The Class-Specific Classifier: Avoiding the Curse of Dimensionality
Tutorial Presented at
OCEANS 2003 Conference, San Diego, Sep 22-26 2003

Dr. Paul M. Baggenstoss *
Naval Undersea Warfare Center
Newport RI, 02841
401-832-8240 (TEL)
p.m.baggenstoss@ieee.org (EMAIL)

August 15, 2003

Abstract

This article describes a new probabilistic method called the "class-specific method" (CSM). CSM has the potential to avoid the "curse of dimensionality" which plagues most classifiers which attempt to determine the decision boundaries in a high-dimensional featue space. In contrast, in CSM, it is possible to build classifiers without a common feature space. Separate low-dimensional features sets may be defined for each class, while the decision functions are projected back to the common *raw data* space. CSM effectively extends the classical classification theory to handle multiple feature spaces. It is completely general, and requires no simplifying assumption such as Gaussianity or that data lies in linear subspaces.

1 Introduction and Background

The purpose of this article is to introduce the reader to the basic principles of classification with class-specific features. It is written both for readers interested in only the basic concepts as well as those interested in getting started in applying the method. For in-depth coverage, the reader is referred to a more detailed article [1].

Classification is the process of assigning data to one of a set of pre-determined class labels [2]. Classification is a fundamental problem that has to be solved if machines are to approximate the human functions of recognizing sounds, images, or other sensory inputs. This is why classification is so important for automation in today's commercial and military arenas.

Many of us have first-hand knowledge of successful automated recognition systems from cameras that recognize faces in airports to computers that can scan and read printed and handwritten text, or systems that can recognize human speech. These systems are becoming more and more reliable and accurate. Given reasonably clean input data, the performance is often quite good if not perfect. But many of these systems fail in applications where clean, uncorrupted data is not available or if the problem is complicated by variability of conditions or by proliferation of inputs from unknown sources. In military environments, the targets to be recognized are often uncooperative and hidden in clutter

*This work supported by Office of Naval Research

and interference. In short, military uses of such systems still fall far short of what a well-trained alert human operator can achieve.

We are often perplexed by the wide gap of performance between humans and automated systems. Allow a human listener to hear two or three examples of a sound - such as a car door slamming. From these few examples, the human can recognize the sound again and not confuse it with similar interfering sounds. But try the same experiment with general-purpose classifiers using neural networks and the story is quite different. Depending on the problem, the automated system may require hundreds, thousands, even millions of examples for training before it becomes both robust and reliable.

Why? The answer lies in what is known as the "curse of dimensionality". General-purpose classifiers need to extract a large number of measurements, or *features*, from the data to account for all the different possibilities of data types. The large collection of features form a high-dimensional space that the classifier has to sub-divide into decision boundaries. It is well-known that the complexity of a high-dimensional space increases exponentially with the number of measurements [3] - and so does the difficulty of finding the best decision boundaries from a fixed amount of training data. Unless a lot is known about the data, allowing the features to be suitably conditioned so that the data samples fall in nicely organized patterns in the feature space, finding the "optimum" decision boundaries in a feature space above about 5 dimensions is futile [4]. Optimum decision boundaries require finding the probability distributions (probability density functions or PDFs) of each class in the feature space [2]. Sub-optimal decision boundaries, that is based on simplified probability models, or simple search procedures such as "nearest neighbor", can achieve very good performance if the data from the various classes are well separated in the feature space, but fail dramatically if there is any degradation of training data or input data quality.

These problems can potentially be avoided if we avoid working in a high-dimensional space. But how can we avoid working in high dimensions if all the measurements (features) carry pertinent information? One way is to keep a large number of features, but divide up the features according to their relevance to a particular class (class-specific features) and process them separately. Many schemes have been invented to try to find suitable rules for combining the processors [5], [6], [7], [8], [9], [10]. While they are on the right track, the problem with these classifiers is that they generally are unable to combine the results of the individual decisions in a way that is both theoretically optimal and completely general at the same time. What is needed is an extension to the classical theory of hypothesis testing that can account for class-specific features.

In answer to this need, the author proposed the class-specific method (CSM) in 1998 [11], [12], [13]. The initial formulation of the method suffered from several difficulties which were solved with the publication of the PDF projection theorem in 2000 [14], [15]. Further enhancements of the theory have resulted from the chain-rule [16],[1] a recursive application of the PDF projection theorem. The resulting classifier architecture, called the chain-rule processor [16],[1], blends the best aspects of signal processing and classification. CSM is completely general and makes no assumptions about the data such as that it yields to linear subspace decomposition. Nor does it require any special topology such as a binary tree of decisions. In fact, the classical feature classifier is a special case CSM that occurs when all classes are represented by a common feature set. But, unlike the classical classifier, CSM can circumvent the curse of dimensionality if each class can be represented (statistically described) using a separate low-dimensional feature set.

2 The Classical Approach

The classical Bayesian classifier selects the most likely class hypothesis given the data according to

$$j^* = \arg\max_{j=1}^{M} \; p(H_j|\mathbf{x}),$$

where $\mathbf{x}$ is the data and $\{H_1, H_2 \ldots H_M\}$ are the M class hypotheses. Using Bayes rule, this may be written

$$j^* = \arg\max_{j} \; p(\mathbf{x}|H_j)\; p(H_j). \tag{1}$$

This classifier has, in theory, the lowest probability of error of all classifiers [2],[17]. Unfortunately, the likelihood functions, or probability density functions (PDFs), denoted by $p(\mathbf{x}|H_j)$ are unknown and need to be estimated from training data. Because the dimension of the raw data is too high, $\mathbf{x}$ has to be reduced to a set of information-bearing features using a feature transformation $\mathbf{z} = T(\mathbf{x})$. If it is possible to find a low-dimensional feature set that contains most or all of the necessary information, the problem can then be re-formulated in terms of $\mathbf{z}$. By regarding $\mathbf{z}$ as the data, the Bayesian feature classifier becomes

$$j^* = \arg\max_{j} \; \hat{p}(\mathbf{z}|H_j)p(H_j),$$

where $\hat{p}(\mathbf{z}|H_j)$ are the feature PDFs estimated from training data.

The classical approach to classification is summarized graphically in Figure 1 for two data classes. The original raw data space ($\mathbf{X}$) is mapped to a feature space ($\mathbf{Z}$) where the PDFs are estimated and the decision boundaries are constructed. The curse of dimensionality forces the following trade-off: If the feature dimension is too high, there are severe errors in PDF estimation causing classification errors. If the feature dimension is too low, the loss of information causes the classes to become overlapped in $\mathbf{Z}$, also causing classification errors. There may be no feature dimension where the performance is acceptable. In short, the curse of dimensionality cannot be overcome. Once the raw data is discarded in favor of a common set of features, all hope is lost for achieving the best possible performance.

3 The Class-Specific Method (CSM)

Because this is a tutorial paper, we present only the most basic mathematical concepts of CSM. For further reading, the reader is referred to the most recent publications [1].

The classical approach loses the fight against the curse of dimensionality because it puts "all of its eggs in one basket". It requires a low-dimensional feature set that contains all of the necessary information - an impossible request. Instead of discarding the raw data, CSM actually operates in the raw data domain - but it estimates the PDFs in low-dimensional feature spaces. This requires a two-step procedure.

3.1 Step 1: Feature transformation and PDF estimation.

First, the raw data is transformed into class-specific low-dimensional feature spaces. Let

$$\begin{aligned} \mathbf{z}_1 &= T_1(\mathbf{x}) \\ \mathbf{z}_2 &= T_2(\mathbf{x}) \\ &\vdots \\ \mathbf{z}_M &= T_M(\mathbf{x}) \end{aligned}$$

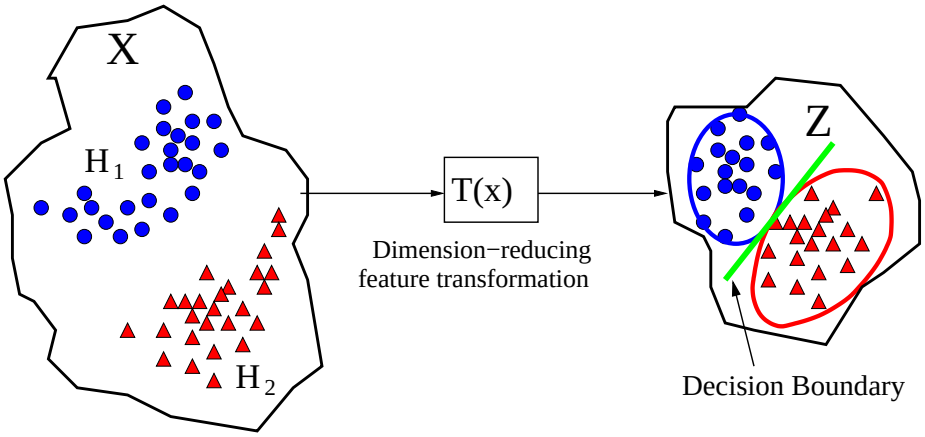

Figure 1: Illustration of the classical approach to classification. The original raw data space ($\mathbf{X}$) is mapped to a feature space ($\mathbf{Z}$) where the PDFs are approximated (ellipses) and decision boundaries (green line) are constructed. Because of potential information loss, the classes can become overlapped in $\mathbf{Z}$, causing classification errors.

be the M different feature sets and feature transformations. The PDFs $p(\mathbf{z}_m|H_m)$, $1 \leq m \leq M$, are then estimated from training data. This first step is illustrated in Figure 2.

3.2 Step 2: PDF projection back to raw data domain.

Next, CSM converts the feature PDFs into raw-data PDFs. It *projects* the PDFs back to the raw data domain where the decision boundaries are constructed. CSM avoids the complexity issues of the raw-data space because the projection operators (functions that transform the PDFs to the raw data domain) are known functions that can be determined exactly from the feature transformations. This last step is illustrated in Figure 3.

