The Laplacian Classifier

Robert Jenssen, *Member, IEEE*, Deniz Erdogmus, *Member, IEEE*, Jose C. Principe, *Fellow, IEEE*, and Torbjørn Eltoft, *Member, IEEE*

***Abstract*—We develop a novel classifier in a kernel feature space related to the eigenspectrum of the Laplacian data matrix. The classification cost function measures the angle between class mean vectors in the kernel feature space, and is derived from an information theoretic divergence measure using Parzen windowing. The classification rule is expressed in terms of a weighted kernel expansion. The weighting associated with a data point is inversely proportional to the probability density at that point, emphasizing the least probable regions. No optimization is needed to determine the weighting scheme, as opposed to the support vector machine. The connection to Parzen windowing also provides a theoretical criterion for kernel size selection, reducing the need for computationally demanding cross-validation. We show that the new classifier performs better than the Parzen window Bayes classifier, and in many cases comparable to the support vector machine, at a computationally lower cost.**

***Index Terms*—Cauchy–Schwarz (CS) divergence, classification, Laplacian matrix, Mercer kernel feature space, Parzen windowing.**

I. Introduction

CLASSIFICATION of the data points which comprise a data set is one of the fundamental problems in machine learning and signal processing [1]–[3]. The classifier is normally constructed based on a cost function and an available *training* data set with known class labels. The goal is to appropriately classify a *test* data set.

One intuitively appealing classification criterion is to construct the classifier such that the probability of classification errors is minimized. It is well known that the Bayes classifier [1]–[3] is optimal with respect to minimum error probability. The problem is that the Bayes classifier assumes *a priori* knowledge of the class probability density functions (pdf's). These pdf's are unfortunately unknown in practice.

One way to implement a suboptimal Bayes classifier is to replace the actual densities by Parzen window estimators [4], obtained using the N_{train} training data points. The Parzen window estimator does not assume any parametric model for the class densities. It is expressed as a sum of equally weighted Parzen windows, or kernels. In order to classify a test data set, these sums, totally consisting of N_{train} terms, have to be evaluated for each test data point. This is a procedure having computational complexity in the order of $O(N_{\text{test}}N_{\text{train}})$

During the last decade, another kernel based classifier, known as the support vector machine (SVM) [5]–[7], has received a lot of attention. The SVM relies on a maximum margin regularization criterion. The criterion is expressed solely in terms of inner-products. The actual maximization of the margin takes place in a so-called Mercer kernel feature space. The inner-products in this space can be computed based on the "kernel-trick," using a Mercer kernel function. The final form of the SVM resembles the Parzen window Bayes classifier. However, the maximum margin criterion leads to a *weighting* of the relevance of the training data points in the construction of the classifier. This is in contrast to the Parzen window Bayes classifier, where all data points are weighted equally. This weighting property has been shown to lead to improved classification performance. To obtain the relevant weighting, which determines the N_{sv} support vectors, a convex optimization problem must be solved. This procedure has a computational complexity in the order $O\left(N_{\text{train}}^3\right)$. More efficient heuristically based optimization methods do exist, such as e.g., sequential minimal optimization [8]. It is also not straightforward to select the two SVM parameters, which are essential for the classifier performance. These parameters are the kernel size, and a constant which regulates the tradeoff between training data misclassification and classifier regularization. It can be a computationally demanding task to perform cross-validation over both these parameters, since the SVM optimization has to be performed for each such pair. However, a desirable feature of the SVM optimization is that is leads to a certain sparseness in the actual classification procedure, since the number of support vectors equals the number of terms in the resulting kernel expansion. Hence, the testing phase complexity is in the order $O(N_{\text{test}}N_{\text{sv}})$.

In this paper, we propose a new classifier which does not require the optimization phase associated with the SVM, but which still produces classification results which in many cases are comparable to the SVM. The new classifier is named the *Laplacian classifier*. It is based on measuring angles between class mean vectors in a Mercer kernel feature space, which may be approximated by the eigenspectrum of the Laplacian data matrix [9] (see Section II-B). The classification cost function is derived from the Cauchy–Schwarz (CS) [10] information theoretic divergence measure between pdf's. Using Parzen windowing for density estimation, the resulting classification rule is expressed in terms of a *weighted* kernel expansion. The weighting associated with a data point is *inversely proportional*

Manuscript received December 5, 2005; revised October 6, 2006. This work was supported in part by the Norwegian Research Council Grant 171125/V30, in part by National Science Foundation (NSF) Grant ECS-0300340, and in part by NSF grant ECS-0524835. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. David J. Miller.

R. Jenssen and T. Eltoft are with the Group of Electrical Engineering, Department of Physics and Technology, University of Tromsø, N-9037 Tromsø, Norway (e-mail: robertj@phys.uit.no; pcte@phys.uit.no).

D. Erdogmus was with the Computational NeuroEngineering Laboratory, University of Florida, Gainesville, FL 32611 USA. He is now with the Computer Science and Electrical Engineering Department and the Biomedical Engineering Department, Oregon Health and Science University, Beaverton, OR 97006 USA (e-mail: derdogmus@ieee.org).

J. C. Principe is with the Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: principe@cnel.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2007.894391

to the probability density at that point, emphasizing the least probable regions. No optimization is needed to determine the weighting scheme, only the evaluation of sums of kernels. This results in an $O\left(N_{\text{train}}^2\right)$ procedure. The computational complexity in the testing phase is given by $O(N_{\text{test}}N_{\text{train}})$, since the kernel expansion in this case consists of N_{train} terms. We discuss how the complexity may be further reduced. The connection to Parzen windowing also provides a theoretical criterion for kernel size selection, which we show reduces the need for cross-validation. We show that the new classifier performs better than the Parzen window Bayes classifier, and in many cases comparable to the SVM, at a computationally lower cost.

Note that some other recent classifiers can also be expressed in terms of weighted kernel expansions, even though the theoretical basis for these methods may be quite different. Kernel Fisher discriminant analysis (KFDA) [11] for example, computes the Fisher discriminant in a Mercer kernel induced feature space as the solution to an optimization problem. Another example is Gaussian process classification (GPC) [12], which solves an optimization problem based on Bayesian inference. The idea here is to place a Gaussian process prior over the input to a sigmoid function which approximates the class probabilities given the data. We compare the Laplacian classifier performance also to these two classifiers.

The remainder of this paper is organized as follows. In Section II, we review the CS classification cost function. In Section III, a new classification rule based on the CS cost function is derived. Section V highlights some properties of the Laplacian classifier. Some classification results are reported in Section VI. We make our concluding remarks in Section VII.

II. Cauchy–Schwarz Cost Function

Initially, we focus the analysis on the two-class case. Consider two data classes, ω_1 and ω_2, corresponding to the probability density functions $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$. The classification cost function we introduce in this paper is based on the CS divergence measure between $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$, defined as [13]

$$D(p_1,p_2) = -\log \frac{\langle p_1,p_2\rangle_f}{\sqrt{\langle p_1,p_1\rangle_f \langle p_2,p_2\rangle_f}} \geq 0 \tag{1}$$

where $\langle p_i,p_j\rangle_f \equiv \int p_i(\mathbf{x})p_j(\mathbf{x})f^{-1}(\mathbf{x})d\mathbf{x}$, $i,j = 1,2$. Here, $f(\mathbf{x}) = P_1p_1(\mathbf{x}) + P_2p_2(\mathbf{x})$ is the overall pdf of the data set (P_1 and P_2 are the class priors). The CS measure is symmetric, and $0 \leq D(p_1,p_2) < \infty$. If $p_1(\mathbf{x}) = p_2(\mathbf{x})$, then $D(p_1,p_2) = 0$. The particular inner-product weighting by $f^{-1}(\mathbf{x})$ connects the CS cost function to the Bayes probability of error, as will be discussed next. It also connects the CS measure to the Laplacian data matrix, as will be discussed subsequently.

