SVM_{Active} — Support Vector Machine Active Learning for Image Retrieval

Edward Chang
Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106
echang@ece.ucsb.edu

Simon Tong
Department of Computer Science
Stanford University
Sanford, CA 94305
simon.tong@cs.stanford.edu

Abstract

Relevance feedback is often a critical component when designing image databases. With these databases it is difficult to specify queries directly and explicitly. Relevance feedback interactively determinines a user's desired output or *query concept* by asking the user whether certain proposed images are relevant or not. For a relevance feedback algorithm to be effective, it must grasp a user's query concept accurately and quickly, while also asking the user to label only a small number of images. We propose the use of a *support vector machine active learning* (SVM_{Active}) algorithm for conducting effective relevance feedback for image retrieval. To support efficient query-concept learning and image retrieval, we also present our multi-resolution image-characterization and high-dimensional indexing methods. We further show that SVM_{Active} can be effectively seeded by MEGA, another active learning algorighm that we developed, or by keyword searches. Experimental results show that our algorithm achieves significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.

Keywords: active learning, image retrieval, query concept, relevance feedback, support vector machines.

1 Introduction

One key design task, when constructing image databases, is the creation of an effective relevance feedback component. While it is sometimes possible to arrange images within an image database by creating a hierarchy, or by hand-labeling each image with descriptive words, these methods are time-consuming, costly, and subjective. Alternatively, requiring an end-user to specify an image query in terms of low level features (such as color and spatial relationships) is challenging to the end-user, because an image query is hard to articulate, and articulation can vary from one user to another.

Thus, we need a way for a user to implicitly inform a database of his or her desired output or *query concept*. To address this requirement, relevance feedback can be used as a query refinement scheme to derive or learn a user's query concept. To solicit feedback, the refinement scheme displays a few image instances and the user labels each image as "relevant" or "not relevant." Based on the responses, another set of images from the database is presented to the user for labeling. After a few such querying rounds, the refinement scheme returns a number of items from the database that seem to fit the needs of the user.

The construction of such a query refinement scheme (hereafter called a *query concept learner* or *learner*) can be regarded as a machine learning task. In particular, it can be seen as a case of *pool-based active learning* [39, 43].

In pool-based active learning the learner has access to a pool of unlabeled data and can request the user's label for a certain number of instances in the pool. In the image retrieval domain, the unlabeled pool would be the entire database of images. An instance would be an image, and the two possible labelings for each image would be "relevant" or "not relevant." The goal for the learner is to learn the user's *query concept* — in other words, to label each image within the database in such a manner that the learner's labeling and the user's labeling will agree.

The main issue with active learning is finding a method for choosing informative images within the pool to ask the user to label. We call such a request for the label of an image a *pool-query*. Most machine learning algorithms are *passive* in the sense that they are generally applied using a randomly selected training set. The key idea with *active* learning is that it should choose its next pool-query based upon the past answers to previous pool-queries.

In general, and for the image retrieval task in particular, such a learner must meet two critical design goals. First, the learner must learn target concepts accurately. Second, the learner must grasp a concept quickly, with only a small number of labeled instances, since most users do not wait around to provide a great deal of feedback. In this study, we propose using a *support vector machine active learner* (SVM_{Active}) to achieve these goals. SVM_{Active} combines active learning with support vector machines (SVMs). SVMs [64, 6] have met with significant success in numerous real-world learning tasks. Like most machine learning algorithms, they use a randomly selected training set, which is not very useful in the relevance feedback setting. Recently, general purpose methods for active learning with SVMs have been independently developed by a number of researchers [7, 55, 63]. We shall use the work and theoretical motivation of [63] on active learning with SVMs to extend the use of support vector machines to the task of relevance feedback for image databases.

Intuitively, SVM_{Active} works by combining the following three ideas:

1. SVM_{Active} regards the task of learning a target concept as one of learning an SVM binary classifier. An SVM captures the query concept by separating the relevant images from irrelevant ones with a hyperplane in a projected space, usually a very high-dimensional one. The projected points on one side of the hyperplane are considered relevant to the query concept and the rest irrelevant.

