PERGAMON

Pattern Recognition 34 (2001) 139–150

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Local discriminative learning for pattern recognition

Jing Peng[1], Bir Bhanu*

Center for Research in Intelligent Systems, University of California, Riverside, CA 92521, USA

Received 13 January 1999; received in revised form 21 June 1999; accepted 9 August 1999

Abstract

Local discriminative learning methods approximate a target function (a posteriori class probability function) directly by partitioning the feature space into a set of local regions, and appropriately modeling a simple input–output relationship (function) in each one. This paper presents a new method for judiciously partitioning the input feature space in order to accurately represent the target function. The method accomplishes this by approximating not only the target function itself but also its derivatives. As such, the method partitions the input feature space along those dimensions for which the class probability function changes most rapidly, thus minimizing bias. The efficacy of the method is validated using a variety of simulated and real-world data. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Discriminative learning; Feature relevance; Local learning; Recursive partitioning; Statistical pattern classification

1. Introduction

In pattern classification, a feature vector $\mathbf{x} \in \Re^p$, representing an object, is assumed to be in one of J classes $\{i\}_{i=1}^J$, and the objective is to build classifiers that assign $\mathbf{x}$ to the correct class. For a given cost matrix L_{ij}, where L_{ij} denotes the cost associated with assigning $\mathbf{x}$ to class i when, in fact, $\mathbf{x}$ is in class j, the Bayes classifier assigns $\mathbf{x}$ to class i such that the expected cost

$$L_i = \sum_{j=1}^{J} L_{ij} \Pr(j|\mathbf{x}) \tag{1}$$

is minimized. Here $\Pr(j|\mathbf{x})$ is the probability that $\mathbf{x}$ is in class j given $\mathbf{x}$. If all misclassifications induce an equal cost, it can be shown that the Bayes classifier reduces to

$$i^* = \arg\max_{1 \leqslant i \leqslant J} \Pr(i|\mathbf{x}). \tag{2}$$

That is, the Bayes classifier assigns $\mathbf{x}$ to the most probable class. It is clear that in order to apply the Bayes classifier (Eq. (2)), the unknown class (conditional) probabilities $\{\Pr(i|\mathbf{x})\}_{i=1}^J$ must be estimated.

Two general learning approaches exist for constructing such pattern classifiers [1]. The *informative* approach models the class conditional densities and assign $\mathbf{x}$ to the most likely class by examining the likelihood of each class generating the features via the Bayes rule

$$\Pr(i|\mathbf{x}) = p(\mathbf{x}|i)\Pr(i) \Big/ \sum_{j=1}^{J} p(\mathbf{x}|j)\Pr(j), \tag{3}$$

where $p(\mathbf{x}|i)$ is the conditional density for $\mathbf{x}$ given class i. An example of this approach is the learning vector quantization methods of Kohonen [2]. In practice, $p(\mathbf{x}|i)$'s are often assumed to take a known form, such as the Gaussian. Then training data within each class can be used to estimate the mean and covariance matrix, and class proportions within the training data can be used to estimate the priors $\Pr(i)$. These two estimates can be substituted into Eq. (3) to obtain a posterior probability for each class for the given input $\mathbf{x}$.

The *discriminative* approach, on the other hand, assigns $\mathbf{x}$ to the most likely class by directly estimating the class conditional probabilities $\{\Pr(i|\mathbf{x})\}_{i=1}^J$ based on regression analysis, such as the least-squares estimate, from

*Corresponding author. Tel.: +1-909-787-3954; fax: +1-909-787-3188.

E-mail addresses: jp@vislab.ucr.edu (J. Peng), bhanu@vislab.ucr.edu (B. Bhanu).

[1] Current address: Computer Science Department, Oklahoma State University, Stillwater, OK 74078, USA.

training data. This is possible because these quantities are the only things that need be computed in order to make correct classification. Examples of this approach include neural networks [3], nearest-neighbor methods [4–7] and decision tree methods [8–15].

This paper is concerned with local discriminative learning methods for building pattern classifiers. These methods first partition the input space into a set of local regions. They then construct a simple pattern classifier within each region. For a given input, classification is made by first locating the region the input is in and then classifying the input using the classifier associated with that region. The major appeals for such methods reside in their ability to perform classification by a sequence of simple tests whose meanings are easy to understand and their computational efficiency.

One of the keys to constructing successful pattern classifiers using these methods is to determine how the input feature space should be partitioned. The original contributions of this paper are to provide a novel partitioning criterion that allows these methods to build high performance classifiers and to compare its performance with that of other existing methods. In particular, the partitioning criterion, when maximized, produces an approximation to not only the class probability function itself but also its derivatives, which often results in better performance.

Note that this paper is focused on a class of local learning methods that may be called *univariate*, because only one input feature at each node is chosen for partitioning. However, there are methods [8,10,12,14,15] that compute new features for partitioning that are a linear combination of the input features. These methods are called *multivariate*. One would prefer such a method when the underlying class is defined by a polygonal space partitioning. While multivariate decision trees can perform significantly better than univariate ones, this improvement is at the expense of increased computation. It should be clear later that the technique presented here can benefit multivariate decision trees as well.

While local discriminative learning methods discussed above show promise in classification with high-dimensional inputs, a new emerging class of appearance-based methods [14,16,17] in computer vision have demonstrated effectiveness in object recognition tasks involving image data with high dimensionality. The key to the success of appearance-based methods lies in their ability to locate subspaces in which input data reside, thereby mitigating the "curse-of-dimensionality" [18]. We do not discuss further issues concerning appearance-based methods in the rest of the paper, except to state that decision tree techniques can be employed in appearance-based systems to achieve superior performance, as demonstrated in [8,14,15].

The paper is organized as follows. We first describe local discriminative learning for pattern classification. We then introduce a novel differential splitting criterion for achieving effective local discriminative learning. After that, we present several experimental studies evaluating the efficacy of our method and its use for pattern classification using both simulated and real-world data. Finally, we conclude with the key aspects of this paper and its contributions.

2. Local discriminative learning

One broad class of methods for estimating $\{\Pr(i|\mathbf{x})\}_{i=1}^{J}$, based on supervised learning, is to treat the class label $C \in \{1, \ldots, J\}$ at query $\mathbf{x}$ (test case) as a random variable from a distribution with the probabilities $\{\Pr(i|\mathbf{x})\}_{i=1}^{J}$. Following Friedman's notation [11], C at $\mathbf{x}$ can be characterized by introducing an additional variable, y, as

$$y_i|_{\mathbf{x}} = \begin{cases} 1, & C|_{\mathbf{x}} = i, \\ 0, & C|_{\mathbf{x}} \neq i. \end{cases} \tag{4}$$

This gives rise to

$$f_i(\mathbf{x}) \overset{def}{=} \Pr(i\,|\,\mathbf{x}) = \Pr(y_i = 1|\mathbf{x}) = E(y_i|\mathbf{x}). \tag{5}$$

Note that

$$0 \leqslant f_i(\mathbf{x}) \leqslant 1$$

and

$$\sum_{i=1}^{J} f_i(\mathbf{x}) = 1.$$

The learning methods can then be applied to estimate each $f_i(\mathbf{x})$ from the corresponding training data

$$\{x_k, y_i\}_{k=1}^{n}, \tag{6}$$

where $i = 1, \ldots, J$. These estimates, $\hat{f}_i(\mathbf{x})$, can then be plugged into Eq. (2) to obtain the optimal classification for input $\mathbf{x}$. This learning paradigm has been a basis for many techniques developed in neural networks, machine learning, and memory-based regression for pattern classification. It is the target function $f_i(\mathbf{x})$, hence $\Pr(i|\mathbf{x})$, that we seek to approximate here in order to achieve optimal classification performance. In what follows, we drop the subscript i from $f_i(\mathbf{x})$ in Eq. (5) for clarity.

