Neural Networks for Density Estimation in Financial Markets

Malik Magdon-Ismail[1] and Amir Atiya[2]

[1] malik@work.caltech.edu
[2] amir@work.caltech.edu
Learning Systems Group, California Institute of Technology
136-93 Caltech, Pasadena, CA, USA, 91125

Abstract. We introduce two new techniques for density estimation. Our approach poses the problem as a supervised learning task which can be performed using Neural Networks. We introduce a stochastic method for learning the cumulative distribution and an analogous deterministic technique. We use these techniques to estimate the densities of log stock price changes, demonstrating that the density is fat-tailed contrary to the Black-Scholes model which assumes it to be Gaussian.

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. 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 [6]).

According to the Black-Scholes model for option pricing [1], the log stock price changes are assumed Gaussian. What if the true distribution is not Gaussian? The option prices can be computed (numerically) using the true distribution with a possibility of detecting mispricing. This could have large financial implications.

Traditional density estimation methods can be grouped into two broad categories (see [6]). 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 non parametric approach where the density is determined according to a formula involving the data points available.

The most common non parametric methods are the kernel density estimator, also known as the Parzen window estimator [4] and the k-nearest neighbor technique [2]. The basic problem with these methods is their sensitivity to the choice of smoothing parameter. A wrong choice can lead to either under smoothing or over smoothing.

In spite of the importance of the density estimation problem, proposed methods using neural networks have been very sporadic in the literature. We propose two new methods for density estimation which can be implemented using multilayer networks. Multilayer networks in addition to being able to implement any distribution function to an arbitrary precision, 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.

We will develop two methods for the estimation of densities by first learning the cumulative and then taking the derivative of the resulting function. One of the techniques is based on a stochastic algorithm (SLC), and the second is based on a deterministic technique based on learning the cumulative (SIC).The stochastic technique will generally be smoother on smaller numbers of data points, however, the deterministic technique is faster and applies to more that one dimension.

1 New Density Estimation Techniques

To illustrate our methods, we will use neural networks, but stress that any sufficiently general learning model 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. We will now proceed to a description of the two methods.

1.1 SLC (Stochastic Learning of the Cumulative)

Let $x_n \in \mathbf{R},\ n = 1, ..., N$ be the data points. Let the underlying density be $g(x)$ and its distribution function $G(x) = \int_{-\infty}^{x} g(t)dt$. 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 $H(x, w) = G(x)$. Learning requires a set of targets. To determine a target for the network training, we make the following observation. It can easily be shown that the density of the random variable $y \equiv G(x)$ (x being generated according to $g(x)$) is uniform in $[0, 1]$. Thus, if $H(x, w)$ is to be as close as possible to $G(x)$, then the network output should have a density that is close 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 $G(x)$.

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. In our simulations, we used a hint penalty to enforce monotonicity [5]. The algorithm is as follows.

1. Let $x_1, x_2, ..., x_N$ be the data points. Without loss of generality, assume the points are sorted so that $x_1 \leq x_2 \leq \cdots \leq x_N$. Set $t = 1$, where t is the training cycle number. Initialize the weights to $w(1)$. This is usually done randomly.

2. Generate randomly from a uniform distribution in $[0, 1]$ N points $u'_1, u'_2, ..., u'_N$. These are the network targets for this cycle. Sort the targets in ascending order, i.e. $u_1 \leq u_2 \leq ... \leq u_N$ (where we have renamed the ordered targets u_n). Then, the point u_n is the target output for x_n.

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

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

where $\mathcal{E}$ is the objective function that includes the error term and the monotonicity hint penalty term [5]:

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

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.

4. Set $t = t + 1$, and go to step 2 to perform another cycle until the error is small enough. Upon convergence, the density estimate becomes

$$\hat{g}(x) = \frac{\partial H(x, w)}{\partial x} \tag{3}$$

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 will generally improve 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.

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.

