Multivariate Wavelet Density Estimation for Streaming Data: A Parallel Programming Approach

Kyle A. Caudle[*] Christer Karlsson[†] Larry D. Pyeatt[‡]

Abstract
Data streams provide unique challenges that are not normally encountered during standard statistical analysis. Foremost is the fact that data is arriving at such a high rate that storing the data and analyzing later is no longer feasible. Methods must be in place to make sense of data on a near real-time basis. Typical methods involve streaming queries and model building. Density estimation is an essential tool used to make sense of data collected by large scale systems. Due to the curse of dimensionality, density estimation in higher dimensions becomes problematic. In this paper, we present a recursive method for constructing and updating an estimate of the non-stationary probability density function in a high dimensional input space using parallel programming. We demonstrate the effectiveness of the approach via the use of simulated data as well as data from Internet header packets.

Key Words: Data Streams, Density Estimation, Wavelets, Parallel Programming

1. Introduction

Streaming data is a challenging area of computational statistics and has been the focus of many recent papers. If data quantity becomes so large that storage becomes an issue, processing will need to occur in real time. Modern databases are continually updating, receiving and processing information. As a result, standard statistical techniques must have mechanisms in place that update queries and models as new data is received. In most data stream settings data order is relevant. By taking order into account we can extract and analyze information as it is received and often data storage is not required. In the age of “Big Data”, analyzing data in real time is becoming more and more prevalent. Applications of data streaming are seen in web-based analytics, finance, network security and engineering applications that involved remote sensing [1, 7].

Due to the curse of dimensionality, density estimation in multiple dimensions becomes problematic [11]. The theory behind estimating a density by kernels has been well researched, see for example [2, 8]. In practice, computing a multivariate density by kernels is computationally infeasible in a sequential manner when the input dimension is large (i.e. > 3). In this paper our approach is to use multivariate kernel density estimation by orthogonal series and use multiple threads to split up the processing. Orthogonal series density estimation was suggested by Čencov [9]. Under moderate regularity conditions, a density function f can be written as a series of orthogonal basis functions

$$f(x) = \sum_{j \in Z} b_j \psi_j(x), \tag{1}$$

where $b_j = < f, \psi_j > = \int \psi_j(x) f(x)\, dx = E[\psi_j(X)]$. This amazingly simple idea allows us to approximate the series coefficients by simple sample means. Let $X_1, X_2, \ldots, X_n$ be a random sample from the unknown density $f(x)$. Because $b_j = E[\psi_j(X)]$, the jth coefficient can be approximated by

$$\hat{b}_j = \frac{1}{n} \sum_{i=1}^{n} \psi_j(X_i). \tag{2}$$

[*]South Dakota School of Mines and Technology, Rapid City, SD 57701, USA
[†]South Dakota School of Mines and Technology, Rapid City, SD 57701, USA
[‡]South Dakota School of Mines and Technology, Rapid City, SD 57701, USA

Although many different orthogonal series bases could have been used, we use wavelets to construct our density estimate. Wavelets are an excellent choice for streaming density estimation because the coefficients decrease very rapidly implying that we obtain an excellent function reconstruction using only a few coefficients. In addition, wavelets are excellent for identifying transients and discontinuities. This makes them particularly well suited to the streaming data application, where abrupt changes in the data stream may be expected.

Walter [13], Vannucci [12] and Neumann [10] have written on the theory of multivariate wavelet density estimation. We do not repeat the theoretical development here but mention that multivariate wavelet density estimation involves the products of univariate wavelets. One can form a d-dimensional wavelet basis by using $\phi(\mathbf{x}) = \phi(x_1)\phi(x_2)\ldots\phi(x_d)$ where $\phi(\cdot)$ is uni-dimensional wavelet function and $2^d - 1$ wavelet functions having the form

$$\psi_l(\mathbf{x}) = \prod_{i=1}^{d} \gamma(x_i) \tag{3}$$

with $\gamma(\cdot) = \phi(\cdot)$ or $\psi(\cdot)$ but not all $\gamma(\cdot) = \phi(\cdot)$
As an example, in the bivariate case there would be $2^2 - 1 = 3$ wavelet functions

