(i)}\right| = M_i$ for $i = 1 \ldots m$. Then*

$$M_1 \leq 2^{m-1} M_0^{1-\frac{1}{m}} M_m^{\frac{1}{m}} \tag{95}$$

PROOF: We prove the lemma by induction on m. For $m = 2$, the lemma is equivalent to lemma B.6. Suppose (95) holds for $m = P$. We show that it holds for $m = P + 1$. From lemma B.6 we have

$$M_1 \leq 2M_0^{\frac{1}{2}} M_2^{\frac{1}{2}} \tag{96}$$

By applying the induction hypothesis to G' we have the following equation

$$M_2 \leq 2^{P-1} M_1^{1-\frac{1}{P}} M_{P+1}^{\frac{1}{P}} \tag{97}$$

Combining these two equations we find that

$$M_1 \leq \left(2^{\frac{P+1}{2}} M_0^{\frac{1}{2}} M_{P+1}^{\frac{1}{2P}}\right)^{\frac{2P}{P+1}}, \tag{98}$$

and thus (95) holds for $m = P + 1$. This concludes the proof. ■

Corollary B.9 *Let $G(x) \in \mathcal{C}^\infty$ on the interval (a, ∞). Let $\sup_x \left|G^{(i)}\right| = M_i$. Define the sequence $\{s_k\}_{k=0}^{\infty}$ by $s_k = 2^{k-1} M_k^{1/k}$. Suppose that $\lim_{k\to\infty} s_k = \beta < \infty$. Then*

$$M_1 \leq \beta M_0 \tag{99}$$

PROOF: Taking the limit $m \to \infty$ in lemma B.8 and using the fact that the term by term product of two convergent sequences converges to the product of the limits, the corollary follows. ■

We assume that the true distribution function has bounded derivatives up to order K. Then, in the asymptotic limit $N \to \infty$, one expects that with probability 1, an interpolator should exist with bounded derivatives with the same bounds, because in the asymptotic limit, $G(x)$ itself becomes an interpolator. To proceed in a more formal manner, let $A_i = \sup_x \left|G^{(i)}\right|$, $i = 1 \ldots K$ and define $B_i^\nu(D)$ by

$$B_i^\nu(D) = \inf_{H \in \mathcal{H}_{D_N}^\nu} \sup_x \left|H^{(i)}\right| \tag{100}$$

for fixed $\nu(N) = \nu > 0$. Note that by definition, for all $\epsilon > 0$, $\exists\ H \in \mathcal{H}_{D_N}^\nu$ such that $\sup_x \left|H^{(i)}(x)\right| \leq B_i^\nu + \epsilon$. $B_i^\nu(D)$ is the lowest possible bound on the i^{th} derivative for the ν-approximate sample distribution functions given a particular data set. In a sense, the "smoothest" approximating sample distribution function with respect to the i^{th} derivative has an i^{th} derivative bounded by $B_i^\nu(D)$ One expects that $B_i^\nu \leq A_i$, at least in the limit $N \to \infty$. This is the content of the next lemma. Let $B_i^\nu = A_i + \eta_i^\nu(D)$.

Lemma B.10 *Let $\nu(N) > 2$. Then, for all $\delta > 0$*

$$\lim_{N \to \infty} P[\eta_i^\nu(D) < \delta] = 1 \tag{101}$$

PROOF: This is a straightforward application of theorem 3.2 as follows. Suppose the lemma were false. Then there exists $\delta > 0$ such that $\lim_{N \to \infty} P[\eta_i^\nu(D) \geq \delta] \geq \gamma > 0$. Therefore with probability$\geq \gamma$, $G \notin \mathcal{H}_{D_N}^\nu$ as $N \to \infty$, as if $G \in \mathcal{H}_{D_N}^\nu$ then $\eta \leq 0$. Thus, with probability $\geq \gamma$,

$$\mathcal{D}_N = \sup_x |G_N(x) - G(x)| > (\nu - 1)\sqrt{\frac{\log\log(N)}{2N}} \tag{102}$$

as $N \to \infty$ (the $(\nu - 1)$ arises because our distribution function differs from G_N at x_i by at most $1/N < \sqrt{\log\log N/2N}$). Thus,

$$\lim_{N \to \infty} D_N \sqrt{\frac{N}{2\log\log N}} \geq \lim_{N \to \infty} \frac{\nu - 1}{2} > \frac{1}{2}$$

