Computer Vision and Image Understanding
Vol. 75, Nos. 1/2, July/August, pp. 150–164, 1999
Article ID cviu.1999.0770, available online at http://www.idealibrary.com on IDEAL®

Probabilistic Feature Relevance Learning for Content-Based Image Retrieval

Jing Peng,[1] Bir Bhanu, and Shan Qing

Center for Research in Intelligent Systems, University of California, Riverside, California 92521
E-mail: {jp,bhanu,shan}@vislab.ucr.edu

Most of the current image retrieval systems use "one-shot" queries to a database to retrieve similar images. Typically a K-nearest neighbor kind of algorithm is used, where weights measuring feature importance along each input dimension remain fixed (or manually tweaked by the user), in the computation of a given similarity metric. However, the similarity does not vary with equal strength or in the same proportion in all directions in the feature space emanating from the query image. The manual adjustment of these weights is time consuming and exhausting. Moreover, it requires a very sophisticated user. In this paper, we present a novel probabilistic method that enables image retrieval procedures to automatically capture feature relevance based on user's feedback and that is highly adaptive to query locations. Experimental results are presented that demonstrate the efficacy of our technique using both simulated and real-world data. © 1999 Academic Press

1. INTRODUCTION

The rapid advance in digital imaging technology makes possible the wide spread use of image libraries and databases. This in turn demands effective means for access to such databases. It is well known that simple textual annotations for images are often ambiguous and inadequate for image database search. Thus, retrieval based on image "content" becomes very attractive [5, 8, 10]. Generally, a set of features (color, shape, texture, etc.) are extracted from an image to represent its content. Then the image database retrieval procedure becomes a K-nearest neighbor (K-NN) search in a multidimensional space defined by the set of features under a given similarity metric, such as the Euclidean distance.

There are several fundamental problems associated with this simple content-based image retrieval scheme:

First. Features are unequal in their differential relevance for computing the similarity between images. Feature relevance may change from image to image and from location to location [1]. When a user says that two images are similar, the user really means that the images are similar in an individual feature, some combination of the features, or some features still unknown to the user. This implies that the similarity does not vary with equal strength or in the same proportion in all directions in the feature space emanating from the query image. Figure 1 illustrates a case in point, where class boundaries are parallel to the coordinate axes. For query a, dimension X is more relevant, because a slight move along the X axis may change the class label, while for query b, dimension Y is more relevant. For query c, however, both dimensions are equally relevant.

Second. The user understands more about the query, whereas the database system can only "guess" what the user is looking for during the retrieval process. As the query examples in Fig. 1 show, any fixed feature relevance scheme is incapable of being customized to each individual query so as to achieve overall optimal performance. As such, the system must interact with the user to learn feature differential relevance for each individual query to guide its search and to iteratively refine its retrieval at run-time.

Finally. Different similarity (closeness) measures capture different aspects of perceptual similarity between images [8]. Humans do seem to selectively attend to features to optimize their visual behaviors [13]. In general, what similarity metric to use is image dependent and plays an important role in the outcome of the retrieval process. For example, in the context of "eigenfaces" for recognition of human faces, the quadratic or *Mahalanobis* distance is more appropriate on the assumption of a Gaussian distribution. While determining similarity metrics is an important research issue, here we are mainly concerned with learning feature relevance. It turns out that there is a unique correspondence between our relative feature relevance measure and a weighted distance metric.

In this paper, we propose a novel method that provides a solution to the problems discussed above. With this method an image retrieval system is able to learn differential feature relevance in an efficient manner by estimating the strength of each feature dimension in predicting the class (1) of a given query. In addition, since the estimation process is carried out locally in the vicinity of the input query, the method is highly adaptive to query locations.

[1] Corresponding author.

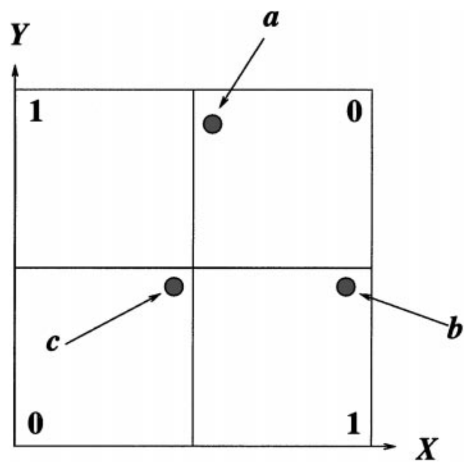

FIG. 1. Feature relevance varies with query locations.

We describe next the functional architecture of our retrieval system. Section 2 describes related work addressing issues of feature relevance computation. Section 3 discusses a simple K-NN search technique and its limitations as a procedure for image retrieval. Section 4 presents our approach to content-based image retrieval that overcomes some of the limitations associated with the simple K-NN search by capturing the notion of local feature relevance. Section 5 describes an efficient procedure for estimating local feature relevance. After that, we present in Section 6 experimental results demonstrating the efficacy of our technique using both simulated and real-world data. Finally, Section 7 concludes this paper by pointing out possible extensions to the current work and future research directions. Appendix provides details on the Gabor features that are used in this research.

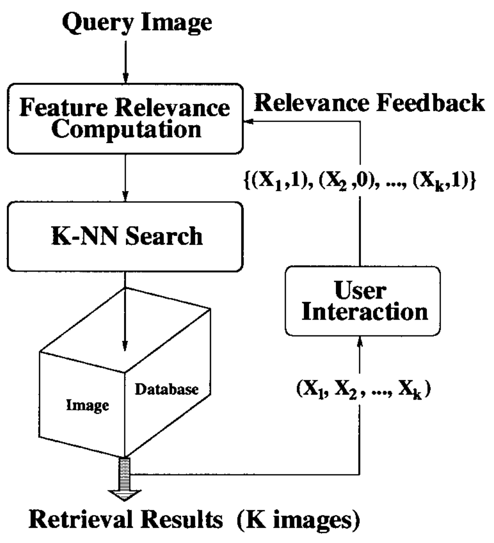
Query Image
Feature Relevance Computation
Relevance Feedback
{(X1,1), (X2,0), ..., (Xk,1)}
K-NN Search
User Interaction
Image
Database
(X1, X2, ..., Xk)
Retrieval Results (K images)

FIG. 2. System for learning feature relevance.

System overview. Figure 2 shows the functional architecture of our system. Images in the database are represented by feature vectors, such as normalized mean and standard deviations of responses from Gabor filters. The user presents a query image to the system. At this time feature relevance along each dimension is assumed to be equal, and its associated weighting is initialized to $1/q$, where q is the dimension of the feature space. The system carries out image retrieval using a K-NN search, based on current weightings to compute the similarity between the query and all images in the database, and returns the top K nearest images. The user then marks the retrieved images as positive (class 1) (e.g., click on an image by using the left mouse button) or negative (class 0) (e.g., click on an image by using the right button). The user "thinks" that the positive images look similar to the query image but the negative ones do not. Note that in practice, only images that are dissimilar to the query (negative images) need to be marked. These marked images constitute training data. From the query and the training data, the system computes local feature relevance in terms of weights, from which a new round of retrieval begins. The above process repeats until the user is satisfied with the results or the system cannot improve the results from one iteration to the next.

2. RELATED WORK

Friedman [6] describes an approach for learning local feature relevance that combines some of the best features of K-NN learning and recursive partitioning. This approach recursively homes in on a query along the most (locally) relevant dimension, where local relevance is computed from a reduction in prediction error given that query's value along that dimension. This method performs well on a number of classification tasks. In contrast, our method, inspired by [6], computes local feature relevance directly from the conditional probabilities, since in a retrieval problem the "label" of the query is known.

A recent work [15] describes an image retrieval system that makes use of retrieval techniques developed in the field of *information retrieval* (IR) for text-based information. In this system, images are represented by weight vectors in the term space, where weights capture the importance of components within a vector as well as importance across different vectors over the entire data set. The system then uses relevance feedback to update queries so as to place more weights on relevant terms and less weights on irrelevant ones. This query updating mechanism amounts to rotating the query vector toward relevant retrievals and, at the same time, away from irrelevant ones. One limitation of this system is that it is variant to translation and general linear transformation because of its use of the nonmetric similarity function. Another limitation with the technique is that in many situations the mere rotation of the query vector is insufficient to achieve desired goals (see problems shown in Section 6.1.1). We will further describe the technique in Section 6, where it is compared to the method introduced in this paper in a variety of tasks.

PCF, *per category feature importance* [3], technique computes feature relevance based on conditional probabilities. This method estimates the conditional probability $p(c \mid f)$ for feature f in every category and uses it as a weight for f in category c. PCF assigns large weights to features having high correlation with the class. Clearly, feature importance as such is nonlocal, and therefore, insensitive to query locations. In addition, these global averaging correlation techniques do not work well on tasks such as the XOR problem shown in Fig. 1.