$$\psi_1(\mathbf{x}) = \phi(x_1)\psi(x_2), \tag{4}$$

$$\psi_2(\mathbf{x}) = \psi(x_1)\phi(x_2), \tag{5}$$

and

$$\psi_3(\mathbf{x}) = \psi(x_1)\psi(x_2) \tag{6}$$

The multivariate density function can be represented as an orthogonal series of the scaling function $\phi(\mathbf{x})$ at the lowest level of detail and wavelet functions $\psi_l(\mathbf{x})$,

$$\Phi(x_1, x_2, \ldots, x_d) = \sum_{k_1} \cdots \sum_{k_d} c_{jo,k_1\ ,\ldots,k_d} 2^{jo\ d/2} \phi\left(2^{jo} x_1 - k_1\right) \ldots \phi\left(2^{jo} x_d - k_d\right) \tag{7}$$

$$\psi_l(x_1, x_2, \ldots, x_d) = \sum_{k_1} \cdots \sum_{k_d} \sum_{j} d_{j,k_1,..,k_d} 2^{jd/2} \gamma\left(2^j x_1 - k_1\right) \ldots \gamma(2^j x_d - k_d) \tag{8}$$

$$f(x_1, x_2, \ldots, x_d) = \Phi(x_1, x_2, \ldots, x_d) + \sum_{l=1}^{2^d - 1} \psi_l(x_1, x_2, \ldots, x_d) \tag{9}$$

The coefficients c_* and d_* are estimated by sample averages in a manner analogous to Equation 2.

$$\hat{c}_{jo,k_1\ ,\ldots,k_d} = \frac{1}{n} \sum_{i=1}^{n} \phi_{jo,k_1,..,k_d}(X_{i1}, X_{i2}, \ldots, X_{id}) \tag{10}$$

$$\hat{d}_{j,k_1\ ,\ldots,k_d} = \frac{1}{n} \sum_{i=1}^{n} \psi_{j,k_1,..,k_d}(X_{i1}, X_{i2}, \ldots, X_{id}) \tag{11}$$

2. Multivariate Density Estimation for Streaming Data

Consider a data stream of n independent and identically distributed d-tuples of random variables of the form $\mathbf{X}_i = (X_{1i}, X_{2i}, \ldots, X_{di})$ with $i = 1, 2, \ldots, n$. Without loss of generality further assume that data element $\mathbf{X}_k$ arrives prior to element $\mathbf{X}_{k+1}$. Let $\hat{f}_2(x_1, x_2, \ldots, x_d)$ be the multivariate density estimate using the first two d-tuples and let $\hat{f}_3(x_1, x_2, \ldots, x_d)$ be the multivariate density estimation using the first three d-tuples. Continuing in this manner the goal is to obtain a sequence of estimates $\{\hat{f}_i(x_1, x_2, \ldots, x_d)\}$ that converges to $f(x_1, x_2, \ldots, x_d)$. As new data arrives, the density is updated. Wegman and Marchette [14] suggested a recursive method for updating kernel density estimates. Caudle and Wegman [3] adapted this technique to wavelet density estimators. This technique updates the wavelet coefficients as each new data element arrives. The same technique can be applied to the multivariate wavelet density estimation as well. It is easy to verify the following updating equation for the wavelet and scaling function coefficients,

$$\hat{b}_{j,n+1} = \frac{n}{n+1}\hat{b}_{j,n} + \frac{\gamma_j(\mathbf{X}_{n+1})}{n+1} \tag{12}$$

where $\gamma(\cdot) = \phi(\cdot)$ or $\psi(\cdot)$.

Once the coefficients are updated, we can produce an updated estimate of the multivariate density. As the number of points in the data stream increases the estimated coefficients asymptotically approach the true coefficients. Also, since the coefficients (c_* and d_*) are the sum of independent and identically distributed terms $\gamma(\cdot)$, strong consistency and asymptotic normality also hold. With streaming data however, traditional limit theorems are of limited value because streaming data is generally non-stationary. For short periods of time however, the data are quasi-stationary. And, in terms of estimating a density that accurately reflects the current situation we need to discard older data that is no longer relevant. Hence, there is a trade-off between statistical accuracy, and estimating a density that accurately reflects current reality.

