VI. Conclusion

This brief paper presents a novel localization algorithm, named DANN. We build a WLAN-based location fingerprinting system based on DANN architecture. Our approach incrementally inserts DCs, and recursively updates the weightings in the network until no further improvement is required. In this way, the network adapts the weights depending on the discriminative information from the inserted DCs. In other words, the redundant information is discarded because it is regarded as noise, which cannot progress the network learning.

Our localization system is developed in a real-world WLAN environment, where the realistic measurement of signal strength is collected to perform the experiments. We implement and compare the traditional approaches on the same test- bed, including WKNN, ML, and MLP. Three different performance metrics including mean of error, standard deviation of error, and accuracy are calculated and compared among different techniques. The experimental results indicate that the proposed algorithm is much higher in all the performance metrics. Finally, we analyze the number of inserted DCs from the cumulative percentage of eigenvalues obtained in our algorithm. It supports the fact that our network can intelligently throw out the redundant information for training and present better performance than other approaches do, because the network efficiently adapts the weights depending on only the useful discriminative information.

References

[1] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Comput. Mag.*, vol. 34, no. 8, pp. 57–66, Aug. 2001.

[2] K. Pahlavan, X. Li, and J. Makela, "Indoor geolocation science and technology," *IEEE Commun. Mag.*, vol. 40, no. 2, pp. 112–118, Feb. 2002.

[3] S. Tekinay, "Wireless geolocation systems and services," *IEEE Commun. Mag.*, vol. 36, no. 4, p. 28, Apr. 1998.

[4] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: Challenges faced in developing techniques for accurate wireless location information," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 24–40, Jul. 2005.

[5] R. Sharaf and A. Noureldin, "Sensor integration for satellite-based vehicular navigation using neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 589–594, Mar. 2007.

[6] G. Sun, J. Chen, W. Guo, and K. Liu, "Signal processing techniques in network-aided positioning: A survey of state-of-the-art positioning designs," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 12–23, Jul. 2005.

[7] J. Yin, Q. Yang, and L. M. Ni, "Learning adaptive temporal radio maps for signal-strength-based location estimation," *IEEE Trans. Mobile Comput.*, vol. 7, no. 7, pp. 869–883, Jul. 2008.

[8] S.-H. Fang, T.-N. Lin, and P.-C. Lin, "Location fingerprinting in a decorrelated space," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 5, pp. 685–691, May 2008.

[9] T. Roos, P. Myllymaki, H. Tirri, P. Misikangas, and J. Sievanen, "A probabilistic approach to WLAN user location estimation," *Int. J. Wireless Inf. Netw.*, vol. 9, no. 3, pp. 155–164, 2002.

[10] A. Kushki, N. Plataniotis, Konstantinos, N. Venetsanopoulos, and Anastasios, "Kernel-based positioning in wireless local area networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 689–705, Jun. 2007.

[11] S.-P. Kuo and Y.-C. Tseng, "A scrambling method for fingerprint positioning based on temporal diversity and spatial dependency," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 5, pp. 678–684, May 2008.

[12] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. New York: Wiely, 2000.

[13] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. INFOCOM*, 2000, pp. 775–784.

[14] R. Battiti, T. L. Nhat, and A. Villani, "Location-aware computing: A neural network model for determining location in wireless LANs," Dept. Inf. Commun. Technol., Univ. Trento, Trento, Italy, Tech. Rep. DIT-02-0083, 2002.

[15] M. Youssef, A. Agrawala, and A. U. Shankar, "WLAN location determination via clustering and probability distributions," in *Pervasive Comput. Commun.*, 2003, pp. 143–150.

[16] M. Brunato and R. Battiti, "Statistical learning theory for location fingerprinting in wireless LANs," *Comput. Netw.*, vol. 47, no. 6, pp. 825–845, 2005.

[17] A. M. Edgar, C. Raul, and F. Jesus, "Estimating user location in a WLAN using backpropagation neural networks," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2004, vol. 3315, pp. 737–746.

[18] C. Nerguizian, C. Despins, and S. Affes, "Geolocation in mines with an impulse response fingerprinting technique and neural networks," *IEEE Trans. Wireless Commun.*, vol. 5, no. 3, pp. 603–611, Mar. 2006.

[19] M. Kyperountas, A. Tefas, and I. Pitas, "Weighted piecewise LDA for solving the small sample size problem in face verification," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 506–519, Mar. 2007.

[20] Z. Manli and A. Martinez, "Pruning noisy bases in discriminant analysis," *IEEE Trans. Neural Netw.*, vol. 19, no. 1, pp. 148–157, Jan. 2008.

Continuously Differentiable Sample-Spacing Entropy Estimation

Umut Ozertem, Ismail Uysal, and Deniz Erdogmus

***Abstract*—The insufficiency of using only second-order statistics and premise of exploiting higher order statistics of the data has been well understood, and more advanced objectives including higher order statistics, especially those stemming from information theory, such as error entropy minimization, are now being studied and applied in many contexts of machine learning and signal processing. In the adaptive system training context, the main drawback of utilizing output error entropy as compared to correlation-estimation-based second-order statistics is the computational load of the entropy estimation, which is usually obtained via a plug-in kernel estimator. Sample-spacing estimates offer computationally inexpensive entropy estimators; however, resulting estimates are not differentiable, hence, not suitable for gradient-based adaptation. In this brief paper, we propose a nonparametric entropy estimator that captures the desirable properties of both approaches. The resulting estimator yields continuously differentiable estimates with a computational complexity at the order of those of the sample-spacing techniques. The proposed estimator is compared with the kernel density estimation (KDE)-based entropy estimator in the supervised neural network training framework with computation time and performance comparisons.**

***Index Terms*—Entropy estimation, minimum error entropy (MEE) criterion, supervised neural network training.**

I. Introduction

Mean square error (MSE) has been the workhorse of adaptive system training [1]–[3]. The simple analyzable structure, and low computational requirements are the most important advantages of the MSE cri-

Manuscript received November 09, 2007; accepted January 24, 2008. Current version published November 5, 2008. This work was supported in part by the National Science Foundation under Grants ECS-0524835 and ECS-0622239.