A. Connection to the Bayes Probability of Error

Note that since the logarithm is a monotonic function, we may just as well focus on the argument of the log in (1), which we may denote P_f. Let two regions, $\mathcal{R}_1$ and $\mathcal{R}_2$, in the data space be defined. If a test data point $\mathbf{x}_t \in \mathcal{R}_1$, it will be assigned to ω_1. If, on the other hand, $\mathbf{x}_t \in \mathcal{R}_2$, it will be assigned to ω_2. The regions $\mathcal{R}_1$ and $\mathcal{R}_2$ must be determined such that a classification cost is optimized. Let us assume that the two classes are relatively well separated. Hence, $f(\mathbf{x}_t) \approx P_1p_1(\mathbf{x}_t)$ for $\mathbf{x}_t \in \mathcal{R}_1$ and $f(\mathbf{x}_t) \approx P_2p_2(\mathbf{x}_t)$ for $\mathbf{x}_t \in \mathcal{R}_2$. Considering now the three terms which comprise P_f, we have

$$\begin{aligned}
\text{a)}\quad & \int \frac{p_1(\mathbf{x})p_2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} \\
&= \int_{\mathcal{R}_1} \frac{p_1(\mathbf{x})p_2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} + \int_{\mathcal{R}_2} \frac{p_1(\mathbf{x})p_2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} \\
&\approx \frac{1}{P_1}\int_{\mathcal{R}_1} p_2(\mathbf{x})d\mathbf{x} + \frac{1}{P_2}\int_{\mathcal{R}_2} p_1(\mathbf{x})d\mathbf{x}. \qquad (2)\\
\text{b)}\quad & \int \frac{p_1^2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} = \int_{\mathcal{R}_1} \frac{p_1^2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} + \int_{\mathcal{R}_2} \frac{p_1^2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} \\
&\approx \frac{1}{P_1}. \qquad (3)\\
\text{c)}\quad & \int \frac{p_2^2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} \\
&= \int_{\mathcal{R}_1} \frac{p_2^2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} + \int_{\mathcal{R}_2} \frac{p_2^2(\mathbf{x})}{f(\mathbf{x})}d\mathbf{x} \\
&\approx \frac{1}{P_2}. \qquad (4)
\end{aligned}$$

Finally, we obtain

$$P_f \approx \sqrt{\frac{P_1}{P_2}}\int_{\mathcal{R}_2} p_1(\mathbf{x})d\mathbf{x} + \sqrt{\frac{P_2}{P_1}}\int_{\mathcal{R}_1} p_2(\mathbf{x})d\mathbf{x}. \tag{5}$$

This expression can be compared to the expression for the Bayes probability or error, given by

$$P_e = P_1\int_{\mathcal{R}_2} p_1(\mathbf{x})d\mathbf{x} + P_2\int_{\mathcal{R}_1} p_2(\mathbf{x})d\mathbf{x}. \tag{6}$$

Hence, it can be seen that $P_f \approx (P_e/\sqrt{P_1P_2})$. Therefore, in this well separated situation, the CS classification cost function is connected to the Bayes error probability. The regions $\mathcal{R}_1$ and $\mathcal{R}_2$ which minimize P_e also minimize P_f. When the two classes are not well separated, it seems as if the CS cost function emphasizes more to classify correctly the least probable class, even though this may increase the error rate, as will be discussed in a later section.

B. Laplacian Matrix and Kernel Space Representation

In this section we briefly review the connection between a Parzen window-based estimator for the CS divergence and the Laplacian data matrix, as noted in [13].

Express P_f as

$$P_f = \frac{\int h_1(\mathbf{x})h_2(\mathbf{x})d\mathbf{x}}{\sqrt{\int h_1^2(\mathbf{x})d\mathbf{x}\int h_2^2(\mathbf{x})d\mathbf{x}}} \tag{7}$$

where $h_1(\mathbf{x}) = f^{-(1/2)}(\mathbf{x})p_1(\mathbf{x})$ and $h_2(\mathbf{x}) = f^{-(1/2)}(\mathbf{x})p_2(\mathbf{x})$. We are given a training data set $\mathbf{x}_l$, $l = 1,\ldots,N$. This data set consists of $\mathbf{x}_i$, $i = 1,\ldots,N_1$, the class one data points and $\mathbf{x}_j$, $j = 1,\ldots,N_2$, the class

two data points. Based on the data samples, define the Parzen window-based estimators

$$\hat{f}(\mathbf{x}) = \frac{1}{N}\sum_{l=1}^{N} W_\sigma(\mathbf{x},\mathbf{x}_l)$$
$$\hat{h}_1(\mathbf{x}) = \frac{1}{N_1}\sum_{i=1}^{N_1} \hat{f}^{-(1/2)}(\mathbf{x}_i) W_\sigma(\mathbf{x},\mathbf{x}_i)$$
$$\hat{h}_2(\mathbf{x}) = \frac{1}{N_2}\sum_{j=1}^{N_2} \hat{f}^{(-1/2)}(\mathbf{x}_j) W_\sigma(\mathbf{x},\mathbf{x}_j). \tag{8}$$

Here, $W_\sigma(\cdot,\cdot)$ is the Parzen window. It is often (but not necessarily) given by a unimodal Gaussian density, i.e.,

$$W_\sigma(\mathbf{x},\mathbf{x}_l) = \frac{1}{(2\pi\sigma^2)^{(d/2)}} \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_l\|^2}{2\sigma^2}\right\}. \tag{9}$$

For simplicity, we assume the same kernel size σ is used in all the Parzen estimators. It is easy to incorporate different kernel sizes in the derivation.

Using these estimators, we have

$$\begin{aligned}
&\int \hat{h}_1(\mathbf{x})\hat{h}_2(\mathbf{x})d\mathbf{x} \\
&= \int \frac{1}{N_1}\sum_{i=1}^{N_1} \frac{W_\sigma(\mathbf{x},\mathbf{x}_i)}{\hat{f}^{(1/2)}(\mathbf{x}_i)} \frac{1}{N_2}\sum_{j=1}^{N_2} \frac{W_\sigma(\mathbf{x},\mathbf{x}_j)}{\hat{f}^{(1/2)}(\mathbf{x}_j)} d\mathbf{x} \\
&= \frac{1}{N_1 N_2}\sum_{i,j=1}^{N_1,N_2} \frac{1}{\hat{f}^{(1/2)}(\mathbf{x}_i)\hat{f}^{(1/2)}(\mathbf{x}_j)} \\
&\quad \times \int W_\sigma(\mathbf{x},\mathbf{x}_i) W_\sigma(\mathbf{x},\mathbf{x}_j) d\mathbf{x} \\
&= \frac{1}{N_1 N_2}\sum_{i,j=1}^{N_1,N_2} \frac{W_{\sqrt{2}\sigma}(\mathbf{x}_i,\mathbf{x}_j)}{\hat{f}^{(1/2)}(\mathbf{x}_i)\hat{f}^{(1/2)}(\mathbf{x}_j)}
\end{aligned} \tag{10}$$

where in the last step, the convolution theorem for Gaussians has been employed. For any pair of data points in the data set, say $\mathbf{x}_l$ and $\mathbf{x}_{l'}$, we define the matrix $\mathbf{K}$ such that element (l,l') equals $W_{\sqrt{2}\sigma}(\mathbf{x}_l,\mathbf{x}_{l'})$. Moreover, a matrix $\mathbf{D} = \mathrm{diag}(\hat{f}(\mathbf{x}_1),\ldots,\hat{f}(\mathbf{x}_N))$ is defined. Now, all $\hat{f}^{-(1/2)}(\mathbf{x}_l) W_{\sqrt{2}\sigma}(\mathbf{x}_l,\mathbf{x}_{l'}) \hat{f}^{-(1/2)}(\mathbf{x}_{l'})$ can be represented by element (l,l') of the matrix $\mathbf{K}_f = \mathbf{D}^{-(1/2)}\mathbf{K}\mathbf{D}^{-(1/2)}$. The matrix $\mathbf{K}_f$ is known as the Laplacian matrix [9].