3.3 How the projection works.

Projecting the PDF from the feature domain back to the raw data domain is made possible by the PDF projection theorem [15],[14]. This theorem may be thought of as a generalization of the well-known change-of-variables theorem which relates the PDF of $\mathbf{y}$ to the PDF of $\mathbf{x}$ when related by the 1:1 transformation $\mathbf{y} = f(\mathbf{x})$. For continuous invertible transformations, it is a simple matter to recover the PDF of $\mathbf{x}$ from the PDF of $\mathbf{y}$ using the formula

$$p_x(\mathbf{x}) = \left|\frac{\partial y}{\partial x}\right| \; p_y(\mathbf{y}). \tag{2}$$

The PDF projection theorem (PPT) is a generalization of (2) for many-to-one transformations. Under certain conditions (having to do with sufficient statistics) it is possible to actually recover $p_x(\mathbf{x})$ from the feature PDF. In general, however, the PPT can only find a particular one of the many possible PDFs of $\mathbf{x}$ that could have produced the given feature PDF. This particular choice has some nice properties. The projection operation is illustrated in Figure 4. Projection can only be accomplished if it is possible to know both the raw data PDF and feature PDF under some reference hypothesis H_0. In general, it is impossible to determine the raw data PDF if all we know is the feature PDF.

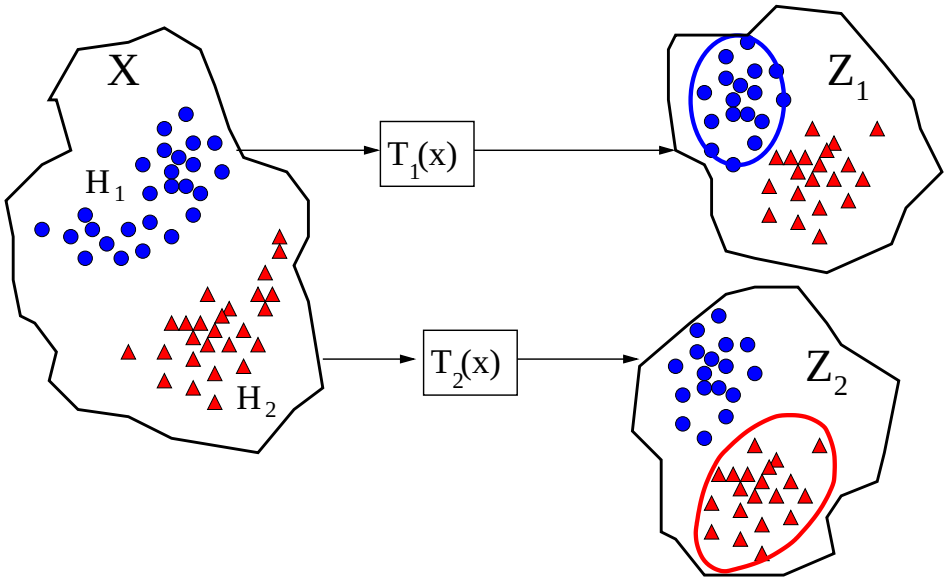

Figure 2: Illustration of the first step of CSM. A separate feature transformation is designed for each class. The feature PDFs for each class are estimated on the corresponding feature space (ellipses).

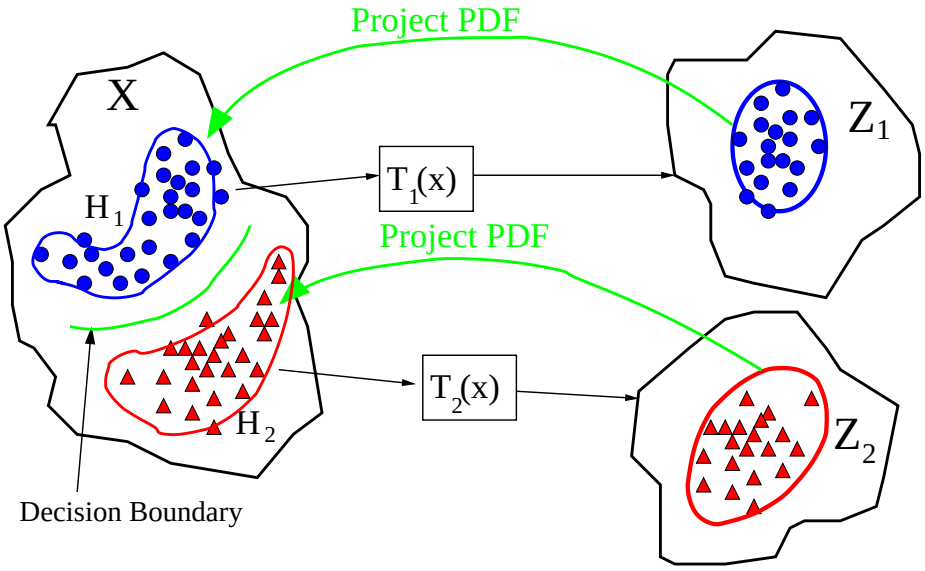

Figure 3: Illustration of the second step of CSM. The feature PDFs are projected back to the raw data space where the decision boundaries are constructed.

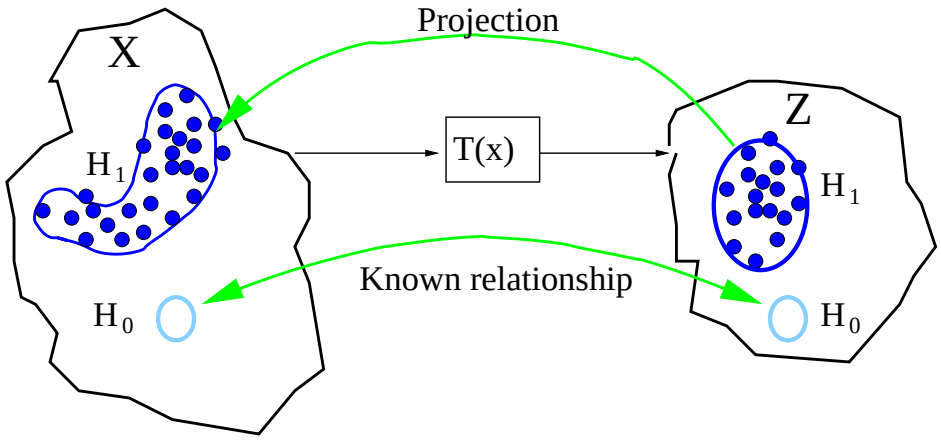

Figure 4: Illustration of the PDF projection operation. The projection can be accomplished only if is possible to know both the raw data PDF and feature PDF for some reference hypothesis H_0.

The projection operation finds only an approximation to the true raw data PDF of the given class hypothesis. The projected PDF defined on the raw data domain is given mathematically by

$$p_p(\mathbf{x}|H_j) \triangleq \left[\frac{p(\mathbf{x}|H_{0,j})}{p(\mathbf{z}_j|H_{0,j})}\right] \hat{p}(\mathbf{z}_j|H_j), \tag{3}$$

where $H_{0,j}$ are the class-specific reference hypotheses. Thus, the partial derivative (which generalizes to the determinant of the Jacobian matrix for multi-dimensions) in equation (2) is replaced by a ratio of PDFs. As expected, the PDF projection theorem simplifies to (2) for continuous invertible transformations. It may be proved [15],[14] that $p_p(\mathbf{x}|H_j)$ is a PDF, so it integrates to 1 on the raw data space, and that it is a member of the class of PDFs which *generate* the original feature PDF $\hat{p}(\mathbf{z}|H_j)$. This means that if a random variable $\mathbf{x}$ is drawn from the PDF in equation (3), and the result is transformed by the feature transformation $\mathbf{z}_j = T_j(\mathbf{x})$, then the PDF of $\mathbf{z}_j$ will be precisely $\hat{p}(\mathbf{z}_j|H_j)$, i.e. the projection process comes full circle.

Various interpretations of the projection theorem can be suggested. One interpretation is that since there are an infinite number of raw data PDFs that generate the feature PDF, it is necessary to invoke a constraint so that one unique raw data PDF can be found. The applicable constraint is that the likelihood ratio with respect to the reference hypothesis remains constant in either domain:

$$\frac{p_p(\mathbf{x}|H_j)}{p(\mathbf{x}|H_{0,j})} = \frac{\hat{p}(\mathbf{z}_j|H_j)}{p(\mathbf{z}_j|H_{0,j})}.$$

Another interpretation is possible if we reverse the usual thinking. Normally we start with two statistical hypothesis, then seek a sufficient statistic for differentiating between them. But we could also start with just one hypothesis ($H_{0,j}$) and a statistic ($\mathbf{z}_j$) and ask "what would be a second hypothesis for which $\mathbf{z}_j$ is sufficient against $H_{0,j}$?". The PDF constructed according to (3) is the second hypothesis we seek. Thus, $\mathbf{z}_j$ is sufficient for $H_{0,j}$ vs. the hypothesis that $p_p(\mathbf{x}|H_j)$ is true.