Trott and Leng [16] introduce an *integer programming* (IP) method for computing feature weights for an interactive retrieval task. In this task query's feature values are interactively supplied by the user. The method imposes a fixed ratio between the weight of the last feature and the sum of the weights of all other features for training samples. This ratio is then used to generate a set of linear equations that can be solved using IP to produce feature weights. While this method could be used for interactive retrieval to emancipate the user from manually adjusting feature weights for weighted similarity computation, its performance is not yet known.

3. SIMPLE *K*-NEAREST NEIGHBOR SEARCH

Simple K-nearest neighbor search, as an image retrieval procedure, returns the K images closest to the query. Obviously this involves the issue of measuring the closeness or similarity between two images. The most common measure of similarity between the two images is the distance between them. If the Euclidean distance

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{q} (x_i - y_i)^2} \tag{1}$$

is used as the measure of similarity, then the K closest images to the query $\mathbf{x}_Q$ are computed according to

$$\{\mathbf{x} \mid D(\mathbf{x}, \mathbf{x}_Q) \le d_K\},$$

where d_K is the Kth order statistic of $\{D(\mathbf{x}_i, \mathbf{x}_Q)\}_1^N$ and N is the number of images in the database. The major appeal for simple K-NN search methods resides in their ability to produce continuous and overlapping, rather than fixed, neighborhoods and to use a different neighborhood for each individual query so that all points in the neighborhood are close to the query.

One problem with Eq. (1) is that it does not take into account the influence of the scale of each feature variable in the distance computation. Changing the scale of a feature dimension in different amounts alters the overall contribution of that dimension in the distance computation, hence its influence in the nearest neighbors retrieved. This is usually considered undesirable. One way to overcome this is to rescale each feature dimension so that the scales of all feature dimensions are the same after rescaling, thereby causing them to contribute equally to the distance calculation in Eq. (1).

An additional limitation of Eq. (1) is that the use of the Euclidean distance, while simple computationally, implies that the input space is isotropic or homogeneous. However, the assumption for isotropy is often invalid and generally undesirable in many practical applications. Figure 1 illustrates a case in point. For query a, the preferred direction along which to search for similar samples is the Y axis. That is, the search space in the vicinity of the query should be elongated along the Y direction and constricted along the the X direction. Capturing such information, therefore, is of great importance to any retrieval procedure in large database systems. The following sections describe a novel technique to achieve just that goal.

4. WEIGHTED *K*-NEAREST NEIGHBOR SEARCH

Simple K-NN search clearly has its limitations as a procedure for content-based image retrieval. The objective of our approach is to develop a retrieval method that inherits the appealing properties of K-NN search, while at the same time overcomes its limitations by capturing the notion of local feature relevance.

4.1. Local Feature Relevance

The retrieval performance for image databases can be characterized by two key factors. *First*, for a given query image, the relevance of all the features input to the database system may not be equal for retrieving similar images. Irrelevant features often hurt retrieval performance. *Second*, feature relevance depends on the location at which the query is made in the feature space. Capturing such relevance information is a prerequisite for constructing successful retrieval procedures in image databases.

We note at the outset that this problem is opposite to typical classification problems based on lazy learning techniques [11], such as nearest-neighbor kernel methods. While the goal in classification is to predict the class label of an input query from nearby samples, the goal in retrieval is to find samples having the same "class label" as that of the query. Moreover, many lazy learning techniques for classification lend themselves to the kind of problems retrieval tasks may face. It is important to realize that, unlike classification problems, the notion of classes in image databases is a user-centered concept that is dependent on the query image. There is no labelling of images in the database. Nonetheless, the "class label" of an image is simply used here as a vehicle to facilitate the theoretical derivation of our feature relevance measure for content-based image retrieval. As we shall see later, the resulting relevance measure and its associated weightings are independent of image labels in the database and, thus, fit our goals nicely here.

4.2. Local Relevance Measure

We begin this section by introducing some classification concepts essential to our probabilistic derivation.

In a two class (1/0) classification problem, the class label $y \in \{0, 1\}$ for query $\mathbf{x}$ is treated as a random variable from a distribution with the probabilities $\{\Pr(1 \mid \mathbf{x}), \Pr(0 \mid \mathbf{x})\}$ [6]. We then have

$$f(\mathbf{x}) \doteq \Pr(1 \mid \mathbf{x}) = \Pr(y = 1 \mid \mathbf{x}) = E(y \mid \mathbf{x}). \tag{2}$$

To predict y at $\mathbf{x}$, $f(\mathbf{x})$ is first estimated from a set of training data using techniques based on regression, such as the least-squares estimate.[2] Decision tree methods, neural networks, and nearest-neighbor kernel methods are examples of using this regression paradigm to the classification problem. The Bayes classifier can then be applied to achieve optimal classification performance. In image retrieval, however, the "class label" of $\mathbf{x}$ is known, which is 1 in terms of the notation given above. All that is required is to exploit the differential relevance of input features for image retrieval. Consider the least-squares estimate for $f(\mathbf{x})$. In the absence of values for any variable assignments, it is simply

$$E[f] = \int f(\mathbf{x})p(\mathbf{x})\,d\mathbf{x}. \tag{3}$$

That is, the optimal prediction (in the least-squares sense) for f is its average value. Now given only that $\mathbf{x}$ is known at dimension $x_i = z$. The least-squares estimate becomes

$$E[f \mid x_i = z] = \int f(\mathbf{x})p(\mathbf{x} \mid x_i = z)\,d\mathbf{x}. \tag{4}$$

Here $p(\mathbf{x} \mid x_i = z)$ is the conditional density of the other input variables defined as

$$p(\mathbf{x} \mid x_i = z) = p(\mathbf{x})\delta(x_i - z) \Big/ \int p(\mathbf{x})\delta(x_i - z)\,d\mathbf{x}, \tag{5}$$

where $\delta(x - z)$ is the Dirac delta function having the properties

$$\delta(x - z) = 0 \quad \text{if } x \neq z$$

and

$$\int_{-\infty}^{\infty} \delta(x - z)\,dx = 1.$$

It is evident that

$$0 \leq E[f \mid x_i = z] \leq 1.$$

Furthermore, Eq. (4) shows the predictive strength (probability) once the value of just one of the input features x_i is known.

Let $\mathbf{z}$ be the query. Since $f(\mathbf{z}) = 1$ (recall that query $\mathbf{z}$ has the same class label (class one) as the positive images), it follows that

$$f(\mathbf{z}) - 0$$

[2] Note that one can also use techniques based on density estimation to compute $f(\mathbf{x})$.

is the largest error one makes in predicting f at $\mathbf{z}$. That is, $f(\mathbf{z}) - 0$ is the error incurred when one predicts $f(\mathbf{z})$ to be zero ($\mathbf{z}$ is not in class one), but in fact $f(\mathbf{z})$ is in class one with probability 1. On the other hand,

$$f(\mathbf{z}) - E[f \mid x_i = z]$$

is the error one makes by predicting the probability of $\mathbf{z}$ being in class one as $E[f \mid x_i = z]$, conditioned on feature x_i taking the value z. Then

$$[(f(\mathbf{z}) - 0) - (f(\mathbf{z}) - E[f \mid x_i = z])] = E[f \mid x_i = z] \tag{6}$$

represents a reduction in error between the two predictions. We can now define a measure of feature relevance for query $\mathbf{z}$ as

$$r_i(\mathbf{z}) = E[f \mid x_i = z]. \tag{7}$$

That is, feature x_i is more relevant for query $\mathbf{z}$ if it contributes more to the reduction in prediction error (6).

The relative relevance, as a weighting scheme, can then be given by

$$w_i(\mathbf{z}) = (r_i(\mathbf{z}))^t \Big/ \sum_{l=1}^{q} (r_l(\mathbf{z}))^t, \tag{8}$$

where $t = 1, 2$, giving rise to linear and quadratic weightings, respectively. In this paper we propose the exponential weighting scheme

$$w_i(\mathbf{z}) = \exp(Tr_i(\mathbf{z})) \Big/ \sum_{l=1}^{q} \exp(Tr_l(\mathbf{z})), \tag{9}$$

where T is a parameter that can be chosen to maximize (minimize) the influence of r_i on w_i. When $T = 0$ we have $w_i = 1/q$, thereby ignoring any difference between the r_i's. On the other hand, when T is large a change in r_i will be exponentially reflected in w_i. In this case, w_i is said to follow the Boltzmann distribution. The exponential weighting is more sensitive to changes in local feature relevance (7) and gives rise to better performance improvement, as we shall see later.