When using simulated density where the underlying density is known, the error in a density estimate is typically estimated by either the mean squared error (MSE) or mean integrated squared error (MISE). In higher dimensions however, the MISE does not scale well in an intuitive fashion, and according to Devroye and László [4] the mean integrated absolute error (MIAE) has some statistical advantages over the MISE. Because of this, and a better intuitive feel as we estimate densities in higher dimensions, we choose the MAE as a measure of accuracy of our density.

$$\text{MAE} \approx \frac{1}{n}\sum_{i=1}^{n}\left|\hat{f}(\mathbf{X}_i) - f(\mathbf{X}_i)\right|, \tag{13}$$

The variance of the density estimate can be reduced by setting those wavelet coefficients that fall below a threshold value to zero. This process is called wavelet shrinkage. Reduction in variance however, will create an estimate that is slightly more biased. Donoho et al. [5] have shown that the reduction in variance is well worth the slight increase in bias.

Threshold values can be obtained by experimentation or they can also be chosen automatically. There are several different approaches for automatically setting the threshold value, see for example [5, 6] In our demonstrations (Section 4) we automatically threshold wavelet coefficients by using the universal threshold value of

$$t = \hat{\sigma}\sqrt{2\log(n)}\ , \tag{14}$$

where the standard deviation (σ) is estimated by the standard deviation of all computed wavelet coefficients and n is the sample size. In addition to achieving variance reduction in our estimate, we have also found that this results in a significant computational savings as a large majority of the wavelet coefficients are now zero.

Algorithm 1 MDE Main Algorithm

```
1: Read the input file, or prepare a set of simulated test points.
2: Create a set of $d$ counters, to be shared by the worker threads.
3: Spawn one worker thread for each processor core to calculate the $\phi(\cdot)$ coefficients
4: Wait for the worker threads to terminate
5: Spawn one worker thread for each core to calculate the $\psi(\cdot)$ coefficients
6: Wait for the worker threads to terminate
7: Threshold the coefficients as necessary
8: Calculate the density value at each sample point in the domain of the density function
9: For simulated data, test the density heights against the density heights of the known distribution.
```

Algorithm 2 MDE Worker Thread Algorithm

```
1: while Data remaining do
2:     Lock the shared counters.
3:     Copy the counters into a local set $L$.
4:     Increment the shared counters by $\Delta$.
5:     Unlock the shared counters.
6:     for $i \leftarrow L, L+\Delta$ do
7:         Calculate coefficient $i$
8: exit thread
```

3. Parallel Approach for Multivariate Density Estimation

The software used to compute our density estimates in Section 4, divides the work between the multiple threads of a parallel computer. One benefit of our algorithm is that neither the input dimension nor the number of threads needs to be specified. In other words, the algorithm determines both of these values at run time. Algorithm 1 outlines the main parallel algorithm for density estimation. Each of the worker threads performs the steps outlined in Algorithm 2. The program was written in C using the pthreads library to create and manage the worker threads The program achieved $> 99\%$ CPU utilization on all of the computers that we tested it on. The code was tested on four computers, with 4, 8, 32, and 64 cores, respectively. The speedup achieved was almost very near linear with the number of cores. Linear speedup could not be achieved due to the overhead involved in setting up the data, managing the threads, performing thresholding, and calculating and producing the final results. If these non-parallel tasks are taken into account, then we achieved above 99% of the theoretical maximum possible speedup.

4. Simulation Results

The simulations outlined in this section were run on a 16 core Intel Xeon E5-2687W running at 3.10GHz with 32GB of RAM. Each core is capable of supporting two simultaneous threads (Hyperthreading). The first simulation performed was estimating a standard Gaussian distribution with a randomized mean vector where the mean for each dimension is chosen from a uniform distribution $U(-5,5)$, and a covariance matrix Σ equal to the d x d identity matrix. As the sample size increases, the estimated wavelet coefficients approach the true coefficients and therefore the MAE decreases.

