Similarity Learning via Dissimilarity Space in CBIR

Giang P. Nguyen, Marcel Worring, and Arnold W.M. Smeulders
Intelligent Systems Lab Amsterdam, University of Amsterdam
Kruislaan 403, 1098SJ Amsterdam, The Netherlands.
{giangnp,worring,smeulders}@science.uva.nl

ABSTRACT

In this paper, we introduce a new approach to learn dissimilarity for interactive search in content based image retrieval. In literature, dissimilarity is often learned via the feature space by feature selection, feature weighting or a parameterized function of the features. Different from existing techniques, we use relevance feedback to adjust dissimilarity in a dissimilarity space. To create a dissimilarity space, we use Pekalska's method [15]. After the user gives feedback, we apply active learning with one-class SVM on this space. Results on a Corel dataset of 10000 images and a TrecVid collection of 43907 keyframes show that our proposed approach can improve the retrieval performance over the feature space based approach.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: [Search process, Relevance feedback]

General Terms

Experimentation, Algorithm

Keywords

Dissimilarity learning, interactive search, visualization

1. INTRODUCTION

Interactive search tasks in content-based image retrieval (CBIR) are classified into three types namely association search, target search and category search [22]. Search by association is a class of searching images where the user starts the search with no specific aim other than interesting findings. Target search aims at finding one specific image. Finally, category search looks for all images belonging to a specific class. In any of the three tasks, during the search process, the system aims to find relevant images and discard irrelevant ones. To do so, a dissimilarity measure is needed to compare and sort images. However, the dissimilarity between images strongly depends on the search context. The system's pre-definition of dissimilarity is based on objective interpretation of the image content, whereas the user interprets the image in its semantic context. This problem is known as the semantic gap [22]. Let us give an example, where we have two sets of pictures, one of "dogs" and the other of "birds". In a search task looking for images of the "animal" category, images in these two sets should be considered similar. However, if the task is searching images of "dogs", the pictures with "birds" are not relevant. Only the user knows exactly what she is searching for and the systems needs to learn the dissimilarity based on the user's relevance feedback.

In literature, many different methods have been developed to learn dissimilarity measures from relevance feedback. For an overview see [19, 29]. To provide feedback, a *manipulation space* with which the users can interact is essential. This space, must show a selection of images to the user in a way suited for interaction. The user may interact in the space by labelling images as relevant and/or irrelevant, by giving dissimilarity scores, or moving relevant images close to one another and so on. When feedback is given, the dissimilarity is commonly learned via the feature space [1, 11, 9]. Using feature space has two disadvantages. First, to effectively learn the dissimilarity, a lot of generic features should be defined or a small set of specific features. A large set of generic features, provides the possibility to find a feature combination suited for describing the dissimilarities among the images using a form of feature weighting. However, a large set of features leads to a high computational load, especially when the dimension of the feature space goes to thousands of features. For interactive search, immediate response is important, the computational expense should be restricted. A small selection of specific features usually works for a narrow domain only. Second, the individual features have no meaning to the user so the feature space can not be mapped to manipulation in an intuitive manner. Hence, for efficient and intuitive search in broad domains a different method is needed.

Let us reconsider the above example. It is difficult to define effective features to assign the two images to the "animal" group. However, if the user points out that the images searched for are similar to the example picture of a "dog" and an example of a"bird" we might group them based on the observation that they are close to either one of them. Defining concepts on the basis of examples is also far more intuitive for the user, hence it can be used in manipulation

MIR'06, October 26–27, 2006, Santa Barbara, California, USA.

space directly. In our approach, rather than considering the feature space, we will focus on the *dissimilarity space*, where images are represented by their relations to other images.

A similar approach has been applied in [2]. In this reference, the authors also consider dissimilarity space as a replacement of the feature space. They create dissimilarity spaces following the technique of Duin and Pekalska [5, 15]. They first select a set of images, named prototypes. The dissimilarity space is created such that images are represented in their relative dissimilarities to the prototypes. They then explore the optimal way of fusing these spaces, which they call multi-modal dissimilarity spaces. However, no specific comparison of this approach to the learning on feature space has been given as well as examining factors related to the performance of the dissimilarity space based approach.

In this paper, we not only present the learning on dissimilarity space, but also explicitly experiment on the performance of our approach over existing methods. The paper is organized as follows. In section 2, we will describe in more detail existing research in learning dissimilarity. Next, in section 3, we present our approach with active learning on the dissimilarity. Results of the system with two different image collections are shown in section 4. Finally, conclusions are presented in section 6.

2. BACKGROUND AND RELATED WORK

In this section, we introduce some essential notation and give an overview of existing literature on learning dissimilarity.

2.1 Essential notations

Given a collection of images $\mathcal{I} = \{I_1, I_2, ..., I_n\}$, a r- dimensional feature space $\mathcal{F}$ is defined in which each image $I_i \in \mathcal{I}$ is represented by a feature vector $\vec{F_i}$ of length r. Dissimilarities between every pair of images I_i and I_j are stored in a matrix $\mathcal{S} = \{S(I_i, I_j)\}_{i=1,n;j=1,n}$. Let $\vec{W} = \{w_1, w_2, \ldots, w_r\}$ denote a set of r values weighting the different elements of $\mathcal{F}$. Finally, let $\vec{\zeta}$ denote a set of parameters steering the dissimilarity function.

Now when the user is interacting with the system he has a goal which can be defined as a set of desired images $\mathcal{I}_+ \subset \mathcal{I}$ to be found. Given this goal, learning of dissimilarity can be viewed as an iterative process where the systems learns to identify the set $\mathcal{I}_+$ from feedback given by the user.

2.2 Methods for learning dissimilarity

In most existing methods, learning is done via the feature space either by feature selection [1, 11], feature weighting [9, 28], or using a parameter based function of features [20, 7].

In general, for existing methods, learning the dissimilarity in iteration $t+1$ from analyzing the feedback from the user in iteration t can be formulated as follows:

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{W}^t, \vec{\zeta}^t), \quad (1)$$

where $\vec{W}^0$ and $\vec{\zeta}^0$ are set to default values.