It is clear from (8) and (9) that

$$0 \leq w_i(\mathbf{z}) \leq 1,$$

where $w_i(\mathbf{z}) = 0$ indicates that knowing x_i at $\mathbf{z}$ does not to help predict the query. On the other hand, $w_i(\mathbf{z}) = 1$ states that having the knowledge of x_i at $\mathbf{z}$ is sufficient to predict the query. Values in between show the degrees of relevance that x_i exerts at $\mathbf{z}$. It reflects the influence of feature x_i on the variation of $f(\mathbf{x})$ at query location $\mathbf{z}$. Thus, (8) and (9) can be used as weights

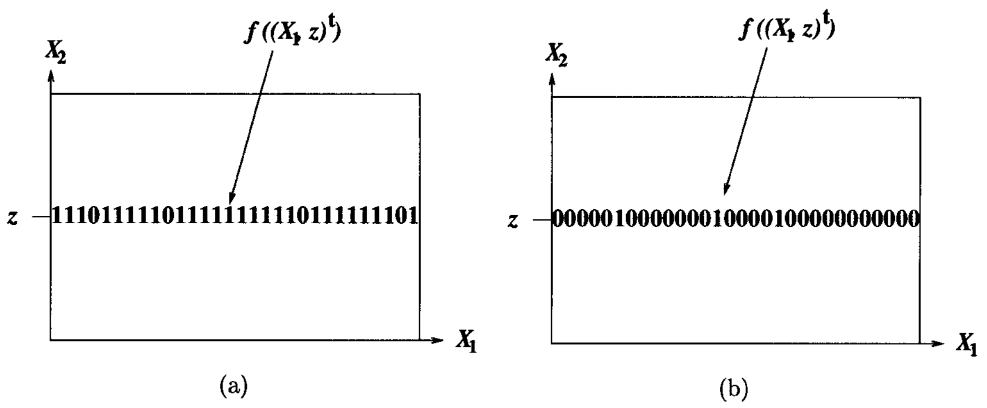

FIG. 3. (a) Large $E[f \mid x_2 = z]$ implies that the subspace spanned by X_1 at z is likely to contain samples having the same class label as the query. (b) Small $E[f \mid x_2 = z]$ indicates that the subspace is unlikely to have samples similar to the query.

associated with features for weighted similarity computation

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{q} w_i (x_i - y_i)^2}. \tag{10}$$

It can be shown that (10) is indeed a metric. These weights enable the similarity computation to elongate less important feature dimensions and, at the same time, to constrict the most influential ones. Note that the technique is *query-based* because weightings depend on the query [2].

A justification for (7) and, hence, (8) and (9), goes like this. Suppose that the value of $E[f \mid x_i = z]$ (4) is large, which implies a large weight along dimension x_i. This, in turn, penalizes points along x_i that are moving away from z. Now $E[f \mid x_i = z]$ can be large only if the subspace spanned by the other input dimensions at $x_i = z$ likely contains samples coming from class 1, assuming a uniform distribution. This scenario is illustrated in Fig. 3a. Then a large weight assigned to x_i based on (8) says that moving away from the subspace, hence from the data in class 1, is a bad thing to do. Similarly, a small value of $E[f \mid x_i = z]$, hence a small weight, indicates that in the vicinity of x_i at z one is unlikely to find samples similar to the query, as shown in Fig. 3b. Therefore, in this situation in order to find samples resembling the query, one must look farther away from z.

Another way to look at (4) is that the conditional expectation can be considered as a "measure" indicating the denseness (sparseness) of data points having the same label as the query in a small neighborhood of feature x_i at z. When data are dense (large conditional expectation) along feature x_i at z, it is given a large weight. Likewise, sparseness (small conditional expectation) results in a smaller weight. This is in direct analogy to the use of the quadratic distance

$$\|\mathbf{y} - \mathbf{x}\|^2 = (\mathbf{y} - \mathbf{x})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \mathbf{x}), \tag{11}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix, assuming a Gaussian distribution. If $\boldsymbol{\Sigma}$ is diagonal, (11) gives rise to a (squared) weighted Euclidean distance (10), where weights along feature dimensions are inversely proportional to the variance (denseness) along that dimension. That is, Eqs. (4) and (8) capture the notion of variable scaling in that they rescale each input feature so that the contribution of all the features to the distance computation is "equal."

We have so far only considered estimating feature relevance along each individual dimension one at a time. However, there are situations where feature relevance can only be captured by examining several feature variables simultaneously. That is, feature variables are not independent, and there is a degree of correlation among them. There is no doubt that such a capability is highly desirable in many database applications. However, it is clear that, in the absence of any other information, determining which feature(s) should be examined to estimate local relevance adds additional complexities to feature relevance computation. Furthermore, it is unclear as to how much gain one can expect when there are insufficient data[3] for conducting such joint estimation. One way to decorrelate association among the features is to rotate the feature dimensions so that they coincide with the eigenvectors of a sample covariance matrix. Note that, even without such a transformation to the eigen space, the technique described here can be readily extended to estimating local relevance, conditioned on multiple feature variables simultaneously. We do not address this issue further in the rest of this paper.

5. ESTIMATION OF RELEVANCE

In order to estimate (8) and (9), one must first compute (7). The retrieved images with relevance feedback from the user can be used as training data to obtain estimates for (7), hence, (8) and (9). Let

$$\{\mathbf{x}_j, y_j\}_1^K$$

be the training data. Here $\mathbf{x}_j$ denotes the feature vector

[3] Recall that there is a very limited number of returns at each retrievel.

representing jth retrieved image, and y_j is either 1 (relevant) or 0 (irrelevant) marked by the user as the class label associated with $\mathbf{x}_j$. To compute $E[f \mid x_i = z]$, recall that $f(\mathbf{x}) = E[y \mid \mathbf{x}]$. Thus, it follows that

$$E[f \mid x_i = z] = E[y \mid x_i = z].$$

However, since there may not be any data at $x_i = z$, the data from the vicinity of x_i at z are used to estimate $E[y \mid x_i = z]$, a strategy suggested in [6]. Therefore, (7) can be estimated according to

$$\hat{E}[y \mid x_i = z] = \sum_{j=1}^{K} y_j 1(|x_{ji} - z| \leq \Omega) \Big/ \sum_{j=1}^{K} 1(|x_{ji} - z| \leq \Omega), \tag{12}$$

where $1(\cdot)$ is an indicator function. That is, $1(\cdot)$ returns 1 if its argument is true, and 0 otherwise. Ω can be chosen so that there are sufficient data for the estimation of (4). In this paper, Ω is chosen such that

$$\sum_{j=1}^{K} 1(|x_{ji} - z| \leq \Omega) = C, \tag{13}$$

where $C \leq K$ is a constant. It represents a trade-off between bias and variance. In addition, (12) can be computed within a subregion, thus making the relevance measure more local. Note that it is possible to extend this technique to multiple class situations where the user can grade the retrieved images. Our retrieval and feature relevance computation algorithm is summarized in Fig. 4, where *precision* denotes the average retrieval precision (18).

The bulk of the computational expense involved in the feature relevance estimation algorithm shown in Fig. 4 is consumed by the K-nearest neighbor search, while the relevance estimation is quite efficient. This is particularly pronounced when the image database is large. In practice, however, the amount of computation associated with the nearest neighbor search can be significantly reduced by partitioning or indexing the image database in such a way that the nearest neighbor search can be localized within a given partition. This issue, however, is beyond the scope of this paper.

We use a simple two-class problem, shown in Fig. 5, to illustrate the feature relevance computation process. In this problem, the data for both classes are generated from a uniform distribution. The number of data points for both classes is roughly the same. The red square, located at (1, 0.1), represents the query. Figure 6a shows the 200 nearest neighbors (red squares) of the query found by the unweighted K-NN method (1). The resulting shape of the neighborhood is circular, as expected. In contrast, Fig. 6b shows the 200 nearest neighbors of the query, computed by the technique described above. That is, the retrievals (with relevance feedback) shown in Fig. 6a are used to compute (7) and, hence, (9) with estimated new weights: $w_1 = 0.134$ and $w_2 = 0.866$. As a result, the new (elliptical) neighborhood is elongated along the horizontal axis (the less important one) and constricted along the vertical axis (the more important one). The effect is that there is a dramatic increase in the retrieved nearest neighbors that are similar to the query.

```
1. Let $i$ be current query; initialize weight vector $\mathbf{w}$ to $\{1/q\}_1^q$.
2. Compute $K$ nearest images using $\mathbf{w}$.
3. User marks the $K$ images as positive or negative.
4. While $precision < \theta$ Do
   (a) $Tset \leftarrow \{marked\ K\ images\}$.
   (b) Update $\mathbf{w}$ from Eqs. (12) and (9) using training data in $Tset$.
   (c) Compute $K$ nearest images using $\mathbf{w}$.
   (d) User marks the $K$ images as positive or negative.
```

FIG. 4. The probabilistic feature relevance learning (PFRL) Algorithm.

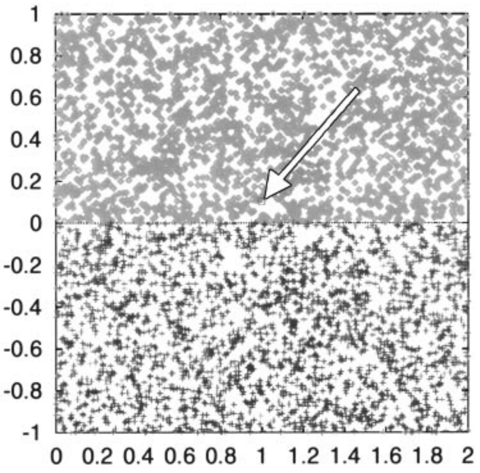

FIG. 5. A simple two-class problem with a uniform distribution. The square (as indicated by an arrow) shows the query.