2.1. Local learning

The principle of local learning is given by the Taylor Theorem. According to this, any complex function can be approximated by a simple function, such as a linear function, in the vicinity of a given query. As such, some of the difficulties often associated with global function approximation methods, such as negative spatial crosstalk [4], can be avoided by local approximation, thereby improving performance.

Given the above training data (Eq. (6)), local learning methods [4,5,9,13,19–23] attempt to learn the target function $f(\mathbf{x})$, Eq. (5), by first partitioning the space $\Re^p$ of input variables into a set of local regions

$$\{R_m\}_{m=1}^{M}$$

where

$$R_m \subset \Re^p \quad \text{and} \quad \bigcup_{m=1}^{M} R_m = \Re^p.$$

One such procedure [9,13] is shown in Fig. 1. Here function *SplitDecision*() computes the direction α^* and the split point sp^* by optimizing some split criterion based on the training data in the region R under consideration, and $SplitFun(\mathbf{x}, \alpha)$ is a splitting function of $\mathbf{x}$ and α, which is usually linear:

$$SplitFun(\mathbf{x}, \alpha) = \alpha^t \mathbf{x}. \tag{7}$$

The region R can thus be split by thresholding the splitting function along the direction α^* at the split point sp^*. The procedure terminates when a stopping criterion, *Stop*(), is satisfied. Note that in practice, split directions, α, to be optimized are usually specified. Furthermore, they are often restricted to the coordinate axes for computational simplicity. That is,

$$\alpha \in \{\mathbf{e}_1, \ldots, \mathbf{e}_p\}, \tag{8}$$

where $\mathbf{e}_i$ is a unit vector along the ith coordinate. Also, the split point sp is usually taken to be the median or mean value of $\{\alpha^t \mathbf{x} : \mathbf{x} \in R\}$. In this case, *SplitDecision*() simply chooses one of the coordinate axes as the split direction.

We can characterize the "local" notion of a region R_m more formally by its size according to

$$size(R_m) = \underset{\mathbf{x},\mathbf{x}' \in R_m}{ave} \|\mathbf{x} - \mathbf{x}'\|, \tag{9}$$

where *ave* is an averaging function, and $\|\cdot\|$ represents a vector norm. That is, smaller values of $size(R_m)$ indicate that R_m is more "local". As pointed out in Ref. [11], regions can be sufficiently described by their locations and sizes for $p = 1$, i.e. $\mathbf{x} \in \Re$. For $p \geqslant 2$, however, the regions have "shape" (s) as well as location and size. One way to define the shape of a region is by its size along all directions in the input feature space, that is, by the function [11]

$$s_m(\alpha) = \underset{\mathbf{x} \in R_m}{ave} |\alpha^t(\mathbf{x} - \mathbf{u}_m)|, \tag{10}$$

where α is a unit vector in $\Re^p$ and $\mathbf{u}_m$ is the center of R_m. Clearly, there are other ways to define the shape of a region. However, Eq. (10) appears to be a simple one and is adequate to serve as a basis for deriving new splitting criteria developed in Section 3. Region shapes play an important role in developing successful local learning methods for pattern classification, as we shall see later.

These methods learn a separate approximator $\hat{f}_m(\mathbf{x})$ individually in each local region. These local approximators are usually low order polynomials (see Ref. [24] for other possibilities). The accuracy of learning can be measured by the typical quadratic error function

$$Err(f(\mathbf{x}), \hat{f}(\mathbf{x})) = |f(\mathbf{x}) - \hat{f}(\mathbf{x})|^2. \tag{11}$$

The goal, therefore, is to develop learning methods such that the expectation of Eq. (11) is minimized in the limit.

Local learning methods have shown promise in low-dimensional settings [7,11,21,22]. However, these methods tend to be less effective as dimensionality p increases. This is a result of the so called "curse-of-dimensionality" [18], which in essence states that the expectation of Eq. (11) can remain large in problems with high dimensional feature spaces even for a reasonably large number of training data.

Given the fact that the expectation of the error (11) can be decomposed into

$$\begin{aligned} E[f(\mathbf{x}) - \hat{f}(\mathbf{x})]^2 &= E[f(\mathbf{x}) - E\hat{f}(\mathbf{x}) + E\hat{f}(\mathbf{x}) - \hat{f}(\mathbf{x})]^2 \\ &= [f(\mathbf{x}) - E\hat{f}(\mathbf{x})]^2 + E[\hat{f}(\mathbf{x}) - E\hat{f}(\mathbf{x})]^2, \end{aligned} \tag{12}$$

where the first term in Eq. (12) is the squared bias and the second term is the variance. Thus, in order to minimize Eq. (12) one must minimize the sum of squared bias and variance. However, these are competing goals. Minimizing squared bias demands small region size. On the other hand, small regions tend to create large variance. Techniques, such as recursive covering [11] and "soft" partitioning [25], exist that accommodate, to the extent possible, these competing demands. The goal of this paper is to introduce a novel technique that minimizes squared bias by choosing region shape (Eq. (10)) more judiciously. It represents an attempt to mitigate the curse-of-dimensionality.

```
Partition($R$)
    if NOT $Stop(R)$ then $(R_l, R_r) \leftarrow$ Split($R$)
        Partition($R_l$)
        Partition($R_r$)
    return partition of $R$
Split($R$)
    $(\alpha^*, sp^*) \leftarrow SplitDecision(\{\mathbf{x}, y | \mathbf{x} \in R\})$
    for all $\mathbf{x} \in R$ do
        if $SplitFun(\mathbf{x}, \alpha^*) \leq sp^*$ then $\mathbf{x} \in R_l$
        if $SplitFun(\mathbf{x}, \alpha^*) > sp^*$ then $\mathbf{x} \in R_r$
    return $(R_l, R_r)$
```

Fig. 1. A recursive partitioning procedure.

3. Splitting criteria

In order to produce a good solution to the classification problem based on local learning, a splitting strategy is required. Such a strategy, when combined with recursive partitioning, selects region shape (Eq. (10)) such that $\hat{f}_m(\mathbf{x})$ approximates the target $f(\mathbf{x})$ (Eq. (5)) well within each region, thereby minimizing the overall error function (Eq. (11)). It is a step where care must be taken in order to achieve optimal pattern classification in a high-dimensional feature space.

3.1. Residual splitting

It is argued in Ref. [11] that the optimal shape for a region R_m should be governed by the properties of the target function $f(\mathbf{x})$ within it. Specifically, its shape in any direction α, Eq. (10), should be inversely proportional to the rate of change of the absolute bias in that direction

$$s_m(\alpha) \sim 1 \Big/ \underset{\mathbf{x}\in R_m}{ave} \left(\alpha^{\mathrm{t}} \frac{\partial}{\partial \mathbf{x}} |f(\mathbf{x}) - \hat{f}_m(\mathbf{x})| \right). \tag{13}$$

It is clear that the shape depends on the target function $f(\mathbf{x})$, the local approximator $\hat{f}_m(\mathbf{x})$ used, and the location $\mathbf{x}$ of the region in the input feature space. Such dependence on the query location is considered highly desirable because the resulting shape for a region is adapted to each individual query, instead of using the same shape for all regions. The partition strategy suggested by Eq. (13) states that the input feature space should be split along directions where the target $f(\mathbf{x})$ changes most rapidly. That is, the input feature space should be constricted along directions of large variations of $f(\mathbf{x})$, and enlongated in directions of small variations.