Not all features are equally important. Hence, in the feature weighting approach weights are set for each feature. Feature selection is a special case of feature weighting, where the weights of the eliminated features are set to 0. As the update changes $\vec{W}$ only, Eq.1 can be rewritten as:

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{W}^t). \quad (2)$$

In [28], the authors concentrate on exploring the distribution of the datas set. A subspace of the feature space is found, and a quadratic similarity functions is learnt. From there, the dissimilarity matrix between images is updated. A similar approach is presented in [1], where the authors propose a weighted Minkowski similarity that continuously learns the weight of each feature based on positive and negative examples. The dissimilarity matrix is adjusted on the basis of the new set of weights. In [11], the similarity is also recalculated by selecting a subspace. Updating is based on the configuration of images on the screen resulting from the user's manipulation of the position of images on the screen. The system re-estimates the layout of the images, such that the similarity function gets closer to the user's desire. A similar approach but with dynamic selection of the feature subspace is done in [9]. In this reference, starting with a number of features, a dynamic function is proposed where at each step the optimal number of features is found. From there, the weighted and non-weighted perceptual dynamic function is built based on the Minkowski distance. In [6], the authors propose a system which allows the user to score the similarity between given pairs of images. The system then predicts the similarity coefficient from the user feedback and learns the similarity of the others. Within the same school of thought, work has been reported in [3, 8, 21, 25]. In general, this approach requires a large set of features in order to select an efficient subset best representing the semantic similarity between images. However, the selection of large number of features has a major disadvantage for interactive search because of its computational expense, especially with complicated dissimilarity functions.

Another class of methods is formed by the parameter based approaches. In these approaches, the matrix in the feature space does not change during learning, but rather a parameterized function of the features is adjusted to fit the user's feedback. Eq.1 is reformulated as:

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{\zeta}^t), \quad (3)$$

For example, in [20], the authors introduce an interface where the user adjusts the similarity between images in the manipulation space by moving them around. New positions of images displayed are used as relevance feedback. Using Fuzzy Feature Contrast and Tversky's similarity measure, the authors define a similarity function where $\vec{\zeta}$ contains around 100 different parameters. Based on user feedback, the system then adjusts the set of parameters in the dissimilarity function such that the dissimilarity decreases between images which according to the user are close. In [7], given a set of images as query examples, a restricted similarity measure is formulated which recalculates dissimilarity between all images and queries depending on their positions compared to the classification boundary. The boundary is characterized by a parameter set. Given a set of positive and negative examples, SVM and AdaBoost are used to learn a classification boundary. The top ranked images are then labelled as positive and negative examples to repeat the refinement of dissimilarity.

The parameter based approaches do not require a large set of features. However, to effectively learn the dissimilarity matrix either the features should be well chosen or the system should have a wide range of parameters.

3. OUR APPROACH

In this section, we present our approach in learning dissimilarity in the dissimilarity space based on user's relevance feedback. Using the same type of notation as in eq.1 our method can be described as:

$$S^{t+1}(I_i, I_j) = f(S^t(I_i, I_j)), \text{ with } S^0(I_i, I_j) = f(\vec{F}_i, \vec{F}_j). \tag{4}$$

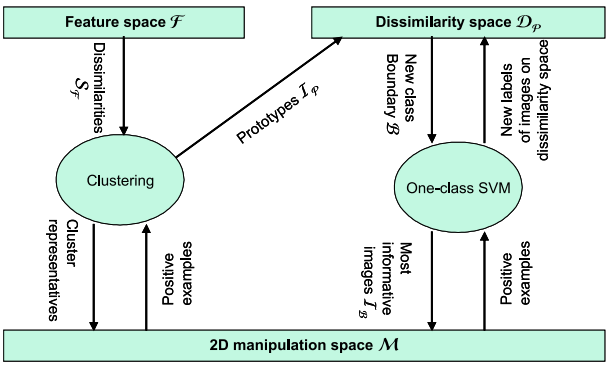


Figure 1: Schematic overview of the proposed approach.

An overview of our proposed approach is in figure 1. First, a dissimilarity matrix is obtained by comparing feature vectors in the feature space $\mathcal{F}$. A projection from the high dimensional space is used to create a manipulation space $\mathcal{M}$ (to be discussed in section 3.2). Images are presented in $\mathcal{M}$ to the user for interaction and feedback. A set of images, named the prototype set $\mathcal{I}_\mathcal{P}$ is selected. When sufficient prototypes have found, a dissimilarity space $\mathcal{D}_\mathcal{P}$ is then created in section 3.1. The learning process is then started. The manipulation space $\mathcal{M}$ is now a projection of $\mathcal{D}_\mathcal{P}$. The learning phase is presented in section 3.3. At each iteration, a set of most informative images is returned. The user then labels positive images for another round of feedback. The learning phase is finished when the user stops the search.

3.1 Prototype-based dissimilarity space

To create a dissimilarity space, we employ the method proposed by Pekalska [15]. In the reference, the goal is data classification with no user interaction or relevance feedback, we extend it to interactive search.

The first step is to select a set of p images $\mathcal{I}_\mathcal{P} \subset \mathcal{I}$, called the *prototypes*:

$$\mathcal{I}_\mathcal{P} = \{I_{P_1}, I_{P_2}, \ldots, I_{P_p}\} \tag{5}$$

The role of the prototypes is to create a dissimilarity space where relevant and irrelevant images are well separated. Hence, careful selection of prototypes is important. The mapping from dissimilarity matrix to dissimilarity space by selection of prototypes is equivalent to choosing a set of columns (or rows) in the dissimilarity matrix. $\mathcal{D}_\mathcal{P}$ denotes the dissimilarity space, and Φ the mapping from a dissimilarity matrix $\mathcal{S}$ to $\mathcal{D}_\mathcal{P}$:

$$\Phi : \mathcal{S} \overset{\mathcal{I}_\mathcal{P}}{\longmapsto} \mathcal{D}_\mathcal{P}. \tag{6}$$

This means that for each image I_i, we have a p-dimensional vector

$$\mathcal{D}_i = \{S(I_i, I_{P_1}), S(I_i, I_{P_2}), \ldots, S(I_i, I_{P_p})\} \tag{7}$$