This example demonstrates that even a simple problem in which the class boundary ideally separates two classes can benefit from the feature relevance learning technique just described, especially when the query approaches the class boundary. It is important to note that for a given distance metric the shape of a neighborhood is fixed, independent of query locations. Furthermore, any distance calculation with equal contribution from each feature variable will always produce spherical neighborhoods. Only by capturing the relevant contribution of the feature variables can a desired neighborhood be realized that is highly customized to query locations.

Experimental validation. We now present an experimental validation of the feature relevance computation algorithm described above. In this experiment, the problem is designed in such a way that all feature dimensions have the same global relevance. However, they have unequal local differential relevance, depending on query locations. There are $q = 5$ feature dimensions and two classes. The data are generated from a normal distribution $\mathbf{x} \sim N(\mathbf{0}, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is given by

$$\mathbf{\Sigma} = \text{diag}\{0.75^2\}_1^q.$$

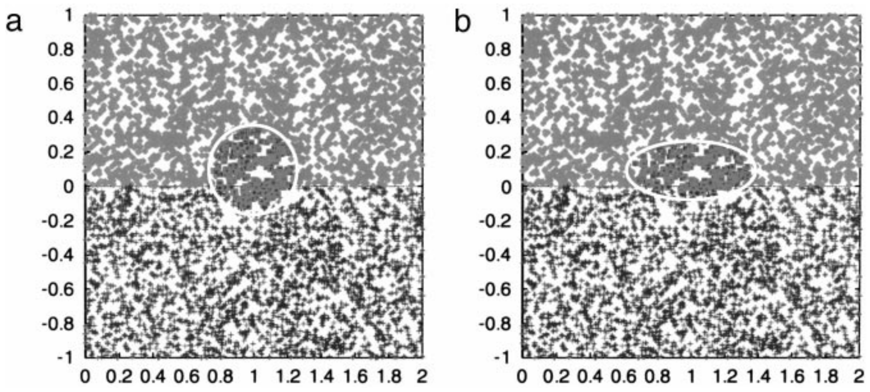

FIG. 6. Effect of feature relevance on retrieval: (a) circular neighborhood (no learning); (b) elliptical neighborhood (after learning).

The classes are defined by

$$\sum_{i=1}^{5} x_i^2 \leq 2.3 \Rightarrow \text{class } 0, \quad \text{otherwise} \Rightarrow \text{class } 1.$$

That is, class 0 is completely surrounded by class 1 in the feature space. There are 10,000 data points total in the database with roughly an equal number of data points in each class.

Four queries from class 1 are generated such that some feature dimensions are more relevant than others. The four representative queries are shown in Table 1. Clearly, feature x_5 is the most discriminating dimension for query q_1. Similarly, features x_4 and x_5 are the most influential dimensions for query q_2 and features x_3, x_4, and x_5, for query q_3. For query q_4 all the dimensions except x_1 are important.

Figure 7 shows the performance of the three weighting schemes (exponential (9), quadratic and linear (8)) on these queries. The performance is measured using the retrieval precision (the number of positive (class 1) retrievals divided by the total number of retrievals) as a function of time. Note that at the first iteration, the retrieval precision is produced by unweighted K-NN search (1). All three weightings demonstrate significant performance improvement over the simple unweighted K-NN method on all the queries. However, the improvement is most pronounced when the number of the relevant features is small. As the number of the relevant features increases, our technique reduces to the unweighted K-NN method as expected. This is correct since when all the features are equally relevant the unweighted Euclidean distance metric (1) is the correct one to use.

It is interesting to note that the exponential weighting (9) seems to produce the best performance on all queries. This is largely due to its sensitivity to changes in conditional expectation (7). Furthermore, it can be seen from Fig. 7 that there is an inherent trade-off between sensitivity and performance. As the number of relevant features increases, the exponential weighting managed to capture the subtle difference between the relevant and irrelevant features, as evidenced by sharp increases in retrieval precision. However, a dramatic increase in retrieval precision increases positive (relevant) images in the resulting retrievals, which in turn makes it highly likely, based on (12), that every dimension becomes relevant (7) at next iteration, thereby lowering retrieval performance. This is particularly true when more features are relevant. This type of sensitivity, however, can be a huge advantage in practical applications, for it reduces the amount of interaction required between the user and the image retrieval process.

Figure 8 illustrates weight changes as a function of iterations based on the quadratic weighting (8). Here two horizontal axes represent input features and iterations, respectively, while the vertical axis represents the magnitude of the weights. It can seen that for all the queries the weights associated with the relevant features are increased and the weights associated with the irrelevant ones are decreased after learning has taken place. These results show convincingly that our method can indeed capture local feature relevance.

We desire that the algorithm be robust in that it produces similar results with similar queries, where by similar queries we mean those queries having the same number of relevant features. In order to see if the algorithm can achieve performance robustness, four additional query points are randomly generated within a neighborhood of each representative query. Figure 9 shows the performance of the quadratic weighting on these queries. It can be seen clearly that the algorithm is indeed capable of producing similar results and capturing corresponding feature(s) as relevant for the given similar queries. Moreover, additional experiments were carried out to determine how the size of Ω in

TABLE 1
Four Representative Queries

Query	x_1	x_2	x_3	x_4	x_5
q_1	0.01	0.005	0.02	0.005	1.531
q_2	0.01	0.02	0.015	1.081	1.081
q_3	0.03	0.08	0.88	0.88	0.88
q_4	0.02	0.76	0.76	0.76	0.76

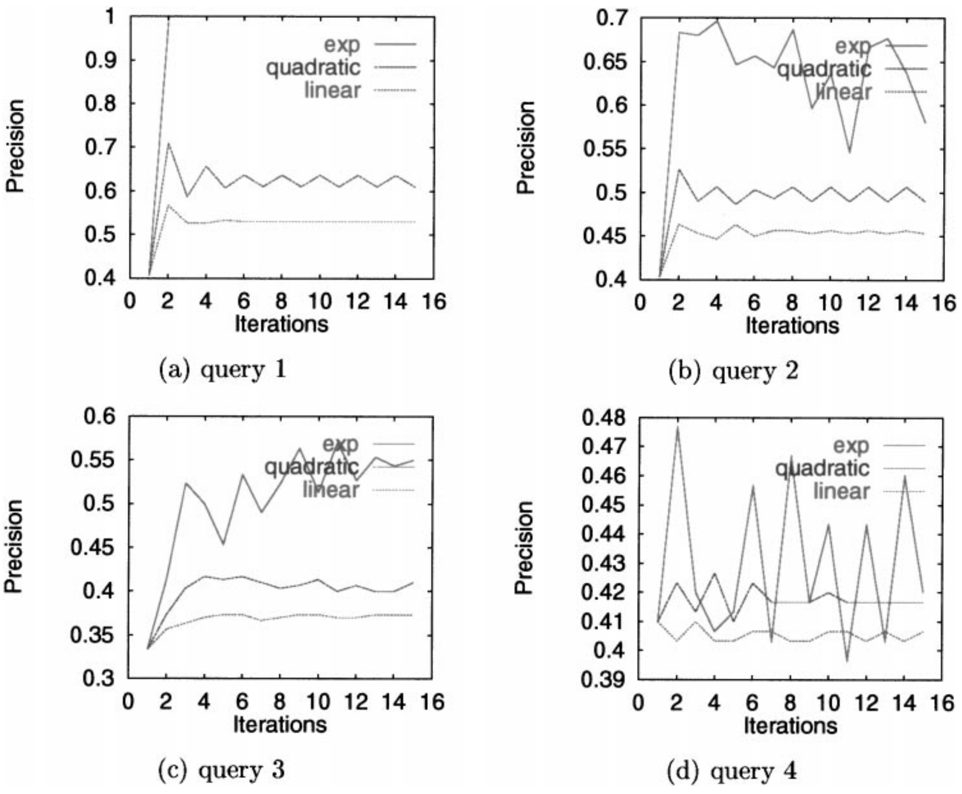

FIG. 7. Performance of exponential, quadratic, and linear weightings on four queries.

(12) will affect the performance of the proposed technique. We omit the details of the experiments here, except to state that our method can tolerate a wide range of values for Ω.

6. EMPIRICAL RESULTS

In the following we compare two competing retrieval methods using both simulated and real data. The simulated data experiments allow us to reliably predict the strengths and limitations of algorithms because the precise nature of the problem the algorithms are facing is known.

METHOD 1. Probabilistic feature relevance learning (PFRL) described in Fig. 4, coupled with the exponential weighting scheme (9).