U. Ozertem is with Yahoo! Inc., Sunnyvale, CA 95054 USA (e-mail: umut@yahoo-inc.com).

I. Uysal is with the Electrical and Computer Engineering Department, University of Florida, Gainesville, FL 32611 USA (e-mail: uysal@ufl.edu).

D. Erdogmus is with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02215 USA (e-mail: derdogmus@ieee.org).

Digital Object Identifier 10.1109/TNN.2008.2006167

terion. Under Gaussianity assumption, MSE, which solely constraints the second-order statistics, is capable of extracting all possible information from the data, whose characteristics are uniquely defined by its mean and variance.

In many applications, data densities take complex forms; hence, the Gaussianity assumption becomes restrictive and many real life phenomena cannot be sufficiently described by only second-order statistics. At this point, constraining the information content of the signals is more promising as compared to using simply their energy. In fact, the insufficiency of mere second-order statistics has been discovered, and more advanced information theoretic objectives are now being studied [4]–[7]. Due to its conceptual simplicity, error entropy minimization merits special attention here.

Entropy is introduced by Shannon as a measure of the average information in a given probability density function (pdf) [8], [9]. It is an explicit function of the pdf itself; hence, it includes all the high-order statistical properties defined in the pdf. Consequently, as an optimality criterion, entropy is superior to MSE, as minimizing the entropy constrains all moments of the error pdf, whereas MSE constrains only the first and second moment of the pdf. In this context, minimizing the output error entropy is equivalent to minimizing the distance between the probability densities of the output and the desired signal [6].

Entropy is a function of the data pdf, and analytical data distributions are never available in real applications. Therefore, entropy must be estimated from the data samples. One common approach is to directly substitute an estimate of the pdf of the signal into the sample mean approximation for the expectation [10], and here, kernel density estimation (KDE) is the typical density estimation scheme [11], [12]. If the kernel function itself is continuously differentiable, KDE yields a continuously differentiable density estimate, which is crucial for gradient-based adaptation. Another alternative is employing density estimators based on sample spacing. This is another nonparametric approach, which is based on the distance between pairs, or generally m-tuples, of the data samples. In this approach, there is no problem such as kernel selection; however, the resulting estimates are not differentiable, hence, not suitable for gradient-based adaptive learning.

The computational bottleneck for these entropy estimators is the computational complexity of the density estimation method employed. Typically, KDE results in $O(N^2)$ complexity, whereas the sample-spacing methods result in $O(Nm)$, where N is the number of data samples, and m is the number of samples used for sample spacing. We propose a blend of these two methods to be used in the entropy estimation that captures the desirable properties of KDE and sample-spacing methods. Proposed approach yields a continuously differentiable density estimate such as KDE, yet it has a low computational complexity, equivalent to the sample-spacing methods.

The high computational requirement of KDE makes it impractical to use KDE-based entropy estimators with many samples, and one of the most important applications that suffers from this limitation is the error-entropy-minimization-based learning. Therefore, we will present the results regarding to the proposed entropy estimator in the context of output error-entropy-based supervised neural network training.

II. Entropy Estimation Methods

A. KDE-Based Entropy Estimates

In KDE, the density estimate is obtained as a sum of kernel functions, where the kernels are centered at the data samples. A crucial point is the selection of the kernel function, and the most commonly used kernel function is the Gaussian. The Gaussian kernel function itself is continuously differentiable; hence, it results in continuously differentiable density estimates. For a given kernel function $K(\cdot)$, kernel density estimate of a random variable e with samples $\{e_i\}_{i=1}^N$ becomes

$$\hat{p}(e) = \frac{1}{N}\sum_{i=1}^{N} K(e - e_i). \tag{1}$$

There is a wide literature on how to select the bandwidth of the Gaussian kernel and there are supervised and unsupervised solutions to this problem [11]–[14]. Methods vary from nearest-neighbor-based heuristics to principled maximum-likelihood (ML)-based approaches. Still, the selection of the *optimal* bandwidth is an open ended problem. For the same random variable e, Renyi's order-α entropy is given by

$$H_\alpha(e) = \frac{1}{1-\alpha}\log\int p^\alpha(e)\,de. \tag{2}$$

Substituting (1) into (2), one can obtain the KDE-based entropy estimate as

$$\hat{H}_\alpha(e) = \frac{1}{1-\alpha}\log\frac{1}{N^\alpha}\sum_{j=1}^{N}\left(\sum_{i=1}^{N} K(e_j - e_i)\right)^{\alpha-1}. \tag{3}$$

Shannon's entropy is the limiting case of Renyi's entropy where $\alpha \to 1$, and the KDE-based estimator for Shannon's entropy can be obtained in a similar way. In our experiments, we will focus on Renyi's second-order entropy ($\alpha = 2$) in line with the earlier results presented by Erdogmus [6], [7].