Based on Eq. (13) the splitting criterion for recursive partitioning proposed in Ref. [11] is the following. Let

$$\{r_i = y_i - \hat{f}_R(\mathbf{x}_i) \,|\, \mathbf{x}_i \in R\} \tag{14}$$

be the residuals resulting from the approximator $\hat{f}_R(\mathbf{x})$ fit to the data in R. Then the criterion (called *residual criterion* here) to be maximized for choosing the split direction α is

$$\mathit{ResidualCri}(\alpha) = \left| \underset{\mathbf{x}_i\in R}{ave} \{r_i | \mathbf{x}_i \leqslant s(\alpha)\} \right| + \left| \underset{\mathbf{x}_i\in R}{ave} \{r_i | \mathbf{x}_i > s(\alpha)\} \right|, \tag{15}$$

where $s(\alpha)$ is the split point (the median of $\{\alpha^{\mathrm{t}}\mathbf{x}_i \,|\, \mathbf{x}_i \in R\}$). Maximizing Eq. (15) produces the direction in which the points to be removed (as part of another partition) deviate most from the local approximation in the region being split. That is, the split direction given by Eq. (15) is the direction along which the local approximator $\hat{f}_R(\mathbf{x})$ is the worst in approximating the target function in R. Therefore, splitting along it can be expected to give rise to better approximation to the target function in the resulting partitions. Note that the split criterion, Eq. (15), is motivated by Eq. (13) and represents a computationally feasible approximation to the simple strategy implied by Eq. (13). It has been demonstrated that the residual splitting criterion achieved impressive performance in a variety of target functions [11].

3.2. Limitation of residual splitting

Let us consider the following quadratic target function defined over the two-dimensional input space $\mathbf{x} \in [-4, 4]^2$

$$f(\mathbf{x}) = -x_1^2. \tag{16}$$

The target function $f(\mathbf{x})$ is clearly (radial) asymmetric in the input features as shown in Fig. 2. That is, for an arbitrary rotation matrix $\mathbf{R}$ (i.e., the column vectors of $\mathbf{R}$ are orthogonal.),

$$f(\mathbf{x}) \neq f(\mathbf{R}\mathbf{x}).$$

Furthermore, the target function $f(\mathbf{x})$ changes rapidly along the X_1-axis while remaining constant along X_2 for a given x_1. Suppose one uses a local constant approximator (zero-degree polynomial or mean value) in each region. Then it can be shown that the optimal recursive split should be carried out along X_1 only. Assume we are given the following training data:

$$\mathbf{x}_1 = (-3, -2)^{\mathrm{t}} \quad y_1 = -9$$
$$\mathbf{x}_2 = (-1, -2)^{\mathrm{t}} \quad y_2 = -1$$
$$\mathbf{x}_3 = (1, -2)^{\mathrm{t}} \quad y_3 = -1$$
$$\mathbf{x}_4 = (3, -2)^{\mathrm{t}} \quad y_4 = -9,$$
$$\mathbf{x}_5 = (-3, +2)^{\mathrm{t}} \quad y_5 = -9$$
$$\mathbf{x}_6 = (-1, +2)^{\mathrm{t}} \quad y_6 = -1$$
$$\mathbf{x}_7 = (1, +2)^{\mathrm{t}} \quad y_7 = -1$$
$$\mathbf{x}_8 = (3, +2)^{\mathrm{t}} \quad y_8 = -9.$$

Assume further, without loss of generality, that the directions to be optimized are taken to be the original coordinate axes and that the split points are taken to be mean values along each axis. Note that the assumptions we are making here do not in any way introduce bias either against or in favor of the residual splitting criterion, Eq. (15). They simply serve to simplify the computational process. When the split criterion is applied to the above data, it fails to yield the optimal direction (X_1 in this case) along which the input space should be split. In fact, with $\hat{f}_R(\mathbf{x}) = -5$ for all $\mathbf{x}$, we have

$$\{r_1 = -4, r_2 = 4, r_3 = 4, r_4 = -4, r_5 = -4,$$
$$r_6 = 4, r_7 = 4, r_8 = -4\}.$$

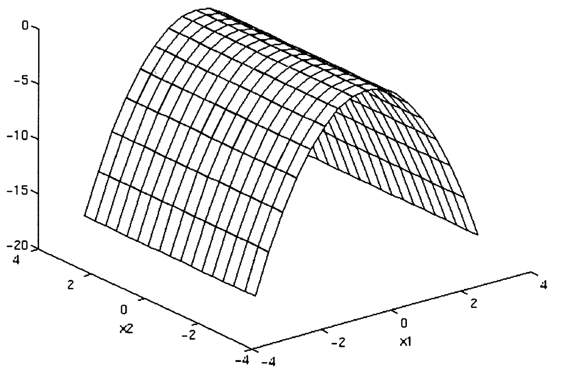

Fig. 2. A quadratic function.

It follows that

$$ResidualCri(\mathbf{e}_1) = ResidualCri(\mathbf{e}_2) = 0.$$

where $\mathbf{e}_i$ is a unit vector along the ith input coordinate. That is, Eq. (15) cannot distinguish high differential relevance between the two input variables.

While the target function presented here is an especially simple one, the local constant approximator is quite general. The above example is meant to illustrate the problems involved. It demonstrates that the residual splitting criterion is quite limited in producing the desired goal in a simple setting. Furthermore, while computationally simple, it is far from approximating the split strategy suggested by Eq. (13). Better criteria that are sensitive to input differential relevance in input variables must be sought.

3.3. *Differential splitting*

The optimal partition for any region should be driven by the properties of the target function $f(\mathbf{x})$ within it, particularly the derivatives (first order), as indicated by the split strategy suggested by Eq. (13). In this sense, the residual splitting criterion presented in the earlier section represents only a zero-order approximation to that strategy in that it finds a direction along which the points to be removed (to create a new partition) minimize the residuals from the local fit to the data within the region being split. However, it is possible that even though the input variables have high differential relevance, the average residual remains the same along the directions under consideration, as evidenced by the target function Eq. (16) in the above example.

The new splitting criterion we propose here represents a computationally feasible approximation to the split strategy suggested by Eq. (13) and is a generalization of residual splitting (Eq. (15)). It attempts to capture differential relevance in the input variables by estimating the (first order) derivatives of target functions. As will be shown later the new criterion is effective in those situations where differential relevance exists.