Therefore, dissimilarities between all images in $\mathcal{I}$ to $\mathcal{I}_\mathcal{P}$ are represented by a matrix with size $n \times p$. The collection $\mathcal{I}$ then builds up a p-dimensional dissimilarity space $\mathcal{D}_\mathcal{P}$, named as *prototype-based dissimilarity space*:

$$\mathcal{D}_\mathcal{P} = \begin{bmatrix} S(I_1, I_{P_1}), & S(I_1, I_{P_2}), & \ldots, & S(I_1, I_{P_p}) \\ S(I_2, I_{P_1}), & S(I_2, I_{P_2}), & \ldots, & S(I_2, I_{P_p}) \\ \vdots & & \vdots & \\ S(I_n, I_{P_1}), & S(I_n, I_{P_2}), & \ldots, & S(I_n, I_{P_p}) \end{bmatrix} \tag{8}$$

An illustration of creating a dissimilarity space is shown in figure 2.

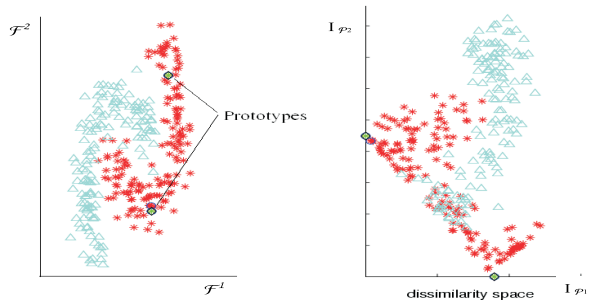


Figure 2: An example to illustrate the creation of a dissimilarity space. In this example, for simplicity, images are represented in a 2D feature space $\mathcal{F} = \{\mathcal{F}1, \mathcal{F}2\}$, with dissimilarities among them obtained by the Euclidean distance between feature vectors. Two images are selected as prototypes. Based on distances between all images to the prototypes I_{P_1} and I_{P_2}, we create a 2D dissimilarity space.

In $\mathcal{D}_\mathcal{P}$ the similarity $S_\mathcal{P}(I_i, I_j)$ of two images is defined by the Euclidean distance between D_i and D_j.

For selecting prototypes, it is argued in [16] that systematic selection of prototypes such as k-means gives a better representation for of dissimilarity space than random selection. We, therefore, apply k-means for clustering the collection to find a set of prototypes $\mathcal{I}_\mathcal{P}$ such that the mapping Φ preserves the information in the similarity matrix $\mathcal{S}$ as good as possible. In interactive search, we are not able to select a set of prototypes as we do not know which images the user will search for. In practice, there are cases where the user starts the search with relevant and/or irrelevant images. In general this set of images is not a good set of prototypes. For CBIR, a browsing strategy is needed for finding a good set of prototypes. For that goal, we group the collection into a number of clusters. In the browsing, a set of images, where each image is selected from a cluster as a representative, is displayed to the user. If the user marks an image is relevant, it will be collected as a prototype. The browsing process is finished when enough prototypes found.

3.2 The manipulation space

To provide relevance feedback a 2-dimensional manipulation space $\mathcal{M}$ is needed in which the user interacts with the images. Ideally, there is a direct relation between the similarities defined in the high dimensional space, being it

feature space or dissimilarity space, and the manipulation space.

A projection from the high dimensional space to a 2-dimensional manipulation space is needed. Similarity based visualization, [11, 18, 12, 17] which aims to preserve the dissimilarities between every pair of images in the manipulation space is highly appropriate for this task.

Let x_i denote the position of image I_i in manipulation space. Furthermore, let $\mathcal{S}_{\mathcal{M}}(I_i, I_j)$ be the Euclidean distance between the images in manipulation space $\mathcal{M}$. In similarity based visualization, to faithfully represent the dissimilarity space, $\mathcal{S}_{\mathcal{M}}$ should reflect the similarity $\mathcal{S}_{\mathcal{P}}$ in dissimilarity space. We define:

$$\Psi : \mathbb{S} \mapsto \mathcal{M} \tag{9}$$

With $\mathbb{S}$ a matrix containing dissimilarities. In [12], we have compared four different projection techniques Ψ. From the reference, the projection method giving the best performance in terms of preserving original relations is called ISOSNE, a combination of isometric mapping (ISOMAP) and stochastic neighbor embedding (SNE). ISOSNE is chosen as our Ψ.

ISOSNE contains two main steps (see [12] for more details). A graph-based distance between images is first computed using k nearest neighbors. For each image I_i, the algorithm creates links to its k nearest neighbors based on Euclidean distance. Based on the graph, distances between images are redefined. If an image I_j is not in the k nearest neighbor list of I_i, i.e. there is no direct link between them, their distance will be computed via the intermediate links. Dijkstra's algorithm is employed to compute the shortest path between I_i and I_j. The second step is to project relations obtained from the graph based distance to the 2D manipulation space. To preserve the relations, the algorithm optimizes a cost function $\mathbf{C}$ measuring the difference between the probability distribution in $\mathcal{M}$ and the distribution in the original space, denoted as $P^{\mathcal{M}}$ and $P^{\mathcal{O}}$, respectively. Based on Kullback-Leibler distance, $\mathbf{C}$ is computed as:

$$\mathbf{C} = \sum_i \sum_j P_{ij}^{\mathcal{O}} \log \frac{P_{ij}^{\mathcal{O}}}{P_{ij}^{\mathcal{M}}} \tag{10}$$

where the probability distributions are calculated as follows:

$$P_{ij}^{(.)} = \frac{exp(-S_{(.)}^2(I_i, I_j))}{\sum_{l \neq i} exp(-S_{(.)}^2(I_i, I_l))} \tag{11}$$

with $S_{(.)}(I_i, I_j)$ denoting similarity in the high dimensional original space, or a Euclidean distance in 2D $\mathcal{M}$ between image I_i and I_j.

To find the optimal placement of images in manipulation space, they are first initialized at random positions. These positions are then adjusted after each gradient descent iteration such that it reduces the cost function $\mathbf{C}$. When $\mathbf{C}$ is optimized, with positions found, distances between images $\mathcal{S}_{\mathcal{M}}$ are the ones optimally preserving the original relation in collection.