3.4 How to choose the reference hypothesis.

A detailed mathematical treatment of the issues surrounding the reference hypothesis are given in [1]. Briefly, the conditions that H_0 must satisfy for the projection (3) to result in a valid PDF are that

$p(\mathbf{z}|H_0)$ must never be precisely zero at any place where sample data can lie - otherwise, J cannot be evaluated. This is a rather mild constraint, easily satisfied by the most common PDFs such as Gaussian and exponential (assuming negative values are illegal). As long as this condition holds, equation (3) will be a PDF and will be among the class of PDFs which give rise to the specified feature PDF $\hat{p}(\mathbf{z}_j|H_j)$ when transformed by the given feature transformation. That being said, there are good and bad choices for $H_{0,j}$. A good choice of $H_{0,j}$ (one that will result in a good approximation to $p(\mathbf{x}|H_j)$) is one for which the features $\mathbf{z}_j = T_j(\mathbf{x})$ are approximately sufficient statistics for testing H_j vs. $H_{0,j}$. Sufficiency is meant in the statistical sense, and does not mean "just good enough". It means that *all* information necessary to separate $H_{0,j}$ from H_j is present. Remember, though, that this condition is a goal, not a requirement and should not discourage anyone from trying a particular feature set. The closer the sufficiency condition can be approximated, the better the projected PDF will approximate $p(\mathbf{x}|H_j)$. It is also advisable that $H_{0,j}$ be such that $p(\mathbf{x}|H_{0,j})$ and $p(\mathbf{z}|H_{0,j})$ can both be determined either in closed form, or else to a good approximation, even in the (far) tails.

3.5 How to build a CSM classifier

By substitution of (3) into the Bayesian classifier (1), the CSM classifier results:

$$j^* = \arg\max_{j=1}^{M} \left[\frac{p(\mathbf{x}|H_{0,j})}{p(\mathbf{z}_j|H_{0,j})}\right] \hat{p}(\mathbf{z}_j|H_j)\; p(H_j), \tag{4}$$

The ratio

$$J(\mathbf{x}, T_j, H_{0,j}) \triangleq \left[\frac{p(\mathbf{x}|H_{0,j})}{p(\mathbf{z}_j|H_{0,j})}\right] \tag{5}$$

we call the "J-function" and may be considered generalized Jacobian or *correction* term necessary to create the optimal Bayes classifier from the various feature PDFs.

3.6 When is CSM optimal?

Clearly if the projected PDFs (3) are valid PDFs, no matter if they are accurate approximations to the desired PDFs $p(\mathbf{x}|H_j)$, the classifier (4) is a valid probabilistic classifier. Optimality occurs when the projected PDFs are equal to the desired PDFs. This happens when (1) the estimated feature PDFs, $\hat{p}(\mathbf{z}_j|H_j)$, are equal to the true feature PDFs, and (2) when the class-specific features, $\mathbf{z}_j = T_j(\mathbf{x})$ are sufficient statistics for deciding between the given class H_j and the chosen reference hypotheses $H_{0,j}$. Because the designer can choose both $T_j(\ )$ and $H_{0,j}$, it is to the designer's benefit to choose them jointly to approximate this condition. Note also that $H_{0,j}$ must be chosen from those hypotheses for which it is possible to solve for both $p(\mathbf{x}|H_{0,j})$ and $p(\mathbf{z}_j|H_{0,j})$. It is not always easy, but great strides have been made in recent years in being able to solve for the feature PDFs for many useful types of features [18], [19].

3.7 Why is CSM better than the classical approach?

Both CSM and the classical approach have the same theoretical performance because they are both based on the optimal Bayesian classifier (1). Indeed, this is demonstrated in an experiment where a class-specific classifier was compared to a classical classifier using exactly the same features [11]. In the 9-class synthetic data experiment, the class-specific classifier used feature sets of dimension between 1 and 2, while the classical (full-dimensional) classifier operated on an 11-dimensional feature set (the union of the class-specific features). The performance was plotted as a function of the number of training samples and is repeated in Figure 5. It shows that although the maximum performance

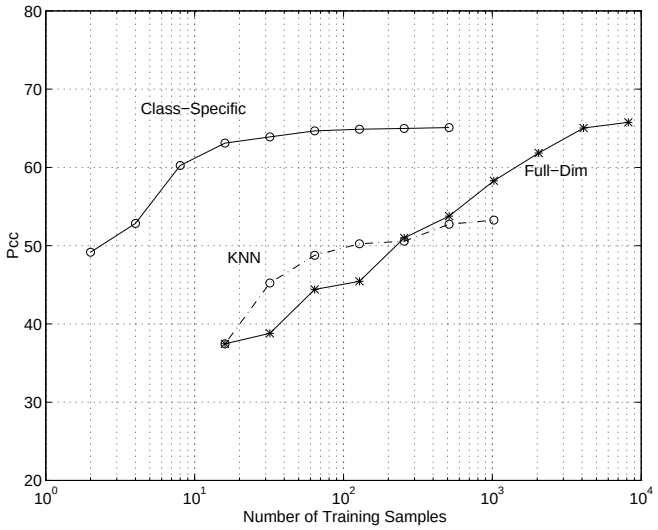

Figure 5: Classification performance of CSM compared with classical approach in a 9-class synthetic data experiment. The classical approach used an 11-dimensional feature set composed of the union of all the class-specific features.

of the classical classifier was the same, it required more than two orders of magnitude more training data to achieve it. Now imagine that only about 100 samples were available - observe on the graph the gap in performance that would exist. But the classical approach can never attain the promised performance because it needs to form a common feature set where the PDFs are estimated. The curse of dimensionality exacts a heavy toll on performance. For a given maximum feature dimension, CSM can collect much more information from the raw data because it can divide the information up according to class.

3.8 The paradigm shift

Those that have worked with the classical approach have difficulty changing over to CSM which is an entirely new paradigm. Someone trained to view features as carrying information to distinguish one class from another may have a difficult time viewing features in a way that ignores the other classes. A simple geometric example can illustrate the paradigm shift. Figure 6 shows a notional 4-class problem involving sets of geometric shapes. The classical paradigm involves finding features that are able to discriminate among the four classes. A list of six measurements or "features" are provided in the yellow box on the right side of the figure. These six features are adequate for discrimination among the four classes. The mathematical implementation of the classical feature paradigm involves the maximization of the feature PDF:

$$j^* = \arg\max_j \; p(\mathbf{z}|H_j),$$

where $\mathbf{z} = T(\mathbf{x})$ is a common feature set.

Figure 7 illustrates the class-specific paradigm using a fixed reference hypothesis. The features are required to discriminate each class from the common reference hypothesis. This is the original formulation of CSM but has a number of difficulties arising from the use of a common fixed reference hypothesis. The mathematical implementation of the fixed-reference class-specific paradigm involves

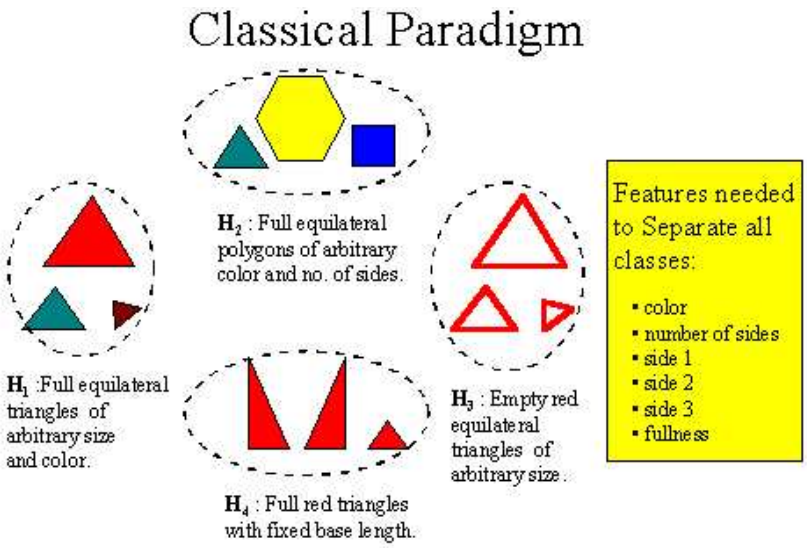

Figure 6: The classical paradigm for a notional 4-class classification problem. The yellow box on the right lists six measurements or "features" useful for classifying the four classes.

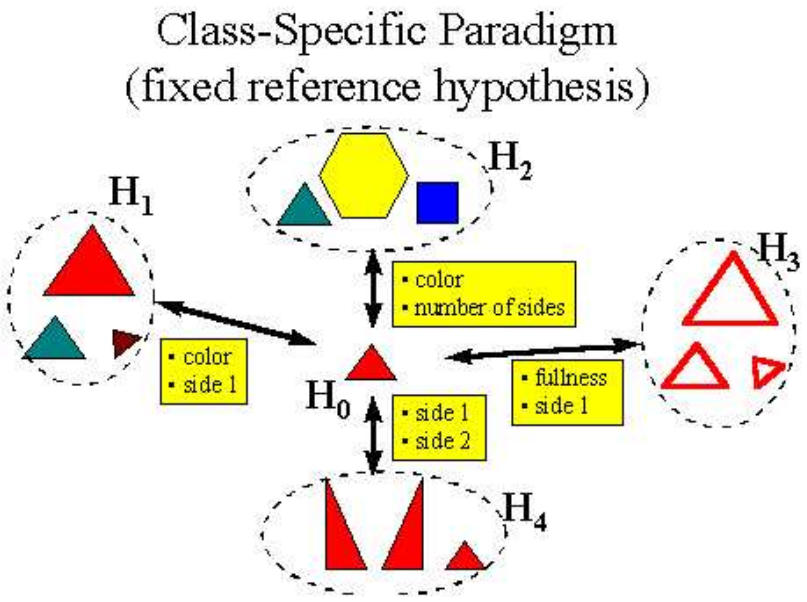

Figure 7: The class-specific paradigm for a fixed reference hypothesis. The features (in yellow boxes) are required to discriminate each class from the common reference hypothesis. Note that fewer features are required for the simpler binary problems.