METHOD 2. The relevance feedback method (RFM) described in [15]. RFM requires that features be normalized according to the following. Let

$$F_i = (f_{i1}, \ldots, f_{ik}, \ldots, f_{iq})$$

be the feature vector representing the ith image in the database. F_i is first transformed into

$$CI_i = \left(\frac{f_{i1}}{\text{mean}_1}, \ldots, \frac{f_{ik}}{\text{mean}_k}, \ldots, \frac{f_{iq}}{\text{mean}_q}\right) \tag{14}$$

where mean_k is the mean value of the kth feature dimension. Then

$$ICI_i = (\log_2(\sigma_{i1} + 2), \ldots, \log_2(\sigma_{ik} + 2), \ldots, \log_2(\sigma_{iq} + 2)), \tag{15}$$

where σ_{ik} is the standard deviation of the kth feature dimension in CI_i. Finally, the normalized feature vector is simply a product of CI_i and ICI_i:

$$\tilde{F}_i = CI_i \times ICI_i. \tag{16}$$

If $\mathbf{x}$ represents the current query, RFM computes a new query according to

$$\mathbf{x} = \alpha\mathbf{x} + \beta\left(\frac{1}{N_r}\sum_{\mathbf{y}_i \in N_r} \mathbf{y}_i\right) - \gamma\left(\frac{1}{N_{ir}}\sum_{\mathbf{y}_i \in N_{ir}} \mathbf{y}_i\right), \tag{17}$$

where N_r represents the set of relevant retrievals and N_{ir} irrelevant ones.

Note that there is a third method, the unweighted K-NN method, that is being compared against implicitly. The first retrieval by PRFL is based on the unweighted K-NN method. Also, in all the experiments, the performance is measured using

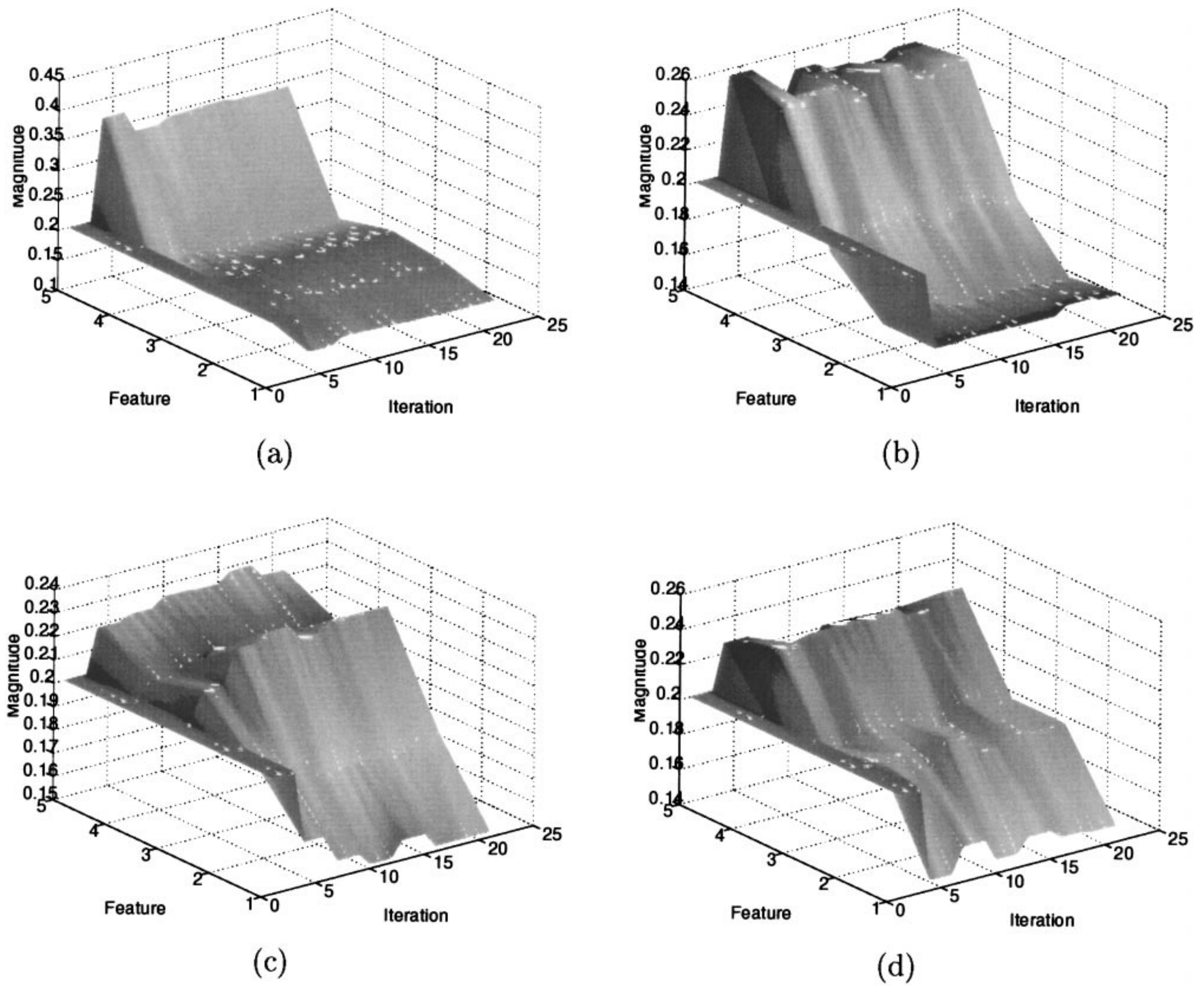

FIG. 8. Weight change for features as a function of iteration: (a) q_1; (b) q_2; (c) q_3; (d) q_4.

the average retrieval precision

$$\text{precision} = \frac{\text{Positive Retrievals}}{\text{Total Retrievals}} \times 100\%. \qquad (18)$$

6.1. Experiments on Simulated Data

For all the experiments reported in this subsection, the features are first normalized over the entire data according to (14), (15) and (16) for RFM, while no normalization takes place for PFRL. There are 500 data in each database.

6.1.1. The Problems

Problem 1 (Two-dimensional XOR). This is the problem shown in Fig. 1. Two classes are distributed in diagonal quadrants. The data $\in [0, 2]^2$ for both classes are generated from a uniform distribution. While the features are rescaled to lie between 0 and 1 for PRFL, they are normalized according to (14), (15), and (16) for RFM.

Problem 2 (Four-dimensional spheres with six noise features). This problem is taken from [7]. There are 10 features and two classes in this problem. The last six features are noise variables, with standard Gaussian distributions, independent of each other and the class membership. The data for both classes are generated from a standard normal distribution. The data for class one have the property that the radius, computed from the first four features, is greater than 3 while the data for class two do not have such restriction. Class one basically surrounds class two in the subspace spanned by the first four features.

Problem 3 (Ten-dimensional spheres). Like Problem 3, there are 10 features and two classes. All 10 features are independent standard normal. All data in class one have the property that their radius is greater than 3 and less than 6, while data in the second class do not have such restrictions. There are no noise variables in this problem. Discriminating information occurs along only one direction in the feature space. Further, this direction changes when moving across the input space and every

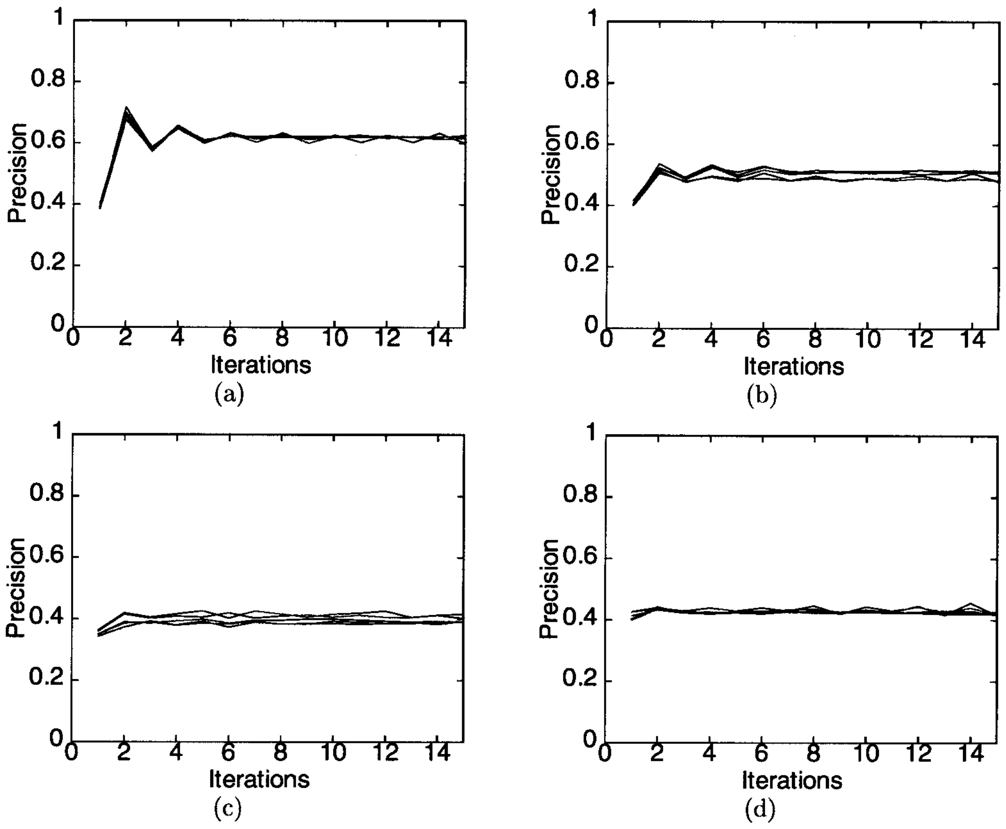

FIG. 9. System performance on simulated data. Relevant features: (a) x_5, (b) x_4, and x_5; (c) x_3, x_4, and x_5; (d) x_2, x_3, x_4, and x_5.

feature becomes important at some point in the space. Again this problem is taken from [7].