with probability$\geq \gamma > 0$, implying that

$$\limsup_{N \to \infty} D_N \sqrt{\frac{N}{2\log\log N}} > \frac{1}{2} \tag{103}$$

with probability$\geq \gamma > 0$, contradicting theorem 3.2. ■

Theorem B.11 (Convergence in probability to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0 \ldots K$, where $K \geq 2$. Let $\nu(N) > 2$ and fix $\epsilon > 0$. Let $B_K^\nu(D) = \inf_{H \in \mathcal{H}_{D_N}^\nu} \sup_x \left|H^{(K)}\right|$. Let $H \in \mathcal{H}_{D_N}^\nu$ be a ν-approximate distribution function with $B_K = \sup_x \left|H^{(K)}\right| \leq B_K^\nu + \epsilon$ (by the definition of B_K^ν, such a ν-approximate distribution function must exist). Then, the inequality*

$$\sup_x |H'(x) - G'(x)| \leq \mathcal{F}(N) \tag{104}$$

where

$$\mathcal{F}(N) = 2^{(K-1)}(2A_K+\epsilon)^{\frac{1}{K}}\left[(1+\nu(N))\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1}\right]^{1-\frac{1}{K}} \tag{105}$$

holds with probability 1, as $N \to \infty$. By this is meant

$$\lim_{N\to\infty} P\left[\sup_x |H'(x) - G'(x)| \le \mathcal{F}(N)\right] = 1 \tag{106}$$

PROOF: Let $S(x) = G(x) - H(x)$ and let $M_K = \sup_x \left|S^{(K)}\right| \le A_K + B_K$. By lemma B.8

$$\sup_x |S'(x)| \le 2^{(K-1)}(A_K+B_K)^{\frac{1}{K}} \sup_x |S(x)|^{1-\frac{1}{K}} \tag{107}$$

where $B_K = \sup_x \left|H^{(K)}\right|$ and by construction, $B_K \le B_K^\nu(D) + \epsilon$. Therefore,

$$P\left[\sup_x |S'(x)| \le \mathcal{F}(N)\right] \ge P\left[2^{(K-1)}(A_K+B_K)^{\frac{1}{K}} \sup_x |S(x)|^{1-\frac{1}{K}} \le \mathcal{F}(N)\right]$$

If we can show that the right hand side tends to 1 as $N \to \infty$ then we are done. We look at the probability of the complementary event and show that it tends to 0. We can bound $P[2^{(K-1)}(A_K+B_K)^{\frac{1}{K}} \sup_x |S(x)|^{1-\frac{1}{K}} > \mathcal{F}(N)]$ by

$$P\left[\left\{(A_K+B_K) > (2A_K+\epsilon)\right\} \; OR \; \left\{\sup_x |S(x)| > (1+\nu(N))\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1}\right\}\right]$$

which in turn can be bounded by the sum of probabilities of the individual events. By lemma B.10, as $N \to \infty$, $B_K^\nu(D) \le A_K$ with probability 1, so,

$$\lim_{N\to\infty} P[(A_K+B_K) > (2A_K+\epsilon)] = 0 \tag{108}$$

Similar reasoning to that which led to (74) can now be applied to show that for $x \in [x_i, x_{i+1}]$,

$$\epsilon_i + \eta_{i+1} - \frac{1}{N+1} \le G(x) - H(x) \le \epsilon_{i+1} + \eta_i + \frac{1}{N+1} \tag{109}$$

therefore

$$\sup_x |S(x)| \le \max_i\{|\epsilon_i| + |\epsilon_{i+1}|\} + \nu(N)\left(\frac{2\log\log N}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1} \tag{110}$$

thus, we see that

$$\lim_{N\to\infty} P\left[\sup_x |S(x)| > \left[(1+\nu(N))\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1}\right]\right] = 0 \tag{111}$$

because $\lim_{N\to\infty} P[\max_i |\epsilon_i| > \sqrt{\log\log N/N}] = 0$, by an application of theorem 3.2. Therefore we have shown that $\lim_{N\to\infty} P[2^{(K-1)}(A_K+B_K)^{\frac{1}{K}} \sup_x |S(x)|^{1-\frac{1}{K}} > \mathcal{F}(N)] \le 0$ thus proving the theorem. ■

The following is an immediate corollary of the theorem.