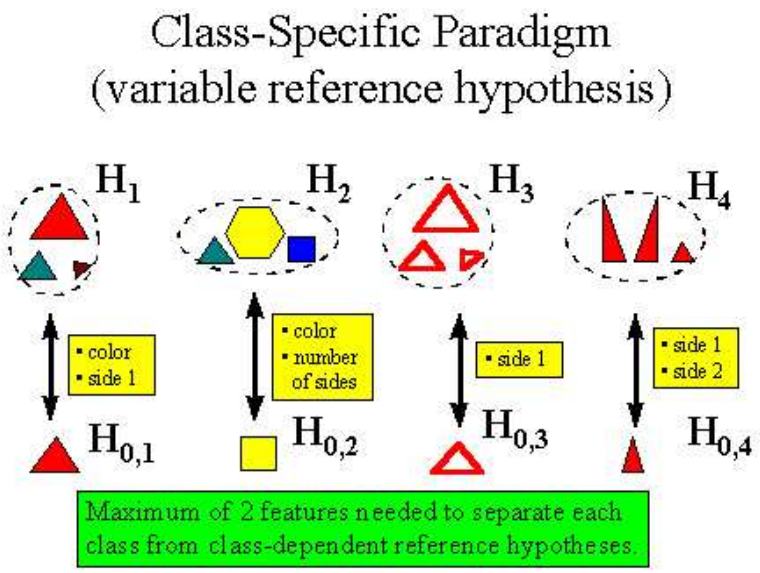

Figure 8: The class-specific paradigm for class-specific reference hypotheses. The features are required to discriminate each class from the corresponding class-specific reference hypothesis.

the maximization of the likelihood ratios

$$j^* = \arg\max_j \frac{p(\mathbf{z}_j|H_j)}{p(\mathbf{z}_j|H_0)},$$

where $\mathbf{z}_j = T_j(\mathbf{x})$, for $j = 1 \ldots M$, are class-specific feature sets.

Figure 8 illustrates the class-specific paradigm using class-specific reference hypotheses. Class-specific features are not chosen to discriminate a class from other classes, they are chosen to discriminate each class from the corresponding class-specific reference hypothesis which may be regarded as a special member of the class. In effect, this means the features are chosen to describe the class. It is important to remember that discrimination happens automatically if each class is well described. In effect, choosing features for description results in the same (or more) feature information content as discrimination but it assigns the information only to those classes for which it is relevant. In spite of this, it is a difficult paradigm shift to make for many people who have been taught to choose features for discriminatory power. The mathematical implementation of the class-specific paradigm involves the maximization of the projected PDFs

$$j^* = \arg\max_j p_p(\mathbf{x}|H_j),$$

where

$$p_p(\mathbf{x}|H_j) = \frac{p(\mathbf{x}|H_{0,j})}{p(\mathbf{z}_j|H_{0,j})}\, p(\mathbf{z}_j|H_j),$$

where $\mathbf{z}_j = T_j(\mathbf{x})$, for $j = 1 \ldots M$, are class-specific feature sets and $H_{0,j}$, for $j = 1 \ldots M$, are class-specific reference hypotheses.

3.9 Working in the raw data domain

CSM creates decision boundaries in the raw data domain instead of in a common feature domain. This sounds troublesome at first. After all, the raw data dimension can be very large and we are interested

in *reducing* the dimension! But remember that PDF estimation happens on the low-dimensional feature space, not in the raw data space. Projecting to the raw data domain is done for us by the J-function (5) which does not suffer from the dimensionality curse because it does not need to be found empirically. The J-function can be determined exactly by analysis of the feature transformations.

There is a clear advantage to working in the raw data domain because it is a common ground where everything can be compared fairly. Interestingly, CSM is not the first attempt to work in the raw data domain. For example, Bishop [20] creates raw data PDFs, but his approach requires linear transformations to be tractable and amounts to something akin to principal component analysis (PCA). CSM, on the other hand, is completely general. The "ace in the hole" is the fact that the projected PDF *is* indeed a PDF and it depends only upon a few parameters - the parameters of the feature PDF and of the feature transformation and reference hypothesis. All of these parameters are "fair game" in a maximum likelihood maximization. Have you an idea for a better feature set? Compare it to the existing feature set based on the maximum likelihood principle. Have you an idea for a better reference hypothesis? Compare it to the existing reference hypothesis based on the maximum likelihood principle. This idea can is represented mathematically as

$$L(\mathbf{x}_1 \dots \mathbf{x}_K, H_0, T, \theta) = \max_{H_0,T,\theta} \sum_{k=1}^{K} \log \left\{ \left[\frac{p(\mathbf{x}_k|H_0)}{p_z(T(\mathbf{x}_k)|H_0)} \right] \; p_z(T(\mathbf{x}_k); \boldsymbol{\theta}) \right\}, \tag{6}$$

K is the number of independent data samples and the subscript z is a reminder that the PDF $p_z(\,)$ is a function of the features $\mathbf{z} = T(\mathbf{x})$. To avoid "over-training", when implementing (6) in practice, it is recommended that the data be partitioned into separate training and testing sets for cross-validation.

3.10 Classifying without training.

The PPT (3) is a decomposition of the raw data PDF into a trained and an untrained factors. The trained factor is the feature PDF $\hat{p}(\mathbf{z}_j|H_j)$ which needs to be estimated from training data of the corresponding class. The untrained factor $[p(\mathbf{x}|H_{0,j})/p(\mathbf{z}_j|H_{0,j})]$ is a known function of the input raw data $\mathbf{x}$, feature transformation $T_j(\,)$, and reference hypothesis $H_{0,j}$. But, for a fixed $\mathbf{x}$, it also can be viewed as a function of j. Thus, it contributes, sometimes in a dominant role, to the classification decision. While the trained component asks "how does this sample compare with trained patterns?", the untrained component asks "how well does this feature set represent this raw data sample?". The untrained component can be seen as a generalization of a matched filter.

To see this, we consider a bank of linear matched filters as a set of class-specific feature extractors

$$z_j = T_j(\mathbf{x}) = |\mathbf{w}_j' \mathbf{x}|^2,$$

where $\mathbf{w}_j$ is a signal template. Let $\mathbf{w}_j$ be normalized such that $\mathbf{w}_j' \mathbf{w}_j = 1$. The simple matched filter bank classifier is given by

$$j^* = \arg\max_j \; z_j.$$

Let us now design a class-specific classifier for these features. Under the reference hypothesis of independent Gaussian noise of variance 1, z_j is distributed $\chi^2(1)$,

$$p(z_j|H_0) = \frac{1}{\sqrt{2\pi z_j}} \; \exp\left\{ -\frac{z_j}{2} \right\}.$$

The PDF of $\mathbf{x}$ under H_0 is the Gaussian PDF

$$p(\mathbf{x}|H_0) = (2\pi)^{-N/2} \; \exp\left\{ -\frac{1}{2} \sum_{n=1}^{N} x_n^2 \right\}.$$

The log-J-function is easily shown to be

$$\log J_j(\mathbf{x}, z_j) = \log p(\mathbf{x}|H_0) - \log p(z_j|H_0) = \frac{1}{2}\ (\log z_j + z_j) + C(\mathbf{x}),$$

where $C(\mathbf{x})$ does not depend on j. The complete class-specific classifier is:

$$\arg\max_j\ \log J_j(\mathbf{x}, z_j) + \log p(\mathbf{z}_j|H_j). \tag{7}$$

Since $\log J_j$ is a monotonic increasing function of z_j, using only $\log J_j(\mathbf{x}, z_j)$ as a classifier is equivalent to the matched filter bank classifier. Note that by ignoring the last term in (7) effectively assumes that each class has the same expected amplitude distribution.

It is clear that classification is quite possible without training as long as each class requires a distinctly different feature set for representation. This idea should not be taken literally without some care. Generalizing the "J-function-only" classifier to cases where the features are not matched filters, requires that some kind of *a priori* feature PDF should be used to account for differences in feature dimension and scaling. Note that this requirement is relaxed if the J-function is highly dominant.

4 The chain rule and the chain-rule processor

As part of the paradigm shift from the classical architecture, we recommend looking at a sophisticated general-purpose classifier as a bank of signal processors. Each signal processor may be thought of as an optimal detector for differentiating the given class from the corresponding reference hypothesis. Each signal processor may be composed of multiple processing stages. If we regard the feature transformation $\mathbf{z} = T(\mathbf{x})$ as a single step, we write the projected PDF as

$$p_p(\mathbf{x}|H_1) = \left[\frac{p(\mathbf{x}|H_0)}{p(\mathbf{z}|H_0)}\right] \hat{p}(\mathbf{z}|H_1),$$

However, if we regard the feature transformation as three separate stages, $\mathbf{y} = T'(\mathbf{x})$, $\mathbf{w} = T''(\mathbf{y})$, then $\mathbf{z} = T'''(\mathbf{w})$, we may apply the PDF projection theorem recursively. For the first stage, we have

$$p_p(\mathbf{x}|H_1) = \left[\frac{p(\mathbf{x}|H_0)}{p(\mathbf{y}|H_0)}\right] p_p(\mathbf{y}|H_1).$$

Applying the same concept to $p_p(\mathbf{y}|H_1)$, we have

$$p_p(\mathbf{y}|H_1) = \left[\frac{p(\mathbf{y}|H_0')}{p(\mathbf{w}|H_0')}\right] p_p(\mathbf{w}|H_1),$$

and so on. The complete break-down is written

$$p_p(\mathbf{x}|H_1) = \left[\frac{p(\mathbf{x}|H_0)}{p(\mathbf{y}|H_0)}\right] \left[\frac{p(\mathbf{y}|H_0')}{p(\mathbf{w}|H_0')}\right] \left[\frac{p(\mathbf{w}|H_0'')}{p(\mathbf{z}|H_0'')}\right] \hat{p}(\mathbf{z}|H_1), \tag{8}$$

where H_0, H_0', H_0'' are reference hypotheses suited to each stage in the processing chain. The advantage of this approach is first that many processing chains may share the same first stages of processing, thus saving processing. Furthermore, analyzing just one stage at a time simplifies the analysis. Finally, there is great advantage in software modularity because each stage of processing can be encapsulated as a module.