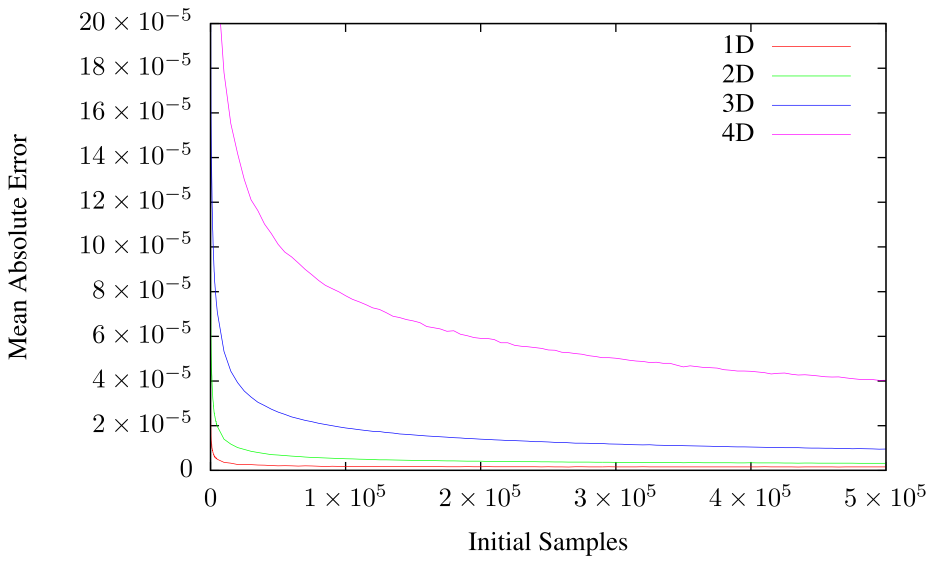

Figure 1: Sample Size Determination

Table 1: Computation Time

Dimension	MAE	Volume	CPU sec	Clock Time (sec)
1	$< 1 \times 10^{-5}$	9.990×10^{-1}	2.3×10^{-1}	7.3×10^{-3}
2	$< 1 \times 10^{-5}$	9.989×10^{-1}	5.00×10^{0}	1.563×10^{-1}
3	$< 1 \times 10^{-5}$	9.990×10^{-1}	1.725×10^{2}	5.392×10^{0}
4	5×10^{-5}	1.082×10^{0}	4.646×10^{3}	1.452×10^{2}

Table 2: Distribution Comparisons

1

			MAE		
Dimension	Sample Size (n)	CPU Time (s)	Standard	Mixture	Laplace
1	100	$< 1.000 \times 10^{-6}$	3.398×10^{-3}	4.162×10^{-3}	9.424×10^{-3}
2	500	1.000×10^{-2}	5.563×10^{-5}	8.902×10^{-5}	5.872×10^{-5}
3	1000	5.900×10^{-1}	9.523×10^{-6}	1.679×10^{-5}	1.342×10^{-5}
4	2500	4.569×10^{1}	1.262×10^{-5}	1.578×10^{-5}	1.129×10^{-5}
5	5000	2.596×10^{3}	5.190×10^{-5}	4.309×10^{-5}	4.445×10^{-5}

4.1 Determining Reasonable Sample Size

Figure 1 shows the relationship between Mean Absolute Error and the number of samples used to create the approximation for problems with one to four dimensions. Each plot in the graph was created by running the algorithm 32 times for each of 100 sample sizes, and averaging the results. The minimum sample size was 5000 samples. On each iteration, the sample size was increased by 5000 points, and the algorithm was run 32 times. The results of those 32 runs were averaged to create one point on the graph in Figure 1.

As can be seen from the graph, each curve shows that at some point increasing the sample size has limited value. The number of points needed to achieve a reasonably close approximation increases with the number of dimensions. For example, in a single dimension, 1000 points may be adequate, while for three dimensions, 100,000 samples provides a reasonably good approximation. Reasonably good approximations for two and four dimensions can be obtained using 10,000 and 200,000 samples, respectively. These numbers are to be used as a general rule-of-thumb and further investigation may provide better guidelines.

Table 1 provides an indication of the computation time required to generate one run with 500,000 samples for each of the estimates in Figure 1. The algorithm makes very efficient use of the available processing resources, but the total amount of computation increases exponentially with the dimensionality of the data. Still, the table indicates that, depending on the application domain, the algorithm may be able to process data in real-time on problems with significantly more than 4 dimensions.

4.2 Estimating More Complex Distributions

In the previous section, we estimated the standard Gaussian distribution, a relatively simple density to estimate. In this section, we compare the estimation errors for the standard Gaussian distribution estimate with a multivariate Gaussian mixture and a multivariate Laplace distribution. We chose the mixture distribution because it has multiple modes and we chose the Laplace distribution because it has a sharp peak in the center.