3.3 Active learning on dissimilarity space

At this point, the initial dissimilarity space is in place. The user has selected the set of prototypes $\mathcal{I}_{\mathcal{P}}$ treated as query examples to start the search process. The search task is to find other relevant images using these examples.

Different learning strategies can be employed [29]. We select active learning with support vector machines (SVM) for its capability of boosting retrieval results [26, 29, 14]. In interactive search, there is an unbalance between the size of the category searched for and the size of the collection. We therefore follow [10, 4, 13] and use one-class SVM.

The prototypes I_{P_i} are used as positive examples. From there, the one-class SVM defines a boundary $\mathcal{B}$ covering as much as possible the positive examples. Let us denote:
$\mathcal{B}_+$ the set of images inside $\mathcal{B}$ predicted relevant to the search.
$\mathcal{B}_-$ the set of images outside $\mathcal{B}$ predicted as irrelevant to the search.
$\mathcal{I}_{\mathcal{B}}$ the set of images closest to $\mathcal{B}$ according to distance function $d_{\mathcal{B}}(.)$.

For SVM $d_{\mathcal{B}}(.)$ is computed as a decision function that decides the probability of an image belonging to the relevant or irrelevant class.

In the next iterations, to improve the search, the system aims at refining $\mathcal{B}$. The refinement is such that it eliminates irrelevant images from $\mathcal{B}_+$ and adds new relevant images to $\mathcal{B}_-$. To do so, $\mathcal{I}_{\mathcal{B}}$ is chosen as set I_D^t to display as images in this set are the most uncertain and the user feedback on those yields the most information. This is known as "close-to-boundary" feedback approach [26, 14]. The user will label relevant images if they exist. Unlabelled images are treated as irrelevant and removed from the collection. Let

$$I_{\mathcal{B}_+^t} = I_D^t \cap I_+. \tag{12}$$

$$I_{\mathcal{B}_-^t} = I_D^t \backslash \mathcal{B}_+^t \tag{13}$$

When new feedback is given the SVM is recomputed on the new set of positive examples to update the boundary:

$$\mathcal{B}_+^{t+1} = (\mathcal{B}_+^t \cup I_{\mathcal{B}_+^t}) \backslash I_{\mathcal{B}_-^t} \tag{14}$$

The process is repeated until the user stops the search.

4. EXPERIMENTS

4.1 Setup

4.1.1 Overview of experiments

We now present experiments to show the performance of our proposed approach.

The first experiment considers the creation of the dissimilarity space. As described in section 3.1, to create $\mathcal{D}_{\mathcal{P}}$ we need to determine the prototype set $\mathcal{I}_{\mathcal{P}}$ and the dissimilarity between images and prototypes $\mathcal{S}(I_i, I_{P_i})$. At the beginning of the search, two spaces are available, namely the feature space $\mathcal{F}$ and its projected space, the manipulation space $\mathcal{M}$.

To create a dissimilarity space both spaces can be used. Therefore, we have two options for creating a dissimilarity space:

$$\Phi^1 : \mathcal{S}_{\mathcal{F}} \overset{\mathcal{I}_{\mathcal{P}}}{\longmapsto} \mathcal{D}_{\mathcal{P}}^r \tag{15}$$

$$\Phi^2 : \mathcal{S}_{\mathcal{M}=\Psi(\mathcal{S}_{\mathcal{F}})} \overset{\mathcal{I}_{\mathcal{P}}}{\longmapsto} \mathcal{D}_{\mathcal{P}}^2 \tag{16}$$

where $\mathcal{D}_{\mathcal{P}}^r$ is the dissimilarity space based on r- dimensional prototypes, and $\mathcal{D}_{\mathcal{P}}^2$ the dissimilarity space based on 2- dimensional prototypes. This experiment leads to the choice of the proper dissimilarity space $\mathcal{D}_{\mathcal{P}}$.

In the second experiment, the main goal is to compare the search performance on the selected dissimilarity space against the feature space. For a fair comparison, the starting points are the same for both approaches. This means that they use the same set of prototypes as initial positive examples. With the selected dissimilarity space, the proposed approach is implemented. We want to see whether the performance of active learning on dissimilarity space is better than the performance of active learning applied to feature space directly.

4.1.2 Image collections

We select two different image collections to implement the experiments. The first one is the well-known Corel collection. We select a set of 10000 images. This set contains 100 non-overlapping categories defined by Corel, the size of each category is 100 images.

The second collection is obtained from the TrecVid 2005 benchmark [23]. This set contains 43907 images, which are extracted from news video archives. We define 29 different categories such as boat, basketball, car, chair to classify the collection. Different from the Corel collection, one image in this collection can belong to different categories. The number of images in each category varies from tens to thousands. In the evaluation of the system performance, those information of the search categories are used as ground truth.

4.1.3 Features

For these two image collections, we extract the contexture feature introduced in [27]. The contexture feature is a combination of texture and color invariance of images. This feature is evaluated in the reference that it effectively learns the categorization in different image collections. The authors define 15 proto-concepts such as water, sky, snow. Note that these proto-concepts are in low level compared to higher level of the 29 categories. They learn probability values that each image contains those proto-concepts. This contexture features are computed for different parameter settings which are scales σ of Gaussian filter in computing the textures, and the size of regions where they compute the features. In particular, for the two collections in our experiment, 8 different parameter settings are used ($\sigma = 1, \sigma = 3$ and different region sizes with ratios of $\frac{1}{2}$ and $\frac{1}{6}$ of the x and y dimensions of the image). Therefore, for each image, we extract a feature vector containing of 15×8 values. We then have a feature space of 120 dimensions. To obtain the manipulation space, ISOSNE is applied to project the feature space to 2D space. The Euclidean distance is used for comparing two feature vectors.

