Simulation and bootstrap on a pRAM architecture

B. Apolloni, A. Brenna , D. de Falco
Dipartimento di Scienze dell'Informazione, Università di Milano
Via Comelico 39-20135, Milano, ITALY
apolloni@dsi.unimi.it

J. G. Taylor
Department of Mathematics, King's College London,
Strand, London WC2R 2LS, ENGLAND
john.g.taylor@kcl.ac.uk

Abstract We discuss the task of interpolating discrete probability densities resulting from the truncation of the binary representation of continuous random variables. We consider both the case in which the initial density is known and a simulator is to be programmed, and the case in which a random sample is given and a bootstrap resampling procedure is to be developed. We point out that the parametrisation of the discrete density suggested by the pRAM (probabilistic Random Access Memory) architecture and the use of Bernstein polynomials for its interpolation constitute an excellent hardware-software combination, able to provide a sagacious smoothing of the histogram of the truncated data.
Keywords: pRAM, Bernstein polynomial, density estimation, bootstrap. *CSCC'99 Proceedings:*-Pages 5361-5366

1. Introduction

Let X be a continuous random variable, taking values in $[0,1]$, $F_X(x) = P(X \le x)$ its cumulative distribution function, and $f_X(x) = dF_X(x)/dx$ its probability density.

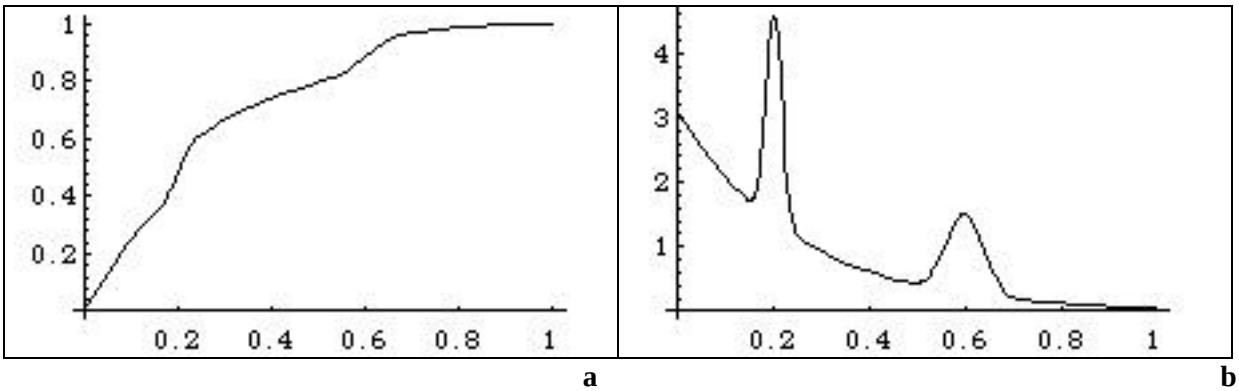
In most of this paper we shall use, as an example, the cumulative distribution function and the probability density plotted in Figure 1.

Let $X = \sum_{j=1}^{\infty} X_j / 2^j = (0.X_1X_2....)_2$ be the binary representation of X. Having fixed the positive integer n, the joint probability law of $(X_1, X_2, \ldots, X_n)$, the first n digits in the binary representation of X , is easily written in terms of F_X as

$$P(X_1 = x_1; X_2 = x_2; \ldots; X_n = x_n)$$
$$= P\left(\sum_{j=1}^{n} \frac{x_j}{2^j} \le X < \sum_{j=1}^{n} \frac{x_j}{2^j} + \frac{1}{2^n} \right)$$
$$= F_X\left(\sum_{j=1}^{n} \frac{x_j}{2^j} + \frac{1}{2^n} \right) - F_X\left(\sum_{j=1}^{n} \frac{x_j}{2^j} \right)$$