Note that each element of $\mathbf{K}$ represents an inner-product in a Mercer kernel feature space, since the Gaussian Parzen window obeys the Mercer conditions [6], [7], [14]. Hence, each element of the Laplacian matrix, $\mathbf{K}_f$, also represents an inner-product, which we may denote

$$\langle \mathbf{\Phi}_f(\mathbf{x}_l), \mathbf{\Phi}_f(\mathbf{x}_{l'}) \rangle = \hat{f}^{-(1/2)}(\mathbf{x}_l) W_{\sqrt{2}\sigma}(\mathbf{x}_l,\mathbf{x}_{l'}) \hat{f}^{-(1/2)}(\mathbf{x}_{l'}). \tag{11}$$

Thus

$$\begin{aligned}
&\int \hat{h}_1(\mathbf{x})\hat{h}_2(\mathbf{x})d\mathbf{x} \\
&= \frac{1}{N_1 N_2}\sum_{i,j=1}^{N_1,N_2} \langle \mathbf{\Phi}_f(\mathbf{x}_i), \mathbf{\Phi}_f(\mathbf{x}_j) \rangle \\
&= \left\langle \frac{1}{N_1}\sum_{i=1}^{N_1} \mathbf{\Phi}_f(\mathbf{x}_i), \frac{1}{N_2}\sum_{j=1}^{N_2} \mathbf{\Phi}_f(\mathbf{x}_j) \right\rangle \\
&= \langle \mathbf{m}_{1_f}, \mathbf{m}_{2_f} \rangle
\end{aligned} \tag{12}$$

where $\mathbf{m}_{1_f}$ and $\mathbf{m}_{2_f}$ are the class mean vectors of the data after the mapping to the *Mercer kernel induced feature space*. Here $\mathbf{m}_{1_f} = (1/N_1)\sum_{j=1}^{N_1} \mathbf{\Phi}_f(\mathbf{x}_i)$, $\mathbf{m}_{2_f} = (1/N_2)\sum_{i=1}^{N_2} \mathbf{\Phi}_f(\mathbf{x}_j)$, where the nonlinear mapping is given by $\mathbf{\Phi}_f(\cdot)$.

By performing an exactly similar analysis for $\int \hat{h}_1^2(\mathbf{x})d\mathbf{x}$ and $\int \hat{h}_2^2(\mathbf{x})d\mathbf{x}$, such a Parzen window estimate of P_f can be expressed as

$$\hat{P}_f \equiv \mathrm{LIC} = \cos\angle(\mathbf{m}_{1_f}, \mathbf{m}_{2_f}) \tag{13}$$

where LIC stands for *Laplacian information cut* because of a connection to the graph cut [15]. Hence, the CS divergence measure turns out to correspond to a measure of the *cosine of the angle* between class mean vectors in a kernel induced feature space related to the Laplacian data matrix.

C. Multiclass CS Measure

The CS divergence can easily be extended to more than two classes, given by

$$D(p_1,\ldots,p_C) = -\log \sum_{i=1}^{C-1}\sum_{j>i} \frac{\langle p_i, p_j \rangle_f}{\kappa\sqrt{\langle p_i, p_i \rangle_f \langle p_j, p_j \rangle_f}} \tag{14}$$

where $\kappa = \sum_{c=1}^{C-1} c$. In the Mercer kernel feature space, this corresponds to a multiway Laplacian information cut, as follows

$$\mathrm{LIC} = \sum_{i=1}^{C-1}\sum_{j>i} \frac{1}{\kappa} \cos\angle(\mathbf{m}_{i_f}, \mathbf{m}_{j_f}). \tag{15}$$

Thus, the LIC measures the sum of pairwise cosines between class mean vectors.

III. New Classification Rule

Based on the training data set, we may define the class mean vectors, $\mathbf{m}_{1_f},\ldots,\mathbf{m}_{C_f}$, corresponding to the classes $\omega_1,\ldots,\omega_C$. We wish to classify a test data point $\mathbf{x}_t$ to the class which minimizes the CS classification cost function. This can obviously be achieved by measuring the angle between $\mathbf{\Phi}_f(\mathbf{x}_t)$ and each of the mean vectors, for then to assign the data point to the class for which the angle is the smallest. This corresponds to the following classification rule

$$\mathbf{x}_t \to \omega_c : \max_c \cos\angle(\mathbf{\Phi}_f(\mathbf{x}_t), \mathbf{m}_{c_f}) \tag{16}$$

$c = 1, \ldots, C$. Each $\cos \angle(\mathbf{\Phi}_f(\mathbf{x}_t), \mathbf{m}_{c_f})$ may be evaluated as follows:

$$\begin{aligned}
&\cos \angle(\mathbf{\Phi}_f(\mathbf{x}_t), \mathbf{m}_{c_f}) \\
&= \frac{\mathbf{\Phi}_f^T(\mathbf{x}_t)\mathbf{m}_{c_f}}{\sqrt{\|\mathbf{\Phi}_f(\mathbf{x}_t)\|^2 \|\mathbf{m}_{c_f}\|^2}} \\
&= \frac{\frac{1}{N_c}\sum_{i=1}^{N_c} \mathbf{\Phi}_f^T(\mathbf{x}_t)\mathbf{\Phi}_f(\mathbf{x}_i)}{\sqrt{\|\mathbf{\Phi}_f(\mathbf{x}_t)\|^2 \frac{1}{N_c^2}\sum_{j,j'=1}^{N_c,N_c} \mathbf{\Phi}_f^T(\mathbf{x}_j)\mathbf{\Phi}_f(\mathbf{x}_{j'})}} \\
&= \frac{\sum_{i=1}^{N_c} \hat{f}^{-(1/2)}(\mathbf{x}_t) W_\sigma(\mathbf{x}_t, \mathbf{x}_i) \hat{f}^{-(1/2)}(\mathbf{x}_i)}{\|\mathbf{\Phi}_f(\mathbf{x}_t)\| \sqrt{\sum_{j,j'=1}^{N_c,N_c} \hat{f}^{-(1/2)}(\mathbf{x}_j) W_\sigma(\mathbf{x}_j, \mathbf{x}_{j'}) \hat{f}^{-(1/2)}(\mathbf{x}_{j'})}}.
\end{aligned} \tag{17}$$

In (17), $\hat{f}^{-(1/2)}(\mathbf{x}_t)$ and $\|\mathbf{\Phi}_f(\mathbf{x}_t)\|$ will be common for all $\cos \angle(\mathbf{\Phi}_f(\mathbf{x}_t), \mathbf{m}_{c_f})$, $c = 1, \ldots, C$. Therefore, we define the test statistic γ_c, where

$$\gamma_c = \frac{\sum_{i=1}^{N_c} \hat{f}^{-(1/2)}(\mathbf{x}_i) W_{\sqrt{2}\sigma}(\mathbf{x}_t, \mathbf{x}_i)}{V_c} \tag{18}$$

where the constant

$$V_c = \sqrt{\sum_{j,j'=1}^{N_c,N_c} \hat{f}^{-(1/2)}(\mathbf{x}_j) W_{\sqrt{2}\sigma}(\mathbf{x}_j, \mathbf{x}_{j'}) \hat{f}^{-(1/2)}(\mathbf{x}_{j'})}. \tag{19}$$

The resulting classification rule therefore becomes $\mathbf{x}_t \rightarrow \omega_c : \max_c \gamma_c$, $c = 1, \ldots, C$.

We name the classifier we have derived *the Laplacian classifier*, because of the connection to the Laplacian matrix and its eigenspectrum (see also Section V-A).

Observe that the constants $\hat{f}^{-(1/2)}(\mathbf{x}_i)$ in (18) induce a *weighting* of the importance of $W_{\sqrt{2}\sigma}(\mathbf{x}_t, \mathbf{x}_i)$, the kernel function at $\mathbf{x}_i$. The weighting factor is *inversely proportional* to the value of the overall pdf estimated at $\mathbf{x}_i$. Hence, a data point $\mathbf{x}_i$ for which the value $\hat{f}(\mathbf{x}_i)$ is high, corresponds to a low weighting of the corresponding kernel function. This implies that training data points corresponding to regions of low probability density are given higher weights in the classifier design. The constant V_c acts as a normalization.