4.1 Feature Modules.

Feature modules are pre-packaged software modules that contain both feature calculation and J-function calculation. The three modules necessary for implementing the above three-stage chain would be

$$\text{Module 1:} \quad \mathbf{y} = T'(\mathbf{x}), \qquad j_1 = \log \frac{p(\mathbf{x}|H_0)}{p(\mathbf{y}|H_0)},$$

$$\text{Module 2:} \quad \mathbf{w} = T''(\mathbf{y}), \qquad j_2 = \log \frac{p(\mathbf{y}|H_0')}{p(\mathbf{w}|H_0')},$$

$$\text{Module 3:} \quad \mathbf{z} = T'''(\mathbf{w}), \qquad j_3 = \log \frac{p(\mathbf{w}|H_0'')}{p(\mathbf{z}|H_0'')}.$$

Completion of the processing chain is accomplished by accumulating the "correction terms"

$$\log p_p(\mathbf{x}|H_1) = j_1 + j_2 + j_3 + \log \hat{p}(\mathbf{z}|H_1). \tag{9}$$

Class-specific classifiers can be rapidly designed by stringing together chains of pre-designed modules and accumulating the log J-function values.

4.2 Classifier Architecture.

Implementation of a classifier is illustrated in Figure 9. Each horizontal chain corresponds to one class. The chains are made up of series of modules. In accordance with equation (9), each module adds the corresponding correction term (J-function) to the stream. At the end, the aggregate J-function is added to the log feature PDF to arrive at the class output value.

5 Building a Classifier

Because CSM is new, there is a large learning curve for those being introduced to it. There are many difficulties and pitfalls associated with building a classifier that should be mentioned.

5.1 Common Problems.

The following is a list of problems and difficulties that are often encountered in designing and implementing a class-specific classifier.

1. **Sufficiency**. Recall that the designer should, as a goal, strive for a feature set/reference hypothesis combination where the features are approximately sufficient to discriminate the class of interest from the reference hypothesis. Sufficiency is does not mean "just enough", i.e. sufficient to get the job done. Sufficiency means all of the information has been extracted for discrimination. But this is a goal, not a requirement. It should not discourage anyone from using a set of features that is reasonable. A common mistake is to leave out a significant amount of information relating to the discrimination of a given class from the fixed reference hypothesis simply because it is not necessary to discriminate the data most if not all of the time. Here's an example. Consider discriminating a sinewave in additive correlated noise from a reference hypothesis of independent noise. While it may be adequate to concentrate on the sinewave, do not lose sight of the fact that the background noise also is different from H_0 and can significantly contribute to discrimination. It would be better in this case to use the correlated noise as the reference hypothesis.

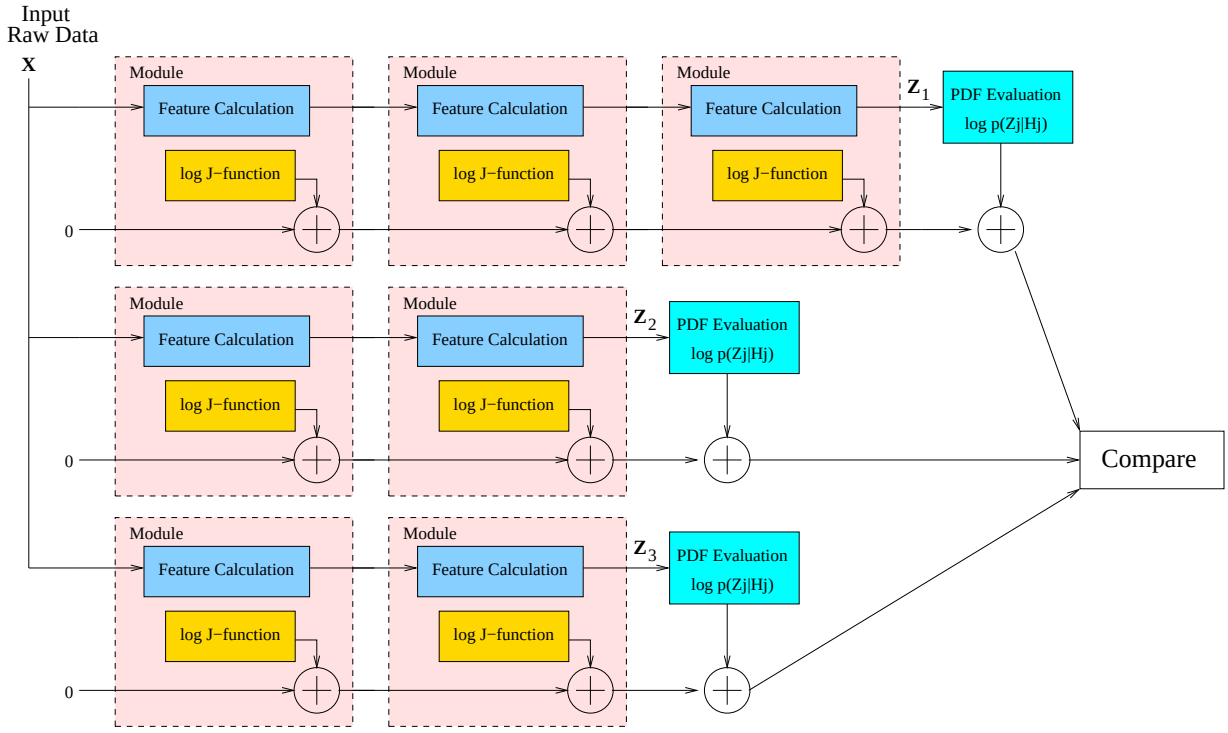

Figure 9: Block diagram of a class-specific classifier using chain rule processors.

2. **Using "all classes" as a reference hypothesis.** The suggestion that the reference hypothesis H_0 can be defined as a combination of "all classes" has been made several times. While in principle, H_0 can be any hypothesis, even this one, it defeats the purpose of class-specific features. This is because all the features are needed for discrimination of one class from all the others (see above item). Furthermore, the "all class" does not yield to mathematical analysis, needed to compute the J-function.

3. **Tail Probability Errors.** A common misconception is that the denominator PDF in the J-function, $p(\mathbf{z}|H_0)$, can be estimated from training data. This is only true if all possible realizations of input data will be within the central part of the distribution and not highly unlikely. This could work, for example, with low-SNR signals. But such a system would perform poorly against high-SNR signals. It may be possible to position the reference hypothesis "close" to the data sample, then attempt to estimate the PDF of the features by random trials. Note that this would need to be re-done for each sample to be tested. It also must meet the requirements for a variable reference hypothesis.

4. **Segmentation.** Segmentation is the practice of carving up data into fixed-sized segments, then extracting features from each segment. This is an important first step in processing. The choice of segmentation size is often a difficult choice in traditional classifiers, because it is necessary to choose the segment size that is "good enough" for all classes. The class-specific method affords us the luxury of using different segmentation sizes for each class. This is because the likelihood comparisons are made on the raw data, which is always the same. A common error people make is that because of the different segment sizes used across different classes, the amount of raw data varies slightly due to the fact that the input data size is not divisible by all segment sizes. This

can be a fatal error. It is necessary to only use an input data record size that can be divisible perfectly by each considered segment size.

5. **Failure to Validate Analysis**. Some form of absolute validation is necessary before using a module. In section 7, a method of validating the J-function analysis is provided. There is no obvious way to locate errors except with this approach.

5.2 Module Design

There are more than one method of module design. The designer should not give up on using a good set of features because one module design approach fails - there may be another that works.

1. **Fixed reference hypothesis.** In this approach, a fixed reference hypothesis, such as independent Gaussian noise of a fixed variance is chosen. Then, the numerator and denominator densities of the J-function must be known exactly or approximated with the saddlepoint approximation [18] to insure accurate tail values.

2. **Floating reference hypothesis.** Floating the reference hypothesis by positioning it "close" to the data sample to be tested is a means of avoiding the tails. In general, a reference hypothesis cannot be made dependent on the data - this violates the concept of a statistical hypothesis. But under certain conditions, the dependence of the numerator and denominator of the J-function on changes in the reference hypothesis cancel out making the approach feasible [1]. The reference hypothesis may be floated as a function of the data as long as the features are sufficient statistics to distinguish all the possible hypotheses that may result. Floating the hypothesis may be a simple as adjusting the variance of the Gaussian assumption to agree with the sample variance of the data. Or, it may be as sophisticated as controlling the noise spectrum of an autoregressive model to agree with the observed autocorrelation function. The designer must insure that the features are sufficient or approximately sufficient to discriminate between the various reference hypotheses. For example, any feature set that contains the sample variance explicitly as a component or where the sample variance can be inferred from the features is fully sufficient to discriminate between any pair of variance hypotheses. Therefore, the variance of the reference hypothesis can be "floated".

3. **On-the-fly analysis.** It is possible to make a rapid Montecarlo-type analysis of the feature PDFs under a floating reference hypothesis [1]. This is useful when the PDF of the features defies analysis.

5.3 NUWC Module Library

The class-specific module is the building block of a class-specific classifier. It can be a source of frustration if a classifier designer wishes to use a feature set and cannot because no analysis is available. This is why a library of pre-tested class-specific modules is useful. A central repository of class-specific modules is being collected at a web-site at NUWC:

`http://www.npt.nuwc.navy.mil/csf/index.html`

To date, this collection includes the following feature transformations:

1. Various invertible transformations.

[1] The author wishes to thank Mario Fritz for this suggestion

2. Spectrogram.
3. Arbitrary linear functions of exponential RVs.
4. Autocorrelation function (contiguous and non-contiguous).
5. Autoregressive parameters (Reflection coefficients).
6. Cepstrum (including MEL Cepstrum).
7. Order statistics of independent RVs.
8. Sets of quadratic forms.