We can also write, in terms of the conditional probabilities

$$a(t, (x_1, x_2, \ldots, x_{t-1}))$$
$$= P(X_t = 1 | X_1 = x_1; X_2 = x_2; \ldots; X_{t-1} = x_{t-1}),$$



a **b**

Figure 1 All our numerical examples will refer to the example (a mixture of normal and exponential laws) shown above. a. cumulative distribution function; b. probability density

this probability law as:

$$P\left(X_{(n)}=\sum_{j=1}^{n}\frac{x_j}{2^j}\right)=P(X_1=x_1;X_2=x_2;...;X_n=x_n)$$
$$=P(X_1=x_1)\cdot P(X_2=x_2|X_1=x_1)$$
$$\cdot P(X_3=x_3|X_1=x_1;X_2=x_2)$$
$$\cdot...\cdot P(X_n=x_n|X_1=x_1;X_2=x_2;...;X_{n-1}=x_{n-1})$$
$$=a(1,(\,))^{x_1}(1-a(1,(\,)))\,a(2,(x_1))^{x_2}(1-a(2,(x_1)))^{1-x_2}$$
$$\cdot a(3,(x_1,x_2))^{x_3}(1-a(3,(x_1,x_2)))^{1-x_3}\cdot$$
$$...\cdot a(n,(x_1,...,x_{n-1}))^{x_n}(1-a(n,(x_1,...,x_{n-1})))^{1-x_n}$$

where

$$a(1,(\,))\equiv P(X_1=1)=P(X>1/2)=1-F_X(1/2)$$

$$a(t,(x_1,x_2,...,x_{t-1}))$$
$$=\frac{F_X\left(\sum_{j=1}^{t-1}\frac{x_j}{2^j}+\frac{1}{2^{t-1}}\right)-F_X\left(\sum_{j=1}^{t-1}\frac{x_j}{2^j}+\frac{1}{2^t}\right)}{F_X\left(\sum_{j=1}^{t-1}\frac{x_j}{2^j}+\frac{1}{2^{t-1}}\right)-F_X\left(\sum_{j=1}^{t-1}\frac{x_j}{2^j}\right)}\qquad(°)$$

The main point we wish to make here is the following: for fixed n, and known values of the parameters $a(t,(x_1,x_2,...,x_{t-1}))$, $1\le t\le n$, $(x_1,x_2,...,x_{t-1})\in\{0,1\}^{t-1}$, the simulation of $(X_1,X_2,...,X_n)$ and, therefore of $X_{(n)}$, is easily performed in hardware by a pRAM [1] with n nodes. This machine is a RAM-based hardware that implements a stochastic neural network. Each RAM address contains a real number (between 0 & 1) which is used to emit a one with that probability when the address is accessed. Using this bit to form the address of the next accessed memory, the pRAM output is a string of random bits, where the content of a memory constitutes the conditional probability of a bit given certain others in the string. These probabilities can be modified, either by the operator or through learning algorithms.

This observation opens up two perspectives of application of the pRAM architecture, summarised in Table 1.

The two applications share the following interesting theoretical problems:

- what does "n suitably fixed" mean ?
- how to cope with hardware limitations, that impose severe restrictions on the fan-in of each pRAM node ? Notice that the representation of $P\left(X_{(n)}=\sum_{j=1}^{n}x_j/2^j\right)$ given above requires that node i has fan-in $i-1$.

Here we experiment with the two applications in order, mainly, to gain an insight into the operational meaning of the theoretical problems.

In the next two section we stress the above two points by working a suitable Bernstein polynomial approximation of the template distribution, both in simulation and in bootstrap modality. Then, in section 4, we extend our approximation method to a bidimensional distribution law. There is a brief conclusion at the end of the paper.

2. Simulation

Simulating F_X with a pRAM consists in loading its memory contents a according to (°).

Simulation	Bootstrap [2]
F_X is known;	F_X unknown;
n is suitably fixed;	a random sample of X is observed and digitised to a suitable number n of bits;
the parameters $a(t,(x_1,x_2,...,x_{t-1}))$ are computed from F_X.	the parameters $a(t,(x_1,x_2,...,x_{t-1}))$ are estimated from the random sample.
A pRAM running with the computed $a's$ gives a simulator of $X_{(n)}$.	A pRAM running with the estimated $a's$ gives a generator of bootstrap samples of $X_{(n)}$.