4.1.4 Evaluation criteria

For comparison, we define a baseline which is used to see whether it is indeed worth the trouble to go for the more difficult approaches presented. The system displays a set of images and relevant images are selected if they presents in the displayed set. To compute the baseline, we calculate the number of relevant images possibly found at iteration $t+1$. We have:

$$n^{t+1} = n^t + \frac{n_+ - n^t}{n - n^t} * n_D \tag{17}$$

where n is the size of the collection, n_+ is number of images in $\mathcal{I}_+$, and n_D is number of displayed images.

For the Corel collection, we have $n = 10000$, $n_D = 100$, and the value $n_+ = 100$ for each category. From this equation, for example, at the first iteration, on average the user can find one relevant image. For the TrecVid2005 collection, the category sizes are varied. Therefore, the baseline for each category is different. We average over all categories to obtain the final baseline.

For evaluating the performance of the system, we report recall values R of the top ranked 100 images $\mathcal{I}^{100}$:

$$R = \frac{\|\mathcal{I}^{100} \bigcap \mathcal{I}_+\|}{\|\mathcal{I}_+\|}. \tag{18}$$

where $\|.\|$ denotes the size of a set. From there, we calculate the relative improvement measuring the improvement of a method over the baseline. Assume, a method X at iteration t yields a recall value R_X^t, the baseline at the same iteration returns a recall R_B^t. The relative improvement is given as:

$$\phi^t(X, B) = \frac{R_X^t - R_B^t}{R_B^t} * 100 \tag{19}$$

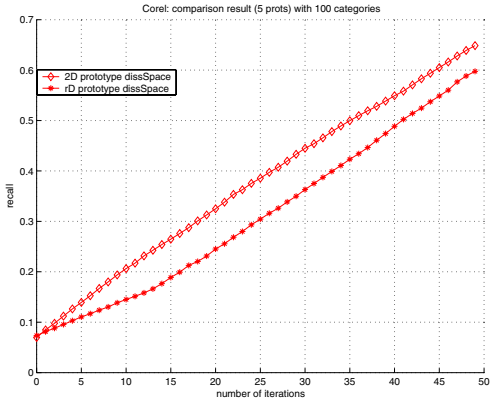
4.2 Experiment on creation of dissimilarity space

To create the projection of the 120 dimensional feature space $\mathcal{F}$ to $\mathcal{M}$, the 2D manipulation space, we apply ISOSNE. We compute two dissimilarity spaces $\mathcal{D}_{\mathcal{P}}^{120}$ and $\mathcal{D}_{\mathcal{P}}^{2}$. To do so, first the prototype set $\mathcal{I}_{\mathcal{P}}$ is selected. We test with two sets of prototypes in the creation of $\mathcal{D}_{\mathcal{P}}$ with $p = 5$ or $p = 10$.

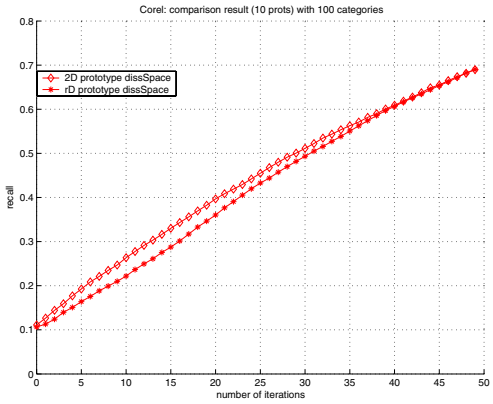
On each dissimilarity space, prototypes I_{P_i} are used as initial positive examples. One-class SVM is applied to define the boundary $\mathcal{B}$ covering these examples. The set of images close-to-boundary $\mathcal{I}_{\mathcal{B}}$ is returned for another round of feedback. We use the evaluation criteria in section 4.1.4 at each iteration, producing a ranked list based on distances to boundary. Recall values are reported with the top 100 images.

Figure 3 and 4 show the performance on $\mathcal{D}_{\mathcal{P}}^{120}$ and $\mathcal{D}_{\mathcal{P}}^{2}$ with 5 or with 10 prototypes for Corel and TrecVid collection. Based on equation 19, for both collections, the performance of learning on dissimilarity space is on average 60% relative improvement over the baseline.

Because of the projection of $\mathcal{I}$ from 120 dimensions to 2 dimensions for creating the manipulation space, relations between images cannot be kept perfectly even though the projection is optimal in keeping these relations. If the relation is not preserved on $\mathcal{M}$, performance of the $\mathcal{D}_{\mathcal{P}}^{2}$ will get worse when compared to $\mathcal{D}_{\mathcal{P}}^{120}$. However, it is interesting to observe from the results that the search performance on $\mathcal{D}_{\mathcal{P}}^{2}$ is always better than on $\mathcal{D}_{\mathcal{P}}^{120}$ whether having 5 or 10 prototypes. These results show that the ISOSNE performs very well in preserving relations between images. Moreover, because of ISOSNE extracts the structure of the collection by first computing the graph-based distance, it takes an advantage over the direct distance computation on the feature space. In other words, dissimilarity between images in the feature space is computed by directly comparing two feature vectors, whereas in the manipulation space, as a results of ISOSNE, dissimilarity is obtained by preserving a graph-based distance on the feature space. That explains why the performance of learning on $\mathcal{D}_{\mathcal{P}}^{2}$ is better than learning on $\mathcal{D}_{\mathcal{P}}^{120}$.

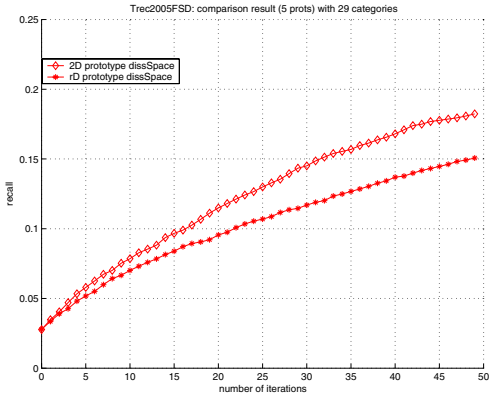


(a)

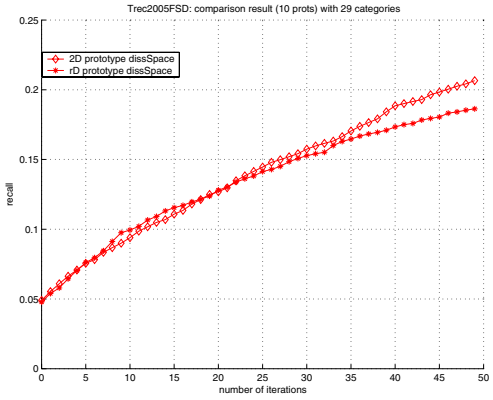


(b)

Figure 3: Comparison results on using different dissimilarity spaces with the Corel collection averaged over 100 categories. a) Dissimilarity space created by 5 prototypes. b) Dissimilarity space created by 10 prototypes.