Corollary B.12 (L_p convergence to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0 \dots K$, where $K \ge 2$. Let $\nu(N) > 1$ and fix $\epsilon > 0$. Let $B_K^\nu(D) = \inf_{H\in\mathcal{H}_{D_N}^\nu} \sup_x \left|H^{(K)}\right|$. Let $H \in \mathcal{H}_{D_N}^\nu$ be a ν-approximate distribution function with $B_K = \sup_x \left|H^{(K)}\right| \le B_K^\nu + \epsilon$*

(by the definition of B_K^ν, such a ν-approximate sample distribution function must exist). Then, for any $F \in \mathcal{G}$, as $N \to \infty$,

$$\lim_{N\to\infty} P\left[\left(\int |H'(x) - G'(x)|^p dF(x)\right)^{1/p} \leq \mathcal{F}(N)\right] = 1 \tag{112}$$

where $\mathcal{F}(N)$ is as defined in theorem 3.6.

B.2 Convergence of the Random Variate Generation Techniques

Theorem B.13 *Let the data $x_1 \dots x_N$ be generated according to the true distribution G. Then, SLCI converges with probability 1 to a local minimum of the error function*

$$\mathcal{E} = E\left[\sum_{n=1}^{N}\Big[H(u_n, w) - x_n\Big]^2\right], \tag{113}$$

(where the expectation is with respect $u_1, \dots, u_N$, the N order statistics of a uniform distribution), provided that the learning rate $\eta(t)$ is a decreasing sequence, that satisfies the following conditions

a. $\sum_{t=1}^{\infty} \eta(t) = \infty$,

b. $\sum_{t=1}^{\infty} \eta^p(t) < \infty$, *for some* $p > 1$,

c. $\lim_{t\to\infty} \sup\big[1/\eta(t) - 1/\eta(t-1)\big] < \infty$.

Proof:(sketch) We will use ideas very similar to those used in the proof of SLC→SIC. The update rule for SLCI is

$$w(t) = w(t+1) - \eta(t) \cdot \underbrace{2\sum_{n=1}^{N}(H(u_n(t), w(t)) - x_n)\frac{\partial H(u_n(t), w(t))}{\partial w}}_{Q(w(t),u(t))} \tag{114}$$

Just as in section 3.1, $Q(w(t), u(t))$ satisfies requirements $A1$, $A2'$, $A3$ and $A4$ of [41]. Therefore, convergence to a stable stationary point of the differential equation

$$\frac{dw}{dt} = \lim_{t\to\infty} E[2\sum_{n=1}^{N}(H(u_n(t), w(t)) - x_n)\frac{\partial H(u_n(t), w(t))}{\partial w}] \tag{115}$$

occurs with probability 1, provided $\eta(t)$ satisfies the conditions of the theorem. ■

Theorem B.14 *Let the data $x_1 \dots x_N$ be generated according to G. If the conditions of theorem B.13 hold, then, in the limit $N \to \infty$, SLCI converges with probability 1 to a minimum of the error function*

$$\mathcal{E} = \sum_{n=1}^{N}\left[H\left(\frac{n}{N+1}, w\right) - x_n\right]^2, \tag{116}$$

Proof: (sketch) In the limit $N \to \infty$, we expect $u_n \to n/(N+1)$ with probability 1. This, however, is just an application of theorem 3.2. Another way to see this is to write $u_i = i/(N+1) + \epsilon_i$ in (113) and expand as a power series in ϵ_i. The ϵ_i terms disappear and the higher order expectations decay to zero as $N \to \infty$. ■

Theorem B.15 **(L_2 convergence to G^{-1})** *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. Assume that G^{-1} is continuous on the closed interval $[0,1]$ and let its derivative be bounded by S. For every generalized inverse distribution function that is zero outside the range of G^{-1} and for any continuous measure $d\mu(u)$ on $[0,1]$ (with bound B on $[0,1]$), the inequality*

$$E\left[\| H^{-1} - G^{-1} \|_{\mu}^{2}\right]_D \leq \mathcal{Q}_3(N) \tag{117}$$

holds, where

$$\mathcal{Q}_3(N) = \frac{1}{N}\left(\frac{S^2}{2} + \nu(N)^2 \log\log N + \sqrt{\frac{S^2\nu(N)^2\log\log N}{N}} + \frac{2S^2}{N} + 8BS^2\right) \tag{118}$$