2. SVM_{Active} learns the classifier quickly via active learning. The active part of SVM_{Active} selects the most informative instances with which to train the SVM classifier. This step ensures fast convergence to the query concept in a small number of feedback rounds.

3. Once the classifier is trained, SVM_{Active} returns the top-k most relevant images. These are the k images farthest from the hyperplane on the query concept side.

SVM_{Active} needs at least one positive and one negative example to start its learning process. We propose two seeding methods: by MEGA [11], and by keywords (Section 7). To make both concept-learning and image retrieval efficient, we employ a multi-resolution image-feature extractor (Section 6.1), and a high-dimensional indexer (Section 6.2). Through examples and empirical study, we show that combining SVM_{Active} with these other components produces a search engine particularly well suited to the query refinement task in image retrieval, significantly outperforming traditional systems.

The rest of this paper is organized into seven sections. Section 2 surveys related work. Section 3 introduces SVMs. Section 4 then introduces the notion of a *version space* which in Section 5 provides theoretical motivation for a method of performing active learning with SVMs. Section 6 depicts our multi-resolution image characterization and high-dimensional indexer. Section 7 presents two options of finding relevant images before switching to SVM_{Active}.

In Section 8, we report experimental results showing that our SVM active learner significantly outperforms traditional methods. Finally, we offer our conclusions in Section 9.

2 Related Work

Machine learning and relevance feedback techniques have been proposed to learn and to refine query concepts. The problem is that most traditional techniques require a large number of training instances [3, 37, 45, 73, 74], and they require seeding a query with "good" examples [36, 51, 71, 49]. Unfortunately, in many practical scenarios, a learning algorithm must work with a scarcity of training data and a limited amount of training time.

2.1 Machine Learning

Of late, ensemble techniques such as *bagging* [4], *arcing* [5], and *boosting* [28, 54, 70] have been proposed to improve learning accuracy for decision trees and neural networks. These ensemble schemes enjoy success in improving classification accuracy through bias or variance reduction, but they do not help reduce the number of samples and time required to learn a query concept. In fact, most ensemble schemes actually increase learning time because they introduce learning redundancy in order to improve prediction accuracy [20, 28, 34, 46].

To reduce the number of required samples, researchers have conducted several studies of active learning [14, 8, 33, 62, 72] for classification. Active learning can be modeled formally as follows: Given a dataset S consisting of an unlabeled subset U and a labeled subset X, an active learner L has two components: f and s. The f component is a classifier that is trained on the current set of labeled data X. The second component s is the sampling function that, given a current labeled set X, decides which subset u in U to select to query the user. The active learner returns a new f after each round of relevance feedback. The sampling techniques employed by the active learner determine the selection of the next batch of unlabeled instances to be labeled by the user.

The *query by committee* (QBC) algorithm [23, 56] is a representative active learning scheme. QBC uses a distribution over all possible classifiers and attempts greedily to reduce the entropy of this distribution. This general purpose algorithm has been applied in a number of domains (although, to our knowledge, not to the image retrieval domain) using classifiers (such as Naive Bayes classifiers [19, 43]) for which specifying and sampling classifiers from a distribution is natural. Probabilistic models such as the Naive Bayes classifier provide interpretable results and principled ways to incorporate prior knowledge. However, they typically do not perform as well as discriminative methods such as SVMs [35, 21], especially when the amount of training data is scarce. (See our SVM-based approach in Section 3.) For image retrieval where a query concept is typical non-linear[1], our MEGA and SVM_{Active} with kernel mapping provide more flexible and accurate concept modeling.