B. Sample-Spacing Estimates

Consider a random variable e, and its *order statistics*. Using the order statistics, one can rearrange the samples in a nondecreasing order to obtain $\{e_1 < e_2 < \ldots < e_N\}$. The 1-spacing density estimate is given by

$$\hat{p}(e) = \begin{cases} \dfrac{1}{(N-1)(e_{i+1} - e_i)}, & \text{if } e_i \le e \le e_{i+1} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Note that the probability mass between two successive samples of the random variable here is $(1)/(N-1)$.[1] The statistical variance of this estimator can be decreased by a factor of m by using successive m-sample intervals rather than each successive pair

$$\hat{p}(e) = \begin{cases} \dfrac{1}{(N-m)(e_{i+m} - e_i)}, & \text{if } e_i \le e \le e_{i+m} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

This is the m-spacing estimator. Similarly, the expected value of the probability mass between successive samples is $(1)/(N-m)$. The m-spacing estimator of the Shannon's entropy is given by [16], [15]

$$H(e) = \frac{1}{N}\sum_{i=1}^{N-m}\log\left(\frac{N}{m}(e_{i+m} - e_i)\right). \tag{6}$$

In Section III, we will write Renyi's second-order entropy for continuously differentiable sample-spacing (CDSS) estimator using the connection to KDE. Similarly, m-spacing estimator of Renyi's entropy can be written. We skip this derivation here, because it is not central to the focus of this paper.

III. Continuously Differentiable Sample-Spacing Entropy Estimation

To build our continuously differentiable sample-spacing estimator, we reinterpret the sample-spacing-based pdf estimation as a *sum of finite support uniform kernel functions centered at the midpoint of successive m-sample intervals*, where the weight of each kernel is determined inversely proportional to the corresponding sample spacing—as

[1]Note that there is a slight difference in the density estimator that Miller and Fisher use [15].

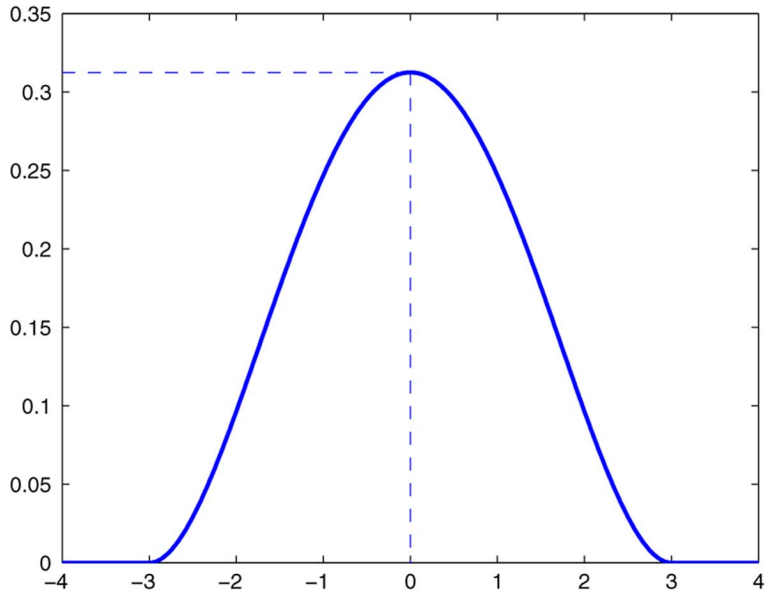

Fig. 1. Polynomial kernel for $\sigma_i = 3$ and $\bar{e}_i = 0$.

in (4) or (5). Although this fact is easier to observe in 1-spacing estimator, it is also valid for any m-spacing estimator. Using this interpretation, one can replace each uniform kernel with a continuously differentiable kernel function to obtain a continuously differentiable estimate. Defining the midpoints of each sample pair $\{e_i, e_{i+m}\}$ as $\tilde{e}_i$, one can obtain the kernel centers

$$\tilde{e}_i = (e_{i+m} + e_i)/2. \tag{7}$$

Using $\tilde{e}$, the m-spacing pdf estimate in (5) can be rewritten as

$$\hat{p}(e) = \frac{1}{(N-m)} \sum_{i=1}^{N-m} K_{\sigma_i}(e - \tilde{e}_i) \tag{8}$$

where the kernel function $K_{\sigma_i}(\cdot)$ is uniform at the value $(1)/((e_{i+m} - e_i))$ in the interval $[e_i, e_{i+m}]$, and zero otherwise. (Note that $\tilde{e}$ is by definition in a nondecreasing order.) To obtain a continuously differentiable density, one should substitute this uniform kernel with a kernel function that meets the following conditions: 1) it is zero outside the interval $[e_i, e_{i+m}]$; 2) it satisfies the boundary conditions at e_i and e_{i+m}, that is, the limit and the derivative of the kernel function at these point has to be 0; 3) it integrates to 1; and 4) it is continuously differentiable.

A fourth-order polynomial—with double roots at e_i and e_{i+m}—is the minimum-order polynomial that meets the above four conditions

$$K_{\sigma_i} = \begin{cases} A_i(e - \sigma_i)^2(e + \sigma_i)^2, & e \in [-\sigma_i, \sigma_i] \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where σ_i is selected to be

$$\sigma_i = (e_{i+m} - e_i)/2 \tag{10}$$

and the normalization factor A_i can be evaluated by integrating the kernel and equating it to 1. For our selection in (9), A_i becomes

$$A_i = 15\sigma_i^{-5}/16. \tag{11}$$

An illustration of the selected kernel function is shown in Fig. 1.