The Laplacian classifier weights are obtained without having to solve any complex optimization problem, rather only the evaluation of sums of kernel functions is needed. Evaluating all these sums is an $O(N_{\text{train}}^2)$ procedure. However, further complexity reduction is possible. Stochastic sampling ideas may be employed, as advocated in a related setting in [16]. Using stochastic sampling, only a subset of the training data points are used in the Parzen window estimation procedure. This reduces the training phase complexity to $O(N_{\text{subset}} N_{\text{train}})$, $N_{\text{subset}} \ll N_{\text{train}}$. Another possibility is to evaluate the kernel expansions with less computational complexity using the fast Gauss transform [17], as demonstrated on a similar problem in [18]. This reduces the complexity to $O(pkN_{\text{train}})$ per data dimension, where k is the number of clusters used in the approximation and p is the truncation order. This may be important, especially for large N_{train}. This is an issue for further study in future work.

A. Connection to Parzen Window Bayes Classifier

The Laplacian classifier contains the Bayes classifier as a special case, where the class densities are estimated using the Parzen window technique. Recall the Parzen window Bayes classifier

$$\mathbf{x}_t \rightarrow \omega_c : \max_c \sum_{i=1}^{N_c} W_\sigma(\mathbf{x}_t, \mathbf{x}_i) \tag{20}$$

$c = 1, \ldots, C$. Consider the case where no assumptions are made about $f(\mathbf{x})$, such that $f(\mathbf{x}) \equiv 1$, $\forall\, \mathbf{x}$, and it is assumed that $V_1 = \cdots = V_C$. In that case, the Laplacian classifier reduces to the Parzen window Bayes classifier.

IV. Kernel Size Selection

The Laplacian classifier is explicitly derived using Parzen windowing. Therefore, in theory an optimal way for selecting σ for Parzen window density estimation would provide the appropriate kernel size to be used in the classifier. Parzen kernel size selection has been thoroughly studied in the statistics literature [19]–[21]. The optimal kernel size is usually selected so as to minimize the mean integrated squared error (MISE) between $\hat{f}(\mathbf{x})$ and the target density $f(\mathbf{x})$. It is easily shown [19]–[21] that the MISE decomposes into a bias term and a variance term. For a fixed sample size, the bias term is minimized by minimizing the kernel size, while the variance term is minimized by maximizing the kernel size. This is the inherent bias-variance tradeoff in the Parzen window technique.

A simple formula for (suboptimal) MISE kernel-size selection is given by Silverman's rule [19]

$$\hat{\sigma} = \sigma_X \left[\frac{4}{(2d+1)N}\right]^{(1/d+4)} \tag{21}$$

where $\sigma_X^2 = d^{-1}\sum_i \mathbf{\Sigma}_{X_{ii}}$, and $\mathbf{\Sigma}_{X_{ii}}$ are the diagonal elements of the sample covariance matrix. We will use this rule as an initial value for the kernel size.

V. Properties of the New Classifier

A. Measuring Angles in Kernel Space

We wish to illustrate by an example that a classification cost function which is based on the angle between a test data point in the kernel feature space and the class mean vectors makes sense. In order to achieve this, we will in this example *approximate* the nonlinear data mapping from input space to the Mercer kernel feature space. Such an approximation to $\mathbf{\Phi}_f(\cdot)$ may be accomplished using the C largest eigenvalues and corresponding eigenvectors of the matrix $\mathbf{K}_f$, as follows [22]–[24]:

$$\mathbf{\Phi}_f(\mathbf{x}_l) \approx \left[\sqrt{\lambda_1} e_{1l}, \ldots, \sqrt{\lambda_C} e_{Cl}\right]^T \tag{22}$$

where e_{ml} denotes the lth element of the mth eigenvector of $\mathbf{K}_f$ and λ_m is the corresponding eigenvalue, where $\lambda_1 \geq \ldots \geq \lambda_C$.

This kind of approximation have been used e.g., in spectral clustering methods [25] and in Laplacian eigenmaps [26]. In [27] this data mapping was utilized in a classifier procedure

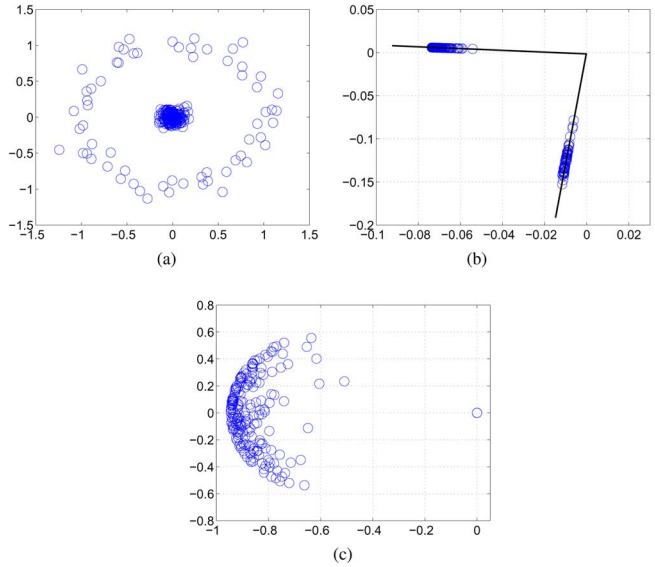


Fig. 1. Approximating a Mercer kernel feature space using the eigenspectrum of the Laplacian matrix. The CS divergence, corresponding to the cosine of the angle between cluster mean vectors in the kernel feature space, clearly makes sense in this example. (a) Two-class ring-shaped data set. (b) Laplacian matrix Kernel feature space mapping. (c) Kernel PCA feature space mapping.

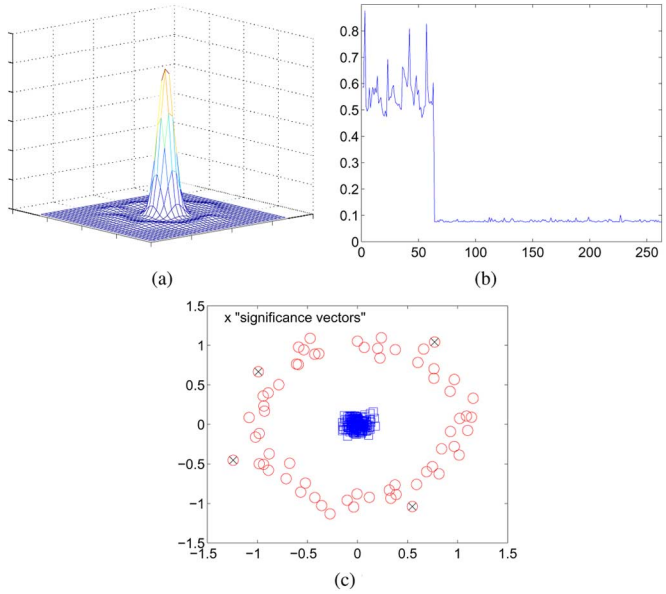


Fig. 2. Illustrating the probability weighting scheme. (a) Parzen window pdf estimate. (b) Weighting for ring-shaped data set. (c) The vectors corresponding to the largest weights may be called "significance vectors."

based on the CS measure. Notice also that in the case where no weighting is performed, i.e., $f(\mathbf{x}) \equiv 1, \forall\, \mathbf{x}$ in (1), the mapping (22) reduces to the kernel principal component analysis (kernel PCA) mapping, as derived in [28]. It can be shown that in the "ideal case," where the clusters are "infinitely" far apart, the data mapping (22) results in C point-clusters, mutually orthogonal to each other, situated on the C first principal axes in the kernel space. This would be the case both using the Laplacian matrix, $\mathbf{K}_f$, and the kernel PCA matrix, $\mathbf{K}$. However, in practice, the mapping (22) based on the Laplacian matrix $\mathbf{K}_f$ is different that the kernel PCA mapping.