Specifically for image retrieval, the PicHunter system [17, 16, 15, 18] uses Bayesian prediction to infer the goal image, based upon users' input. Mathematically, the goal of PicHunter is to find a single goal point in the feature space (e.g., a particular flower image), whereas our goal is to hunt down all points that match a query concept (e.g., the entire flower category, which consists of flowers of different colors, shapes, and textures, and against different backgrounds). Note that the points matching a target concept can be scattered all over the feature space. To find these points quickly with few hints, our learning algorithms must deal with many daunting challenges.

[1] A query such as "animals", "women", and "european architecture" does not reside contiguously in the space formed by the image features.

2.2 Relevance Feedback

The study of [52] puts relevance feedback techniques proposed by the Information Retrieval (IR) into three categories: *query reweighting*, *query point movement* and *query expansion*.

- *Query reweighting* and *query point movement* [32, 48, 47, 51, 53]. Both query reweighting and query point movement use nearest-neighbor sampling: They return top ranked objects to be examined by the user and then refine the query based on the user's feedback. If the initial query example is good and the query concept is convex in the feature space [32, 71], this nearest-neighbor sampling approach works fine. Unfortunately, most users do not have a good example to start a query, and most image-query concepts are non-convex.
- *Query expansion* [52, 71]. The *query expansion* approach can be regarded as a multiple-instances sampling approach. The samples of a subsequent round are selected from the neighborhood (not necessarily the nearest ones) of the positive-labeled instances of the previous round. The study of [52] shows that query expansion achieves only a slim margin of improvement (about 10% in precision/recall) over query point movement.

Almost all traditional relevance feedback methods require seeding the methods with "good" positive examples [22, 29, 31, 42, 59, 60, 67], and most methods do not use negative-labeled instances effectively. For instance, sunset images must be supplied as examples in order to search for sunset pictures. However, finding good examples should be the job of a search engine itself. Our methods ($\mathsf{SVM}_{\mathit{Active}}$ and MEGA) effectively use negative-labeled instances to induce more negative instances, and thereby improve the probability of finding positive instances. At the same time, our active-learning approach selects the most informative unlabeled instances to query the user to gather maximum amount of information to disambiguous the user's query concept. Because of the effective use of negative and unlabeled instances, our method can learn a query concept much faster and more accurately than the traditional relevance-feedback methods.

3 Support Vector Machines

Support vector machines are a core machine learning technology. They have strong theoretical foundations and excellent empirical successes. They have been applied to tasks such as handwritten digit recognition [65], object recognition [50], and text classification [35].

We shall consider SVMs in the binary classification setting. We are given training data $\{\mathbf{x_1} \ldots \mathbf{x_n}\}$ that are vectors in some space $\mathsf{X} \subseteq \mathbb{R}^d$. We are also given their labels $\{y_1 \ldots y_n\}$ where $y_i \in \{-1, 1\}$. In their simplest form, SVMs are hyperplanes that separate the training data by a maximal margin (see Fig. 1). All vectors lying on one side of the hyperplane are labeled as -1, and all vectors lying on the other side are labeled as 1. The training instances that lie closest to the hyperplane are called *support vectors*. More generally, SVMs allow us to project the original training data in space X to a higher dimensional feature space F via a Mercer kernel operator K. In other words, we consider the set of classifiers of the form: $f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_\mathbf{i} \mathbf{K}(\mathbf{x_i}, \mathbf{x})$. When $f(\mathbf{x}) \geq \mathbf{0}$ we classify $\mathbf{x}$ as $+1$, otherwise we classify $\mathbf{x}$ as -1.

When K satisfies Mercer's condition [6] we can write: $K(\mathbf{u}, \mathbf{v}) = \mathbf{\Phi}(\mathbf{u}) \cdot \mathbf{\Phi}(\mathbf{v})$ where $\Phi : \mathsf{X} \rightarrow \mathsf{F}$ and "$\cdot$" denotes an inner product. We can then rewrite f as:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}), \text{ where } \mathbf{w} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_\mathbf{i} \mathbf{\Phi}(\mathbf{x_i}). \tag{1}$$