The resulting density estimate becomes continuously differentiable and suitable for first- and second-order iterative learning algorithms. Besides, the finite support property of the kernels will provide significant computational savings in the entropy estimation. Using the density estimate in (8), one can easily formulate the entropy estimator by approximating the expected value operator with the sample mean. Consider Renyi's second-order entropy

$$\begin{aligned} H_2(e) &= -\log \int p^2(e)de = -\log E[p(e)] \\ &\approx -\log \frac{1}{N} \sum_{j=1}^{N} \frac{1}{N-m} \sum_{i=1}^{N-m} K_{\sigma_i}(e_j - \tilde{e}_i) \\ &\approx -\log \frac{1}{N(N-m)} \sum_{j=1}^{N} \sum_{i=1}^{N-m} K_{\sigma_i}(e_j - \tilde{e}_i). \end{aligned} \tag{12}$$

To write this estimator in terms of the samples only, one should substitute (7), (9), (10), and (11) for $\tilde{e}_i, K_{\sigma_i}(\cdot), \sigma_i$ and A_i, respectively. To implement the finite-support kernels and remove unnecessary evaluations, one should change the summation indices accordingly. This yields

$$H_2(e) \approx -\log \sum_{i=1}^{N-m} \sum_{j=i+1}^{i+m-1} \frac{30(e_j - e_{i+m})^2(e_j - e_i)^2}{N(N-m)(e_{i+m} - e_i)^5} \tag{13}$$

where the estimator is directly given in terms of error samples.

IV. Comparison of Underlying Density Estimates

Before we proceed to the adaptive system training scenario, we will briefly discuss properties of underlying density CDSS density estimation and KDE. Comparison of the characteristics of the underlying density estimates is critical to understand the characteristics of the resulting entropy estimators.

The most important open ended problem in KDE literature is the selection of an efficient kernel bandwidth, and the results severely suffer from improper kernel width selections. CDSS approach automatically eliminates the necessity for searching for the kernel width, and the kernel bandwidths are automatically selected exploiting the sample spacings as shown in (10).

Another well-known fact in the density estimation literature is the superior asymptotic behavior and improved outlier robustness provided by the variable kernel width density estimates. Variable kernel widths are selected in a manner that the width of the kernel increases for the samples that are more likely to be outliers of the distribution. In CDSS density estimation, sample spacings naturally implement each sample's outlier likelihood and the resulting estimator is a variable bandwidth kernel density estimator. Note that the only parameter in the final CDSS entropy estimator in (13) is m, the number of samples

used in sample spacing. This parameter adjusts the tradeoff between the computational complexity and the asymptotic behavior of the estimator. Similar to sample-spacing density estimates as in (5) and (8), a consistent and asymptotically unbiased estimator exists if and only if $\lim_{N\to\infty} m = \infty$ and $\lim_{N\to\infty}(m)/(N) = 0$. Typical selections for m include $m = \sqrt{N}$ and $m = \sqrt[4]{N}$ [15]. The computational complexity of CDSS entropy estimator is $O(mN)$, which becomes $O(N^{3/2})$ and $O(N^{5/4})$ for these choices, both of which are significantly less than $O(N^2)$ computational complexity of the KDE entropy estimator. Moreover, the computational complexity can be further reduced to $O(N)$ if *a priori* information about the data or empirical evaluations yield a sufficient constant value for m.

Selection of m is in line with selection of kernel bandwidth in KDE. At this point, note that in CDSS, computational savings is an important criterion to be considered in the selection of m, whereas in KDE, the selection of bandwidth does not bring any computational savings. In KDE literature, one of the most principled kernel optimization approaches is to use the leave-one-out cross-validation ML procedure. Consider the random variable e with samples $\{e_1, e_2, \ldots, e_N\}$. The KDE that leaves the ith sample out of density estimation is given by

$$p_i(e) = \frac{1}{N} \sum_{j=1, j\neq i}^{N} K_\sigma(e - e_j) \tag{14}$$

and the objective of the kernel bandwidth optimization problem is defined by maximizing the log-likelihood function over all samples

$$\max_{\sigma} \log \prod_{i=1}^{N} p_i(e). \tag{15}$$

This optimization problem can be solved using a line search. For CDSS, a similar approach can be defined over the sample-spacing parameter m. This leads to a discrete optimization problem over m, which is not much more difficult than the analogous problem for KDE for the 1-D case here.

In KDE-based minimum error entropy (MEE) adaptive system training context, implementing any kernel optimization procedure is quite impractical, and computationally more efficient—mostly heuristic—kernel selection methods have been used in the MEE system training literature. The reason is this optimization should be performed at every step of system adaptation. Regardless of the adaptive system used in the training, at every step of system adaptation the error samples of the training set change. This requires an ML kernel optimization procedure at every step of the system training, which is computationally unfeasible. Similarly for CDSS, no kernel optimization technique known to the authors is computationally cheap enough to be used in the system adaptation iterations. If CDSS is going to be used for other applications, the ML approach explained above can be used, but with today's computational power, MEE-based learning algorithms are unfortunately still bound to employ heuristics, visual or empirical inspections for selecting the kernel parameters.

For the kernel function selected in (9), note that 1-spacing density estimator in particular is rather problematic; the pdf is equal to 0 at all data samples.[2] For simplicity, here we used the lowest order polynomial that satisfies the four constraints listed before, and employed an m-spacing estimator. For a 1-spacing estimator, the kernel function should be modified in a way that the finite support of the considered fourth-order polynomial extends beyond the sample pairs. This can be achieved either by modifying the placement of the root of the polynomial or using a piecewise defined function that satisfies the aforementioned constraints.

[2]In sample-spacing entropy estimation, kernels can be centered at the sample of interest or at the midpoint of the convex interval (or center of mass of the volume in multidimensional cases) that contains all m-nearest neighbors. The first choice does not have the counterintuitive zero-likelihood observed samples. We choose the second option and this creates the problem of zero-likelihood estimates for the observed sample set.