In table 2, we show the mean absolute error (MAE) for each of these density estimates for various input dimensions. From table 2, it is not readily apparent which density was easiest to estimate. All estimates were quite good, any differences had more to do with the randomness of our random deviates than with the ability of the algorithm to produce a good estimate. We also note that the computation time in table 2 was for estimation of the standard Gaussian distribution, the computation time for the other distributions was similar. Thus, it appears from this simulation study that the distribution we are estimating is not a major factor in either accuracy or speed.

For visualization purposes, we provide estimates of each of these three distributions with 2-dimensional

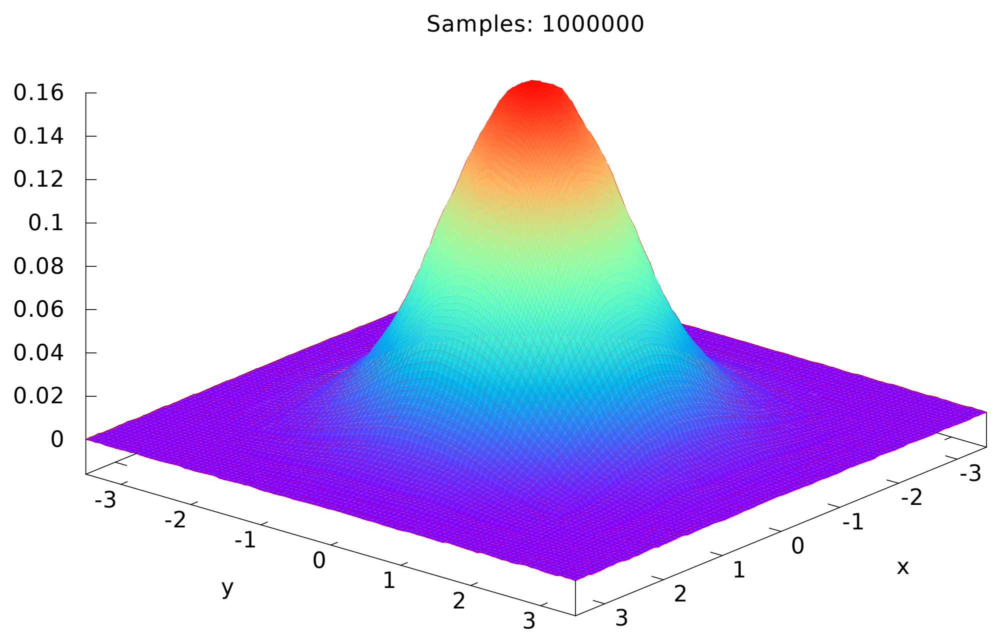

Figure 2: Bivariate Gaussian Distribution

input data. Figure 2 shows an example of a simple multivariate Gaussian distribution with a covariance matrix Σ equal to the d x d identity matrix. This is representative of the distributions that were used for the experiments in Section 4.1.

Figure 3 shows an example of a bivariate Gaussian mixture distribution. The distribution shown in Figure 3 uses data from the distribution

$$f = 0.3N(\mu_1, \Sigma_1) + 0.7N(\mu_2, \Sigma_2) \tag{15}$$

where,

$$\mu_1 = [-1, 1]$$
$$\Sigma_1 = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.4 \end{bmatrix}$$

and

$$\mu_2 = [1.5, -1.5]$$
$$\Sigma_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

Finally, in Figure 4 we see a bivariate Laplace distribution with mean vector $\mu = [0,0]$ and a rate parameter of $\lambda = 1$.

4.3 Real Data

Our final demonstration shows one potential application using data taken from header packets. Our sample data displayed in table 3, consists of eight different parameters. The data in table 3 came from a "class-B"