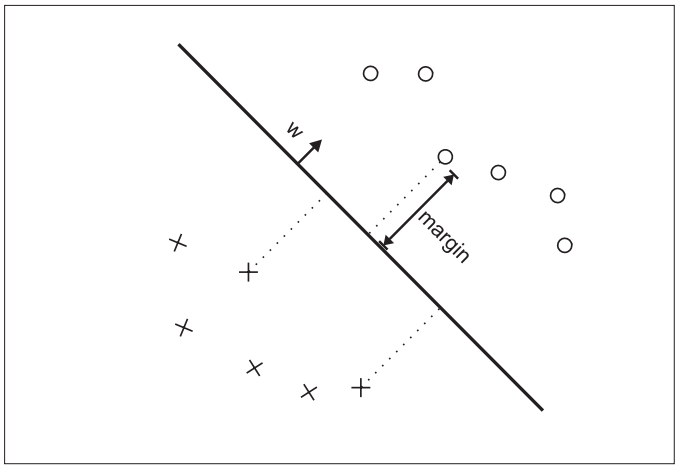


Figure 1: A simple linear Support Vector Machine

Thus, by using K we are implicitly projecting the training data into a different (often higher dimensional) feature space F. The SVM then computes the α_is that correspond to the maximal margin hyperplane in F. By choosing different kernel functions we can implicitly project the training data from X into space F. (Hyperplanes in F correspond to more complex decision boundaries in the original space X.)

Two commonly used kernels are the polynomial kernel $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + \mathbf{1})^{\mathrm{P}}$, which induces polynomial boundaries of degree p in the original space X, and the radial basis function kernel $K(\mathbf{u}, \mathbf{v}) = (\mathbf{e}^{-\gamma(\mathbf{u}-\mathbf{v})\cdot(\mathbf{u}-\mathbf{v})})$, which induces boundaries by placing weighted Gaussians upon key training instances. In the remainder of this paper we will assume that the modulus of the training data feature vectors are constant, i.e., for all training instances $\mathbf{x_i}$, $\|\Phi(\mathbf{x_i})\| = \lambda$ for some fixed λ. The quantity $\|\Phi(\mathbf{x_i})\|$ is always constant for radial basis function kernels, and so the assumption has no effect for this kernel. For $\|\Phi(\mathbf{x_i})\|$ to be constant with the polynomial kernels we require that $\|\mathbf{x_i}\|$ be constant. It is possible to relax this constraint on $\Phi(\mathbf{x_i})$. We shall discuss this option at the end of Section 5.

4 Version Space

Given a set of labeled training data and a Mercer kernel K, there is a set of hyperplanes that separate the data in the induced feature space F. We call this set of consistent hyperplanes or *hypotheses* the *version space* [44]. In other words, hypothesis f is in version space if for every training instance $\mathbf{x_i}$ with label y_i we have that $f(\mathbf{x_i}) > \mathbf{0}$ if $y_i = 1$ and $f(\mathbf{x_i}) < \mathbf{0}$ if $y_i = -1$. More formally:

Definition 4.1 *Our set of possible hypotheses is given as:*

$$\mathsf{H} = \left\{ \mathsf{f} \mid \mathsf{f}(\mathbf{x}) = \frac{\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x})}{\|\mathbf{w}\|} \textit{ where } \mathbf{w} \in \mathsf{W} \right\},$$

where our parameter space W *is simply equal to* F. *The* Version space, V *is then defined as:*

$$\mathsf{V} = \{\mathsf{f} \in \mathsf{H} \mid \forall \mathsf{i} \in \{1 \ldots \mathsf{n}\} \;\; \mathsf{y_i f}(\mathbf{x_i}) > \mathbf{0}\}.$$

(a) (b) (c)

Figure 2: (a) Version space duality. The surface of the hypersphere represents unit weight vectors. Each of the two hyperplanes corresponds to a labeled training instance. Each hyperplane restricts the area on the hypersphere in which consistent hypotheses can lie. Here version space is the surface segment of the hypersphere closest to the camera. (b) An SVM classifier in version space. The dark embedded sphere is the largest radius sphere whose center lies in version space and whose surface does not intersect with the hyperplanes. The center of the embedded sphere corresponds to the SVM, its radius is the margin of the SVM in F, and the training points corresponding to the hyperplanes that it touches are the support vectors. (c) Simple Margin Method.