Finally, note that the CDSS procedure applied here is specifically based on order statistics in one dimension and the estimation cannot be directly generalized into multivariate estimates. Still, the modifications to convert the proposed estimator into multivariate scenarios is very straightforward. In higher dimensional spaces, instead of just the ordering, one needs to generate K-nearest-neighbor statistics for each sample by checking pairwise distances between all samples.[3] This methodology is the basis of Kraskov's work on mutual information estimation [17]. Using the same interpretation of uniform rectangular kernel, one can replace these with continuously differentiable kernel functions to obtain a multivariate CDSS estimator.

Overall, CDSS yields a computationally efficient variable width kernel density estimate that is robust to outliers. The optimal way to select the scale parameter can be achieved by maximizing the data likelihood; however, this is still as computationally expensive as it is in KDE, and therefore, cannot be used in MEE context, because the error distribution is changing at every step of the system training.

V. Minimum Error Entropy and Adaptive System Training

In adaptive learning context, MEE has been shown to be superior to methods based on second-order statistics [6], [7]. Although the error entropy minimization approach is very promising, it can only be applied to small scale problems due to high computational complexity. Therefore, we present CDSS results in minimum-error-entropy-based adaptive system training context. Here, one can define the optimization criterion over the entropy estimate H_2, or equivalently, one can also use the corresponding information potential V_2, which is the term in the logarithm in (13).

For gradient-based adaptation, one needs the derivative of the optimization objective with respect to the system parameters. Using chain rule this can be decomposed into two parts: 1) $(\partial V_2)/(\partial y)$ the derivative of the error entropy with respect to the system output, and 2) $(\partial y)/(\partial w)$ the derivative of the system output with respect to the system parameters. The first part 1) is specific to the CDSS definition here, whereas the second part 2) is independent of the CDSS definition and only depends on the system to be trained. To make the derivation generic for any system selection, here we focus on the first part, and leave the specific system selection choice to the user. Depending on the choice of the adaptive system, the form of $(\partial y)/(\partial w)$ can be decided from the relevant literature [2].

Considering a training set of input and desired output pairs and an adaptive system with parameter vector $\mathbf{w}$, one can design a gradient-based adaptation. Gradient descent is the simplest and the most commonly used choice, whereas second-order alternatives such as the Newton's method are also possible. Let y denote the output of the system, and e_i denote the training error between the ith input–output pair. The gradient of the error entropy with respect to the system parameters can be obtained using the derivative of the CDSS entropy estimator in (13). Because the entropy is a monotonic function of information potential, one can focus on the information potential $V_2(e)$, which is the argument of the logarithm in (13). This gives

$$\begin{aligned}
\frac{\partial V_2(e)}{\partial \omega} &= \frac{30}{N(N-m)} \sum_{i=1}^{N-m} \sum_{j=i+1}^{i+m-1} \Phi_{ij}\Delta_{ij} \\
\Phi_{ij} &= (e_{i+m} - e_i)^{-6}(e_j - e_{i+m})(e_j - e_i) \\
\Delta_{ij} &= \begin{pmatrix} 5\left(\dfrac{\partial y_{i+m}}{\partial w} - \dfrac{\partial y_i}{\partial w}\right)(e_j - e_{i+m})(e_j - e_i) \\ -(e_{i+m} - e_i)\left(\dfrac{\partial y_j}{\partial w} - \dfrac{\partial y_{i+m}}{\partial w}\right)(e_j - e_i) \\ -(e_{i+m} - e_i)(e_j - e_{i+m})\left(\dfrac{\partial y_j}{\partial w} - \dfrac{\partial y_i}{\partial w}\right) \end{pmatrix}.
\end{aligned} \tag{16}$$

[3]Also note that the number of neighbors K used here has to be selected greater than the data dimensionality.

In (16), $(\partial y_i)/(\partial w)$ denotes the derivative of the adaptive system output with respect to the system parameters; therefore, it is independent of the CDSS scheme and it only depends on the adaptive system selected by the user. For the adaptive system to be used, derivative of the adaptive system output with respect to the system parameter vector $(\partial y_i)/(\partial w)$ can be found in the literature; hence, CDSS-based gradient adaptation approaches can be implemented for different adaptive systems using (16) only. We will provide the details of the derivation in the Appendix. In Section VI, we will use standard neural networks and employ a gradient descent with line search to present the optimizer of the CDSS-based MEE cost surface.

In many adaptive learning scenarios, the optimization algorithms are sensitive to local optima. Deterministic annealing is the one of the most commonly used methods to tackle this problem. The approach is based on introducing a smoothing to the cost function—so that the local optima would disappear—and tracking the local optima while slowly reducing the amount of introduced smoothing. Previously, this idea was applied in the KDE-based MEE context by Erdogmus [7], and deterministic annealing is shown to lead to better results. Conceptually, implementing deterministic annealing in KDE-based MEE is quite straightforward using Gaussian kernel functions. In the entropy estimation, one should start with a wide kernel bandwidth, leading to a smooth density function that eliminates local optima in the resulting objective function. Then, the bandwidth of the Gaussian should be decreased slowly as the iterations proceed.

In KDE-based MEE approach, one requires an additional parameter to adjust the rate of change of the kernel bandwidth—either linearly or logarithmically.[4] CDSS inherently implements the kernel annealing approach with no additional parameters. Considering (10), one can see that the kernels adapt the data scale automatically. At a random initialization of system weights in the beginning, the error samples are rather big, and kernel functions turn out relatively wide at the initialization, preventing the gradient-based algorithm from getting trapped in a local optima. However, as the algorithm starts converging and the error distribution starts becoming narrower, the kernel sizes also shrink to prevent oversmoothing. In Section VI, we will provide a comparison of distributions of CDSS kernel sizes throughout iterations.