Fig. 1(a) shows a two-class ring-shaped data set. This may be considered the training data set available. Notice that the class mean vectors for this data set are identical. Using (21) to determine the Parzen window width, σ, and hence the Mercer kernel size, $\tilde{\sigma} = \sqrt{2}\sigma$ by (10), we construct the Laplacian matrix using (11). By computing the eigenvalues and eigenvectors of $\mathbf{K}_f$, we may approximate the mapping to the kernel feature space using (22), obtaining the data set shown in Fig. 1(b). The data structure in the kernel feature space is clearly different to the structure in the original data space. The transformed data points, corresponding to the two classes, are distributed along two lines radially from the origin, in *different angular directions*. It clearly makes sense in this example to classify a *test data point* in the kernel feature space to the class represented by the mean vector closest to it, using an angular measure. However, the reader should bear in mind that in this paper, we do not actually use (22) in the training and testing phases of the Laplacian classifier. Hence, no eigenvalue problem needs to be solved in our procedure. For completeness, we also show in Fig. 1(c) the kernel PCA mapping (based on the affinity matrix, i.e., no weighting) obtained on this data set, using the same kernel size. The transformed data points corresponding to the ring are those which are concentrated near the origo, while the other data points correspond to the Gaussian cluster. A classification based on an angluar measure is no longer appropriate. This shows that the weighting property associated with the Laplacian matrix has a significant effect on the mapping.

B. Probability Weighting Scheme

Using the same ring-shaped data set, we analyze the weighting $\hat{f}^{-(1/2)}(\mathbf{x}_l)$ associated with each data point $\mathbf{x}_l$, $l = 1, \ldots, N$. The data set is constructed such that the first 63 data points correspond to the ring, while the last 200 data points correspond to the Gaussian class in the center. Fig. 2(a) shows the Parzen window pdf estimate obtained using (21) to determine the window width. Fig. 2(b) shows a plot of the resulting weighting constants. The data points corresponding to the ring experiences a higher weighting, since they are situated in low probability density regions in the input space. The data points comprising the dense Gaussian class experiences a low weighting because of the high probability density in that region. The most "important" data points in the classifier design are those having high weights. These may be denoted "significance vectors." Fig. 2(c) shows the top one percent "significance vectors" (marked by x), where also the class labels are indicated (squares and circles). The "significance vectors" are in this case evenly distributed in the low density regions in the input data space.

Note that for many data sets, low values for the overall probability density function will correspond to class boundary regions. It makes sense that training data points situated near class boundaries should be emphasized more in the classifier design, since test data points situated near the class boundaries are the most difficult to classify correctly.

C. Robustness Against Outliers

We have argued that the Laplacian classifier assigns a weight to each data point in the training set, which is inversely propor-

tional to the value of the Parzen window estimated probability density at that point. This raises the question of whether *outliers* may dominate the classification, since they will be assigned very high weights.

Assume that class c consists of N_c data points plus an outlier. For simplicity, we denote the weighting for data point $\mathbf{x}_i$, i.e., $\hat{f}^{-(1/2)}(\mathbf{x}_i)$, by α_i. Also, we denote the kernel $W_{\sqrt{2}\sigma}(\mathbf{x}_i, \mathbf{x}_{i'})$ by $W_{ii'}$ for any $\mathbf{x}_i$ and $\mathbf{x}_{i'}$. Let $\mathbf{x}_o$ be the outlier, and let α_o denote its weight. The test statistic, (18), for class c, with the outlier appended, is thus given by

$$\gamma_c = \frac{\sum_{i=1}^{N_c} \alpha_i W_{ti} + \alpha_o W_{to}}{\sqrt{\sum_{j,j'=1}^{N_c,N_c} \alpha_j W_{jj'} \alpha_{j'} + \sum_{k=1}^{N_c} 2\alpha_o W_{ok} \alpha_k}}. \tag{23}$$

The normalizing constant V_c is expressed by the denominator of this expression. Let $\psi = \sum_{k=1}^{N_c} 2\alpha_o W_{ok}\alpha_k$. We note that ψ is independent of the test data point $\mathbf{x}_t$. If we assume that all the training data point weights α_k, $k = 1, \ldots, N_c$ are approximately similar, we may approximate the α_k's by the constant $\tilde{\alpha}$. The quantity $\psi \approx 2\tilde{\alpha}\alpha_o \sum_{k=1}^{N_c} W_{ok}$, and since $\alpha_o = 1/\sqrt{(1/N_c)\sum_{k=1}^{N_c} W_{ok}}$, we have

$$\psi \approx 2\tilde{\alpha} N_c \sqrt{\frac{1}{N_c}\sum_{k=1}^{N_c} W_{ok}} = \frac{2\tilde{\alpha} N_c}{\alpha_o}. \tag{24}$$

Hence, if the outlier weight α_o is very high, ψ will have a limited effect on the normalizing constant V_c associated with class c.

Let $\xi = \alpha_o W_{to}$. If the test data point $\mathbf{x}_t$ is close to $\mathbf{x}_o$, such that W_{to} will be large, then consequently ξ will attain a large value. Hence, γ_c will be large, and the test data point will probably be assigned to class c. This makes sense. However, if $\mathbf{x}_o$ is truly an outlier, $\mathbf{x}_t$ is more likely to be situated near the training data points. Hence, the proximity measure W_{to} will be comparable to the proximities W_{ok}, $k = 1, \ldots, N_c$, comprising $\alpha_o = 1/\sqrt{(1/N_c)\sum_{k=1}^{N_c} W_{ok}}$. These all have small values. Thus, by approximating these proximities by a constant $\tilde{W}$, we obtain

$$\xi \approx \frac{\tilde{W}}{\sqrt{\frac{1}{N_c}\sum_{k=1}^{N_c} \tilde{W}}} = \frac{\tilde{W}}{\sqrt{\frac{1}{N_c} N_c \tilde{W}}} = \sqrt{\tilde{W}} \tag{25}$$

which is a small value.

These derivations show that an outlier, or a few outliers, will not dominate the Laplacian classifier, despite high outlier weights. The larger the distance from the outlier of class c to the training data set comprising that class, the smaller effect it will have on the normalizing constant V_c. A test data point close to the outlier will produce a high value for the test statistic γ_c. But the value for γ_c will not be much affected by the outlier if the test data point is located in the proximity of the training data points.

These theoretical results are supported by our experimental experience. A simple example is included for illustration purposes. Fig. 3(a) shows a training data set. This is the exact same

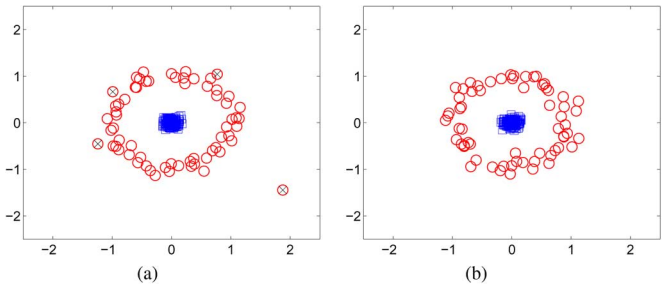


Fig. 3. Illustrating robustness against outliers. The outlier appended to the outermost ring does not dominate the Laplacian classifier performance. (a) Training data set with outlier. (b) Test data data set correctly classified.

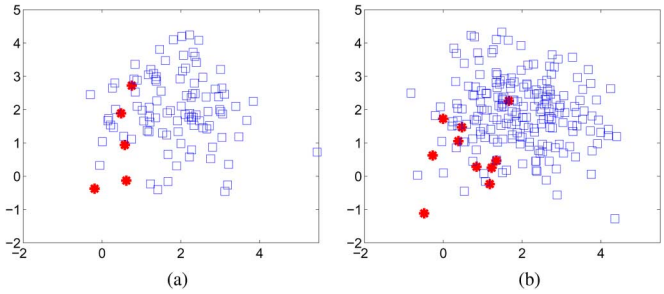


Fig. 4. Data set consisting of two Gaussian classes. (a) Training data ser to for two-Gaussians case. (b) Test data set for two-Gaussians case.

data set as shown in Fig. 2(c), only that in this case an *outlier* is appended to the outermost ring. The top one percent "significance vectors" [see Fig. 2(c)] includes the outlier. The weight assigned to each data point, except the outlier, are exactly the same as those weights shown in Fig. 2(b). However, the outlier weight is very large; $\alpha_o = 47112$. Fig. 3(b) shows the classification result for a test data set generated in a similar manner as the training data set. The classification is not dominated by the outlier. The same result is obtained also if the outlier belonged to the Gaussian class in the middle instead. Other classification experiments seem to support this result.