Proof: Because both G^{-1} and H^{-1} are monotonic, the same arguments that led up to (74) can be used to get that for $u \in [u_i, u_{i+1}]$,

$$\begin{aligned} \xi_i \leq G^{-1}(x) - x_i &\leq \xi_{i+1} + x_{i+1} - x_i \\ -\kappa_i \leq H(x) - x_i &\leq -\kappa_{i+1} + x_{i+1} - x_i \end{aligned} \tag{119}$$

Now, following similar reasoning as in theorem 3.5 we get that for $u \in [u_i, u_{i+1}]$,

$$(H^{-1}(u) - G^{-1}(u))^2 \leq 2\left(\xi_i^2 + (G^{-1}(u_{i+1}) - x_i)^2 + \kappa_i^2 + \kappa_{i+1}^2 + 2\kappa_{i+1}|x_{i+1} - x_i| + (x_{i+1} - x_i)^2\right) \tag{120}$$

The quantities on the right hand side of (120) can be bounded as follows. The density for the order statistic x_i is [12, pg 200]

$$\tilde{g}(x_i) = \frac{N!}{(i-1)!(N-i)!} g(x_i) G(x_i)^{i-1}(1 - G(x_i))^{N-i} \tag{121}$$

We will require the expected value $E[(x_i - G^{-1}(u_{i+1}))^2]$. It can be bounded as follows

$$\begin{aligned} E[(x_i - G^{-1}(u_{i+1}))^2] &= \int_{-\infty}^{\infty} dx_i\ (x_i - G^{-1}(u_{i+1}))^2 \tilde{g}(x_i) && (122)\\ &\overset{(a)}{=} \int_0^1 du\ \beta_{i,N+1-i}(u)(G^{-1}(u) - G^{-1}(u_{i+1}))^2 && (123)\\ &\overset{(b)}{=} \int_0^1 du\ \beta_{i,N+1-i}(u) \frac{dG^{-1}(\zeta)}{du}^2 (u - u_{i+1})^2 && (124)\\ &\leq S^2 E[(u - u_{i+1})^2]_{\beta_{i,N+1-i}} && (125)\\ &\leq \frac{S^2}{4(N+1)} && (126) \end{aligned}$$

where

$$\beta_{\alpha,\beta}(u) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} u^{\alpha-1}(1-u)^{\beta-1} \tag{127}$$

(a) follows by using (121) and making a transformation of variables to $u = G(x_i)$, and (b) by Taylor's theorem, for some $\zeta(u)$. The expectations are with respect to the relevant β-distributions and can be calculated using standard methods. Then, straightforward bounds for these expectations gives (126). Remembering that $G(x_i) = 1/(N+1) + \epsilon_i$, we have

$$E\left[(x_{i+1}-x_i)^2\right] = E\left[\left(G^{-1}\left(\frac{i}{N+1}+\epsilon_i\right)-G^{-1}\left(\frac{i+1}{N+1}+\epsilon_{i+1}\right)\right)^2\right] \tag{128}$$

$$\stackrel{(a)}{\leq} S^2E\left[\left(\frac{1}{N+1}+\epsilon_{i+1}-\epsilon_i\right)^2\right] \tag{129}$$

$$\leq \frac{2}{(N+1)^2} \tag{130}$$

where (a) follows because the derivative of G^{-1} is bounded by S and the last inequality follows by using (21) and obtaining a straight forward bound. Finally we can bound $E\left[\xi_i^2\right]$ by noting that $x_i = G^{-1}(u_i+\epsilon_i) = G^{-1}(u_i) + (dG^{-1}(\zeta)/du)\epsilon_i$ for $\zeta(u)$ some point in $(u_i, u_i+\epsilon_i)$. Thus, $\xi_i^2 \leq S^2\epsilon_i^2$ and using (21) we can bound $\epsilon_i^2 \leq 1/4(N+1)$. Putting all this together, we find that that for $u \in [1/(N+1), N/(N+1)]$,

$$E\left[(H^{-1}(u)-G^{-1}(u))^2\right] \leq \frac{1}{N}\left(\frac{S^2}{2}+\nu(N)^2\log\log N+\sqrt{\frac{S^2\nu(N)^2\log\log N}{N}}+\frac{2S^2}{N}\right) \tag{131}$$