Notice that since H *is a set of hyperplanes, there is a bijection (an exact correspondence) between unit vectors* $\mathbf{w}$ *and hypotheses* f *in* H. *Thus we will redefine* V *as:*

$$\mathsf{V} = \{\mathbf{w} \in \mathsf{W} \mid \|\mathbf{w}\| = \mathbf{1},\ \mathbf{y_i}(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) > \mathbf{0}, \mathbf{i} = \mathbf{1} \ldots \mathbf{n}\}.$$

Note that a version space exists only if the *training* data are linearly separable in the feature space. Thus, we require linear separability of the training data in the feature space. This restriction is much less harsh than it might at first seem. First, the feature space often has a very high dimension and so in many cases it results in the data set being linearly separable. Second, as noted by [57], it is possible to modify any kernel so that the data in the newly induced feature space is linearly separable. This is done by redefining all training instances $\mathbf{x_i}$: $K(\mathbf{x_i}, \mathbf{x_i}) \leftarrow \mathbf{K}(\mathbf{x_i}, \mathbf{x_i}) + \nu$ where ν is a positive regularization constant. The effect of this modification is to permit linear non-separability of the training data in the original feature space.

There exists a duality between the feature space F and the parameter space W [65, 30] which we shall take advantage of in the next section: points in F correspond to hyperplanes in W and *vice versa*.

Clearly, by definition, points in W correspond to hyperplanes in F. The intuition behind the converse is that observing a training instance $\mathbf{x_i}$ in feature space restricts the set of separating hyperplanes to ones that classify $\mathbf{x_i}$ correctly. In fact, we can show that the set of allowable points $\mathbf{w}$ in W is restricted to lie on one side of a hyperplane in W. More formally, to show that points in F correspond to hyperplanes in W, suppose we are given a new training instance $\mathbf{x_i}$ with label y_i. Then any separating hyperplane must satisfy $y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) > \mathbf{0}$. Now, instead of viewing $\mathbf{w}$ as the normal vector of a hyperplane in F, think of $y_i\Phi(\mathbf{x_i})$ as being the normal vector of a hyperplane in W. Thus $y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) = \mathbf{w} \cdot \mathbf{y_i}\mathbf{\Phi}(\mathbf{x_i}) > \mathbf{0}$ defines a half-space in W. Furthermore $\mathbf{w} \cdot \mathbf{y_i}\mathbf{\Phi}(\mathbf{x_i}) = \mathbf{0}$ defines a hyperplane in W that acts as one of the boundaries to version space V. Notice that version space is a connected region on the surface of a hypersphere in parameter space. See Fig.2(a) for an example.

SVMs find the hyperplane that maximizes the margin in feature space F. One way to pose this is as follows:

$$\begin{array}{rc} \text{maximize}_{\mathbf{w} \in \mathsf{F}} & \min_i\{y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}))\} \\ \text{subject to:} & \|\mathbf{w}\| = \mathbf{1} \end{array}$$

$$y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) > 0 \;\; \mathbf{i = 1 \ldots n}.$$

By having the conditions $\|\mathbf{w}\| = \mathbf{1}$ and $y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) > \mathbf{0}$ we cause the solution to lie in version space. Now, we can view the above problem as finding the point $\mathbf{w}$ in version space that maximizes the distance $\min_i\{\mathbf{w} \cdot \mathbf{y_i \Phi}(\mathbf{x_i})\}$. From the duality between feature and parameter space, and since $\|\Phi(\mathbf{x_i})\| = \mathbf{1}$, then each $y_i \Phi(\mathbf{x_i})$ is a unit normal vector of a hyperplane in parameter space and each of these hyperplanes delimits the version space. Thus we want to find the point in version space that maximizes the minimum distance to any of the delineating hyperplanes. That is, SVMs find the center of the largest radius hypersphere whose center can be placed in version space and whose surface does not intersect with the hyperplanes corresponding to the labeled instances, as in Figure 2(b). It can be easily shown that the hyperplanes touched by the maximal radius hypersphere correspond to the support vectors and that the radius of the hypersphere is the margin of the SVM.