PROBLEM 4 (Ten-dimensional ellipsoidals). There are 10 features and two classes in this problem. All of the data are generated according to a standard normal distribution. Two classes are defined by

$$\sum_{i=1}^{10} x_i^2/i \leq 2.5 \Rightarrow \text{class } 1, \quad \text{otherwise} \Rightarrow \text{class } 2.$$

All features are relevant in this problem, but the higher numbered features are more so. This problem is taken from [6].

PROBLEM 5 (Eleven-dimensional hypercube with 10 noise features). There are 11 features and two classes in this problem. All 11 features are uniformly distributed between −1 and 1, independent of each other. The data for class one have the property: $-1 \leq x_1 < -0.5$ or $0 \leq x_1 < 0.5$, whereas, the data for class two have the restriction that $-0.5 \leq x_1 < 0$ or $0.5 \leq x_1 \leq 1$. That is, the two classes separate each other. The class membership is a simple function of x_1 only, and the last 10 features contain no additional information. They serve as noise variables.

6.1.2. Results

Table 2 shows the average retrieval precisions obtained by the two competing methods for the problems described above. The third column in Table 2 shows the average retrieval precision obtained by each method without any relevance feedback (0 rf). For PRFL, this average retrieval precision is obtained by the unweighted K-NN method. The fourth column shows the average retrieval precision computed after learning has taken place once (1 rf). That is, relevance feedback obtained from the previous retrieval is used to compute a new query in case of RFM or probabilistic local feature relevance, hence new weighting, in case of PFRL, respectively. The last column shows the relative

TABLE 2
Average Retrieval Precision for Simulated Data

Problem	Method	0 (rf)	1 (rf)	Improvement
Problem 1	PFRL	96.50	97.75	1.98
	RFM	66.49	67.38	0.89
Problem 2	PFRL	65.37	83.42	38.24
	RFM	57.37	72.57	32.42
Problem 3	PFRL	59.80	76.59	53.22
	RFM	52.23	59.20	16.32
Problem 4	PFRL	61.23	80.80	52.70
	RFM	54.93	66.86	24.86
Problem 5	PFRL	59.82	88.02	53.47
	RFM	57.14	75.46	41.87

performance improvement by the two methods, where the average performance improvement (API) is computed according to

$$\text{API} = \frac{\text{Positive Retrievals}(n+1) - \text{Positive Retrievals}(n)}{\text{Positive Retrievals}(n)} \times 100\% \tag{19}$$

averaged over all queries, where Positive Retrievals(n) represents the positive retrievals at the nth iteration.

There are two procedural parameters (T (9) and C (13) input to the PFRL algorithm. The values of the parameters used to obtain the results reported in Table 2 were determined experimentally. These values are (15, 16), (14, 16), (15, 10), (10, 8), and (22, 8) for problems 1, 2, 3, 4, and 5, respectively. Similarly, we experimented with the parameters (α, β, and γ) [15] input to RFM, and the best performance results found in those experiments are reported in Table 2. These values are (1, 2, 1), (2, 2, 3), (2, 4, 4), (1, 4, 2), and (2, 2, 4) for problems 1, 2, 3, 4, and 5, respectively. Note that these experiments are by no means exhaustive.

It can be seen from Table 2 that both methods show performance improvement across all the tasks. However, the margins of improvement achieved by PRFL are much greater than that obtained by RFM. Furthermore, PRFL did consistently better than RFM. One thing to notice is that RFM performed poorly on the XOR problem (Problem 1). This can be attributed to the nonmetric similarity function employed by RFM [15],

$$\text{Sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^{\text{t}}\mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|}, \tag{20}$$

where $\mathbf{x}$ and $\mathbf{y}$ are feature vectors, t denotes transpose, and $\|\cdot\|$ represents the L_2 norm. This function is invariant to rotation and dilation, but it is variant to translation and general linear transformation [4]. Should we normalize the features so as to have zero mean and unit variance, RFM would have attained the same performance level for problem 1 as that by PRFL.

An important observation one can make from Table 2 is that all the problems, except the first one, do not favor RFM regardless of normalization procedures employed. It is not hard to show that, in these problems, a line in the input space along a query vector will almost always intercept both classes. In this case, the mere rotation of the query vector, which is carried out by RFM for computing a new query vector using relevance feedback (Eq. 17), does not necessarily move the query closer to the relevant class and away from the irrelevant one. On the other hand, in PFRL capturing local feature relevance is sufficient to produce a neighborhood whose shape is tailored to the particular query so that the number of the retrievals similar to the query is increased, as evidenced by the results shown in Table 2.

Note that here we only show results at one iteration after receiving the relevance feedback. The reason is that subsequent relevance feedbacks only give rise to minor performance improvement in the average retrieval precision. This result is corroborated by the experiments performed in [15]. As pointed out in [15], this is considered highly desirable, since acceptable results can be achieved with the minimum number of feedback cycles. As a comparison, however, we performed experiments in which RFM was allowed to receive the second relevance feedback and recompute a new query vector. The average retrieval precisions for the five problems are 67.91, 78.16, 64.72, 70.66, and 83.44, respectively. The results show that given twice the amount of computation, RFM still could not achieve the level of performance obtained by that of PFRL on the problems examined here.

It may be argued that the problems chosen here are not particularly in favor of RFM. Most of the problems are constructed so that one class is surrounded by another in the feature space. This type of situation presents most difficulties to RFM. However, it also poses considerable challenges to the PRFL method. Differential relevance information occurs along one direction. Further, this direction changes when one moves across the feature space. In general, such relevance information is difficult for any method to capture. And it seems likely that this type of situation would occur often in the real world. Thus, it is our view that these problems provide a reasonable basis for comparing the two competing methods.

6.2. Experiments on Real Data

In order to compare the two competing methods more objectively, original features are normalized in three different ways. For the probabilistic feature learning method described above, the normalization is carried out along each feature dimension over the entire data set in such a way that the normalized feature values lie between 0 and 1. We call this normalization process *scale*. This process does not in any way provide inductive bias in favor of the learning method. It simply removes some of artifacts due to different scales of variables that are generally considered undesirable in the absence of any additional information. This is particularly true for retrieval procedures whose distance computation is based on the Euclidean metric (1).

For RFM, the features are normalized according to (14), (15), and (16). This normalization process attempts to explicitly capture feature importance within a feature vector as well as across different feature vectors over the entire data collection, thereby enabling RFM to take advantage of some of the well-known results from information retrieval. Following [15], we denote this normalization procedure by $tf \times idf$.

While the two competing methods perform image retrieval on the same database, they use different input representations. In order for the two methods to receive exactly the same inputs, the features are normalized so that each dimension will have zero mean and unit variance. And as such, the normalized features cannot be further altered by (14), (15), and (16). Thus, the two methods see exactly the same database and input representation. This procedure is denoted by 01.

6.2.1. *The Problems*

Problem 1. The data in the first problem, from the UCI repository [12] consist of images that were drawn randomly from a database of seven outdoor images. The images were hand-segmented by the creators of the database to classify each pixel. Each image is a region. There are seven classes: *brick-face*, *sky*, *foliage*, *cement*, *window*, *path*, and *grass*, each having 330 instances. Thus, there are total 2310 images in the database. These images are represented by 19 real-valued attributes that are described in Table 3. These features are basically statistical moments and line counts. For further details, see [12].

Problem 2. The data in the second problem are obtained from MIT Media Lab at ftp://whitechapel.media.mit.edu/pub/VisTex in the same way as in [15]. There are 40 texture images that are manually classified into 15 classes. Each of these images is then cut into 16 nonoverlapping images of 128×128. Thus, there are total 640 images in the database. Also, the number of images in each class varies from 16 to 80. The images in this database are represented by 16-dimensional feature vectors. We use 16 Gabor filters (two scales and four orientations), described in the Appendix, for feature extraction. The mean μ_{mn} (21) and the standard deviation σ_{mn} (22) of the magnitude of the transform coefficients are used as feature components (23) after being normalized by the standard deviations of the respective features over the entire set of images in the database. Sample images are shown in Fig. 10.