(a)



(b)

Figure 4: Comparison results on using different dissimilarity spaces with the TrecVid collection averaged over 29 categories. a) Dissimilarity space created by 5 prototypes. b) Dissimilarity space created by 10 prototypes.

For the selection of a dissimilarity space, we prefer using $\mathcal{D}^2_{\mathcal{P}}$.

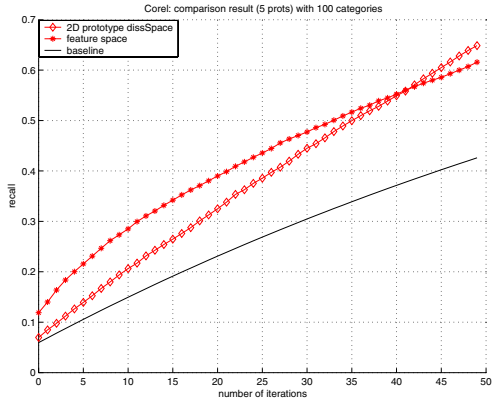
4.3 Experiment on direct manipulation of dissimilarity space vs. indirectly via feature space

In the interaction mode, there are two ways of updating dissimilarity matrix, one is via $\mathcal{D}^2_{\mathcal{P}}$, the other one is via feature space $\mathcal{F}$. We compare whether the learning on dissimilarity space gives a better result than learning via $\mathcal{F}$ as is currently done [1, 11, 9, 28].

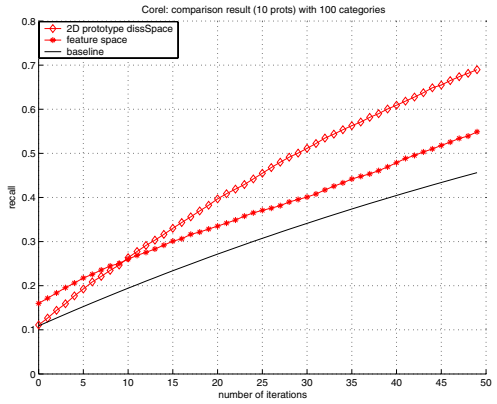
We implement the proposed scheme in section 3 with the dissimilarity space $\mathcal{D}^2_{\mathcal{P}}$. Again, one-class SVM is used with initial positive examples are the prototypes. We report recall values at the top 100 images.

Results are shown in figure 5 and 6 for the Corel and TrecVid. The figures show that with small number of prototypes, the dissimilarity space is not able to maintain the relations between images. Therefore, the improvement of learning on the dissimilarity space is smaller than learning via the feature space. With 10 prototypes, the dissimilarity space covers the image collection better. Hence, on average it gives a higher improvement.

We should note here that, with smaller number of prototypes i.e. smaller number of initial positive examples, the performance of the baseline (Eq. 17) is worse than the one with higher number of examples. Because of the relative improvement over the baseline, the values in comparing the performance with 5 prototypes get higher than with 10 prototypes. From the figures, average performance of learning via feature space get worse when having more initial examples. Because the prototype set $\mathcal{I}_{\mathcal{P}}$ is chosen such that it distributes over the collection, when p gets higher, this set better covers the collection. In the feature space, this means that the more prototypes, the broader the boundary $\mathcal{B}$. This leads to a higher number of irrelevant images fell inside $\mathcal{B}$. This is a main disadvantage of using feature space where

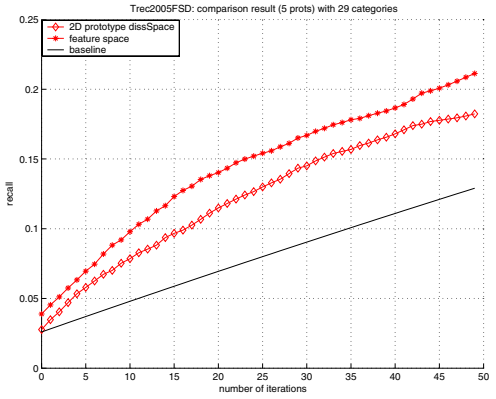


(a)

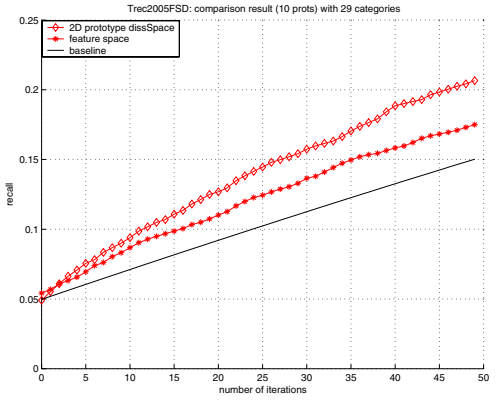


(b)

Figure 5: Comparison of direct learning on dissimilarity spaces and learning via feature space with the Corel collection averaged over 100 categories. The results are evaluated by recall. a) dissimilarity space created by 5 prototypes. b) dissimilarity space created by 10 prototypes.



(a)



(b)

Figure 6: Comparison of direct learning on dissimilarity spaces and learning via feature space with the TrecVid collection averaged over 29 categories (second row). The results are evaluated by recall. a) dissimilarity space created by 5 prototypes. b) dissimilarity space created by 10 prototypes.

selected features are not capable of capturing semantic categorization. On dissimilarity space created from $\mathcal{I}_{\mathcal{P}}$, the set of initial examples groups positive images together. Therefore, the performance of learning on dissimilarity space is improved.