D. Emphasizing Least Probable Class

In Section II-A we showed that when the classes are relatively well separated, the CS cost function has a connection to the Bayes error probability. Thus, in such a situation the Laplacian classifier minimizes the probability of error.

Consider now a data set where the classes are not well separated, rather they have significant overlap. Fig. 4(a) shows such a data set. Both classes originates from Gaussian distributions. These distributions have the same spherical covariance structure with unit variance. The mean vector in the input space of class one is $\mu_1 = [2 \quad 2]^T$. The mean vector of the second class is $\mu_2 = [0.6 \quad 0.6]^T$. The training data is constructed such that class one is represented by 100 data points, compared to only 5 data points from class two. Hence, $P_1 \approx 0.95$, while $P_2 \approx 0.05$. Based on this data set, the new classifier is trained. For comparison, we construct a Parzen window Bayes classifier and also train a SVM. Recall that the Bayes classifier is in theory optimal with respect to the probability of error. The test data set is drawn from the same distributions as the training data set. It consists of 200 data points from class one, and 10 data points from class two. This data set is shown in Fig. 4(b), with the true

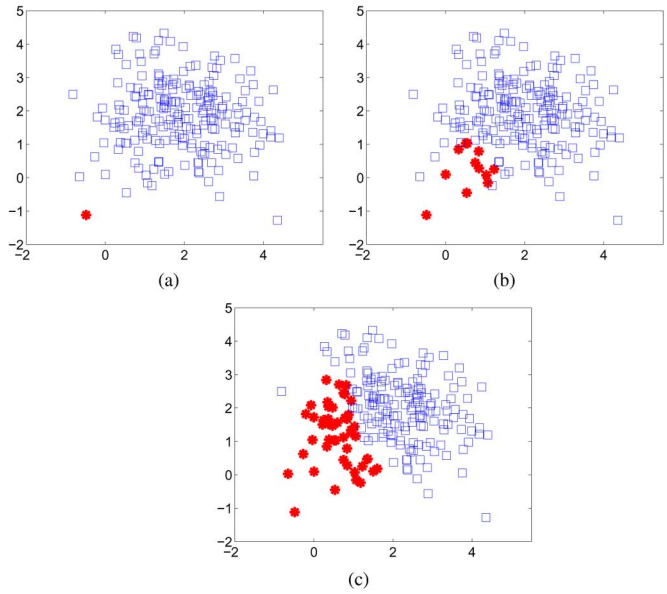


Fig. 5. SVM and Laplacian classifier result on the data set consisting of two Gaussian classes. (a) Parzen window Bayes Classifier. (b) Support vector machine classifier. (c) Laplacian classifier.

TABLE I
CONFUSION MATRIX FOR PARZEN WINDOW BAYES CLASSIFIER

	$\hat{\omega}_1$	$\hat{\omega}_2$
ω_1	1	9
ω_2	0	200

TABLE II
CONFUSION MATRIX FOR SUPPORT VECTOR MACHINE

	$\hat{\omega}_1$	$\hat{\omega}_2$
ω_1	3	7
ω_2	9	191

TABLE III
CONFUSION MATRIX FOR LAPLACIAN CLASSIFIER

	$\hat{\omega}_1$	$\hat{\omega}_2$
ω_1	9	1
ω_2	38	162

labels indicated. Because of the overlap, classification errors are unavoidable. The classification results we show subsequently in Fig. 5(a)–(c) should be compared to Fig. 4(b), as it represents the "ground-truth" in this example.

We use (21) to determine the kernel size. The classification result using the Parzen window Bayes classifier is shown in Fig. 5(a). Although it misclassifies only 9 data points, it only classifies correctly one single class two data point. This is to be expected, since the Bayes classifier will sacrifice the least probable class in order to achieve a low classification error. The confusion matrix summarizing this result is shown in Table I.

Fig. 5(b) shows the result obtained using a SVM. The confusion matrix is shown in Table II. The SVM classifies correctly 3 class two data points, and in fact obtains more errors overall than the Parzen window Bayes classifier.

The Laplacian classifier obtains the result shown in Fig. 5(c), corresponding to the confusion matrix shown in Table III. The result is significantly different from the results obtained by the two previous classifiers. It classifies correctly 9 of the class two

TABLE IV
CONFUSION MATRIX FOR PARZEN WINDOW BAYES CLASSIFIER ON ECHOCARDIOGRAM DATA

	$\hat{\omega}_1$	$\hat{\omega}_2$
ω_1	2	10
ω_2	2	28

TABLE V
CONFUSION MATRIX FOR LAPLACIAN CLASSIFIER ON ECHOCARDIOGRAM DATA

	$\hat{\omega}_1$	$\hat{\omega}_2$
ω_1	9	3
ω_2	10	20

data points. However, the penalty is 38 erroneously classified class one data points. Clearly, the Laplacian classifier emphasizes more *to classify correctly the least probable class*. This property may be useful in many applications, and should be further studied in future work. Note that a similar effect may be generated in the Bayes classifier, and probably in the SVM also, by weighting the importance of the different classes. This requires the user to determine the relative importance of the classes.

As an example of the different emphasis between the Parzen window Bayes classifier and the Laplacian classifier, we consider an echocardiogram data set, obtained from the UCI repository [29]. Each exemplar contains information collected from patients who have had a recent heart attack. The problem is to determine if the patient will survive for one year following their heart attack. We have generated a training data set consisting of 10 data points. Three of these correspond to surviving patients. There are eight features, corresponding to age and various measurements of the heart condition. The classes have considerable overlap. The test data set consist of 42 data points. In this case, the confusion matrix based on the Parzen window Bayes classifier is shown in Table IV. The confusion matrix based on the Laplacian classifier is shown in Table V. The number of errors committed by the two classifiers are almost the same, but the confusion matrices are different. Again, the Laplacian classifier emphasizes more to classify correctly the least probable class.

VI. MORE CLASSIFICATION RESULTS

In this section, we report some classification results on 11 different well-known data sets, using the Laplacian classifier.

First, we consider four of the Rätsch data sets used in [30]: `Banana` (2, 400, 4800), `Thyroid` (5, 140, 75), `Image` (18, 1300, 1010), and `Twonorm` (20, 400, 4600), where the numbers in parenthesis are the dimensionality, the size of the training data set and the size of the test data set, respectively. There are between 20 and 100 realizations of each data set. The data sets have zero mean and unit standard deviation for each feature. We present results using a kernel size given by $\tilde{\sigma}_{\text{mise}} = \sqrt{2}\sigma_{\text{mise}}$, where σ_{mise} is obtained using (21), but also a kernel size given by $\tilde{\sigma}_{cv}$, obtained using three-fold cross-validation over a range of kernel sizes. We compare with the SVM results presented in [30]. In that paper, the SVM was trained using five-fold cross-validation on *both* SVM parameters C and $\tilde{\sigma}$. The results are summarized in Table VI (mean correct classification rate in percent and corresponding standard deviation).

Perhaps surprisingly, the simpler Laplacian classifier obtains results which are comparable and even better in some cases than

TABLE VI
CLASSIFICATION OF RÄTSCH DATA

Data	$\tilde{\sigma}_{cv}$ *Laplacian*	Rätsch *SVM*	$\tilde{\sigma}_{mise}$ *Laplacian*
Banana	$\mathbf{89.4 \pm 0.5}$	89.2 ± 0.7	88.4 ± 0.6
Thyroid	$\mathbf{95.7 \pm 2.2}$	95.2 ± 2.2	94.4 ± 2.6
Image	96.6 ± 0.6	$\mathbf{97.0 \pm 0.6}$	96.2 ± 0.7
Twonorm	$\mathbf{97.4 \pm 0.2}$	97.0 ± 0.2	97.3 ± 0.3