Table 1. Two application frameworks of the pRAM architecture.

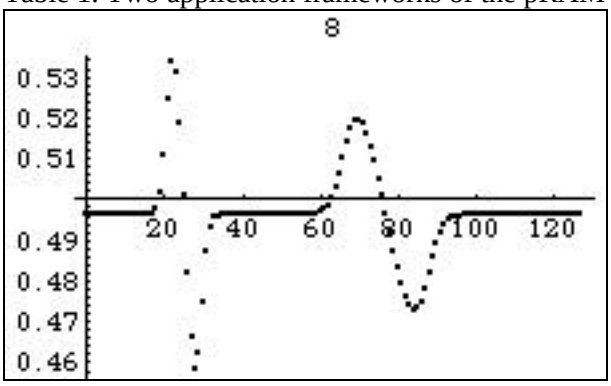


Figure 2. The $2^7 = 128$ conditional probabilities stored in the 8-th pRAM unit.

For instance, values of $a\left(8,(x_1,x_2,...,x_7)\right)$ with the 2^7 possible assignments to $,(x_1,x_2,...,x_7)$ are plotted in Figure 2.

Truncating X to the first n bits corresponds to storing conditional probabilities up to the first n units of our pRAM device, with obvious approximations that, for instance, appear when we rebuild the cumulative distribution function from data simulated by this device, as in Figure 3.

As shown in Fig. 3.b, the simulator gives a discrete random variable with values separated by steps of $1/2^n$ or $1/256$ in our example with 8 nodes.

Notice, however, that from the knowledge of the $a's$ loaded on a pRAM with n nodes, one can reconstruct the values of F_X at all points of the form $k/2^n$, for $k = 0,\ldots,2^n$. Therefore, one has all the coefficients needed to write the Weirstrass-Bernstein [3] polynomial $B[F_X,2^n,x]$ of degree 2^n associated to F_X.

Moreover, denoting by $f_X(x;n)=\frac{d}{dx}B[F_X,n+1,x]$ the n-th degree approximation of f_X coming from the approximation of F_X with $B[F_X,n+1,x]$, we have:

$$f_X(x;n)=\frac{d}{dx}B[F_X,n+1,x]$$
$$=\sum_{k=0}^{n}\frac{(n+1)!}{(k-1)!\cdot(n-k)!}\cdot x^k\cdot(1-x)^{n-k}$$
$$\cdot\left(F_X\left(\frac{k}{n+1}+\frac{1}{n+1}\right)-F_X\left(\frac{k}{n+1}\right)\right)$$

Now, since

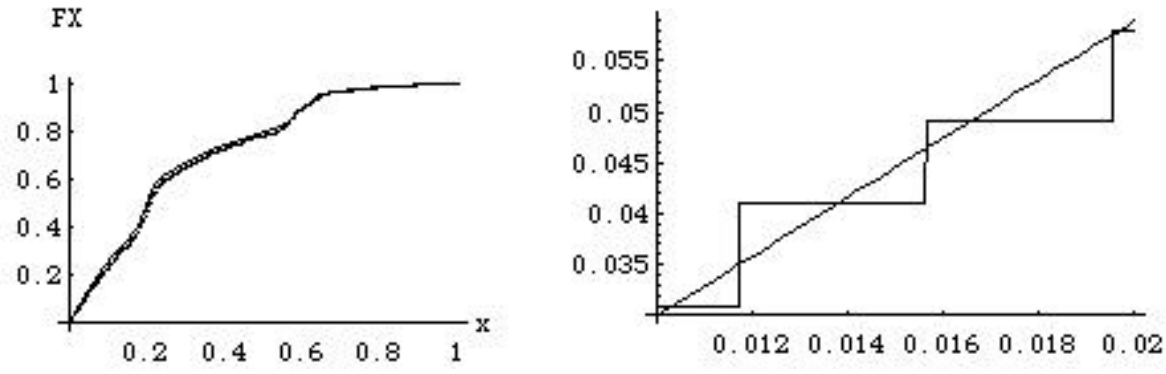
$f_{U[n+1,k+1]}(x)\equiv\frac{(n+1)!}{(k-1)!\cdot(n-k)!}x^k\cdot(1-x)^{n-k}$ is a *Beta* probability density describing the law of the (k+1)-th order statistic of a sample of size n+1 drawn from the uniform population on [0,1], and