From this experiment, we conclude that number of prototypes should not be too small. With 10 prototypes, it is a reasonable for creating the dissimilarity space as well as reasonable in the interactive search where few positive examples are provided. The learning performance on the dissimilarity space is better than updating dissimilarity via feature space.

5. THE INTERACTIVE SEARCH SYSTEM

In this section, we present an interactive search system based on the methodology described in the previous sections (This system is part of the MediaMill search system [24]).

To create the dissimilarity space $\mathcal{D}_{\mathcal{P}}$, a set of relevant images should be selected as prototypes $\mathcal{I}_{\mathcal{P}}$. In our system, the interface is designed supporting the user in browsing and searching through image collections. On this interface, the manipulation space $\mathcal{M}$ is the central part, where images are presented to the user as thumbnails. Their positions x_i on screen are dependent on their relations to one another. In figure 7, the main window shows the manipulation space. This screen is captured at the first round of the browsing phase. Images displayed in the figure, are representing the center points of all clusters. At this stage, positions are obtained by projecting the feature space $\mathcal{S}_{\mathcal{F}}$ to $\mathcal{M}$. The small window on the top-right corner shows the overview of the Corel collection, so called the overview window. The green dots represent the images currently displayed on the main screen. The bottom-right corner window shows the full size of a current image.

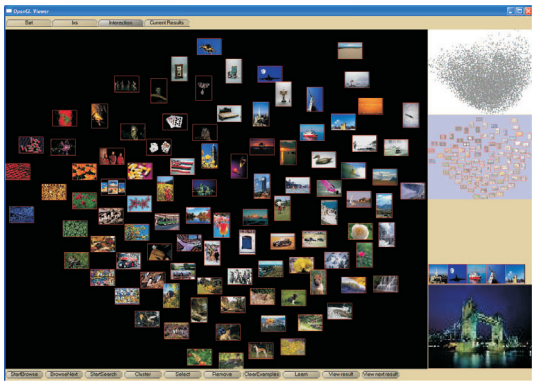


Figure 7: A screen shot of the system when browsing through the Corel collection.

During the interaction, relevant images selected by the user will be kept in the third window on the right side so as to recall the user of what has been selected. For selection of positive examples, the 2D similarity based visualization shows the advantage over the grid based display. For example, instead of selecting one image at a time, in 2D similarity-based visualization similar images stay close together, therefore the user can select a group of images at a time.

When a certain number of positive examples have found, the system then creates $\mathcal{D}_\mathcal{P}$. From this point onwards, the manipulation space $\mathcal{M}$ will be a projection of the dissimilarity space $\mathcal{D}_\mathcal{P}$. In the first round of the learning phase, the SVM learns on $\mathcal{D}_\mathcal{P}$ with $\mathcal{I}_\mathcal{P}$. After the learning process is finished, a set of images closest to boundary $\mathcal{I}_\mathcal{B}$ is returned and displayed. For a better understanding of what images are showed to the user, prototypes are also displayed on the main window. To distinguish from the other images, the prototypes are given in different color. Positions of displayed images with the prototypes gives the user an idea of what happens. This is illustrated in figure 8a. In the overview window, the green dots now show the boundary formed by the current set of images (see figure 8b).

To continue the search, the user selects more positive images. The ones which are not selected are removed from the collection. The learning process is repeated until there are no positive examples left or the user is satisfied with the result. A "View results" button gives the user a set of images which is the final learning results (figure 9).

6. CONCLUSION

In this paper, we have proposed a new approach in interactively learning dissimilarity. Different from existing techniques [1, 11, 9, 28], we learn on the dissimilarity space. This means that instead of collecting a large set of features or choosing well-defined features, only the relations between images are used. By doing that, we avoid the computational problem with large set of features and the difficulty in selecting effective features in interactive category search. Representing images in their relations to others can extract the perceptual meaning of those images, which is difficult to obtain using feature representations.

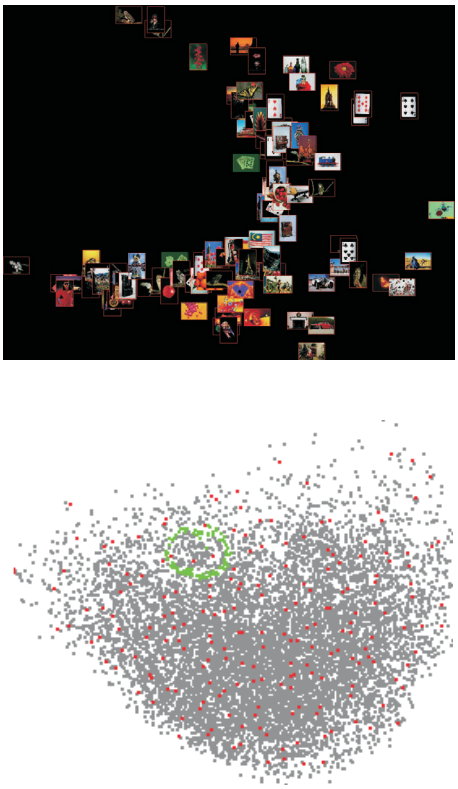


Figure 8: (a) A screen shot of images closest to the border (see (b) the green dots draw the boundary). For making distinction, on the main view those prototypes are painted in green color, and not be able to reselect again. Their appearances just to give the user a clear understanding of what is happening. (b) An example of boundary defined by image closest to the boundary (represented by points in green color). The red points represent for images which have been displayed or removed from the collection.

We have demonstrated by experiments that learning on this dissimilarity space $\mathcal{D}_\mathcal{P}$ in general gives a better performance than the learning on feature space $\mathcal{F}$, under the provision a reasonable number of initial prototypes $\mathcal{I}_\mathcal{P}$ is being used.

In conclusion, interactive learning on dissimilarity space $\mathcal{D}_\mathcal{P}$ rather than via feature space $\mathcal{F}$ is very promising. The simplicity in creating dissimilarity space and its performance is certainly of interest for further more detail research.

Acknowledgments

The work is within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) sponsored by IOP MMI (Man-Machine Interaction). The first author would like to thank J. van Gemert for providing the contexture features.