From the Taylor series expansion, for a sufficiently small δ

$$\begin{aligned} f(\mathbf{x}+\delta\mathbf{e}_i) &\simeq f(\mathbf{x}) + \delta\mathbf{e}_i^t g(\mathbf{x}) + \tfrac{1}{2}\delta^2\mathbf{e}_i^t G(\mathbf{x})\mathbf{e}_i \\ &= f(\mathbf{x}) + \delta[g]_i(\mathbf{x}) + \tfrac{1}{2}\delta^2[G]_{ii}(\mathbf{x}), \end{aligned} \tag{17}$$

where $\mathbf{e}_i$ is the unit vector along the ith coordinate direction, $[g]_i$ is the ith element of g and $[G]_{ii}$ is the iith element of G — the Hessian matrix of size $p \times p$. Thus, $[g]_i$ can be approximated by $[\hat{g}]_i$

$$[\hat{g}]_i(\mathbf{x}) = \frac{f(\mathbf{x}+\delta\mathbf{e}_i) - f(\mathbf{x})}{\delta} - \frac{1}{2}\delta[G]_{ii}(\mathbf{x}). \tag{18}$$

When the second term involving $[G]_{ii}$ is ignored, which is the second-order term in δ in the Taylor series, we have the *forward difference approximation* that is exact for linear functions. The alternative, *central difference approximation*, neglects only third-order terms in δ and is, therefore, exact for quadratic functions [26]. However, the increase in accuracy is at the expense of increased computation.

From Eq. (18), the finite difference approximation to the first-order derivatives of the target function $f(\mathbf{x})$, along input dimension i at $\mathbf{x}$ in the region R being split, can be given by

$$[\hat{g}]_i(\mathbf{x}) = \frac{y - y'}{[\mathbf{x}]_i - [\mathbf{x}']_i}, \tag{19}$$

where $\mathbf{x}, \mathbf{x}' \in R$, $[\mathbf{x}]_i \neq [\mathbf{x}']_i$, $0 < |[\mathbf{x}]_i - [\mathbf{x}']_i| \leqslant \delta$ and $\|\mathbf{x} - \mathbf{x}'\|_\infty \leqslant \theta$. Here δ and θ are the procedural ("meta") parameters. The conditions

$$0 < |[\mathbf{x}]_i - [\mathbf{x}']_i| \leqslant \delta \quad \text{and} \quad \|\mathbf{x} - \mathbf{x}'\|_\infty \leqslant \theta \tag{20}$$

state that two points $\mathbf{x}$ and $\mathbf{x}'$ must be close for the approximation to be valid. It turns out that Eq. (19) does not critically depend on θ, as we shall see later. Note that it is possible for $[\hat{g}]_i(\mathbf{x})$ to have multiple values, in which case the maximum absolute value is taken. It is also possible that such an approximation, $[\hat{g}]_i(\mathbf{x})$, may not exist. Now let

$$\{d(\mathbf{x}) = \hat{g}(\mathbf{x}) - \partial\hat{f}_R/\partial\mathbf{x} \mid \mathbf{x} \in R\} \tag{21}$$

be the differential residuals resulting from the local approximator $\hat{f}_R$ fit to the data in R. If $[\hat{g}]_i(\mathbf{x})$ does not exist, $d_i(\mathbf{x}) = 0$. Also let the derivative criterion

$$\begin{aligned} DeriCri(\alpha) = &\operatorname*{ave}_{\mathbf{x}\in R} \{|\alpha^t d(\mathbf{x})| \mid \mathbf{x} \leqslant s(\alpha)\} \\ &+ \operatorname*{ave}_{\mathbf{x}\in R} \{|\alpha^t d(\mathbf{x})| \mid \mathbf{x} > s(\alpha)\}, \end{aligned} \tag{22}$$

where $s(\alpha)$ is the split point (usually the median or mean value of $\{\alpha^t d(\mathbf{x}) \mid \mathbf{x} \in R\}$). Maximizing Eq. (22) produces the direction along which the first-order derivatives of the local approximation deviate the most on average from those of the target function in the region being split. From Eqs. (22) and (15), the differential splitting criterion (to be maximized) for selecting the split direction α can then be given by

$$SplitCri(\alpha) = \lambda ResidualCri(\alpha) + (1 - \lambda) DeriCri(\alpha), \qquad (23)$$

where $\lambda \in [0, 1]$.

If the input variables are equally relevant, then how well the local approximator fits to the data should determine the split direction. In this case, one prefers $\lambda = 1$. On the other hand, if the input variables are unequal in their differential relevance, then the direction selected for split should reflect the extent to which such relevance can be captured by the local approximation along it. This corresponds to $0 \leqslant \lambda < 1$, and is related to the notion of differential scaling of input variables [27]. At the extreme where $\lambda = 0$, partition is determined solely by Eq. (22). In theory, Eq. (22) by itself can produce the desired goal. In practice, however, insufficient data might impede the direct application of Eq. (22). Our experiments show that differential splitting (Eq. (23)) with $\lambda \in (0,1)$ achieves the best of both residual splitting (Eq. (15)) and derivative splitting (Eq. (22)) worlds.

In order to gain an intuitive perspective on the differential splitting criterion, we apply it to the example described in Section 3.2 with the following parameter settings: $\lambda = 0.9$, $\delta = 4$, $\theta = 0.1$. Once again the split points are taken to be the mean and a local constant approximator is used. From Eq. (21), we have

$$d(\mathbf{x}_1) = (4,0)^t,\ d(\mathbf{x}_2) = (4,0)^t,\ d(\mathbf{x}_3) = (4,0)^t,\ d(\mathbf{x}_4) = (4,0)^t,$$
$$d(\mathbf{x}_5) = (4,0)^t,\ d(\mathbf{x}_6) = (4,0)^t,\ d(\mathbf{x}_7) = (4,0)^t,\ d(\mathbf{x}_8) = (4,0)^t.$$

It follows that $DeriCri(\mathbf{e}_1) = 8$ and $DeriCri(\mathbf{e}_2) = 0$. We also have $ResidualCri(\mathbf{e}_1) = ResidualCri(\mathbf{e}_2) = 0$. Therefore, $SplitCri(\mathbf{e}_1) = 0.8$ and $SplitCri(\mathbf{e}_2) = 0$. Thus, X_1 is the direction selected for splitting according to the differential splitting criterion (Eq. (23)), which is indeed the optimal direction.

4. Empirical results on simulated data

In this section we present results of applying the two splitting criteria, Eqs. (15) and (23), to artificial data that simulate a variety of target functions (Eq. (5)). Two approximation schemes were employed in these experiments. One is local constant approximation where the target function in each local region, R, is approximated by

$$\hat{f}_R(\mathbf{x}) = c, \qquad (24)$$

where $\mathbf{x} \in R$ and $c = ave_{\mathbf{x} \in R}(\mathbf{x})$. Here ave is an averaging function such as the mean function. The other is local linear approximation where the target function is approximated by

$$\hat{f}_R(\mathbf{x}) = \mathbf{a}^t \mathbf{x}, \qquad (25)$$

where $\mathbf{x} = (1, x_1, \ldots, x_p)^t \in R$ and $\mathbf{a} = (a_0, a_1, \ldots, a_p)^t$ is determined from the training observations in region R.

In all the experiments reported in this section, λ (Eq. (23)) was set to 0.9. In addition, the values of δ (Eq. (20)) used to produce the results for each target were selected as the best from among several runs, while θ (Eq. (20)) was set to 0.5 throughout. No attempt was made to search for the best values exhaustively.

4.1. Performance measure

For each target function $f(\mathbf{x})$ and each method k, the mean absolute target error

$$e_k = mean|f(\mathbf{x}) - \hat{f}_k(\mathbf{x})| \qquad (26)$$

was computed over 5000 independently generated test observations. The performance measure used for comparison is

$$r_k = e_k / \min_t e_t. \qquad (27)$$

This quantity is exactly the same as the one used in Ref. [11]. That is, we normalize the target error for each method by that of the best method. Therefore, the best method receives a value of 1, while all others have larger values for each target function. The distribution of r_k values provides relative performance for each method. Thus, for a method that performs the best for all target functions this distribution should be a point mass at the value 1.