VI. Experimental Results and Comparisons

The performance of the proposed supervised learning scheme will be evaluated using a framework consisting of a time delay neural network (TDNN). Three different TDNNs will be trained with three different learning techniques minimizing the following performance metrics: 1) mean square error (MSE), 2) error entropy with kernel density estimation (MEE-KDE), and 3) error entropy with continuously differentiable sample-spacing estimator (MEE-CDSS). TDNNs will be trained to perform two different tasks: the short-term prediction of the Mackey–Glass chaotic time series and a computer generated time series, which was actually used in Santa Fe time-series computation, displaying high-dimensional dynamics with finite states and drifting parameters [18], [19].

A. Case Study I: Mackey–Glass Time Series

The first experiment is the short-term prediction of a Mackey–Glass chaotic time series with parameter $\tau = 30$. The input to the TDNN will consist of the current value of the chaotic time series, followed by the five delayed values in time, whereas the output will be the prediction of the next instant in the series. The TDNN framework for each scheme consists of a single hidden layer with the network nonlinearity chosen as the *tanh* function.

For a comprehensive analysis that avoids convergence to local solutions, a systematic Monte Carlo approach is followed. Each TDNN is trained 1000 times with random realizations of initial weight and bias values whereas the number of neurons in the hidden layer is also varied between 5 and 10. All TDNNs were tested with 10 000 samples with a sampling period of 0.1 s. A well-known backward propagation algorithm with a variable step size is used for error minimization [20]. For MEE-KDE, the kernel size is empirically chosen to be $\sigma = 0.01$. As suggested in the literature, the sample size for MEE-CDSS m is chosen as integer multiples of $\sqrt{N}$ where N is the size of the training input [15].

After empirical analysis, m is chosen to be $3\sqrt{N}$. Note that unlike searching efficient values of the kernel bandwidth in KDE, selection of m affects the computation time as well. To be able to present results for different number of samples while fixing the computation time, we set this parameter empirically.

Table I compares the performances and computational complexities of the three supervised learning schemes. For the first three rows, all TDNNs were trained with the exact same number of samples. MEE-KDE has the best L1 performance while requiring more computation time than the other two algorithms. On the other hand, MEE-CDSS has a very similar performance to MEE-KDE with less than only $1/4$th of its computation time.

The novelty of the MEE-CDSS scheme becomes more obvious when it is allowed to train using more training samples such that its computation time is approximately similar to MEE-KDE. The last two rows of Table I displays the results for such a scenario where both MSE and MEE-CDSS are allowed to use more samples for training to match their computation time that of MEE-KDE. The L1 performance of MSE, while getting closer, still cannot outperform MEE-KDE; on the contrary MEE-CDSS now has a lower error than MEE-KDE while displaying asymptotic stability.

Fig. 2 shows the error distribution for the three schemes for different number of iterations. Even though they are all concentrated around

TABLE I
Computation Time Versus L1 Performance Comparisons and Tradeoffs for the Learning Schemes—Mackey–Glass Time-Series Prediction

Scheme	Computation Time	L1 Performance	# of samples
MSE	0.05 units	0.064	200
MEE - CDSS	**0.45 units**	**0.048**	**200**
MEE - KDE	2.1 units	0.046	200
MSE	2 units	0.052	4000
MEE - CDSS	**1.8 units**	**0.038**	**1250**

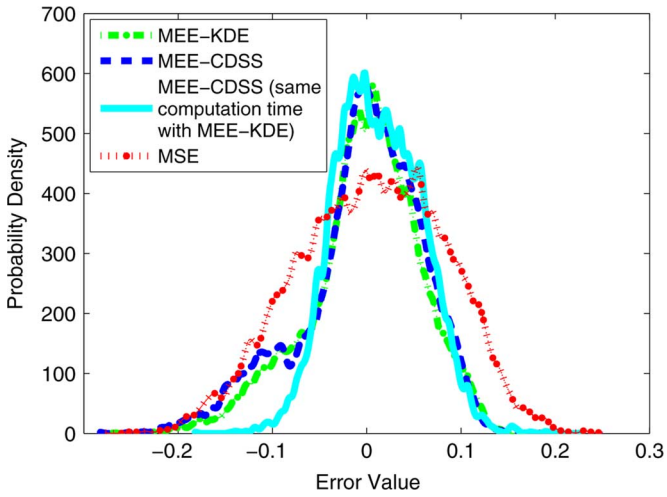

Fig. 2. The error probability densities for MSE, MEE-KDE and MEE-CDSS for the Mackey-Glass time-series prediction.

[4]In annealing, this parameter is generally referred to as the temperature.

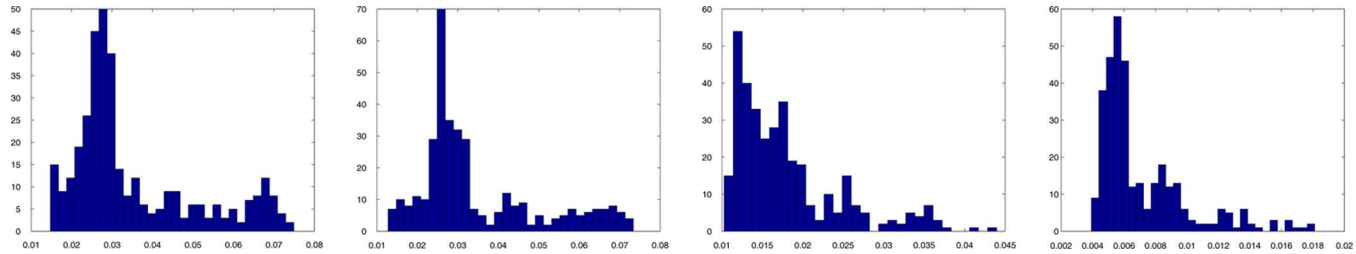

Fig. 3. Comparison of CDSS kernels throughout the TDNN training, at the initialization, 50th, 100th, and 200th iterations, from left to right. As the error samples get smaller, they get close to each other; hence, smaller kernel sizes are required. CDSS kernels automatically adapt the scale of the data.

zero mean error, MSE has a considerably larger variance whereas MEE-KDE and MEE-CDSS distributions display quite similar characteristics. More importantly, when the computation time is fixed and MEE-CDSS is allowed to use more training samples to compensate for reduced complexity, the error variance is decreased. Fig. 3 compares the distribution of the kernel widths throughout the iterations. At the initialization, broader kernels are required to avoid local optimizers. As the adaptive algorithm converges, the error samples get much smaller and smaller kernel bandwidths are required to prevent introducing an oversmoothing to the density estimate. Even though m is kept constant throughout, CDSS implements kernel annealing with no additional parameter.