New feature modules may be designed using the analysis tools of CR bound analysis (for maximum likelihood features) Readers are encouraged to use the library and submit their own contributed modules.

6 Examples of Feature Chains

Examples are necessary to make clear the important point thus far discussed. Each example shows how a feature transformation chain can be analyzed to obtain the correction term for PDF projection. When the feature transformation occurs in more than one step, the examples are broken down into separate modules. For each module, we provide the following information (all enclosed in boxes for clarity),

Feature Calculation: The mathematical expression of the feature calculation.

H_0 : A description of the reference hypothesis.

The class-specific correction term (J-function) is given by

$$J(\mathbf{x}, T, H_0) = \frac{p(\mathbf{x}|H_0)}{p(\mathbf{z}|H_0)}.$$

We separately provide the numerator and denominators:

Numerator PDF: The numerator PDF of the J-function.

Denominator PDF: The denominator PDF of the J-function.

The simplest kind of feature transformation is an invertible transformation. While these are not useful for dimension reduction, they are important for feature conditioning. For invertible transformations, the J-function is just the absolute value of the determinant of the Jacobian matrix of the transformation. Thus,

$$J(\mathbf{x}, T) = |\det(\mathbf{J})|$$

where

$$\mathbf{J} = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} & \cdots \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \frac{\partial z_2}{\partial x_3} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

For invertible transformations, we provide the complete J-function only:

J-Function (Jacobian): The log of the determinant of the Jacobian matrix.

6.1 Log Transformation

An example of an invertible transformation is the log function. Consider the transformation

$$\boxed{\textbf{Feature Calculation: } z_i = \log(x_i), \;\; 1 \leq i \leq M.}$$

We have $dz/dx = 1/x$, thus $\log J = \log(1/x) = -\log x = -z$. For taking the logarithm of a vector of length M, we have

$$\boxed{\textbf{J-Function (Jacobian): } \log J = -\sum_{i=1}^{M} z_i.}$$

6.2 Variance Estimate

A very simple example of a class-specific module is the sample variance. Let $\mathbf{x}$ be a time-series of length N and let $\mathbf{z}$ be the variance estimate

$$\boxed{\textbf{Feature Calculation: } z = T(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} x_i^2.}$$

Let the reference hypothesis be

$$\boxed{\mathbf{H_0}: \text{Independent zero-mean Gaussian noise of variance 1.}}$$

Then the numerator of the J-function is

$$\boxed{\textbf{Numerator PDF: } \log p(\mathbf{x}|H_0) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\left(\sum_{i=1}^{N} x_i^2\right).}$$

Since $\mathbf{z}$ has the Chi-squared distribution with N degrees of freedom (scaled by $1/N$), the denominator of the J-function is

$$\boxed{\textbf{Denominator PDF: } \log p(\mathbf{z}|H_0) = \log N - \log\Gamma(N/2) - \frac{N}{2}\log 2 + \left(\frac{N}{2} - 1\right)\log(Nz) - \frac{Nz}{2}.}$$

6.3 Autocorrelation Function.

A very useful feature set in stationary time-series analysis is the autocorrelation function (ACF). The ACF coefficient of lag τ is an estimate of the mean or expected value of the product $x_t x_{t-\tau}$, which for stationary time-series, is independent of t. The ACF is the fundamental feature extraction behind many spectral estimation techniques with varying names such as linear predictive coding (LPC), autoregressive (AR) modeling, and reflection coefficients (RC). All of these methods are related and begin by estimating the ACF using a variety of methods. The benefit of AR modeling is that the spectral information can be boiled down to but a few coefficients which can hold spectral information with high resolution. The first $P+1$ ACF lags ($\tau = 0, 1 \ldots P$) are required for a P-th order AR model [21]. These ACF lags can then be transformed to RCs or AR coefficients using invertible transformations, thus they are equivalent from a modeling point of view. A good source of information on the topic is the book by Kay [21].

It may also be useful to use arbitrary ACF lags, rather than only the first $P+1$ lags. This is especially true when dealing with periodic time-series such as human voice, where the lag value at the pitch period is also of interest. Let $\mathbf{x} = [x_1, x_2 \ldots x_N]$ be a time-series of length N. We define the M-dimensional feature set $\mathbf{z}$ as the arbitrary ACF lags $k_1, k_2, \ldots k_M$. Thus, the feature calculation is

$$\boxed{\textbf{Feature Calculation: } \mathbf{z} = [r_{k_1}, r_{k_1} \ldots r_{k_M}], \text{ where } r_k = \frac{1}{N}\sum_{i=1}^{N} x_i x_{[i+k]_N},}$$

where the braces $[i-k]_N$ indicates modulo-N. These are known as the *circular* ACF estimates because of the modulo indexing. We choose this form of the ACF because it simplifies the analysis. A solution is available for arbitrary forms of the ACF based on quadratic forms [19], but is more complicated. As before, let the reference hypothesis be

$$\boxed{\mathbf{H_0}: \text{Independent zero-mean Gaussian noise of variance 1.}}$$

Then, as before, the numerator of the J-function is

$$\boxed{\textbf{Numerator PDF: } \log p(\mathbf{x}|H_0) = -\tfrac{N}{2}\log 2\pi - \tfrac{1}{2}\left(\textstyle\sum_{i=1}^{N} x_i^2\right).}$$

There is no known closed-form expression for the joint PDF of $\mathbf{z}$ under H_0, although a cumbersome but exact expression is available for the normalized statistics $\tilde{r}_k = r_k/r_0$ (See [18] section II.B). However, an approximation based on the saddlepoint approximation (SPA) [22] that is valid in the tails has been published. Specifically, in reference [18], section IV.B, the SPA for the scaled ACF estimates $\tilde{\mathbf{z}} = 2N\mathbf{z}$ are derived. The J-function denominator is thus,

$$\boxed{\textbf{Denominator PDF: } \log p(\mathbf{z}|H_0) = M\log(2N) + \log p(\tilde{\mathbf{z}}|H_0),}$$

where $p(\tilde{\mathbf{z}}|H_0)$ is from reference [18], section IV.B.

6.4 Contiguous ACF and Reflection Coefficients.

Reflection coefficients (RCs) are an alternate way of representing the information in an AR model. The RCs can be more convenient and easier to statistically model. Reflection coefficients (RCs) may be calculated from ACF estimates [21], and therefore we may use the results of Section 6.3 followed by a conversion to RCs. However, Section 6.3 is more general since it describes an approach that can handle arbitrary ACF lags; whereas the RCs are computed from a contiguous set of ACF lags (lags 0 through P). The use of contiguous ACF samples allows a different approach to analysis of the ACF features which is both instructive and useful for comparison purposes. If we use the circular ACF estimates as before, we can calculate the ACF samples by first computing the magnitude-squared DFT, then the inverse DFT. A third stage is necessary to convert to RCs and a fourth stage is used for further conditioning. The complete chain provided below has been found to be extremely versatile in modeling time-series. By segmenting the time-series, signals can be converted into sequences of feature vectors that can be statistically modeled using the a hidden Markov model (HMM). These feature sequences can also be converted back into time-series to validate the fidelity of the representation. As an additional check of model fidelity, the trained HMM can be used to generate random feature sequences, then converted into time-series for listening. Because of the versatility of CSM, each signal type can be represented using a particular choice of segment size and AR model order.

6.4.1 Stage 1: Magnitude-Squared DFT

In the first stage, we let $\mathbf{y} = [y_0, y_1 \ldots y_{N/2}]$ be the magnitude-squared DFT of $\mathbf{x}$,

$$\boxed{\textbf{Feature Calculation: } \quad y_k = \left|\sum_{i=1}^{N} x_i \exp\left\{-\frac{j2\pi(i-1)k}{N}\right\}\right|^2, \quad k = 0, 1 \ldots N/2.}$$

As before, we let the reference hypothesis be

$$\boxed{\mathbf{H_0}: \text{Independent zero-mean Gaussian noise of variance 1.}}$$

Also, as before, the numerator of the J-function is

$$\boxed{\textbf{Numerator PDF: } \log p(\mathbf{x}|H_0) = -\tfrac{N}{2}\log 2\pi - \tfrac{1}{2}\left(\textstyle\sum_{i=1}^{N} x_i^2\right).}$$

The DFT bins are independent under H_0, but not identically distributed. DFT bins 0 and $N/2$ are real-valued so y_k have the Chi-squared distribution with 1 degree of freedom scaled by N, which we denote by $p_0(y)$:

$$p_0(y) = \frac{1}{N\sqrt{2\pi}}\,(y_i/N)^{-1/2}\,\exp\left\{-\frac{y_i}{2N}\right\}.$$

DFT bins 1 through $N/2-1$ are complex so y_k have the Chi-squared distribution with 2 degrees of freedom scaled by $N/2$, which we denote by $p_1(y)$:

$$p_1(y) = \frac{1}{N}\,\exp\left\{-\frac{y_i}{N}\right\}.$$

The complete denominator PDF is

$$\boxed{\textbf{Denominator PDF: } \log p(\mathbf{y}|H_0) = \log p_0(y_0) + \textstyle\sum_{k=1}^{N/2}\log p_1(y_k) + \log p_0(y_{N/2}).}$$

6.4.2 Stage 2: Inverse DFT

In the second step, let $\mathbf{r} = [r_0, r_1 \ldots r_P]$ be the first $P+1$ ACF lags, which can be computed from $1/N$ times the first $P+1$ samples of the real part of the inverse DFT of $\mathbf{y}$. This may be written as

$$r_k = \frac{1}{N^2}\sum_{i=0}^{N/2}\epsilon_i\, y_i \cos\left\{\frac{2\pi i k}{N}\right\}, k = 0, 1 \ldots P,$$

where $\epsilon_i = 1$ for $i = 0, N/2$, and $\epsilon_i = 2$ for $i = 1, 2 \ldots N/2-1$. This may be written in the matrix form

$$\boxed{\textbf{Feature Calculation: } \mathbf{r} = \mathbf{C}'\,\mathbf{y},}$$

where matrix $\mathbf{C}$ is defined accordingly.