4.2. Quadratic function

The target function examined in this section is the same quadratic function (Eq. (16)) as in Section 3.2. The two input variables are randomly generated from a uniform distribution: $\mathbf{x} \sim U^2[-4,4]$, and the corresponding output y is computed according to Eq. (16). Fifty sets of training data having 500 samples each were independently generated. The training observations in each terminal region of the final partition were set to 5 (stopping criterion).

Fig. 3 shows the distributions of relative errors for the two methods on an independent test set of 5000 observations. Fig. 3(a) shows the results using a local constant approximator (Eq. (24)), while Fig. 3(b) the results using a local linear approximator (Eq. (25)). It can be seen that high differential relevance is exploited by the differential splitting criterion.

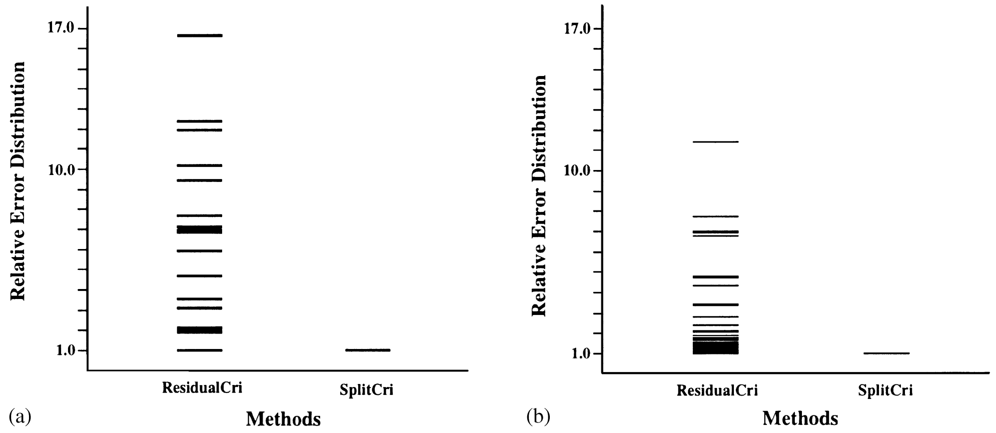

Fig. 3. Distribution of relative errors for the quadratic target function: (a) local constant approximator; (b) local linear approximator.

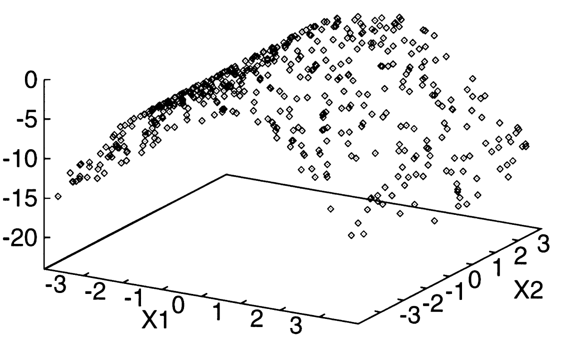

Fig. 4. A set of 500 randomly generated training data.

4.2.1. *Input feature space partition*

Fig. 4 shows one set of 500 randomly generated training data according to the quadratic function Eq. (16). Fig. 5(a) shows the partition of the input space resulting from applying Eq. (15) (residual splitting) as the split criterion, from which a mean absolute target error of 2.01 (Eq. (26)) was obtained. The total number of disjoint regions is 109. In contrast, Fig. 5(b) shows the partition as a result of applying Eq. (23) (differential splitting) as the split criterion. This partition gave rise to a mean absolute target error of 0.09 over 5000 independently randomly generated test data. The number of disjoint regions is 113. A local constant approximator (Eq. (24)) is used in this experiment. The results demonstrate clearly the advantage of the new splitting criterion in that it splits the input space along the direction where the target function changes most rapidly.

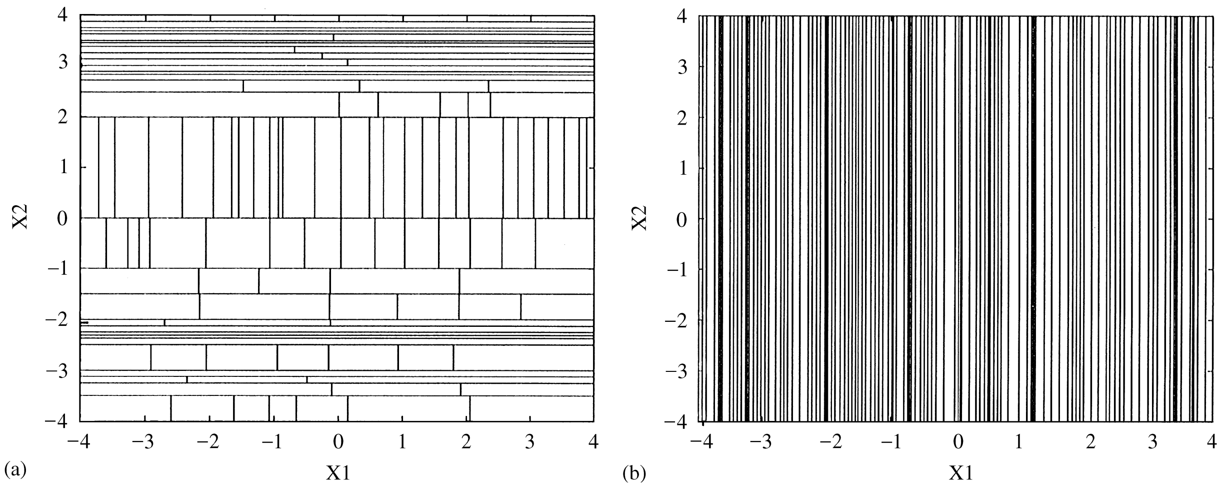

Fig. 5. Input feature space partition: (a) partitions produced by residual splitting (15); (b) partitions produced by new differential splitting (23).

4.3. Randomly generated functions

The target function (Eq. (16)) was purposefully chosen to show the ability of the new splitting criterion to differentiate input relevance in some cases. To further illustrate the relative merits of the new splitting criterion, it is applied to a variety of randomly generated target functions (taken from Ref. [11]). Each one has the form

$$f(\mathbf{x}) = \sum_{k=1}^{K} a_k h(\mathbf{x}, \mathbf{z}_k, \mathbf{V}_k), \tag{28}$$

where

$$h(\mathbf{x}, \mathbf{z}, \mathbf{V}) = \frac{1}{|\mathbf{V}|} \exp\left[-\frac{1}{2} (\mathbf{x} - \mathbf{z})^t \mathbf{V}^{-1} (\mathbf{x} - \mathbf{z}) \right]. \tag{29}$$

That is, $f(\mathbf{x})$ is a linear combination of Gaussian functions, where

$$\{a_k \sim U[-1, 1]\}_{k=1}^{K}$$

and

$$\{\mathbf{z}_k \sim U^n[0, 1]\}_{k=1}^{K}.$$

In addition, the eigenvectors of each $\mathbf{V}_k$ are generated uniformly randomly on the unit p-sphere subject to the orthogonality constraint, and the eigenvalues are generated uniformly from $U[0.525\sqrt{p}, 0.025\sqrt{p}]$. This allowed us to generate a wide variety of target functions in terms of the geometric shapes of their contours.