5 Active Learning

In pool-based active learning we have a pool of unlabeled instances. It is assumed that the instances $\mathbf{x}$ are independently and identically distributed according to some underlying distribution $F(\mathbf{x})$, and the labels are distributed according to some conditional distribution $P(y \mid \mathbf{x})$.

Given an unlabeled pool U, an *active learner* ℓ has three components: (f, q, X). The first component is a classifier, $f : \mathsf{X} \to \{-1, 1\}$, trained on the current set of labeled data X (and possibly unlabeled instances in U too). The second component $q(X)$ is the querying function that, given a current labeled set X, decides which instance in U to query next. The active learner can return a classifier f after each pool-query (*online learning*) or after some fixed number of pool-queries.

The main difference between an active learner and a regular passive learner is the querying component q. This brings us to the issue of how to choose the next unlabeled instance in the pool to query. We use an approach that queries such instances in order to reduce the size of the version space as much as possible. We need one more definition before we can proceed:

Definition 5.1 *Area*(V) *is the surface area that the version space* V *occupies on the hypersphere* $\|\mathbf{w}\| = \mathbf{1}$.

We wish to reduce the version space as fast as possible. Intuitively, one good way of doing this is to choose a pool-query that halves the version space. More formally, we can use the following lemma to motivate which instances to use as our pool-query:

Lemma 5.2 *(Tong & Koller, 2000) Suppose we have an input space* X*, finite dimensional feature space* F *(induced via a kernel* K*), and parameter space* W*. Suppose active learner* ℓ^* *always queries instances whose corresponding hyperplanes in parameter space* W *halves the area of the current version space. Let* ℓ *be any other active learner. Denote the version spaces of* ℓ^* *and* ℓ *after* i *pool-queries as* V_i^* *and* V_i *respectively. Let* P *denote the set of all conditional distributions of* y *given* $\mathbf{x}$*. Then,*

$$\forall i \in \mathbb{N}^+ \;\; \sup_{P \in \mathsf{P}} E_P[\mathit{Area}(\mathsf{V}_i^*)] \leq \sup_{P \in \mathsf{P}} \mathsf{E}_P[\mathit{Area}(\mathsf{V}_i)],$$

with strict inequality whenever there exists a pool-query $j \in \{1 \ldots i\}$ *by* ℓ *that does not halve version space* V_{j-1}.

This lemma says that, for any given number of pool-queries, ℓ^* minimizes the maximum expected size of the version space, where the maximum is taken over all conditional distributions of y given $\mathbf{x}$.

Now, suppose $\mathbf{w}^* \in \mathsf{W}$ is the unit parameter vector corresponding to the SVM that we would have obtained had we known the actual labels of *all* of the data in the pool. We know that $\mathbf{w}^*$ must lie in each of the version spaces $\mathsf{V_1} \supset \mathsf{V_2} \supset \mathsf{V_3} \ldots$, where $\mathsf{V_i}$ denotes the version space after i pool-queries. Thus, by shrinking the size of the version space as much as possible with each pool-query we are reducing as fast as possible the space in which $\mathbf{w}^*$ can lie. Hence, the SVM that we learn from our limited number of pool-queries will lie close to $\mathbf{w}^*$.