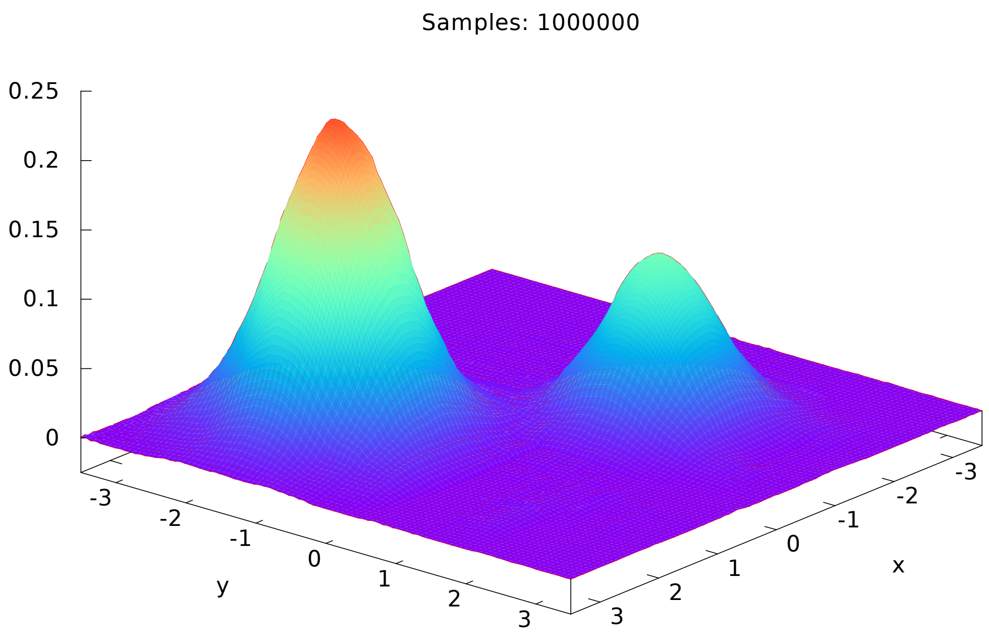

Figure 3: Bivariate Gaussian Mixture Distribution

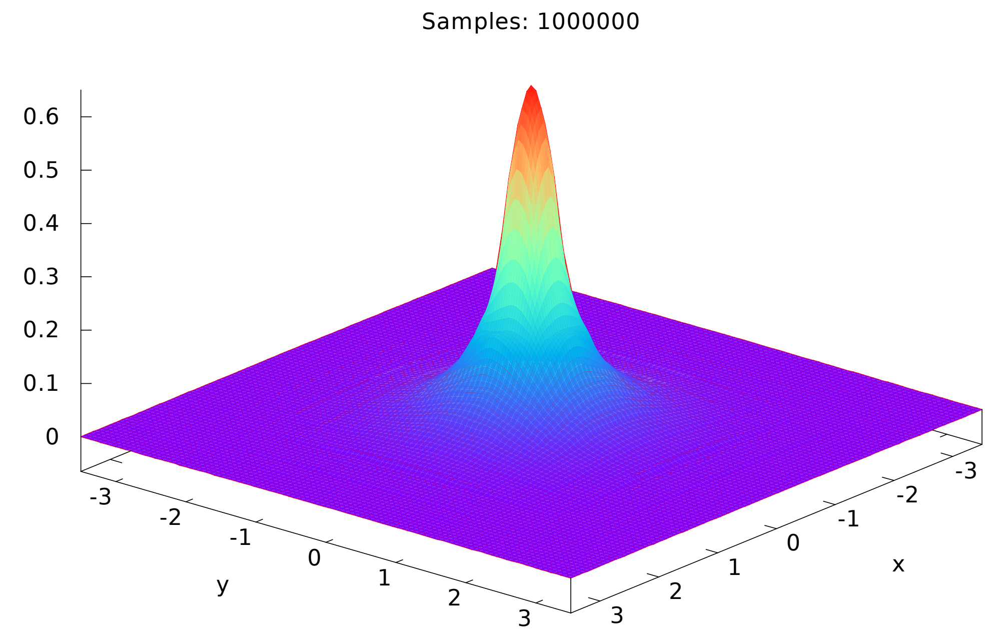

Figure 4: Bivariate Laplace Distribution

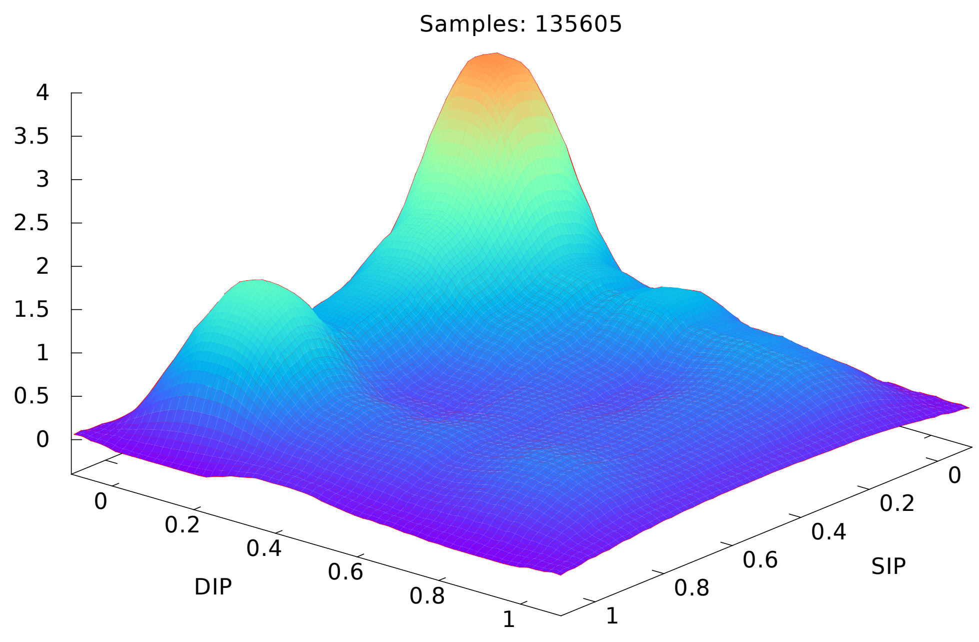

Figure 5: SIP vs. DIP

Table 3: Packet Data

time	duration	SIP	DIP	DPort	SPort	Npacket	Nbyte
39603.64	0.23	4367	54985	443	1631	9	3211
39603.64	0.27	18146	9675	3921	25	15	49
39603.65	0.04	18208	28256	1255	80	6	373
39603.65	1389.10	24159	17171	23	1288	845	5906
39603.65	373.99	60315	37727	2073	80	1759	834778

network so the first two octets are the same, thus allowing for 216 = 65; 536 possible machines (SIP's) and 232 destinations (DIP's). Only the first two octets in the DIP are retained since one is typically only concerned with the destination network and not a specific machine.

The plot in Figure 5 is a bivariate plot of Source IP vs. Destination IP. The rational for this plot is that this might give us an indication of which ports communicate with each other on a routine basis. If this density plot changes over time, this could indicate that an attack on the network is taking place.

5. Conclusions

This paper clearly demonstrates that multivariate density estimation is possible with reasonable speed up to 5 input dimensions. Density estimation applications abound and we made the case for one such application in the area of computer intrusion detection. We have experimented with higher dimensions (i.e. up to 8), and although the creation time becomes substantial, we point out that the update mechanism is fast, so change detection of a much higher dimensional density estimate is possible if one can afford the startup cost.

References

[1] Brian Babcock. Models and issues in data stream systems. In *Proceedings of the twenty first ACM SIGMOD-SIGART symposium on Principles of database systems*, 2002.