1.2 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

$$G(\mathbf{x}) = \int_{-\infty}^{x^1} \cdots \int_{-\infty}^{x^d} g(\mathbf{x}) dx^1 \cdots x^d, \tag{4}$$

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

$$\hat{G}(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \Theta(\mathbf{x} - \mathbf{x}_n), \tag{5}$$

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 method we propose uses such an estimate for the target outputs of the neural network. The estimate given by (5) 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 (5) 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 (5). Another option is to use the data points themselves as examples. The target

for a point $\mathbf{x}_m$ would then be

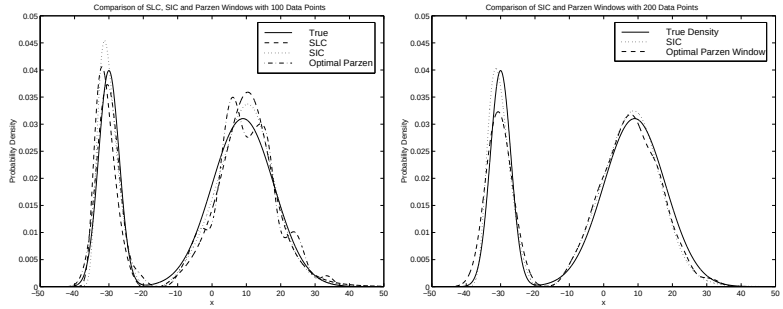
$$\hat{G}(x_m) = \frac{1}{N-1} \sum_{n=1,\ n \neq m}^{N} \Theta(x_m - x_n). \tag{6}$$

We also 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

$$\hat{g}(\mathbf{x}) = \frac{\partial^d H(\mathbf{x}, w)}{\partial x^1 \cdots \partial x^d}. \tag{7}$$

We note that for a few dimensions, a derivation of the derivatives in (7) will be straightforward. For larger dimensions a numerical differentiation scheme such as the simple differencing method is more feasible.

2 Simulation Results



(a) **(b)**

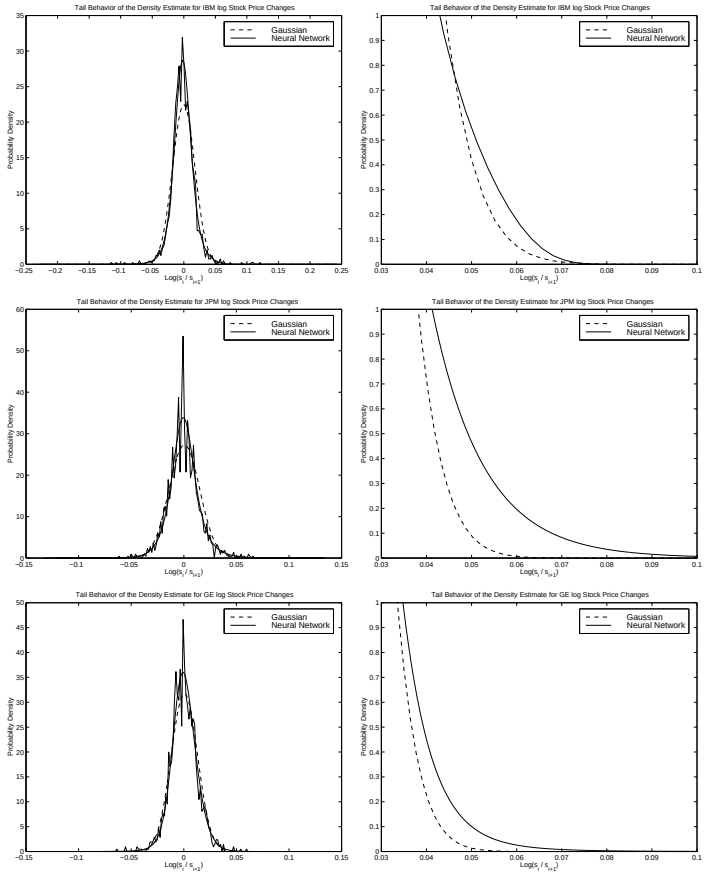
Fig. 1. Comparison of optimal Parzen windows, with neural network estimators. Plotted are the true density and the estimates (SLC, SIC, Parzen). In (a), 100 data points were used, and in (b) 200 data points were used. The optimal Parzen window width (h) can be calculated for the true density as $h \approx 5.12/n^{1/5}$ [6, pg 40]. Though this is not possible in practice, we use it here for comparison. Notice that even the optimal Parzen window is bumpy, whereas the neural networks, which are also capable of implementing any density (by using arbitrarily many hidden units) is smooth.