Now, for the first time, we use a reference hypothesis other than independent Gaussian noise. In fact, we use a floating reference hypothesis - one that depends upon the data sample. The use of a floating reference hypothesis and the constraints on how it may vary are discussed elsewhere [1]. The floating reference hypothesis is the AR spectrum corresponding to the ACF $\mathbf{r}$. Using the Levinson-Durbin recursion [21], we may transform $\mathbf{r}$ into the AR coefficients $\{a_1, a_2 \ldots a_P, \sigma^2\}$. The corresponding AR spectrum is written

$$y_k^r = \sigma^2\left|\sum_{k=0}^{P} a_k \exp\left\{-\frac{j2\pi i k}{N}\right\}\right|^2,$$

where the superscript "r" is a reminder that the AR spectrum depends on $\mathbf{r}$. We let our reference hypothesis, denoted by $H_0(\mathbf{r})$, be that the mean of $\mathbf{y}$ equals the AR spectrum $\mathbf{y}^r = [y_0^r, y_1^r \ldots y_{N/2}^r]$. For simplicity, we assume the elements of $\mathbf{y}$ are independent.

$$\boxed{\mathbf{H_0}(\mathbf{r}) : \text{That } \mathbf{y} \text{ has independent elements with mean } \mathrm{E}(\mathbf{y}) = \mathbf{y}^r.}$$

Under $\mathbf{H_0}(\mathbf{r})$, the elements of $\mathbf{y}$ are independent and Chi-squared with 1 or 2 degrees of freedom with mean y_k^r. Bins $k = 0, N/2$ are distributed according to

$$p_0(y, y^r) = \frac{1}{y^r\sqrt{2\pi}} \, (y_i/y^r)^{-1/2} \, \exp\left\{-\frac{y_i}{2y^r}\right\},$$

while bins 1 through $N/2 - 1$ are exponentially distributed according to

$$p_1(y, y^r) = \frac{1}{y^r} \, \exp\left\{-\frac{y_i}{y^r}\right\}.$$

In summary,

$$\boxed{\textbf{Numerator PDF: } \log p(\mathbf{y}|H_0(\mathbf{r})) = \log p_0(y_0, y_0^r) + \sum_{k=1}^{N/2} \log p_1(y_k, y_k^r) + \log p_0(y_{N/2}, y_{N/2}^r).}$$

We may use the central limit theorem (CLT) to approximate the PDF of $\mathbf{r}$ under $H_0(\mathbf{r})$ because the mean of $\mathbf{r}$ under $H_0(\mathbf{r})$ is very nearly $\mathbf{r}$ itself. Under $H_0(\mathbf{r})$, the elements of $\mathbf{y}$ are independent with mean $\mathbf{y}^r$ and diagonal covariance $\mathbf{\Sigma}_y^r$ given by

$$\mathbf{\Sigma}_y^r(i,i) \triangleq \mathcal{E}((y_i - y_i^r)^2|H_0(\mathbf{r})) = \begin{cases} 2(y_i^r)^2, & i = 0, N/2 \\ \\ (y_i^r)^2, & 1 \le i \le N/2 - 1. \end{cases}$$

Under $H_0(\mathbf{r})$, $\mathbf{r}$ has mean

$$\mathbf{r}^r \triangleq \mathrm{E}(\mathbf{r}|H_0(\mathbf{r})) = \mathbf{C}' \, \mathbf{y}^r,$$

and covariance

$$\mathbf{\Sigma}_r^r = \mathbf{C}' \, \mathbf{\Sigma}_y^r \, \mathbf{C}.$$

$$\begin{aligned} \log p(\mathbf{r}|H_0(\mathbf{r})) \quad &= \quad -\tfrac{(P+1)}{2} \, \log(2\pi) - \tfrac{1}{2} \, \log|\det(\mathbf{\Sigma}_r^r)| \\ & \qquad -\tfrac{1}{2} \, (\mathbf{r} - \mathbf{r}^r)'(\mathbf{\Sigma}_r^r)^{-1}(\mathbf{r} - \mathbf{r}^r). \end{aligned} \tag{10}$$

If we we make the approximation $\mathbf{r}^r \simeq \mathbf{r}$, we obtain

$$\boxed{\textbf{Denominator PDF: } \log p(\mathbf{r}|H_0(\mathbf{r})) = -\tfrac{(P+1)}{2} \, \log(2\pi) - \tfrac{1}{2} \, \log|\det(\mathbf{\Sigma}_r^r)|.}$$

6.4.3 Stage 3: Conversion to RCs.

The conversion from ACF to RCs is an invertible transformation that is characterized by a Jacobian matrix. The determinant of this matrix is the J-function of the transformation.

$$\boxed{\textbf{Feature Calculation: } \mathbf{r} \longrightarrow \text{ (Levinson recursion for reflection coefficients) } \longrightarrow \mathbf{k},}$$

where $\mathbf{r}$ is the ACF vector, $\mathbf{r} \triangleq [r_0, r_1, \ldots r_P]$, and $\mathbf{z}$ is the vector of reflection coefficients augmented by the variance (zero-th lag ACF sample),

$$\mathbf{k} \triangleq [r_0, k_1 \ldots k_P].$$

Note that we use r_0 and *not* the AR prediction error variance σ_0^2. This transformation is invertible and is characterized by the Jacobian

$$\boxed{\textbf{J-function (Jacobian): } \log J = -P \log(r_0) + \sum_{i=1}^{P-1} (P - i) \log(1 - k_i^2).}$$

6.4.4 Stage 4: Log-Bilinear Transformation.

Although the RCs have desirable properties as features, they are subject to the limit $|k_i| < 1$ which produces a discontinuity in the PDF. As a result, the PDF can be difficult to estimate using so-called non-parametric PDF estimators such as Gaussian mixtures. To obtain more Gaussian behavior, the log-bilinear transformation is recommended (thanks to S. Kay for recommending this).

$$\textbf{Feature Calculation: } k_i' = \frac{\log(1-k_i)}{\log(1+k_i)}, \quad 1 \le i \le P, \quad r_0' = \log(r_0)$$

This transformation is invertible and is characterized by the Jacobian

$$\textbf{J-function (Jacobian): } \log J = r_0' - \sum_{i=1}^{P} \log\left(\frac{2}{1-k_i^2}\right).$$

7 Experimental Validation

A very important question in developing a class-specific classifier is how to validate the analysis of a feature transformation. Because the numerator and denominator PDFs of the J-function are often evaluated in the far tails, we can never know if these PDF values are correct by histogram techniques. In Section 6.3 and 6.4 (up to stage 2), two method are presented for calculating the J-function for ACF samples. It may be verified that for contiguous ACF samples, the two approaches produce exactly the same features. The J-function values produced by the two methods are very close, but not exactly the same. Such comparisons are reassuring but are not a complete test and cannot be made for all problems. The following approach is a complete end-to-end test that has proved to be very useful.

Validation of the feature modules amounts to validating the PDF projection theorem (3). To validate equation (3), we design a hypothesis $H_{\rm v}$ for which we know the PDF $p(\mathbf{x}|H_{\rm v})$ exactly and for which we can create a large amount of synthetic raw data samples. We convert the synthetic data to features which we use to obtain the PDF estimate $\hat{p}(\mathbf{z}|H_{\rm v})$. Using this estimate in equation (3), we obtain an estimate $p_p(\mathbf{x}|H_{\rm v})$. To validate the result, we plot the projected PDF values $p_p(\mathbf{x}|H_{\rm v})$ on one axis and the exact values $p(\mathbf{x}|H_{\rm v})$ on the other axis for each sample of synthetic data. The points should lie near the $y = x$ line. An example is shown in Figure 10 where we tested the chain of four feature modules in Section 6.4. The synthetic data used in the experiment were 100 time-series of independent Gaussian noise of variance 100 and length 4096. The features were computed using an AR model order of 4 with segmentation to 64-sample segments, thus producing 64 independent feature vectors of dimension 5 per sample. A Gaussian mixture model was used to statistically model the features.