TABLE VII
CLASSIFICATION OF UCI REPOSITORY DATA

Data	$\tilde{\sigma}_{cv}$ *Laplacian*	$\tilde{\sigma}_{cv}$ *SVM*	$\tilde{\sigma}_{mise}$ *Laplacian*	$\tilde{\sigma}_{mise}$ *SVM*
Wine	97.3 ± 1.4	$\mathbf{97.5 \pm 1.6}$	$\mathbf{96.0 \pm 1.7}$	58.5 ± 11.4
Iris	94.5 ± 2.1	$\mathbf{95.7 \pm 2.0}$	$\mathbf{92.6 \pm 3.6}$	92.4 ± 2.7
Ionos	92.5 ± 1.7	$\mathbf{94.1 \pm 1.2}$	$\mathbf{91.9 \pm 1.7}$	72.1 ± 10.6
WBC	$\mathbf{97.1 \pm 0.7}$	96.9 ± 0.7	$\mathbf{96.9 \pm 0.7}$	94.6 ± 1.1
Pima	73.9 ± 1.7	$\mathbf{76.8 \pm 1.5}$	$\mathbf{73.3 \pm 1.7}$	64.6 ± 1.7
Pen	98.9 ± 0.4	$\mathbf{99.6 \pm 0.2}$	96.6 ± 0.8	$\mathbf{98.3 \pm 0.5}$

the much more complex SVM classifier. Note also that the classification results obtained using (21) to determine the classifier kernel size is quite close to the best results obtained after cross-validation. This indicates that $\tilde{\sigma}_{\text{mise}} = \sqrt{2}\sigma_{\text{mise}}$ is close to $\tilde{\sigma}_{cv}$. Hence, instead of performing cross-validation over a wide range of kernel sizes, this result may show that $\tilde{\sigma}_{\text{mise}}$ can be used as a starting point in the cross-validation procedure, close to the desirable solution. This may be important from a computational complexity perspective.

For additional comparison, we also provide the KFDA classification results reported in [11], on the same four data sets. The rates for KFDA are 89.2 ± 0.5 (`Banana`), 95.2 ± 0.6 (`Thyroid`), 95.8 ± 2.1 (`Image`), and 97.4 ± 0.2 (`Twonorm`). Hence, the Laplacian classifier performs comparably also to this classifier. We note that the Parzen window Bayes classifier performs consistently worse than both the Laplacian classifier, the SVM and the KFDA. For example, for `Banana` it obtains a best (cross-validation) classification rate of 87.4 ± 0.6, while for `Thyroid` it obtains 93.4 ± 2.6.

Next, we focus on some data sets extracted from the UCI [29] repository: `Wine` (13, 178), `Iris` (4, 150), `Ionos` (34, 351), Wisconsin breast-cancer (`WBC`) (9, 683), `Pima` (8, 768), and Pen -based handwritten digit recognition (16, 1091), where the numbers in parenthesis indicate the dimensionality and the size of the data set. The Pen data set is based on the integers 0, 1, and 2. The standard deviation of these data sets are also normalized to one for each feature, since the classifiers use spherical kernel functions. In this case, over 100 trials, we randomly split the data sets into two halves, one for training and the other for testing. Table VII shows the classification results for the Laplacian classifier and the SVM.[1] We have used three-fold cross-validation.

Using cross-validation (on two parameters in the SVM case), the SVM obtains slightly better results than the Laplacian classifier. The results are very sensitive to the tradeoff between training data misclassification and classifier regularization, regulated by the constant C, so cross-validation over both SVM parameters are required. It should be noted that the results obtained by the Laplacian classifier are quite close to those obtained by the SVM, without requiring an optimization procedure, and only performing cross-validation over one parameter.

[1]SVM software downloaded from [31] and [32].

Note also that the results obtained using $\tilde{\sigma}_{\text{mise}}$ are comparable to the cross-validation results for the Laplacian classifier. A kernel size given by $\tilde{\sigma}_{\text{mise}}$ does in general not produce very good results using the SVM. For completeness, we list the kernel sizes used for the different data sets ($\tilde{\sigma}_{\text{mise}}$, Laplacian classifier $\tilde{\sigma}_{cv}$, SVM $\tilde{\sigma}_{cv}$) as `Wine` (1.0, 2.3, 3.0), `Iris` (0.7, 0.4, 3.0), `Ionos` (1.1, 1.4, 2.5), `WBC` (0.7, 1.1, 10), `Pima` (0.7, 1.1, 10), and `Pen` (0.8, 0.3, 3.0).

We also provide the (best) results reported in [33] for some of the data sets in Table VII, using a Gaussian process classifier.[2] These are (correct classification in percent) 92.0 (`Ionos`), 96.8 (`WBC`), and 77.4 (`Pima`). Hence, the Laplacian classifier performs comparably also to a Gaussian process classifier. The Parzen window Bayes classifier consistently performs worse than the other classifiers. For example the classification rate for Ionos is 83.4 ± 3.7 and for `Wine` it is 95.8 ± 2.4.

We also conduct an experiment on the 10-class USPS data set.[3] The USPS data set consists of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service [34]. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in (16×16) grayscale images. The images are represented by 256-dimensional vectors of grayscale values. There are 7291 training observations and 2007 test observations. This data set is known to be notoriously difficult. We obtain 94% classification accuracy using the Laplacian classifier. In [35], 96% accuracy was reported using the SVM.

VII. CONCLUSION

We have derived a novel classifier which operates in a Mercer kernel feature space defined by the eigenspectrum of the Laplacian data matrix. In that space, the classification rule is based on comparing angles between test data points and class mean vectors. The classification cost function is derived from the CS divergence between pdf's, in combination with Parzen window density estimation.

The classification rule is expressed in terms of a weighted kernel expansion. The weighting associated with each data point is inversely proportional to the probability density function at that point, emphasizing the least probable regions. No optimization is needed to determine the weighting constants. The connection to Parzen windowing provides a theoretical criterion for kernel size selection. It has been shown that $\tilde{\sigma}_{\text{mise}}$ provides a good starting point for the kernel size cross-validation procedure.

The Laplacian classifier may be a useful alternative to the SVM, especially for large data sets, where the computationally complex SVM optimization and cross-validation is undesirable. The Laplacian classifier has a lower theoretical computational complexity in the training phase than the SVM. We have also discussed possibilities for further complexity reduction using heuristics, such as stochastic sampling and the use of the fast Gauss transform. The Laplacian classifier is therefore a much simpler classifier than the SVM, but which still

[2]Note that there may be some differences in preprocessing for the data used in [33], compared to this paper.

[3]Downloaded from http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html

produces comparable classification results in many cases. However, the SVM has the desirable property that its computational complexity in the testing phase depends on the number of support vectors, i.e., it exhibits a sparseness property. We believe a similar effect may be produced in the Laplacian classifier, by keeping only the largest "significance vectors" for each class in the kernel expansion comprising the classifier. For example, in preliminary experiments, we have obtained promising results by keeping only the 10%–20% largest weights for each class in (18), hence reducing the testing phase computational complexity significantly. This is, however, an issue which needs further attention in future work.