We tested our techniques for density estimation on data drawn from a mixture of two Gaussians:

$$g(x) = \frac{3}{10} \frac{1}{\sqrt{18\pi}} e^{-\frac{(x+30)^2}{18}} + \frac{7}{10} \frac{1}{\sqrt{162\pi}} e^{-\frac{(x-9)^2}{162}} \tag{8}$$

Data points were randomly generated and the density estimates using SLC or SIC (for 100 and 200 data points) were compared to the Parzen technique.

Learning was performed with a standard 1 hidden layer neural network with 3 hidden units. The hidden unit activation function used was $tanh$ and the output unit was an erf function[3]. A set of typical density estimates are shown in figure 1. A monotonic mapping was obtained by enforcing the monotonicity hint as a penalty with a weight 10000, and the learning algorithm used was conjugate gradient.



As an application we estimated the density of stock price changes for a group of companies. Refer to [7] for a more detailed discussion of modeling stock price change using stochastic differential equations. Essentially, the ratio $\log(S_{i+1}/S_i)$ should have a Gaussian distribution according to the Black-Scholes formulation. Using our techniques we estimate the densities for this ratio, illustrated in the figures above. We plot the estimated density along with the true histograms and the Gaussian with the same mean and variance for three representative companies (IBM, GE, JP Morgan). The tails are magnified on the right. Notice

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

how the true distribution significantly deviates from the Gaussian, displaying the well established fat-tailed behavior.

2.1 Convergence to the True Density

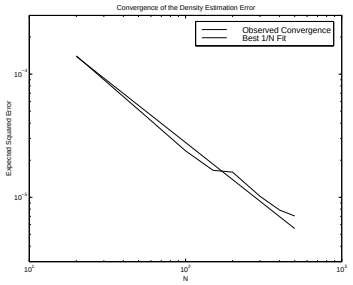


Fig. 2. Convergence of the density estimation error for SICI. A five hidden unit two layer neural network was trained according to SIC. The resulting density estimation error was computed for over 100 runs for various N. Plotted are the results on a Log-Log scale. For comparison, also shown is the best $1/N$ fit.

It can be shown that SLC and SIC are essentially equivalent [3]. We demonstrate the convergence of SIC to the true density with a simulation on the same test density used in section 2. Shown in figure 2 is the behavior of $\log(E)$ versus $\log(N)$, and for comparison, we show a slope -1 curve. The optimal linear fit had slope -0.97. This indicates that the convergence rate is about $1/N$. The detailed analysis of convergence can be found in [3] and is at least as fast as $\log\log(N)/N$.

3 Comments

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 so doing, we placed the density estimation problem within the supervised learning framework. 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). Both methods learn the distribution function by mapping the data to a uniform distribution. 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 1 (a). A similar outcome is obtained by adding small random perturbations to the targets of SIC.

Extensions along these lines are possible: For example, one can replace the uniform targets 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, $Q(H(w,x))$ is the required estimate $G(x)$, from which the density can be obtained by differentiation with respect to x. Thus our methods could be extended by mapping to any standard density for which $Q(x)$ is known analytically. 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 should be easier to learn as it will be close to the identity mapping.

Our simulations demonstrated that the our techniques performed well in comparison to the Parzen technique using the optimal kernel width, which is unrealizable in practice. 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. 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. Using a simulation, we also demonstrated that our estimator converges to the true density at a rate close to $1/N$.

4 Acknowledgments

We would like to acknowledge Yaser Abu-Mostafa and the Caltech Learning Systems Group for their useful input.

References

1. F. Black and M. S. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 3:637–654, 1973.
2. K. Fukunaga and L. D. Hostetler. Optimization of k-nearest neighbor density estimates. *IEEE Transactions on Information Theory*, 19(3):320–326, 1973.
3. M. Magdon-Ismail and A. Atiya. Consistent density estimation from sample distribution functions. *Manuscript in preparation for submission to IEEE Transactions on Information Theory*, 1998.
4. E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.
5. J. Sill and Y. S. Abu-Mostafa. Monotonicity hints. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 9, pages 634–640. Morgan Kaufmann, 1997.
6. B. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, UK, 1993.
7. P. Wilmott, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives.* Cambridge University Press, 1995.