$$F_X\left(\frac{k}{n+1}+\frac{1}{n+1}\right)-F_X\left(\frac{k}{n+1}\right)$$
$$=P\left(\frac{k}{n+1}<X\le\frac{k}{n+1}+\frac{1}{n+1}\right)$$

the above density reads

$$f_X\left(x;2^n-1\right)=\frac{d}{dx}B[F_X,2^n,x]$$
$$=\sum_{k=0}^{2^n-1}f_{U[2^n,k+1]}(x)\cdot P\left(X_{(n)}=\frac{k}{2^n}\right)$$

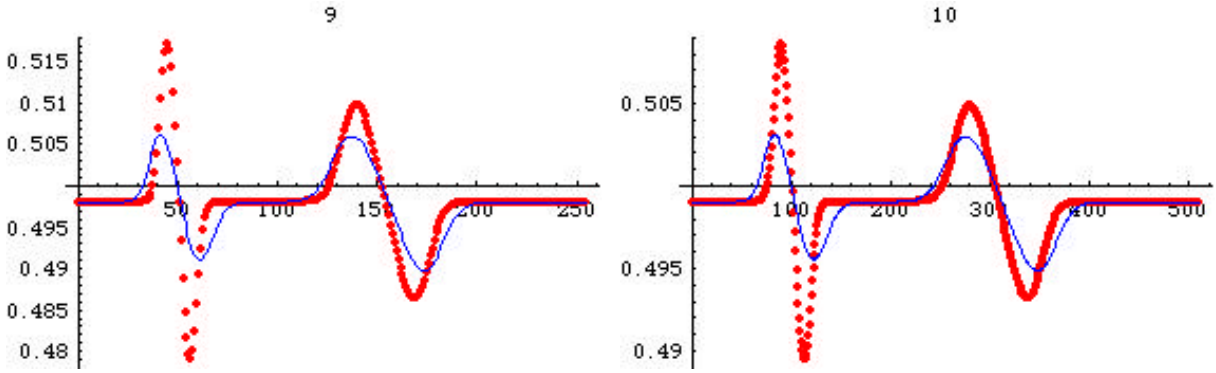
A random variable having this convex combination of probability densities as its probability density is easily simulated.



a **b**

Figure 3. Comparison between the cumulative distribution function in fig. 1 and the empirical distribution function of a sample of size 1000 generated by an 8 unit pRAM.

a. full picture; b. zooming on a piece.



a **b**

Figure 4 Using only information on the contents of nodes 1-8, the Bernstein approximation amounts to making the conjectures shown above on the contents, for instance, of units 9 and 10. dots→ true parameters (not stored in the pRAM before Bernstein smearing); graphically interpolated graphs→Bernstein inferred parameters

Figure 4 shows that constructing, for instance, $B[F_X, 2^8, x]$ from the contents of only the first 8 nodes amounts to making a very educated guess on what the contents of the following nodes should be.