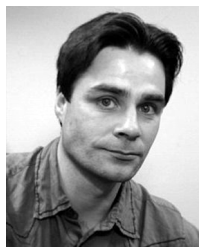
The Laplacian classifier has been shown to minimize the Bayes error probability for relatively well-separated data classes. For data classes which experience a high degree of overlap, the Laplacian classifier seems to have the interesting property that it emphasizes to classify correctly the least probable class, as opposed to the Parzen window Bayes classifier. This may come at the cost of a higher error rate. We consider this as an interesting property, which may be useful in cases where one class is underrepresented compared to the others, for example in some types of medical research.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the four anonymous reviewers for detailed comments which provided new insight and helped improve the paper.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification and Scene Analysis*, 2nd ed ed. New York: Wiley, 2001.
[2] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.
[3] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. San Diego, CA: Academic, 1999.
[4] E. Parzen, "On the estimation of a probability density function and the mode," *Annal. Math. Stat.*, vol. 32, pp. 1065–1076, 1962.
[5] C. Cortes and V. N. Vapnik, "Support vector networks," *Mach. Learning*, vol. 20, pp. 273–297, 1995.
[6] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[7] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[8] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*, Schölkopf and Smola, Eds. Cambridge, MA: MIT Press, 1999.
[9] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI: American Mathematial Society, 1997.
[10] J. Principe, D. Xu, and J. Fisher, "Information theoretic learning," in *Unsupervised Adaptive Filtering*, S. Haykin, Ed. New York: Wiley, 2000, vol. I, ch. 7.
[11] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller, "Fisher discriminant analysis with kernels," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, Madison, WI, Aug. 1999, pp. 41–48.
[12] C. K. Williams and D. Barber, "Bayesian classification with Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 12, pp. 1342–1351, Dec. 1998.
[13] R. Jenssen, D. Erdogmus, J. C. Principe, and T. Eltoft, "The Laplacian PDF distance: A cost function for clustering in a kernel feature space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 625–632.
[14] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to Kernel-Based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, pp. 181–201, 2001.
[15] R. Jenssen, J. C. Principe, and T. Eltoft, "Information cut and information forces for clustering," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, Toulouse, France, Sep. 2003, pp. 459–468.
[16] D. Erdogmus, J. C. Principe, and K. E. Hild, "On-line entropy manipulation: Stochastic information gradient," *IEEE Signal Process. Lett.*, vol. 10, no. 8, pp. 242–245, Aug. 2003.
[17] A. Elgammal, R. Duraiswami, and L. Davis, "The fast Gauss transform for efficient Kernel density evaluation with applications in computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 11, pp. 1499–1504, Nov. 2003.
[18] A. Han, S. Rao, and J. C. Principe, "Estimating the information potential with the fast Gauss transform," in *Proc. Int. Conf. Independent Component Anal.*, Charleston, SC, Mar. 2006, pp. 82–89.
[19] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman and Hall, 1986.
[20] D. W. Scott, *Multivariate Density Estimation*. New York: Wiley, 1992.
[21] M. P. Wand and M. C. Jones, *Kernel Smooting*. London, U.K.: Chapman and Hall, 1995.
[22] C. Williams and M. Seeger, "Using the Nyström method to speed up Kernel machines," in *Proc. Adv. Neural Inf. Processing Syst.*, 2001, vol. 13, pp. 682–688.
[23] Y. Bengio, P. Vincent, and J.-F. Paiement, "Spectral clustering and kernel PCA are learning eigenfunctions," Dépt. d'inf. Rech. Opér., Univ. Montréal, Montréal, QC, Canada, 2003.
[24] M. Brand and K. Huang, "A unifying theorem for spectral embedding and clustering," in *Proc. Int. Workshop Artif. Intell. Statistics*, Key West, FL, Jan. 2003, pp. 297–304.
[25] A. Y. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Processing Syst.*, 2002, vol. 14, pp. 849–856.
[26] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, pp. 1373–1396, 2003.
[27] R. Jenssen, D. Erdogmus, J. C. Principe, and T. Eltoft, "The Laplacian spectral classifier," in *Proc. IEEE Conf. Acoust., Speech Signal Process.*, Philadelphia, PA, Mar. 2005, pp. 325–328.
[28] B. Schölkopf, A. J. Smola, and K. R. Müller, "Nonlinear component analysis as a Kernel Eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.
[29] R. Murphy and D. Ada, "UCI repository of machine learning databases," Tech. Rep, Dept. Comput. Sci., Univ California, Irvine, 1994.
[30] G. Räetsch, T. Onoda, and K. R. Müller, "Soft margins for adaboost," *Mach. Learn.*, vol. 42, pp. 287–320, 2001.
[31] G. C. Cawley, "MATLAB Support Vector Machine Toolbox $(v0.50\beta)$," Univ. East Anglia, School Inf. Syst., Norfolk, U.K, Tech. Rep. NR4 7TJ, 2000 [Online]. Available: http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox
[32] S. Canu, Y. Grandvalet, and A. Rakotomamonjy, "SVM and Kernel methods matlab toolbox Perception Syst. Information, INSA de Rouen", Rouen, France, 2003 [Online]. Available: http://asi.insa-rouen.fr/~arakotom/toolbox/index.html
[33] M. Kuss and C. E. Rasmussen, "Assessing approximate inference for binary Gaussian process classification," *J. Mach. Learn. Res.*, vol. 6, pp. 1679–1704, 2005.
[34] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, Oct. 1994.
[35] B. Schölkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.



Robert Jenssen (M'03) received the M.S. and Ph.D. degrees in electrical engineering, from the University of Tromsø, Tromsø, Norway, in 2001 and 2005, respectively.

In his research, he has focused on developing an information theoretic approach to machine learning, based on statistical information measures and nonparametric density estimation. His research interests also include Mercer kernel methods, spectral clustering, and independent component analysis. He spent the academic year 2002/2003 and March/April 2004 at the University of Florida, Gainesville, as a Visitor at the Computational NeuroEngineering Laboratory. He is currently an Associate Professor of electrical engineering at the University of Tromsø.

Dr. Jenssen received Honorable Mention for the 2003 Pattern Recognition Journal Best Paper Award and the 2005 IEEE ICASSP Outstanding Student Paper Award.

Deniz Erdogmus (M'02) received the B.S. degrees in electrical and electronics engineering and mathematics and the M.S. degree in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 1997 and 1999, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Florida (UF), Gainesville, in 2002.

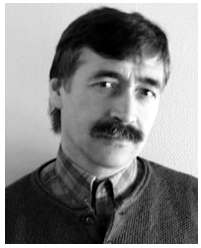
He worked as a Research Engineer at TUBITAK-SAGE, Turkey, from 1997 to 1999, focusing on the design of navigation, guidance, and flight control systems. He was also a Research Assistant and a Postdoctoral Research Associate at UF from 1999 to 2004, concentrating on signal processing, adaptive systems, machine learning, and information theory, specifically with applications in biomedical engineering including brain machine interfaces. Currently, he is an Assistant Professor jointly at the Computer Science and Electrical Engineering Department and the Biomedical Engineering Department of the Oregon Health and Science University, Beaverton, OR. His research focuses on information theoretic adaptive signal processing and its applications to biomedical signal processing problems. He has over 35 articles in international scientific journals and numerous conference papers and book chapters.

Dr. Erdogmus has also served as Associate Editor and Guest Editor for various journals, participated in various conference organization and scientific committees, and he is a member of Tau Beta Pi, Eta Kappa Nu, and IEE (U.K.). He was the recipient of the IEEE-SPS 2003 Best Young Author Paper Award and 2004 INNS Young Investigator Award.

Jose C. Principe (M'83–SM'92–F'00) received the B.S. degree from the University of Porto, Portugal, the M.S. and Ph.D. degrees from the University of Florida, Gainesville, all in electrical engineering, and a Laurea Honoris Causa degree from the Universita Mediterranea, Reggio Calabria, Italy.

He is a Distinguished Professor of electrical and biomedical engineering at the University of Florida, Gainesville, since 2002. He joined the University of Florida in 1987, after an eight-year appointment as Professor at the University of Aveiro, Portugal. His interests lie in nonlinear nonGaussian optimal signal processing and modeling and in biomedical engineering. He created the Computational NeuroEngineering Laboratory, in 1991 to synergistically focus the research in biological information processing models. He holds five patents and has submitted seven more. He was supervisory committee chair of 47 Ph.D. and 61 Master's students, and he is author of more than 400 refereed publications (3 books, 4 edited books, 14 book chapters, 116 journal papers, and 276 conference proceedings).

Dr. Principe is a Past President of the International Neural Network Society, and has been Editor-in-Chief of the TRANSACTIONS OF BIOMEDICAL ENGINEERING since 2001, as well as a former member of the Advisory Science Board of the FDA.



Torbjørn Eltoft (M'92) received the Cand. Real. (M.S.) and Dr. Scient. (Ph.D.) degrees from the University of Tromsø, Tromsø, Norway, in 1981 and 1984, respectively.

His early research was on the application of modern signal processing techniques in experimental ionospheric physics. Since 1984, he has been working with remote sensing, with a special interest in the nonlinear SAR imaging of ocean waves and the scattering of microwaves from the ocean surface. He joined the Faculty of Physics, University of Tromsø, Tromsø, Norway, in 1988, where he is currently a Professor of electrical engineering. His current research interests include remote sensing, image and signal processing, and artificial neural networks.

Dr. Eltoft was awarded the 2000 Outstanding Paper Award in Neural Networks by the IEEE Neural Networks Council and Honorable Mention for the 2003 Pattern Recognition Journal Best Paper Award.