B. Case Study II: Time Series With High-Dimensional Dynamics

For the second task, we used a computer generated data set with high-dimensional dynamics and weak nonstationarity modeled by drifting parameters [19]. This data set was used in the Santa Fe time-series competition directed by N. Gershenfeld and A. Weigend. The internal state dynamics and the output of the system can be summarized by the following equations.

The time series is basically generated by numerically integrating the dynamic equations of motion for a weakly damped and periodically driven particle [21] given below:

$$\frac{d\mathbf{x}^2}{dt^2} + \gamma\frac{d\mathbf{x}}{dt} + \nabla V(\mathbf{x}) = F(t) \tag{17}$$

where $\mathbf{x}$ is a 4-D vector

$$\mathbf{x} = [x_1; x_2; x_3; x_4]^T \tag{18}$$

defining an asymmetric 4-D, four-well potential V

$$V = a_4\sqrt{(x_1^2 + x_2^2 + x_3^2 + x_4^2)} - a_2\left(x_1^2 x_2^2\right) - a_1 x_1. \tag{19}$$

A small drift is introduced into dissipation, $-\gamma$ parameter, by integrating a Gaussian random variable. To solve the equations, an arbitrary period of $(1)/(\omega)$ is enforced in the x_3 direction

$$F(t) = \sin(\omega t). \tag{20}$$

The time series is generated by observing the following variable:

$$f = \sqrt{(x_1^2 + 0.3)^2 + (x_1^2 + 0.3)^2 + x_3^2 + x_4^2} \tag{21}$$

which was sampled at every 0.05 s. The main reason for choosing such a system is the existence of high-dimensional dynamics and the slow drift of parameters to ensure a long-term change in transition probabilities. Moreover, by picking such an observable, the complexity of the problem is increased by having more internal states than externally observable [19].

TABLE II
COMPUTATION TIME VERSUS L1 PERFORMANCE COMPARISONS AND TRADEOFFS FOR THE LEARNING SCHEMES—MOTION OF A DAMPED, DRIVEN PARTICLE

Scheme	Computation Time	L1 Performance	# of samples
MSE	0.15 units	0.092	400
MEE - CDSS	**1.4 units**	**0.04**	**400**
MEE - KDE	7.5 units	0.036	400
MSE	7.2 units	0.078	7500
MEE - CDSS	**7.2 units**	**0.034**	**2250**

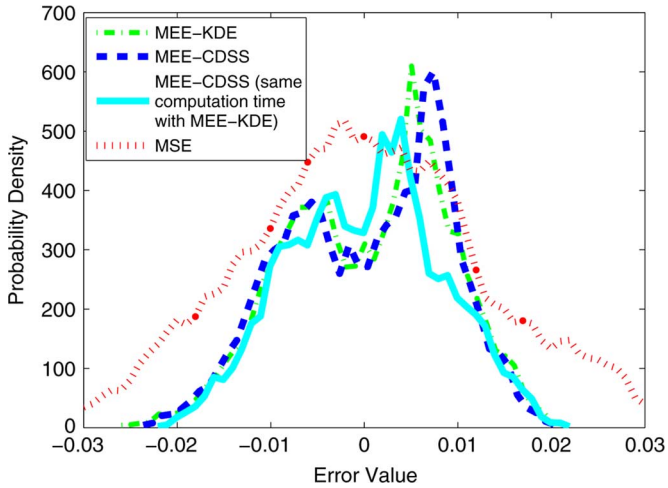

Fig. 4. Error pdfs for MSE, MEE-KDE, and MEE-CDSS for the prediction of the motion of a weakly damped and periodically driven particle.

The specifications/parameters (number of iterations, m with $3\sqrt{N}$, etc.) of TDNNs used and the training and testing process using the Monte Carlo approach are practically identical to the ones used for Mackey–Glass as described in Section VI-A above. For this experiment, 10 000 samples are used for testing.

Table II compares the performances and computation times of the three supervised learning schemes for the second case study. As the table shows, the results confirm those of the first experiment. When the exact same number of iterations and the same size of training data is used, MEE-KDE has the best performance with the tradeoff of having a higher computation time. Also note that, with the increasing size of the training samples available, the ratio of the computation times of MEE-KDE and MEE-CDSS increases; however, the L1 performance of MEE-CDSS is still very close to MEE-KDE. Once again, the performance of MEE-CDSS scheme surpasses MEE-KDE assuming a fixed amount of computation time is allowed for both algorithms. Fig. 4 shows the error distributions for the three training schemes. Given the same number of iterations and the same size of training

data, the error variance of MEE-KDE is noticeably less than that of MSE, however there is only a marginal difference between MEE-KDE and MEE-CDSS. On the other hand, given a fixed computation time and more training samples, the error density of MEE-CDSS achieves a more concentrated peak around zero.