[2] Theophilos Cacoullos. Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics*, 18.1:179–189, 1966.

[3] Kyle A. Caudle and Edward Wegman. Nonparametric density estimaiton of streaming data using orthogonal series. *Computational Statistics & Data Analysis*, 53(12):3980–3986, 2009.

[4] Luc Devroye and László Gyórfi. *Nonparametric density estimation the L_1 view*. John Wiley & Sons, New York, 1985.

[5] David L. Donoho and Iain Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81.3:425–455, 1994.

[6] David L. Donoho and Iain Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90.432:1200–1224, 1995.

[7] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[8] Don O. Loftsgaarden and Charles P. Quesenberry. A nonparametric estimate of a multivariate density function. *Annals of Mathematical Statistics*, pages 1049–1051, 1965.

[9] N. N. Čencov. Evaluation of an unknown density from observations. *Soviet Mathematics*, 3:1559–1562, 1966.

[10] M. Neumann. Multivariate wavelet thresholding: a remedy against the curse of dimensionality. Technical report, Humboldt University, 1996.

[11] David W. Scott. *Multivariate density estimation: theory practice and visualization*. John Wiley & Sons, New York, 2009.

[12] Marina Vannucci. *On the application of wavelets in statistics*. PhD thesis, Dipartimento di Statistica G. Parenti, University of Florence, Italy (in Italian), 1996.

[13] G. G. Walter. Estimation with wavelets and the curse of dimensionality. Technical report, Department of Mathematical Sciences, University of Wisconsin-Milwaukee, 1995.

[14] Edward J. Wegman and David J. Marchette. On some techniques for streaming data: A case study of internet packet headers. *Journal of Computational and Graphical Statistics*, 12.4:893–914, 2003.