In the two end cells $[0, 1/(N+1)]$ and $[N/(N+1), 1]$, the difference between H^{-1} and G^{-1} is bounded by $2S$, the measure is bounded by $2B/(N+1)$ because $d\mu(u)$ was assumed to be a continuous measure and therefore can be bounded on this compact set (by B). Hence the two end cells contribute at most $8BS^2/N$. The theorem now follows by adding this to (131). ■

Theorem B.16 (Convergence in probability to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0\ldots K$, where $K \geq 2$, and let $G^{(1)} \geq \delta$ Fix $\epsilon > 0$ and let $\nu(N) > 2/(\delta-\epsilon)$. Let $B_K^\nu(D) = \inf_{Q^{-1}}\sup_x \left|Q^{(K)}\right|$ where Q^{-1} represents a generalized ν-approximate inverse distribution function and Q the corresponding distribution function. Let H^{-1} be a generalized ν-approximate inverse distribution function with $B_K = \sup_x \left|H^{(K)}\right| \leq B_K^\nu + \epsilon$ (by the definition of B_K^ν, such a ν-approximate inverse distribution function must exist). Then, the inequality*

$$\sup_x |H'(x)-G'(x)| \leq \mathcal{F}(N) \tag{132}$$

where

$$\mathcal{F}(N) = 2^{(K-1)}(2A_K+\epsilon)^{\frac{1}{K}}\left[(1+\nu(N))\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}}+\frac{2}{N+1}\right]^{1-\frac{1}{K}} \tag{133}$$

holds with probability 1, as $N \to \infty$. By this is meant

$$\lim_{N\to\infty} P\left[\sup_x |H'(x)-G'(x)| \leq \mathcal{F}(N)\right] = 1 \tag{134}$$

Proof:(sketch) We see that $|\kappa_i| \leq |\eta_i| \sup_x dH^{-1}(u)/du$ where $H(x_i) = i/(N+1) - \eta_i$. Thus if $|\kappa_i| > 2/\delta$ then an analogous lemma to B.10 will apply: with probability 1 in the limit $N \to \infty$, there will exist generalized inverse distribution functions such that $H^{(i)}$ is bounded above by $A_i + \epsilon$ and $H^{(1)}$ is bounded below by $\delta - \epsilon$ for $\epsilon > 0$. By choice of ν we see that H (corresponding to H^{-1} as chosen) is a generalized approximate sample distribution function that satisfies the conditions of theorem B.11. Thus theorem 3.6 applies to the convergence of H' to G' and we are done. ■

References

[1] Y. Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6:192–198, 1990.

[2] Y. Abu-Mostafa. Hints. *Neural Computation*, 4(7):639–671, 1995.

[3] T. Adali, X. Liu, and K. Sönmez. Conditional distribution learning with neural networks and its application to channel equalization. *IEEE Trans. Signal Processing*, 45(4):1051–1064, 1997.

[4] R. A. Adams. *Sobolev Spaces.* Academic Press, New York, 1975.

[5] A. Atiya and C. Ji. How initial conditions affect generalization performance in large networks. *IEEE Transactions on Neural Networks*, 8(2):448–451, March 1997.

[6] Amir Atiya. On the required size of multilayer networks for implementing real-valued functions. In *Proc. IEEE World Congress on Computational Intelligence*, 1994.

[7] A. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[8] A. Barron and T. Cover. Minimimum complexity density estimation. *IEEE Transactions on Information Theory*, 37(4):1034–1054, July 1991.

[9] A. Barron, L. Györfi, and E. van der Meulen. Distribution estimation consistent in total variation and in two types of information divergence. *IEEE Transactions on Information Theory*, 38(5):1437–1454, September 1992.

[10] A. R. Barron and C-H Sheu. Approximation of density functions by sequences of exponential families. *Annals of Statistics*, 19(3):1347–1369, 1991.

[11] A. Benveniste, M. Métivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations.* Springer-Verlag, New York, 1990.

[12] P. Billingsley. *Probability and Measure.* Wiley Series in Probability and Mathematical Statistics. Wiley, 1986.

[13] C. M. Bishop. Mixture density networks. *Neural Computing Research Group Report, NCRG/4288*, 1994.

[14] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[15] C. M. Bishop and C. Legleye. Estimating conditional probability densities for periodic variables. *Advances in Neural Information Processing Systems*, 7, 1995.

[16] D. L. Bonoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Density estimation by wavelet thresholding. *The Annals of Statistics*, 24(2):508–539, 1996.