The experiments described here consist of 100 such randomly generated functions with $p = 10$ input variables. The study of the 100 target functions is divided into two groups of 50 targets each, the first group with $K = 5$, and the second with $K = 10$. The input points for each target were generated randomly from a uniform distribution

$$\mathbf{x} \sim U^{10}[0, 1].$$

The training observations in each terminal region of the final partition were set to 5 for local constant fitting, while the training observations were set to 20 for local linear fitting. Note that linear fitting requires sufficient training observations to constraint the unknown variables and 20 data points in each partition seem to be adequate for local fitting in these experiments. The quantity for comparison is provided by Eq. (27) over 5000 independently generated test observations.

Fig. 6 shows the distributions of relative performance of the two splitting criteria for the target functions. In both cases, each target was learned using a training set of $n = 1500$ observations. The results show convincingly that the new criterion (Eq. (23)) outperformed residual splitting (Eq. (15)) on all the target functions. It can also be seen that local constant fitting seems to give better performance than local linear fitting, which is consistent with the results reported in Ref. [11].

4.3.1. Effect of sample size

The new criterion (Eq. (23)) requires estimates of the derivatives of target functions. However, in a high-dimensional space there may not be sufficient training data to permit such (accurate) estimates. In order to gain a perspective on this issue, the criterion is applied to the same target functions, but at a reduced sample size. Fig. 7 shows the distributions of relative errors at a sample size 1000 and 500, using a local constant approximator (Eq. (24)). The results show that the improvement of differential splitting (Eq. (23)) over residual splitting (Eq. (15)) becomes less dramatic as the size of training sets decreases.

4.3.2. Effect of λ on performance

Finally, additional experiments were carried out to examine the effect of λ on the performance of differential splitting. Ten target functions were randomly generated from Eq. (28) with $K = 10$. Each target was learned using a training set of 1000 points. The results were averaged over 5000 independently generated test observations. A local constant approximator was used in these experiments. Fig. 8 shows the performance of Eq. (23) as a

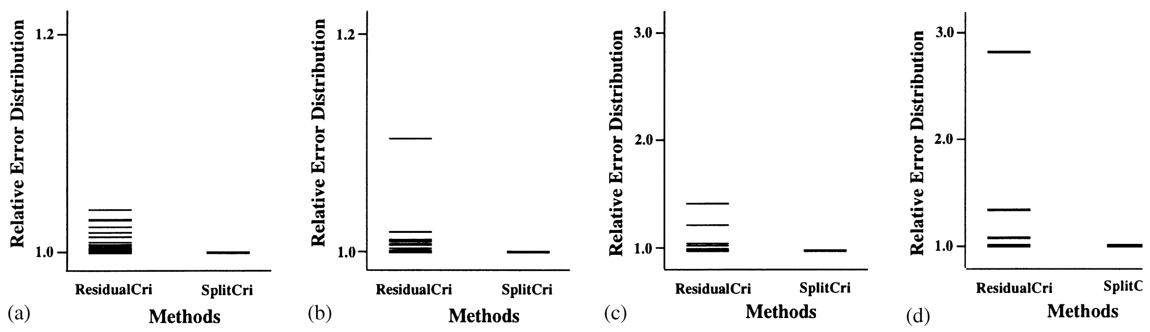

Fig. 6. Distributions of relative errors for the target functions: (a), (c) $K = 5$; (b), (d) $K = 10$; (a), (b) Local constant approximator; (c),(d) Local linear approximator.

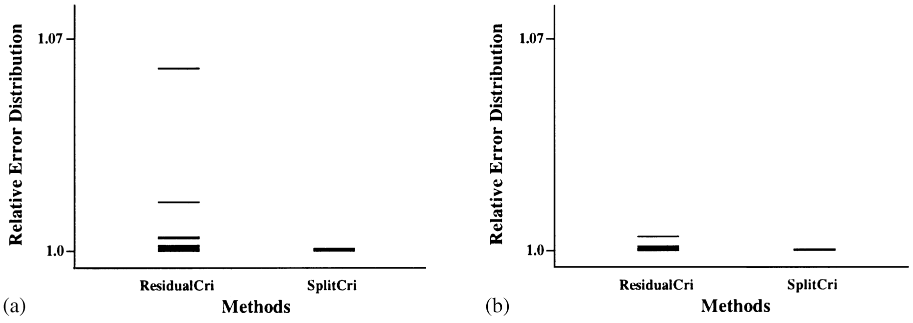

Fig. 7. Distribution of relative errors for the target functions ($K = 5$): (a) training sample size $n = 1000$; (b) training sample size $n = 500$.

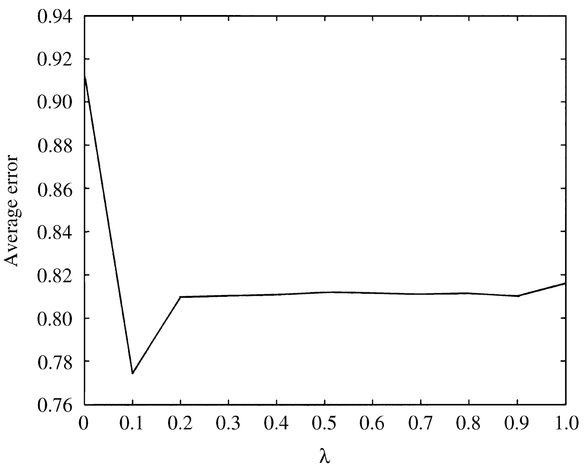

Fig. 8. Performance of *SplitCri* (Eq. (23)) as a function of λ.

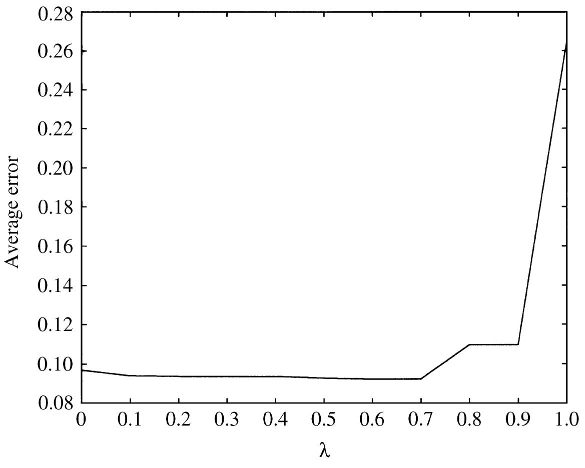

Fig. 9. Performance of *SplitCri* (Eq. (23)) as a function of λ.

function of λ. Again, the values of δ (Eq. (20)) that produced the results were chosen as the best from among several runs, while θ (Eq. (20)) was set to 0.5.

The overall performance of differential splitting as a function of λ shows that the new criterion produces relatively large approximation errors when λ takes on values 0 or 1, thereby indicating neither residual splitting (Eq. (15)) nor derivative splitting (Eq. (22)) alone achieved good performance over target functions under study. The best of both worlds is achieved at the values of λ between 0 and 1, as was expected. In addition, the results show that good performance can be obtained over a wide range of λ values. Similarly, Fig. 9 shows the performance of the differential splitting criterion on the quadratic function (Eq. (16)) as a function of λ. It can be seen that similar performance characteristics to that of (Eq. (23)) on the random target functions were obtained. Although these experiments of 10 random and one quadratic target functions are by no means extensive, we expect the result (best performance when $\lambda \in (0, 1)$) to hold in general. This is because the differential splitting criterion exploits both types of information, namely, residual (Eq. (15)) and derivative (Eq. (22)).