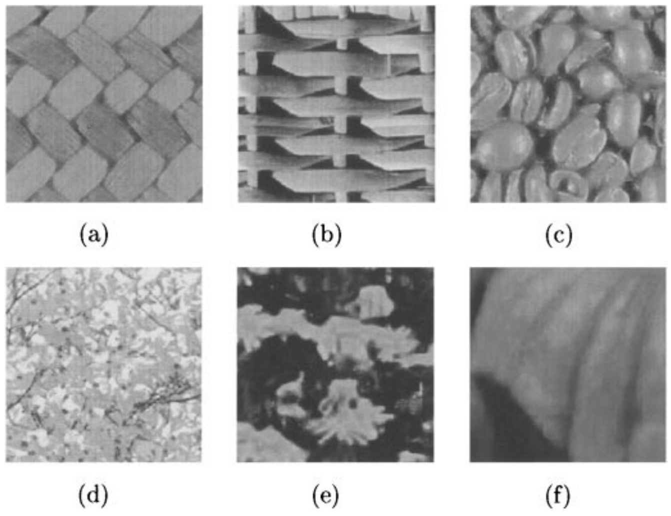
FIG. 10. Sample images from MIT database.

6.2.2. *Results*

For both problems, each image in the database is selected as a query and top 20 nearest neighbors are returned that provide necessary relevance feedback. The average retrieval precision is summarized in Table 4. There are four rows under each problem in the table. The first two rows indicate the performance of the two methods under the condition that the features are normalized so as to have zero mean and unit variance. The third row shows the results obtained by PFRL, conditioned on the features being scaled to lie between 0 and 1. The fourth row shows RFM's performance given that the features are normalized according to (14), (15), and (16).

The second column in Table 4 shows the average retrieval precision obtained by each method without any relevance feedback. The third column shows the average retrieval precision computed after learning has taken place. That is, relevance feedback obtained from the previous retrieval is used to compute a new query in case of RFM, or to estimate local feature relevance, hence a new weighting, in case of PFRL, respectively.

TABLE 3
Feature Information

Feature		Description
1	Region-centroid-col	Center pixel column of the region
2	Region-centroid-row	Center pixel row of the region
3	Region-pixel-count	Number of pixels in a region
4	Short-line-density-5	Line count, low contrast, ≤ 5
5	Short-line-density-2	Line count, high contrast, >5
6	Vedge-mean	Contrast of horizontally adjacent pixels
7	Vedge-sd	See 6
8	Hedge-mean	Contrast of vertically adjacent pixels
9	Hedge-sd	See 8
10	Iintensity-mean	$(R+G+B)/3$
11	Rawred-mean	Average of the R value
12	Rawblue-mean	Average of the B value
13	Rawgreen-mean	Average of the G value
14	Exred-mean	$(2R-(G+B))$
15	Exblue-mean	$(2B-(G+R))$
16	Exgreen-mean	$(2G-(R+B))$
17	Value-mean	3D nonlinear transformation of RGB
18	Saturation-mean	See 17
19	Hue-mean	See 17

TABLE 4
Average Retrieval Precision for Real Data

Method	0 (rf)	1 (rf)	Improvement
UCI database			
PFRL (01)	92.10	95.64	6.82
RFM (01)	91.25	95.12	9.75
PFRL (*scale*)	92.08	96.05	7.66
RFM ($tf \times idf$)	86.39	91.95	15.33
MIT database			
Method	0 (rf)	1 (rf)	Improvement
PFRL (01)	77.05	84.02	14.37
RFM (01)	81.79	89.63	17.76
PFRL (*scale*)	78.27	84.44	12.70
RFM ($tf \times idf$)	83.74	90.23	13.53

The last column shows relative performance improvement by the two methods. It can be seen from Table 4 that both methods demonstrate significant performance improvement across the tasks. In general, PFRL seems to slightly outperform RFM on the UCI data, whereas RFM achieves better results with the MIT data. The two competing methods seem compatible, at least for the problem experiments here. Furthermore, various normalization techniques do not seem to have a significant impact on the overall ability of each method. Finally, similar to the simulated data experiments, when RFM is allowed to update the query after receiving the second relevance feedback, there is a slight improvement in the retrieval results (UCI data: 96.63 (01) and 93.94 ($tf \times idf$); MIT data: 91.81 (01) and 91.88 ($tf \times idf$)). However, this is at the expense of increased computation, which can be prohibitive when the database is very large.

Note that the procedural parameters (T (9) and C (13)) input to PFRL were determined empirically, and they were set to 15 and 16, respectively, in all the experiments reported in this section. Similarly, we experimented with the parameters (α, β, and γ) [15] input to RFM, and the best performance results found in those experiments are reported in Table 4. For the UCI data, these values are (1, 3, 2) for 01 normalization and (1, 4, 3) for ($tf \times idf$). Likewise, for the MIT data these values are (1, 2, 1) for 01 normalization and (1, 4, 2) for ($tf \times idf$), respectively.

Figure 11 shows a particular retrieval result obtained by PFRL from the MIT image database with no learning; that is, each dimension is weighted equally in the distance computation (10), where a retrieval precision of 25 is achieved. Note that Fig. 11a represents the query image. In contrast, Fig. 12 shows the retrieval results after learning has taken place, where the results in Fig. 11 provide relevance feedback. In this case, a retrieval precision of 95 is achieved. This illustrates that capturing local feature relevance indeed helps to improve retrieval performance.

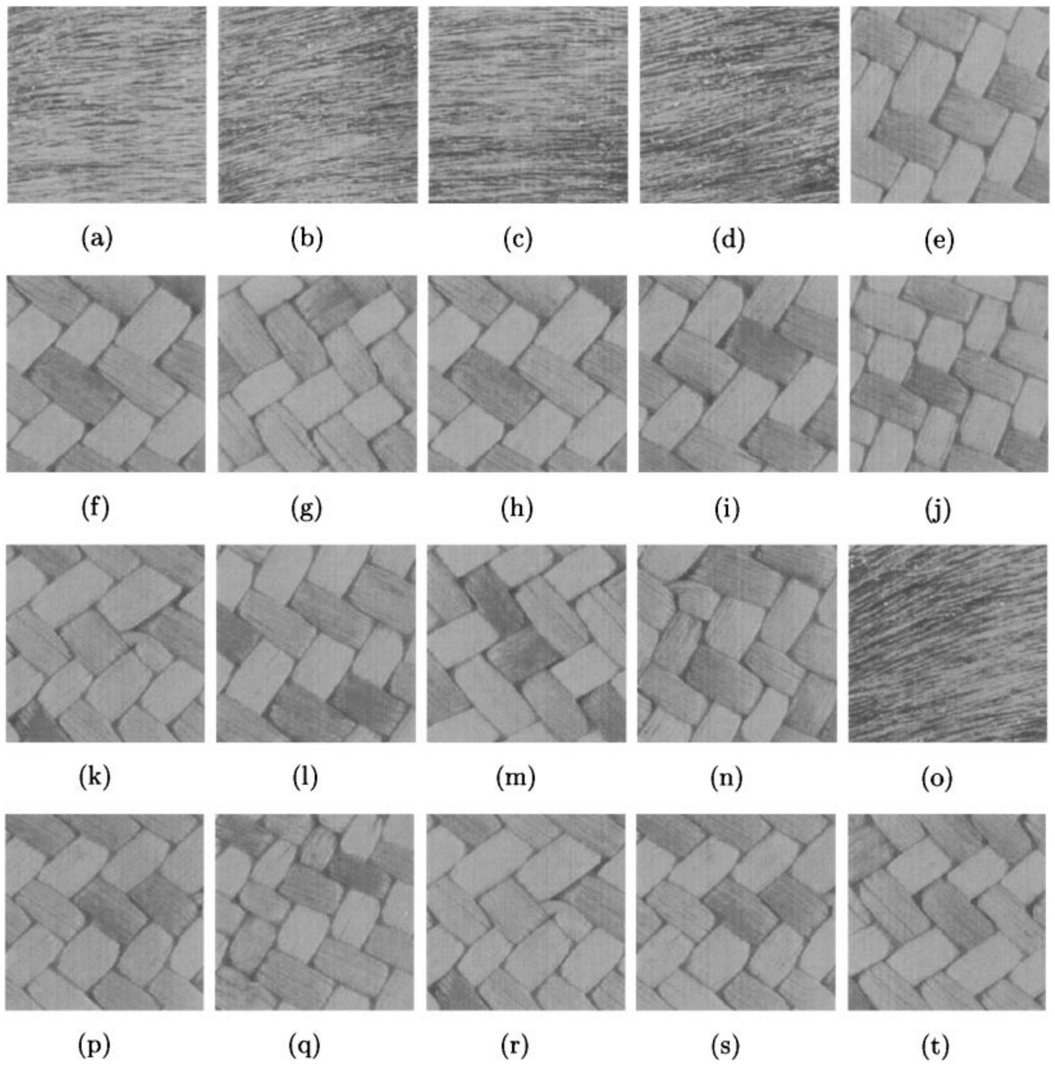

FIG. 11. Retrieval results without learning, where (a) represents the query image. Retrieval precision: 0.25.