[17] J. Bretagnolle and C. Huber. Estimation des densités: Risque minimax. *Z. Wahrsch. Verw. Gebiete*, 47:119–137, 1979.

[18] K. L. Chung. An estimate concerning the Kolmogorov limit distribution. *Transactions of the American Mathematical Society*, 67:36–50, 1949.

[19] B. R. Crain. Estimation of distributions using orthogonal expansions. *Annals of Statistics*, 2:454–463, 1974.

[20] E. Csáki. An iterated logarithm law for semimartingales and its application to the empirical distribution function. *Studia Sci. Math. Hung.*, 3:287–292, 1968.

[21] J. Cwik and J. Koronacki. Probability density estimation using a gaussian clustering algorithm. *Neural Computing & Applications*, 4(3):149–160, 1996.

[22] G. M. de Montricher, R. A. Tapia, and J. R. Thompson. Nonparametric maximum likelihood of probability densities by penalty function methods. *Annals of Statistics*, 3:1329–1348, 1975.

[23] L. Devroye. *Non Uniform Random Variate Generation.* Springer-Verlag, New york, 1986.

[24] L. Devroye and L. Györfi. No empirical probability measure can converge in total variation sense for all distributions. *Annals of Statistics*, 18:1496–1499, 1990.

[25] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, 1973.

[26] S. Y. Efroimovich. Non-parametric estimation of a density of unknown smoothness. *Theory of Probability and its Applications*, 30(3):557–568, 1985.

[27] S. Y. Efroimovich. Sequential non-parametric estimation of a density. *Theory of Probabability and its Applications*, 34(2):228–239, 1986.

[28] S. Y. Efroimovich and M. S. Pinsker. Estimation of square integrable probability density of a random variable. *Problems Inform. Transmission*, 18:175–189, 1983.

[29] H. Finkelstein. The law of the iterated logarithm for empirical distributions. *Annals of Mathematical Statistics*, 42(2):607–615, 1971.

[30] K. Fukunaga and L. D. Hostetler. Optimization of k-nearest neighbor density estimates. *IEEE Transactions on Information Theory*, 19(3):320–326, 1973.

[31] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Scientific Publishers, 1992.

[32] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, CA, 1991.

[33] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[34] K. Hornik, M. Stinchcombe, H. White, and P. Auer. Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Computation*, 6:1262–1275, 1994.

[35] D. Husmeier and J Taylor. Predicting conditional probability densities of stationary stochastic time series. *Neural Networks*, 10(3):479–498, April 1997.

[36] D. Husmeier and J Taylor. Predicting conditional probability densities: improved training scheme combining em and rvfl. *Neural Networks*, 11(1):89–116, January 1998.

[37] G. Johnson. Construction of particular random processes. In *Proceedings IEEE*, volume 82(2), pages 270–281, February 1994.

[38] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1981.

[39] H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, 1978.

[40] R. Larsen and M. Marx. *An Introduction to Mathematical Statistics.* Prentice-Hall, 1986.

[41] L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, August 1977.

[42] D. Martinez. Neural tree density estimation for novelty detection. *IEEE Transactions on Neural Networks*, 9(2):519–523, 1994.

[43] E. Masry. Probability density estimation from dependent observations using wavelets orthonormal bases. *Statistics & Probability Letters*, 21:181–194, 1994.

[44] G. Miller and D. Horn. Probability density estimation using entropy maximization. *Neural Computation*, 10(7):1925–1938, 1998.

[45] D. Modha and Y. Fainman. A learning law for density estimation. *IEEE Transactions on Neural Networks*, 5(3):519–523, May 1994.

[46] D. S. Modha and E. Masry. Rate of convergence in density estimation using neural networks. *Neural Computation*, 8:1107–1122, 1996.

[47] K. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[48] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[49] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, UK, 1988.

[50] J. Rissanen. Density estimation by stochastic complecity. *IEEE Transactions on Information Theory*, 38(2):315–323, 1992.

[51] Z. Roth and Y. Baram. Multidimensional density shaping by sigmoids. *IEEE Transactions on Neural Networks*, 7(5):1291–1298, 1996.

[52] H. Schioler and P. Kulczyki. Neural network for estimating conditional distributions. *IEEE Transactions on Neural Networks*, 8(5):1015–1025, September 1997.