8 A Class-Specific Time-Series Classifier Using Reflection Coefficients and HMM.

We can put to use the material thus-far discussed to arrive at a fully modular, extremely versatile class-specific classifier. A functional block-diagram of this classifier is provided in Figure 11. A given time-series is processed by each class-model to arrive at a raw-data log-likelihood for the class. Each block labeled "RC(P)" computes the reflection coefficients of order P from the associated time-series segment. The figure shows two class-models employing different segmentation lengths as well as different model orders. The log-correction terms of all the segments are added together and the aggregate correction term is added to the HMM log-likelihood (from the forward procedure [23]) to

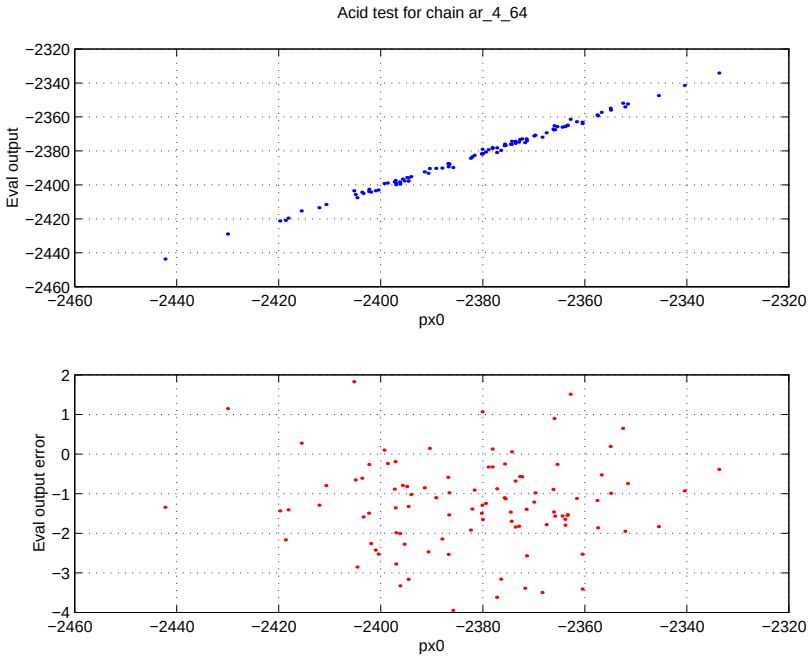

Figure 10: Example of validation test results for 4-th order autoregressive features (Section 6.4 stages 1-4). The upper graph shows theoretical log-PDF values on the x-axis and PDF projection theorem values on the y-axis for 100 synthetic events. The lower graph shows the errors.

arrive at the final raw data log-likelihood for the class. The segmentation sizes and model orders are optimized for each class individually, eliminating the need to "compromise".

Each "RC(P)" block is composed of a series of modules implementing ACF calculation followed by conversion to RCs, and ending with feature conditioning by the log-bilinear transformation. This may be implemented by the three modules described in Sections 6.3, 6.4.3, and 6.4.4. Alternatively, the four modules of Sections 6.4.1, 6.4.2, 6.4.3, and 6.4.4 will produce virtually identical features and J-function values. This classifier has the added benefit that the models may be validated by re-synthesis of time-series from features (either computed from actual data or generated at random by the HMM).

It should be stressed that we are not limited to using RC features and HMM PDF models. As long as care is taken in computing the correction terms, any feature set and any statistical model may be employed. Straight DFT features may be preferable to RC features for sinusoidal signals. Wavelet features may be preferable for certain other types of signals. A particularly good set of features for DFT (or wavelet processing) is to save the largest M bins and residual energy. The correction term for this feature set has been worked out by Nuttall. [24]. Nuttall has also derived the correction term for features that may be written as a set of inter-dependent quadratic forms, [25].

9 Conclusion

Previous to the class-specific method, practitioners in image or signal classification had no guidance from classical theory in dealing with complex problems. The incomplete theory forced practitioners to think of feature extraction from the point of view of class separability. This flawed paradigm led the practitioner down the slippery slope of high dimensionality. Now that the reader has been introduced

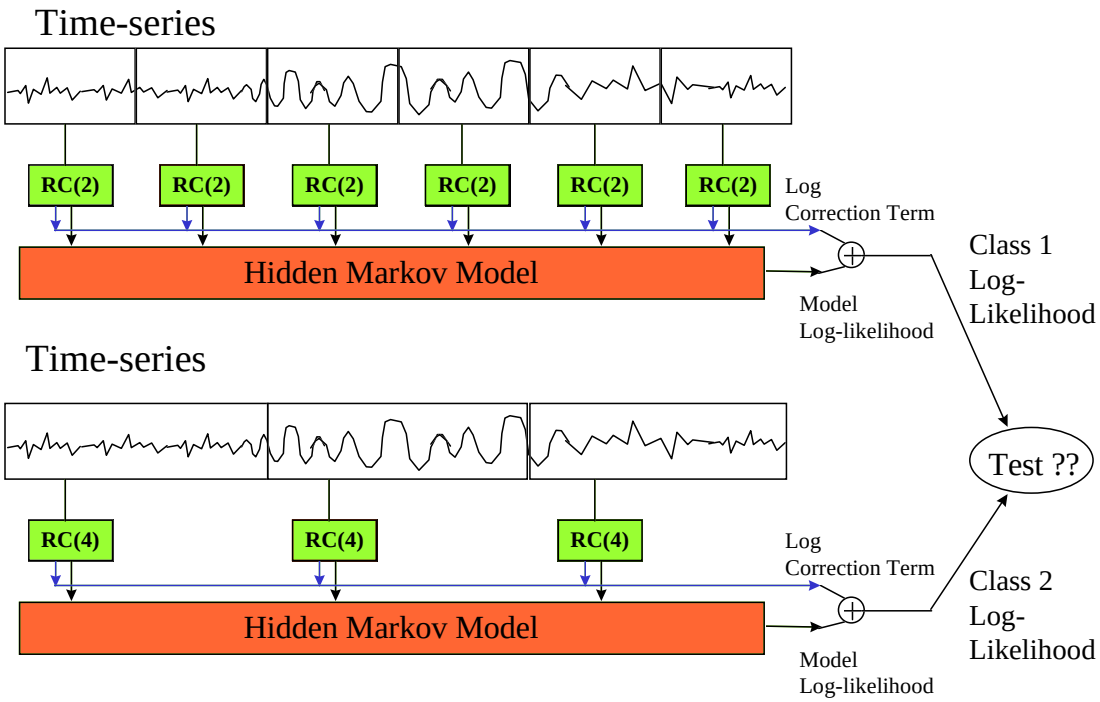

Figure 11: Block diagram of an HMM and RC-based class-specific classifier. A given time-series is processed by each class-model to arrive at a raw-data log-likelihood for the class. Each block labeled "RC(P)" computes the P-th order reflection coefficients from the corresponding time-series segment and is implemented by a series of modules (See text).

to the fundamental concepts of classification theory using class-specific features, he or she has the tools necessary to attack classification problems one class at a time, capturing all the necessary information in the features and not being forced to "make-do" with features that are general enough for *all* classes, but not sufficient for *any* class. The examples provided are enough to build a simple, yet effective class-specific time-series classifier.

References

[1] P. M. Baggenstoss, "The PDF projection theorem and the class-specific method," *Submitted to IEEE Trans. Signal Processing*, 2002.

[2] Duda and Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973.

[3] C. J. Stone, "Optimal rates of convergence for nonparametric estimators," *Annals of Statistics*, vol. 8, no. 6, pp. 1348–1360, 1980.

[4] D. W. Scott, *Multivariate Density Estimation*. Wiley, 1992.

[5] T. Balachander and R. Kothari, "Oriented soft localized subspace classification," *ASSP-99*, vol. 2, pp. 1017–1020, 1999.

[6] Frimpong-Ansah, K. Pearce, D. Holmes, and W. Dixon, "A stochastic/feature based recogniser and its training algorithm," *ICASSP-89*, vol. 1, pp. 401–404, 1989.

[7] S. Kumar, J. Ghosh, and M. Crawford, "A versatile framework for labeling imagery with large number of classes," in *Proceedings of the International Joint Conference on Neural Networks*, (Washington, D.C.), pp. 2829–2833, 1999.

[8] S. Kumar, J. Ghosh, and M. Crawford, "A hierarchical multiclassifier system for hyperspectral data analysis," in *Multiple Classifier Systems* (J. Kittler and F. Roli, eds.), pp. 270–279, Springer, 2000.

[9] H. Watanabe, T. Yamaguchi, and S. Katagiri, "Discriminative metric design for robust pattern recognition," *IEEE Trans. Signal Processing*, vol. 45, no. 11, pp. 2655–2661, 1997.

[10] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *PAMI*, vol. 19, pp. 711–720, July 1997.

[11] P. M. Baggenstoss, "Class-specific feature sets in classification," in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC), 1998*, pp. 413–416, National Institute of Standards and Technology, 1998.

[12] P. M. Baggenstoss, "Class-specific features in classification.," *IEEE Trans Signal Processing*, pp. 3428–3432, December 1999.

[13] S. Kay, "Sufficiency, classification, and the class-specific feature theorem," *IEEE Trans. Information Theory*, vol. 46, pp. 1654–1658, July 2000.

[14] P. M. Baggenstoss, "A theoretically optimum approach to classification using class-specific features.," *Proceedings of ICPR, Barcelona*, 2000.

[15] P. M. Baggenstoss, "A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces.," *IEEE Trans. Speech and Audio*, pp. 411–416, May 2001.

[16] P. M. Baggenstoss, "The chain-rule processor: Optimal classification through signal processing," *Proceedings of ICPR, Quebec*, August 2002.

[17] E. H. Lehmann, *Testing Statistical Hypotheses*. New York: Wiley, 1959.

[18] S. M. Kay, A. H. Nuttall, and P. M. Baggenstoss, "Multidimensional probability density function approximation for detection, classification and model order selection," *IEEE Trans. Signal Processing*, pp. 2240–2252, Oct 2001.

[19] A. H. Nuttall and P. M. Baggenstoss, "The joint distributions for two useful classes of statistics with applications to classification and hypothesis testing," *Submitted to IEEE Trans. Signal Processing*, 2002.

[20] C. M. Bishop, M. Svensen, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, pp. 215–234, 1998.

[21] S. Kay, *Modern Spectral Estimation: Theory and Applications*. Prentice Hall, 1988.

[22] O. E. Barndorff-Nielsen and D. R. Cox, *Asymptotic Techniques for Use in Statistics*. Chapman and Hall, 1989.

[23] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, February 1989.

[24] A. H. Nuttall, "Joint probability density function of selected order statistics and the sum of the remaining random variables," *NUWC Technical Report 11345*, October 2001.

[25] A. H. Nuttall, "Saddlepoint approximation and first-order correction term to the joint probability density function of M quadratic and linear forms in K Gaussian random variables with arbitrary means and covariances," *NUWC Technical Report 11,262*, December 2000.