To summarize, it has already been shown that MEE-KDE prefers a larger and more concentrated peak around zero error and approximates the pdf of the desired output much better than MSE with the cost of a substantially increased computational time during the training of the system [6]. Results presented in this section simply demonstrate that with the use of continuously differentiable sample spacing, MEE-CDSS can achieve a similar performance to MEE-KDE with much less computational complexity.

VII. Conclusion

We present a hybrid entropy estimator that combines the desirable properties of KDE and sample-spacing methods. For this purpose, we reinterpret the sample-spacing method and utilize variable-size finite-support kernels. With a single analytical expression over the nonzero interval, a fourth-order polynomial is the lowest order polynomial that satisfies the boundary conditions. The finite-support nature of the resulting kernel function yields significant computational savings.

Outlier robustness and better asymptotic behavior are well-studied results of variable size KDE. In CDSS, variable kernel sizes that locally fit the data naturally arise after the reinterpretation of the sample-spacing estimates, and the resulting variable kernel size entropy estimator has no parameters except m, which eliminates tedious kernel size optimization requirements to adjust data-dependent variable bandwidths. Furthermore, it is known that deterministic annealing improves the results if there is local optima in the optimization objective. In MEE context, this annealing is performed over the kernel bandwidth, and CDSS density estimator inherently implements this kernel annealing without any additional parameter. However, ML-based optimal selection of the sample spacing m is still as computationally expensive as optimization of the Gaussian bandwidth in KDE, and cannot be implemented in MEE context.

We tested our computationally efficient CDSS entropy estimator in supervised adaptive system training context, which is one of the main areas that suffers from the computational complexity of entropy estimation. The CDSS entropy estimator has proven superior to its KDE-based counterpart by having the best computational complexity-performance tradeoff while displaying asymptotic stability. To sum up, the reduced complexity will allow one to employ error-entropy-minimization-based techniques for mid- or large-scale problems; in addition, when compared to the fixed kernel size KDE-based scheme, CDSS entropy estimator enables training with more samples and achieves better performance in the same amount of computation time.

Appendix

We will briefly present the intermediate steps of the result presented in (16). We start with rewriting the entropy estimator $H_2(e) = -\log V_2(e)$, where V_2 is

$$V_2(e) \approx \sum_{i=1}^{N-m} \sum_{j=i+1}^{i+m-1} \frac{30(e_j - e_{i+m})^2 (e_j - e_i)^2}{N(N-m)(e_{i+m} - e_i)^5}.$$

Taking the derivative of V_2 with respect to system parameters, one should consider $e = d - y$, and $(\partial e)/(\partial w) = -(\partial y)/(\partial w)$, where y is the output of the system, and d is the known desired output. Taking the derivative using chain rule, one obtains

$$\begin{aligned}\frac{\partial V_2(e)}{\partial \omega} = \frac{30}{N(N-m)} \sum_{i=1}^{N-m} \sum_{j=i+1}^{i+m-1} \Omega_{ij}\\ \Omega_{ij} = -5\left(\frac{\partial e_{i+m}}{\partial w} - \frac{\partial e_i}{\partial w}\right)(e_{i+m}-e_i)^{-6}(e_j - e_{i+m})^2(e_j-e_i)^2\\ +2(e_{i+m}-e_i)^{-5}\left(\frac{\partial e_j}{\partial w} - \frac{\partial e_{i+m}}{\partial w}\right)(e_j - e_{i+m})(e_j-e_i)^2\\ +2(e_{i+m}-e_i)^{-5}(e_j-e_{i+m})^2\left(\frac{\partial e_j}{\partial w} - \frac{\partial e_i}{\partial w}\right)(e_j-e_i)^2.\end{aligned}$$

Substituting the expression for $(\partial y)/(\partial w)$ and collecting the common terms in Ω_{ij}, one obtains the result presented in (16).

References

[1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[3] A. H. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley, 2003.

[4] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.

[5] A. Cichocki and S. I. Amari, *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. New York: Wiley, 2002.

[6] D. Erdogmus and J. C. Principe, "An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems," *IEEE Trans. Signal Process.*, vol. 50, no. 7, pp. 1780–1786, Jul. 2002.

[7] D. Erdogmus and J. C. Principe, "Generalized information potential criterion for adaptive system training," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1035–1044, Sep. 2002.

[8] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana, IL: Univ. Illionis Press, 1964.

[9] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[10] J. Berilant, E. J. Dudewicz, L. Gyorfi, and E. C. van der Meulen, "Nonparametric entropy estimation: An overview," *Int. J. Math. Statist. Sci.*, vol. 6, no. 1, pp. 17–39, 1997.

[11] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 1962.

[12] L. Devroye and G. Lugosi, *Combinatorial Methods in Density Estimation*. New York: Springer-Verlag, 2001.

[13] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Champman & Hall, 1986.

[14] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 281–288, Feb. 2003.

[15] E. G. Miller and J. W. Fisher, "ICA using spacings estimates of entropy," in *Proc. 4th Int. Symp. ICA Blind Signal Separat.*, 2003, pp. 1047–1052.

[16] O. Vasicek, "A test for normality based on sample entropy," *J. Roy. Statist. Soc. B*, vol. 38, no. 1, pp. 54–59, 1976.

[17] A. Kraskov, H. Stogbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, 2004, 066138.

[18] D. Kaplan and L. Glass, *Understanding Nonlinear Dynamics*. New York: Springer-Verlag, 1995.

[19] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA: Addison-Wesley, 1994.

[20] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1973.

[21] S. Linkwitz and H. Grabert, "Energy diffusion of a weakly damped and periodically driven particle in an anharmonic potential well," *Phys. Rev. B, Condens. Matter*, vol. 44, no. 21, pp. 11888–-11900, 1991.