[53] I. J. Schoenberg. Splines and histograms. In *Spline functions and approximation theory (Proc. Sympos., Alberta, Edmonton, Alta., 1972)*, volume 21, pages 277–327, Basel, 1973. Internat. Ser. Numer. Math., Birkhäuser.

[54] R. Serfling. *Approximation Theorems of Mathematical Statistics.* Wiley Series in Probability and Mathematical Statistics. Wiley, 1980.

[55] J. Sill. Monotonic networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, 1998.

[56] B. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, UK, 1993.

[57] B. W. Silverman. On the estimation of a probability density function by the maximum penalized likelihood method. *The Annals of Statistics*, 10(3):795–810, 1982.

[58] N. V. Smirnov. An approximation to the distribution laws of random quantiles detremined by empirical data. *Uspehi Mat. Nauk*, 10:179–206, 1944.

[59] P. Smyth and D. Wolpert. Stacked density estimation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, 1998.

[60] C. J. Stone. Optimal rates of convergence for non-parametric estimators. *Annals of Statistics*, 8:1348–1360, 1980.

[61] C. J. Stone. Optimal global rates of convergence for non-parametric regression. *Annals of Statistics*, 10:1040–1053, 1982.

[62] C. J. Stone. Optimal uniform rate of convergence for non-parametric estimators of a density function or its derivatives. In M. H. Rezvi, J. S. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics: Papers in honor of Herman Chernoff on His Sixtieth Birthday*, pages 393–406, New York, 1983. Academic.

[63] C. J. Stone. An asymptotically optimal window selection rule for kernel density estimates. *The Annals of Statistics*, 12(4):1285–1297, 1984.

[64] C. J. Stone. Large sample inference for log-spline models. *Annals of Statistics*, 18(2):717–741, 1990.

[65] J. Struckmeier. Generation of random variates using asymptotic expansions. *COMPUTING*, 59(4):331–347, 1997.

[66] M. Thathachar and M. Arvind. Global boltzmann perceptron network for on-line learning of conditional distributions. *IEEE Transactions on Neural Networks*, 10(5):1090–1098, 1999.

[67] A. N. Tikhonov. On solving ill-posed problems and the method of regularization. *Doklady Akademii Nauk USSR*, 153:501–504, 1963.

[68] A. N. Tikhonov and V. I. Arsenin. *Solutions of Ill-Posed Problems.* Scripta Series in Mathematics. Distributed solely by Halsted Press, Winston; New York, 1977. Translation Editor: Fritz, John.

[69] A. Timmerman, editor. *Special Issue on conditional density forecasting in economics and finance.* Journal of Forecasting, to appear, 2000.

[70] H. Traven. A neural network approach to statistical pattern classification by semiparametric estimation of probability density functions. *IEEE Transactions on Neural Networks*, 2(3):366–377, May 1991.

[71] M. van Hulle. Topographic map formation by maximizing unconditional entropy: a plausible strategy for on-line unsupervised competitive learning and nonparametric density estimation. *IEEE Transactions on Neural Networks*, 7(5):1299–1305, September 1996.

[72] H. van Trees. *Detection, Estimation, and Modulation theory: Part I.* Wiley, New York, 1968.

[73] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer–Verlag, 1995.

[74] V. N. Vapnik. *Statistical Learning Theory.* Adaptive and Learning Systems for Signal Processing, Communications and Control. John Wiley & Sons, Inc., New york, 1998.

[75] G. Wahba. Interpolating spline methods for density estimation I: Equi-spaced knots. *The Annals of Statistics*, 3(1):30–48, 1975.

[76] G. Wahba. Optimal convergence properties of variable knot, kernel, and orthogonal series methods for density estimation. *The Annals of Statistics*, 3(1):15–29, 1975.

[77] G. Wahba. Histosplines with knots which are order statistics. *Journal of the Royal Statistical Society, B*, 38:140–151, 1976.

[78] A. Weigend and A. Srivastava. Predicting conditional probability distributions: a connectionist approach. *International Journal of Neural Systems*, 6(2):109–118, 1995.

[79] P. Williams. Usiing neural networks to model conditional multivariate densities. *Neural Computation*, 8:843–854, 1996.

[80] P. Wilmott, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives.* Cambridge University Press, 1995.

[81] A. Zeevi and R. Meir. Density estimation through convex combinations of densities: approximation and estimation bounds. *Neural Networks*, 10(1):99–110, January 1997.