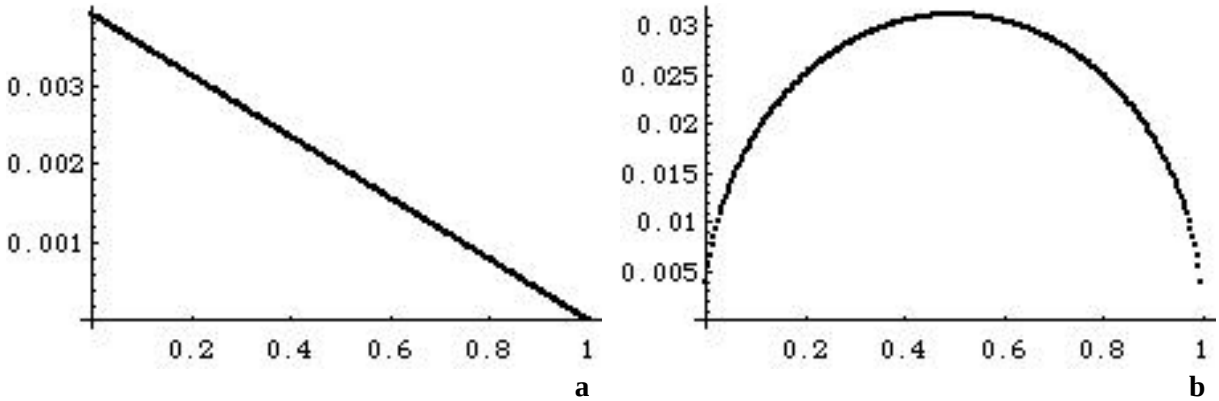
Analysing the Bernstein contribution in filling the gaps in fig 3.b, we realise that over the discrete step necessarily provided by the original pRAM, our smearing algorithm adds a very selected amount of noise, whose mean value and variance change as in fig. 5, depending on $X_{(n)}$. Stated otherwise, starting from the knowledge of f_X in a limited number n of points - possibly the set of n -bit rational numbers in [0,1]- we interpolate these values with a mixture of *Beta* variables [4] arising from the sole assumption that F_X is continuous and the mandatory target that this interpolation converges with increasing n to F_X. This smoothing is not uniform, as could be suggested by the obvious idea of adding equally distributed extra bits, but is decreasing with $X_{(n)}$ with a variance that is a maximum at the centre of the interval.

3. Bootstrap

Here F_X is unknown the task is to determine f, to within a suitable approximation. A random sample of size m, call it $X(1)$, $X(2)$,…., $X(m)$ will be, instead, drawn from the population F_X and the pRAM parameters will be estimated as indicated by the arrow:

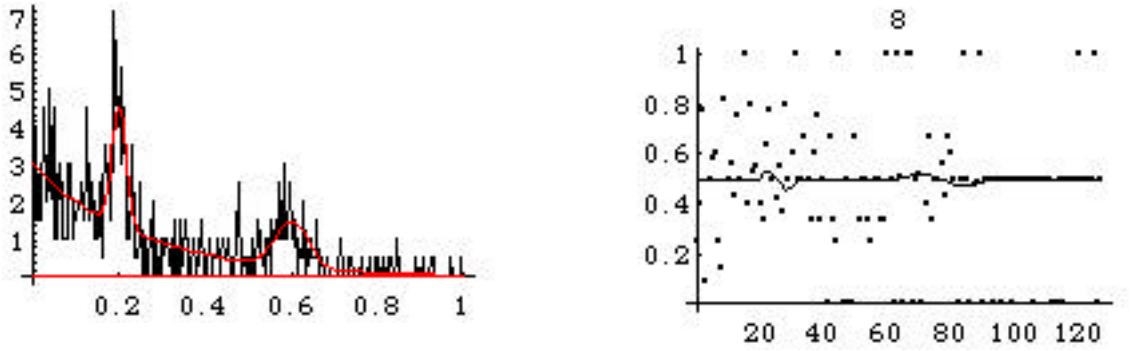
$$a(1,(\,)) \equiv P(X_1 = 1) = P(X > 1/2) = 1 - F_X(1/2)$$

$$\rightarrow A_m(1) = \frac{1}{m}\sum_{i=1}^{m} X_1(i)$$



a **b**

Figure 5 The Bernstein smearing contribution depends on $X_{(n)}$, the raw output of a n node pRAM. a. mean value and b. variance of the Beta noise added to $X_{(n)}$



a **b**

Figure 6. The result of training a pRAM with 8 nodes by parameter estimation based on m = 500 "experimental data".