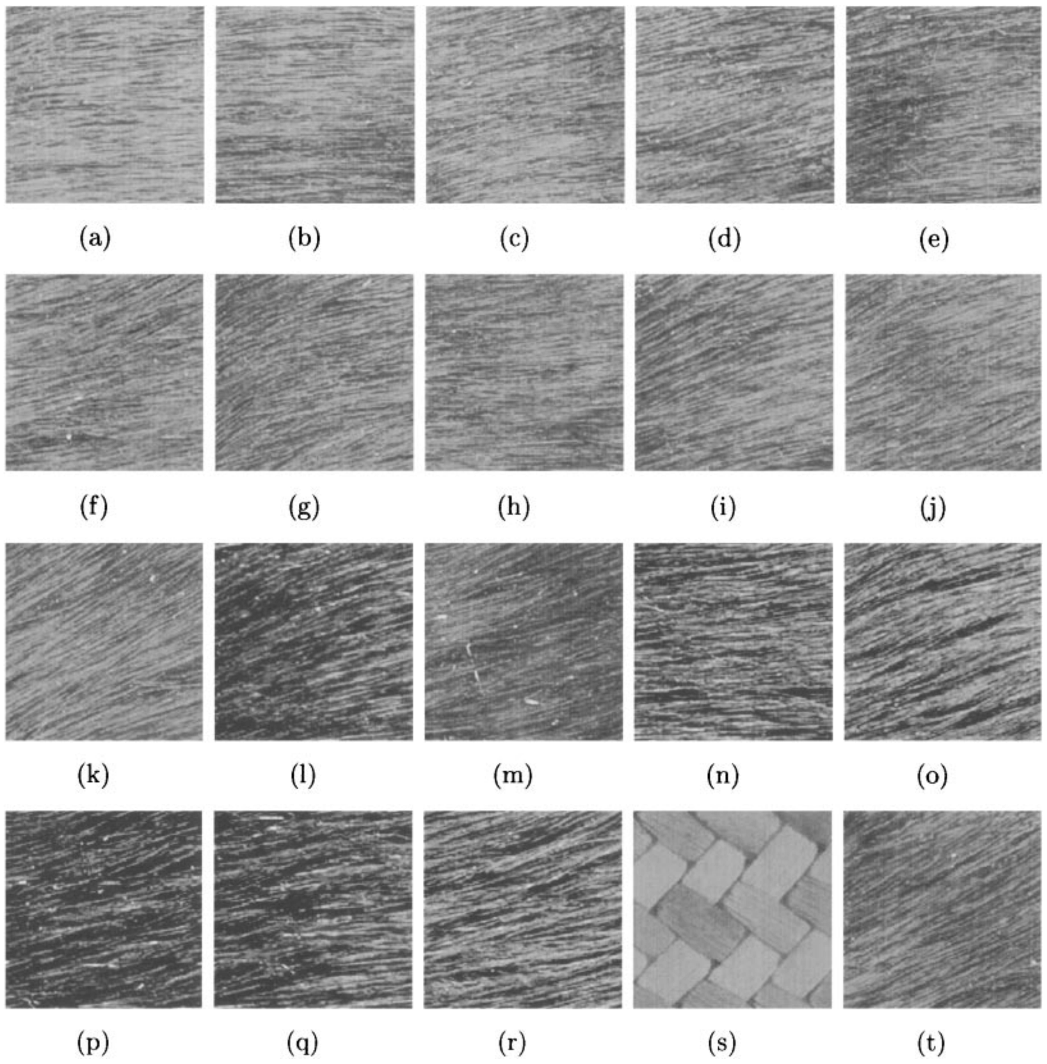

FIG. 12. Retrieval results with learning, where (a) indicates the query image. Retrieval precision: 95.

7. CONCLUSIONS

This paper presents a novel probabilistic feature relevance learning technique for efficient content-based image retrieval. The experimental results using both simulated and real data show convincingly that learning feature relevance based on user's feedback can indeed improve retrieval performance of an image database system. Furthermore, since the relevance estimate is local in nature, the resulting retrieval, in terms of the shape of the neighborhood, is highly adaptive and customized to the query location.

Our retrieval technique learns local feature relevance for each given query. However, it is possible that the knowledge acquired during one retrieval can be gradually collected and it can become part of the database itself through continuous learning. This knowledge can be used in conjunction with case-based learning [1, 2, 14] to achieve generalization in future retrievals in order to further optimize the performance of the system.

A potential extension to the technique described in this paper is to consider additional derived variables (features) for local relevance estimate, thereby contributing to the distance calculation. The derived features are functions, such as linear functions, of the original features. When the derived features are more informative, huge gains may be expected. On the other hand, if they are not informative enough, they may cause retrieval performance to degrade since they add to the dimensionality count. The challenge is to be able to have a mechanism that computes such informative derived features efficiently.

APPENDIX: GABOR WAVELET REPRESENTATION

We use Gabor wavelets [9, 17] to extract texture features. A two-dimensional Gabor function $g(x, y)$ can be written as

$$g(x, y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right)\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right].$$

Using the above formula as the mother function, a set of self-similar filters are derived through the generating function

$$g_{mn}(x, y) = a^{-m} g(x', y')$$

and

$$x' = a^{-m}(x \cos\theta + y \sin\theta), \quad y' = a^{-m}(-x \sin\theta + y \cos\theta),$$

where $a > 1$, $\theta = n\pi/K$, $n = 0, 1, \ldots, K-1$, and $m = 0, 1, \ldots, S-1$. K is the total number of orientations and S is the total number of scales.

Given an image $I(x, y)$, its Gabor wavelet transform is then defined

$$W_{mn}(x, y) = \int\int I(x_1, y_1) g^*_{mn}(x - x_1, y - y_1)\, dx_1\, dy_1,$$

where * indicates the complex conjugate. We then compute

$$\mu_{mn} = \int\int |W_{mn}(x, y)|\, dx\, dy \tag{21}$$

and

$$\sigma_{mn} = \sqrt{\int\int (|W_{mn}(x, y)| - \mu_{mn})^2\, dx\, dy}. \tag{22}$$

In our system, $S = 2$ and $K = 4$, thereby generating a 16-dimensional feature vector of the form for each texture image

$$f = [\mu_{00}\sigma_{00} \cdots \mu_{16}\sigma_{16}]. \tag{23}$$

ACKNOWLEDGMENTS

This work was supported by DARPA/AFOSR Grants F9620-97-1-0184 at the University of California, Riverside. The contents of the information do not necessarily reflect the position or the policy of the Government.

REFERENCES

1. D. W. Aha, D. Kibler, and M. K. Albert, Instance-based learning algorithms, *Mach. Learning* **6**, 1991, 37–66.
2. C. Atkeson, A. W. Moore, and S. Schaal, Locally weighted learning, *AI Rev.* **11**, 1997, 11–73.
3. R. H. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz, Trading mips and memory for knowledge engineering, *Comm. ACM* **35**, 1992, 48–64.
4. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
5. M. Flickner *et al.*, Query by image and video content: The QBIC system, *IEEE Comput.* September 1995, 23–31.
6. J. H. Friedman, *Flexible Metric Nearest Neighbor Classification*, Tech. report, Dept. of Statistics, Stanford University, 1994.
7. T. Hastie and R. Tibshirani, Discriminant adaptive nearest neighbor classification, *IEEE Trans. Pattern Anal. Mach. Intell.* **18**, 6, 1996, 607–616.
8. R. Jain, S. N. J. Murthy, and L. Tran, Similarity measures for image databases, in *IEEE Conference on Fuzzy Logic, 1995*.
9. B. S. Manjunath and W. Y. Ma, Texture feature for browsing and retrieval of image data, *IEEE Trans. Pattern Anal. Mach. Intell.* **18**, 8, 1996, 837–842.
10. T. P. Minka and R. W. Picard, Interactive learning with a "society of models," *Pattern Recognit.* **30**, 4, 1997, 565–581.
11. T. Mitchell, *Mach. Learning*, McGraw–Hill, New York, 1997.
12. P. M. Murphy and D. W. Aha, UCI repository of machine learning databases. http://www.cs.uci.edu/mlearn/MLRepository.html, 1995.
13. R. M. Nosofsky, Attention, similarity, and the identification-categorization relationship, *J. Exp. Psych. General* **15**, 1986, 39–57.
14. J. Peng, Efficient memory-based dynamic programming, in *Proceedings, 12th International Conference on Machine Learning, Tahoe City, CA, July 1995*, pp. 438–446.
15. Y. Rui, T. S. Huang, and S. Mehrotra, Content-based image retrieval with relevance feedback in MARS, in *Proceedings IEEE International Conference on Image Processing, Santa Barbara, CA, October, 1997*, pp. 815–818.
16. J. R. Trott and B. Leng, An engineering approach for troubleshooting case bases, in *Proceedings 2nd ICCBR Conference, Providence, RI*, pp. 178–189, Springer-Verlag, New York, 1997.
17. X. Wu and B. Bhanu, Gabor wavelet representation for 3-D object recognition, *IEEE Trans. Image Process.* **6**, 1, 1997, 47–64.