5. Empirical results on real data

In order to further evaluate our local learning method for pattern classification, we applied it to the letter image recognition data (LIRD) from the UCI repository of machine learning databases [28]. LIRD consists of a large number of black-and-white rectangular pixel arrays as one of the 26 upper-case letters in the English alphabet. The characters are based on 20 Roman alphabet fonts, such as HASTR, HCART, and HCITA, etc. They represent five different stroke styles, such as simplex and triplex, and six different letter styles, such as script and German. Each letter is randomly distorted through a quadratic transformation to produce a set of 20,000 unique letter images that are then converted into 16 primitive numerical features. Basically, these numerical features are statistical moments and edge counts. The last

Table 1
Feature information

Feature	Description	
1	x-bar	mean x of on pixels in box
2	y-bar	mean y of on pixels in box
3	x2bar	mean x variance
4	y2bar	mean y variance
5	xybar	mean xy correlation
6	x2$y$br	mean of $x*x*y$
7	xy2br	mean of $x*y*y$
8	x-ege	mean edge count left to right
9	$x$egvy	correlation of x-edge with y
10	y-ege	mean edge count bottom to top
11	$y$egvx	correlation of y-edge with x

11 of the 16 features[2] are used here and scaled to fit into a range of real values from 0 to 1. A brief description of these features [28] is listed in Table 1. Subsets of LIRD (sample character images are shown in Fig. 10) are used to produce the results reported here.

Instead of recursive partitioning, we used recursive covering [11] as the regression model in these experiments. Similar to recursive partitioning, recursive covering partitions the input space into a set of local regions within which a simple input-output relationship is modeled. Unlike recursive partitioning, however, the local regions produced by recursive covering are not disjoint. They overlap each other instead. Fig. 11 illustrates child regions produced by recursive partitioning vs. recursive covering. The split points, s_1 and s_2, in recursive covering are taken such that there are $(1-\gamma)\%$ of data in each child region, where $0 \leqslant \gamma \leqslant 1/2$ is a procedural "trimming factor". Because more data are passed on to child regions in recursive covering regression, these methods tend to produce better approximation than that produced by recursive partitioning methods, especially when γ is close to zero. However, increased accuracy is at the expense of increased computation. Note that when $\gamma = 1/2$, recursive covering reduces to recursive partitioning.

Similar to recursive partitioning, the recursive covering method employs local constant approximators in these experiments. Fig. 12 summarizes our local learning algorithm for approximating $f_i(\mathbf{x})$ that takes as input a set of training data and produces as output an approximation to $f_i(\mathbf{x})$: $\hat{f}_i(\mathbf{x})$, $i = 1, \ldots, J$. For a given input query $\mathbf{x}$, the decision is to assign $\mathbf{x}$ to class i such that

$$i = \arg\max_{1 \leqslant i \leqslant J} \{\hat{f}_i(\mathbf{x})\}.$$

[2] Our experiment shows that the first 5 features, such as box positions and the size of box, do not significantly contribute to overall recognition performance.

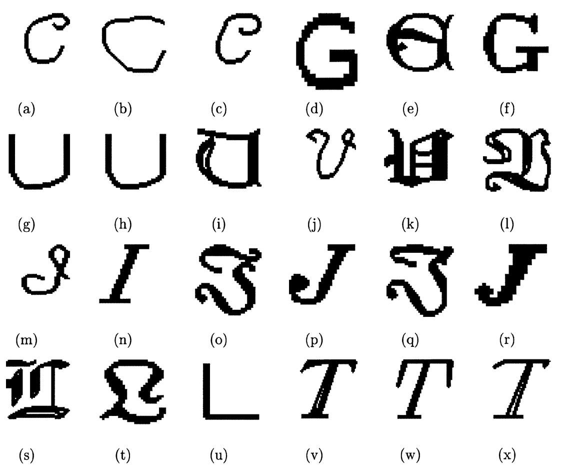

Fig. 10. Sample character images.

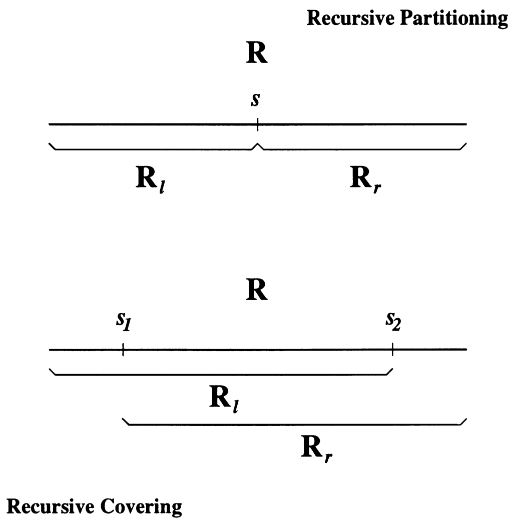

Fig. 11. Child regions generated by recursive partitioning vs. recursive covering.

```
• Let $T = \{\mathbf{x}_i, C_i\}_{i=1}^n$ be the input training data, where $C_i \in \{1, \cdots, J\}$.
• Convert $T$ into $J$ training sets: $T_j = \{\mathbf{x}_k, y_j\}_{k=1}^n, j = 1 \cdots J$ (Eq. (6)) via Eq. (4).
• For $j = 1$ to $J$ do
    – Compute $\hat{f}_j$ from $T_j$ based on recursive covering regression using constant approximators in each local regions.
• Return $\{\hat{f}_j\}_{j=1}^J$.
```

Fig. 12. A local learing algorithm for approximating the class probability function.

Note that the training observations in each terminal region of the final partition are set to 8. For all the experiments reported here, the trimming factor γ was set to 0.35, λ (Eq. (23)) was set to 0.9 and θ (Eq. (20)) to 0.3, while δ (Eq. (20)) was determined experimentally during several trial runs. For comparison, C4.5 [13] was also implemented as a competing method.

5.1. The problems

Problem 1. This is a two class classification problem involving letters C and G. There are total 1509 instances, 736 for C and 773 for G. 1000 instances from among the 1509 instances are uniformly randomly selected (without replacement) as training data, and remaining instances are used as testing data upon which classification performance is evaluated. We denote this problem as **CG1**.

Problem 2. This problem is the same as the first one. However, only 500 instances are uniformly randomly selected as training data, and remaining 1009 instances as testing data. We call this experiment **CG2**.

Problem 3. This problem involves letters U and V. There are total 1577 instances, 813 for U and 764 for V. Similar to the first problem, 1000 instances are uniformly randomly chosen as training data, and remaining instances are used as testing data. We call this experiment **UV**.

Problem 4. The final problem involves letters I, J, L, and T. There are 3059 instances, 1500 of which are uniformly randomly selected as training data, and the rest as testing data. This problem is denoted by **IJLT**.

5.2. Results

For each problem, 10 training sets (and corresponding testing sets), each with a different random seed, were independently generated. The average classification performance of each method on the 10 independent experiments is then reported in Table 2, where μ represents the mean and σ the standard deviation. Note that in Table 2 RSRC denotes the recursive covering method coupled with the residual splitting criterion (Eq. (15)), whereas DSRC represents the recursive covering method employing the differential splitting criterion (Eq. (23)).