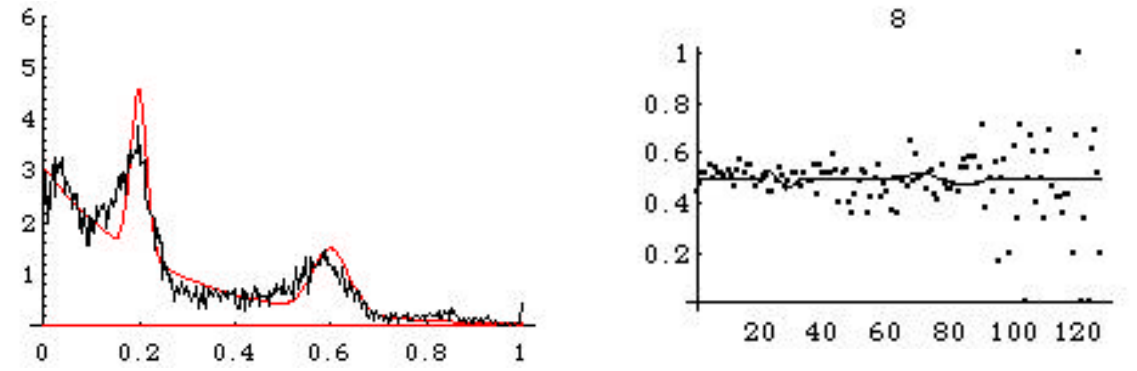
$$a(k,(x)) \to A_m(k,(x))$$
$$= \frac{\sum_{i=1}^{m}\prod_{j=1}^{k-1}\left(X_j(i)\right)^{\left(x_j(i)\right)}\left(1-X_j(i)\right)^{\left(1-x_j(i)\right)}\cdot X_k(i)}{\sum_{i=1}^{m}\prod_{j=1}^{k-1}\left(X_j(i)\right)^{\left(x_j(i)\right)}\left(1-X_j(i)\right)^{\left(1-x_j(i)\right)}}$$

Figure 6, shows the result of the following numerical experiment: an 8-units pRAM has been trained, according to the previous formulas, on a random sample of size m = 500 drawn according to F_X; the probability law of its output has been computed from the estimated parameters and is shown in 6 a.

Three effects make this plot differ from the true (unknown to the pRAM) density function shown in the background: i. the sample points do not load all pRAM parameters (in our actual numerical experiment the pRAM has seen only 161 different 8-bit words, so that $255-161=94$ parameters have remained undefined); ii. the loaded parameters are themselves poor estimates of true conditional probabilities; iii. the pRAM simulates, in fact, a discrete random variable taking only values separated by steps of 1/256.

Figure 7 summarises the results of the following experiment: the network, call it pRAM1, trained as in figure 6, has been asked to simulate the extraction of a sample of size 10000 according to its law of fig. 6.a. These data have been contaminated with Beta noise (coming from Bernstein approximation) and shown to another, initially untrained, 8-node network, call it pRAM2, set in learning mode. pRAM2, without seeing any of the original experimental data, has estimated its parameters on the basis only of the sample issued by pRAM1. Figure 6.b. shows the parameters loaded in node 8 of pRAM1; Figure 7 b. shows the parameters loaded in node 8 of pRAM2; comparison is made, in both cases with the true parameters represented by the graphically interpolated line.

Figures 6 and 7 show the beneficial smoothing effect of "rethinking for a long time (in our case 10000 steps)" the model built from a few (in our case 500) experimental data points, contaminated by a reasonable amount of noise.



a **b**

Figure 7. The result of training a pRAM with 8 nodes by parameter estimation based on m =10000 "bootstrap data" generated by the pRAM of the previous figure, contaminated by the Beta noise of fig. 5.