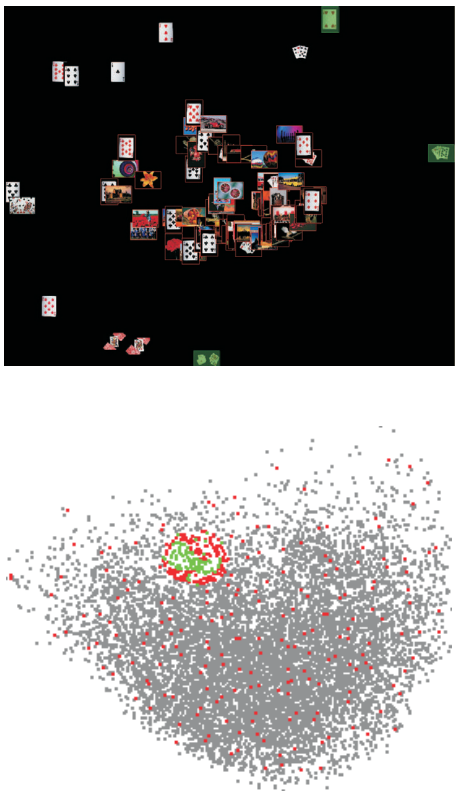


Figure 9: (a) A screen shot of images on the main view when pressing "View results". (b) An example of images with result contains images having the largest distances to the boundary.

7. REFERENCES

[1] B. Bhanu, J. Peng, and S. Qing. Learning feature relevance and similarity metrics in image databases. In *Proceedings of the IEEE Workshop on Content - Based Access of Image and Video Libraries*, page 14, 1998.

[2] E. Bruno, N. Loccoz, and S. Maillet. Learning user queries in multimodal dissimilarity spaces. In *Proceedings of the 3rd International Workshop on Adaptive Multimedia Retrieval*, 2005.

[3] A. Carkacioglu and F. Vural. Learning similarity space. In *International Conference on Image Processing*, 2002.

[4] Y. Chen, X. Zhou, and T. Huang. One-class SVM for learning in image retrieval. In *International Conference on Image Processing*, volume 1, pages 34–37, 2001.

[5] R. Duin, D. Ridder, and D. Tax. Experiments with a featureless approach to pattern recognition. *Pattern Recognition Letters*, 18:1159–1166, 1997.

[6] I. El-Naqa, Y. Yang, N. Galatsanos, R. Nishikawa, and M. Wernick. A similarity learning approach to content based image retrieval: application to digital mammography. *IEEE Transactions on Medical Imaging*, 23(10):1233–1244, 2004.

[7] G. Guo, A. Jain, W. Ma, and H. Zhang. Learning similarity measure for natural image retrieval with relevance feedback. *IEEE Transactions on Neural Networks*, 13(4):811–820, 2002.

[8] X. He, O. King, W. Ma, M. Li, and H. Zhang. Learning a semantic space from user's relevance feedback for image retrieval. *IEEE transactions on Circuits and Systems for Video Technology*, 13(1):39–48, 2003.

[9] B. Li and E. Chang. Discovery of a perceptual distance function for measuring image similarity. *ACM Multimedia Journal Special Issue on Content-Based Image Retrieval*, 8(6):512–522, 2003.

[10] L. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2004.

[11] B. Moghaddam, Q. Tian, N. Lesh, C. Shen, and T. Huang. Visualization and user-modeling for browsing personal photo libraries. *International Journal of Computer Vision*, 56(1/2):109–130, 2004.

[12] G. Nguyen and M. Worring. Similarity based visualization of image collections. In *In proceedings of 7th International Workshop on Audio-Visual Content and Information Visualization in Digital Libraries*, 2005.

[13] G. P. Nguyen and M. Worring. Scenario optimization for interactive category search. In *Proceeding of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, 2005.

[14] H. Nguyen and A. Smeulders. Active learning using pre-clustering. In *In Proceedings of the 21th International Conference on Machine Learning*, 2004.

[15] E. Pekalska and R. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23:943–956, 2002.

[16] E. Pekalska, R. Duin, and P. Paclik. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.

[17] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? *ACM Conference on Human Factors in Computing Systems*, pages 190 – 197, 2001.

[18] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[19] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[20] S. Santini, A. Gupta, and R. Jain. Emergent semantics through interaction in image databases. *IEEE Transactions on Knowledge and Data Engineering archive*, 13(3):1041–4347, 2001.

[21] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern analysis and machine intelligence*, 21(9):871–883, 1999.

[22] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of early years. *IEEE Transaction on Pattern Analysis and Machine Intelligence.*, 22(12):1349–1380, 2000.

[23] C. Snoek, J. van Gemert, J. Geusebroek, B. Huurnink, D. Koelma, G. Nguyen, O. de Rooij, F. Seinstra, A. Smeulders, C. Veenman, and M. Worring. The mediamill trecvid 2005 semantic video search engine. In *Proceedings of the 3th TRECVID Workshop*, 2005.

[24] C. Snoek, M. Worring, J. van Gemert, J. Geusebroek, D. Koelma, G. Nguyen, O. de Rooij, , and F. Seinstra. Mediamill: Exploring news video archives based on learned semantics. In *Proceedings of ACM Multimedia, Best Technical Demonstration Award*, 2005.

[25] D. Squire. Learning a similarity-based distance measure for image database organization from human partitionings of an image set. In *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, page 88, 1998.

[26] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *ACM Internaltional Conference on Multimedia*, volume 9, pages 107–118, 2001.

[27] J. van Gemert, J. Geusebroek, C. Veenman, C. Snoek, and A. Smeulders. Robust scene categorization by learning image statistics in context. In *CVPR Workshop on Semantic Learning Applications in Multimedia (SLAM)*, 2006.

[28] H. Ye and G. Xu. Similarity measure learning for image retrieval using feature subspace analysis. In *Proceedings of the Fifth International Conference on Computational Intelligence and Multimedia Applications.*, pages 131–136, 2003.

[29] X. Zhou and T. Huang. Relevance feedback in image retrieval: A comprehensive overview. *Multimedia systems*, 8:536–544, 2003.