The results show that DSRC registered performance that is consistently better than both RSRC and C4.5 across the tasks. While the margins of improvement are not as dramatic as in the simulated data experiments, they are significant (e.g., the mean differences are at least more than two standard deviations apart, except that between RSRC and DSRC on the IJLT problem). One possible explanation is that the letter image recognition problems (with the features that are used here) do not exhibit significant difference in differential relevance along directions under consideration.

Table 2
Average classification accuracy for LIRD problems

Problem\method	RSRC		DSRC		C4.5	
	μ	σ	μ	σ	μ	σ
CG1	95.1	0.31	96.4	0.24	95.2	0.18
CG2	93.4	0.23	94.2	0.18	93.2	0.32
UV	98.3	0.24	99.1	0.14	98.3	0.14
IJLT	92.2	1.50	94.6	0.17	93.8	0.17

6. Conclusions

The local discriminative learning method developed in this paper for pattern classification establishes its validity by producing the desired goal in situations where high differential relevance exists in input variables. It achieves superior performance in a wide variety of target functions and classification tasks by capturing such differential relevance. A potential limitation of the method presented here is that it requires sufficient data in order to reliably estimate differential relevance of features input to the local learning procedure. This is particularly true when differential relevance of input features to the underlying task is fairly equal. However, when such relevance is unequal, the method works very effectively, as evidenced by the quadratic target function task (Eq. (16)).

It is important to note that, while directions under consideration for splitting in the experiments reported here are those along coordinate axes, other directions, as a function of observations, can be considered as well using the same technique. This will no doubt enhance the discriminating power of resulting classifiers, albeit it is at the expense of increased computation. We believe that the region selection technique presented in this paper lays a solid foundation upon which to construct efficient local discriminative learning methods for pattern classification in high dimensional settings.

Acknowledgements

We wish to thank Dave Slate for providing us the letter images. This work was supported by DARPA/AFOSR grant F49620-97-1-0184. The contents of the information do not necessarily reflect the position or the policy of the U.S. Government.

References

[1] Y.D. Rubinstein, T. Hastie, Discriminative vs informative learning, Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, 1997, pp. 49–53.

[2] T. Kohonen, The self-organizing map, Proc. IEEE 78 (1990) 1464–1480.

[3] R. Lippmann, Pattern classification using neural networks, IEEE Commun. Mag. 11 (1989) 47–64.

[4] C. Atkeson, A. Moore, S. Schaal, Locally weighted learning, Artif. Intell. Rev. 11 (1997) 11–73.

[5] W.S. Cleveland, S.J. Devlin, Locally weighted regression: an approach to regression analysis by local fitting, J. Amer. Statist. Assoc. 83 (1988) 596–610.

[6] T.K. Ho, Nearest neighbors in random subspaces, Lecture Notes in Computer Science: Advances in Pattern Recognition, 1998, pp. 640–648.

[7] D.G. Lowe, Similarity metric learning for a variable-Kernel Classifier, Neural Comput. 7 (1) (1995) 72–85.

[8] S. Baker, S.K. Nayar, Pattern rejection, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 1996, pp. 544–549.

[9] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, 1984.

[10] C.E. Brodley, P.E. Utgoff, Multivariate decision trees, Mach. Learning 19 (1995) 45–77.

[11] J.H. Friedman, Local learning based on recursive covering, Technical Report, Dept. of Statistics, Stanford University, Stanford, CA 94305, 1996.

[12] S.K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, J. Artif. Intell. Res. 2 (1994) 1–32.

[13] J.R. Quinlin, C4.5: Programs for Machine Learning, Morgan-Kaufmann Publishers, Inc., Los Altimos, CA, 1993.

[14] D. Swets, J. Weng, Discriminant analysis and eigenspace parition tree for face and object recognition from views, Proceedings of International Conference on Automatic Face- and Gesture-Recognition, Killington, Vermont, October 1996, pp. 192–197.

[15] J. Weng, Cresceptron and SHOSLIF: toward comprehensive visual learning, in: S.K. Nayar, T. Poggio (Eds.), Early Visual Learning, Oxford University Press, New York, 1996, pp. 183–214.

[16] H. Murase, S.K. Nayar, Visual learning and recognition of 3-D objects from appearance, Int. J. Comput. Vision 14 (1) (1995) 5–24.

[17] M. Turk, A. Pentland, Eigenfaces for recognition, J. Cognitive Neurosci. 3 (1) (1991) 71–86.

[18] R.E. Bellman, Adaptive Control Processes, Princeton Univ. Press, Princeton, NJ, 1961.

[19] L. Bottou, V. Vapnik, Local learning algorithms, Neural Comput. 4 (6) (1992) 888–900.

[20] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[21] J. Peng, Efficient memory-based dynamic programming, Proceedings of the 12th International Conference on Machine Learning, 1995, pp. 438–446.

[22] S. Salzberg, A nearest hyperrectangle learning method, Mach. Learning 6 (1991) 251–276.

[23] C.J. Stone, Nonparametric regression and its applications (with discussion), Ann. Statist. 5 (1977) 595.

[24] J.H. Friedman, Multivariate adaptive regression splines, Ann. Statist. 15 (1) (1991) 1–141.

[25] M.I. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Comput. 6 (1994) 181.

[26] L.E. Scales, Introduction to Non-Linear Optimization, Springer, New York, 1985.

[27] T.J. Hastie, R.J. Tibshirani, Generalized Additive Models, Chapman & Hall, London, 1990.

[28] P.M. Murphy, D.W. Aha, UCI repository of machine learning databases. http://www.cs.uci.edu/~mlearn/MLRepository.html, 1995.

About the Author—DR. PENG received the B.S. degree in Computer Science from Beijing University of Aeronautics and Astronautics, China, the M.A. degree in Computer Science from Brandeis University and the Ph.D. degree in Computer Science from Northeastern University, Boston, Massachusetts. From 1994 to 1995 he was a Post Doctoral Fellow in the Visualization and Intelligent Systems Laboratory at UC Riverside. During 1996 he worked as Senior Research Scientist in the Machine Vision Department at the Amherst Systems in Buffalo, NY. From January 1997 to July 1999 he was Research Scientist in the Center for Research in Intelligent Systems at UC Riverside. Sicne August 1999 he has been a Assistant Professor in the Computer Science Department at the Oklahoma State University. Dr. Peng's interests are in the areas of machine learning, content-based image retrieval, pattern classification, and learning and computer vision.

About the Author—BIR BHANU received the S.M. and E.E. degrees in electrical engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge and the Ph.D. degree in electrical engineering from the Image Processing Institute, University of Southern California, Los Angeles. Currently he is a Professor of electrical engineering and computer science and Director of Center for Research in Intelligent Systems at the University of California, Riverside. Previously, he was a Senior Honeywell Fellow at Honeywell Systems and Research Center, Minneapolis, MN. He has been the principal investigator of various programs for DARPA, NASA, NSF, AFOSR, ARO and other agencies and industries in the areas of learning and vision, image understanding, pattern recognition, target recognition, navigation, image databases, and machine vision applications. He is the co-author of books on "Computational Learning for Adaptive Computer Vision," (Plenum, Forthcoming), "Genetic Learning for Adaptive Image Segmentation" (Kluwer 1994), and "Qualitative Motion Understanding" (Kluwer 1992). He holds 10 U.S. and international patents and over 200 reviewed technical publications in the areas of his interest. Dr. Bhanu is a Fellow of the IEEE and the AAAS. He is a member of ACM, AAAI, Sigma Xi, Pattern Recognition Society, and SPIE.