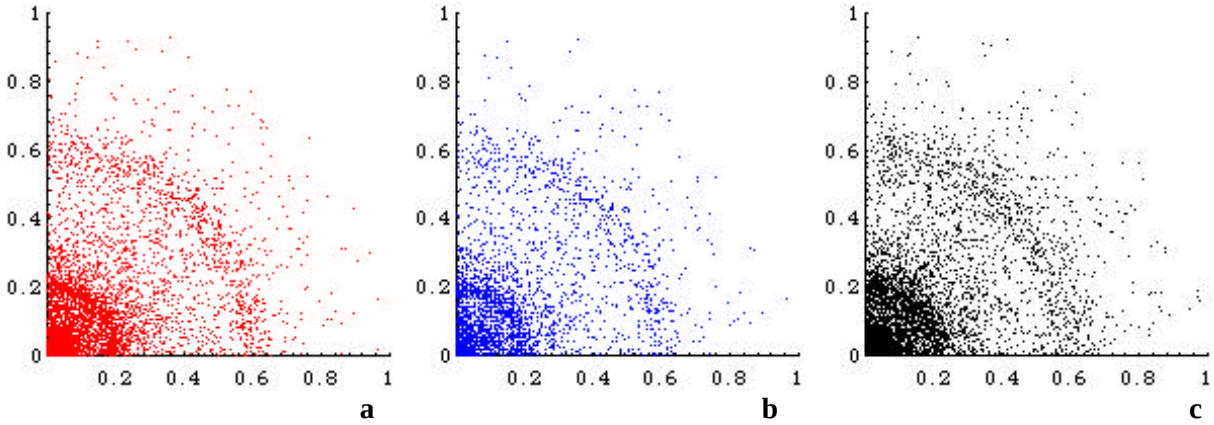


Figure 8.**a**. scatter plot of the "experimental data"; **b**. reconstruction of figure a performed by an 8+8-node pRAM; **c**. the effect of Bernstein smearing on b.

4. From scalar to vectorial variables

A two component random variable (X,Y) can be studied by essentially the same techniques as used above. The probability density corresponding to the Bernstein polynomial approximation (of degree $n+1=2^n$ in each variable) to the joint cumulative function $F_{X,Y}$ can be written as

$$f_{X,Y}(x,y,n)=\sum_{k=0}^{2^n-1}\sum_{h=0}^{2^n-1} f_{U[2^n,k+1]}(x) f_{U[2^n,h+1]}(y) \cdot P\left(X_{(n)}=\frac{k}{2^n}, Y_{(n)}=\frac{h}{2^n}\right) \qquad (°°)$$

A random variable having this density is, in turn, easily simulated as a mixture of pairs of independent order statistics.

Simulation of the weights $P\left(X_{(n)}=\frac{k}{2^n}, Y_{(n)}=\frac{h}{2^n}\right)$ of the mixture is easily performed by a $2\cdot n$ - node pRAM loaded by a sample of $\left(X_{(n)}, Y_{(n)}\right)$ seen in the format $\left(X_1, Y_1, X_2, Y_2 \ldots X_n, Y_n\right)$.

In figure 8. a. we show the scatter plot of a set s of m=5000 "experimental points".

Figure 8.b is the scatter plot of a set s' of 5000 points drawn by an 8+8-node pRAM trained on s. In fact, s' has only 2470 different elements.

Figure 8.c presents the effect of adding bivariate Beta noise to each point of s'.

5. Conclusions

Compressing continuous into discrete variables is a mandatory action in digitally storing and managing signals. Viceversa, rebuilding continuous signals is a routinely job of our mind , for instance in watching TV and in many other, more intelligent, activities. In this paper we present a restoring mechanism that has two features: i. it works efficiently with a limited amount of computational resources and ii. it can be used to complement efficiently the performance of available neuromimetic hardware [5].

References

[1] Clarkson T.G., Gorse D., Taylor J.G., Ng C.K. Learning probabilistic RAM nets using VLSI structures. *IEEE Transactions on Computers. 41*, 1552-1561 (1992).

[2] Efron, B., & Tibshirani, R. (1993). *An introduction to the Bootstrap*. Chapmann and Hall, Freeman, New York.

[3]Lorentz G.C. Bernstein Polynomials. *Universityof Toronto Press* (1953)

[4] Wilks, S.S. (1962). *Mathematical statistics.* John Wiley, N.Y.

[5] pRAM-256 data sheet. *Department of Electronic and Electrical Engineering, King's College, London* (1994).