This discussion provides motivation for an approach in which we query instances that split the current version space into two equal parts insofar as possible. Given an unlabeled instance $\mathbf{x}$ from the pool, it is not practical to explicitly compute the sizes of the new version spaces V^- and V^+ (i.e., the version spaces obtained when $\mathbf{x}$ is labeled as -1 and $+1$ respectively). There is a way of approximating this procedure as noted by [63]:

Simple Method. Recall from Section 4 that, given data $\{\mathbf{x_1} \ldots \mathbf{x_i}\}$ and labels $\{y_1 \ldots y_i\}$, the SVM unit vector $\mathbf{w_i}$ obtained from this data is the center of the largest hypersphere that can fit inside the current version space $\mathsf{V_i}$. The position of $\mathbf{w_i}$ in the version space $\mathsf{V_i}$ clearly depends on the shape of the region $\mathsf{V_i}$; however, it is often approximately in the center of the version space. Now, we can test each of the unlabeled instances $\mathbf{x}$ in the pool to see how close their corresponding hyperplanes in W come to the centrally placed $\mathbf{w_i}$. The closer a hyperplane in W is to the point $\mathbf{w_i}$, the more centrally it is placed in version space, and the more it bisects version space. Thus we can pick the unlabeled instance in the pool whose hyperplane in W comes closest to the vector $\mathbf{w_i}$. For each unlabeled instance $\mathbf{x}$, the shortest distance between its hyperplane in W and the vector $\mathbf{w_i}$ is simply the distance between the feature vector $\Phi(\mathbf{x})$ and the hyperplane $\mathbf{w_i}$ in F — which is easily computed by $|\mathbf{w_i} \cdot \mathbf{\Phi}(\mathbf{x})|$. This results in the natural Simple rule:

- Learn an SVM on the existing labeled data and choose as the next instance to query the pool instance that comes closest to the hyperplane in F.

Figure 2(c) presents an illustration. In the stylized picture we have flattened out the surface of the unit weight vector hypersphere that appears in Figure 2(a). The white area is version space $\mathsf{V_i}$ which is bounded by solid lines corresponding to labeled instances. The five dotted lines represent unlabeled instances in the pool. The circle represents the largest radius hypersphere that can fit in the version space. Note that the edges of the circle do not touch the solid lines — just as the dark sphere in Figure2(b) does not meet the hyperplanes on the surface of the larger hypersphere (they meet somewhere under the surface). The instance $\mathbf{b}$ is closest to the SVM $\mathbf{w_i}$ and so we will choose to query $\mathbf{b}$.

As noted by [63] there exist more sophisticated approximations that we can perform. However, these methods are significantly more computationally intensive. For the task of relevance feedback, a fast response time for determining the next image to present to the user is so critically important that these other approximations are not practical.

Our SVM_{Active} image retrieval system uses radial basis function kernels. As noted in Section 3, radial basis function kernels have the property that $\|\Phi(\mathbf{x_i})\| = \lambda$. The Simple querying method can still be used with other kernels when the training data feature vectors do not have a constant modulus, but the motivating explanation no longer holds since the SVM can no longer be viewed as the center of the largest allowable sphere. However, alternative motivations have recently been proposed by Campbell, Cristianini and Smola (2000) that do not require a constraint on the modulus.

For the image retrieval domain, we also have a need for performing multiple pool-queries at the same time. It is not practical to present one image at a time for the user to label, because he or she is likely to lose patience after a few rounds. To prevent this from happening we present the user with multiple images (say, twenty) at each round of pool-querying. Thus, for each round, the active learner has to choose not just one image to be labeled but twenty. Theoretically it would be possible to consider the size of the resulting version spaces for each possible labeling of each possible set of twenty pool-queries, but clearly this would be impractical. Thus instead, for matters of computational efficiency, our SVM_{Active} system takes the simple approach of choosing the pool-queries to be the twenty images closest to its separating hyperplane. We discuss a couple of extensions in Section 9.

It has been noted [63] that the Simple querying algorithm used by SVM_{Active} can be unstable during the first round of querying. To address this issue, SVM_{Active} always randomly chooses twenty images for the first relevance feedback round. Then it uses the Simple active querying method on the second and subsequent rounds. We discuss two other seeding methods, seeding SVM_{Active} by MEGA and by keywords, in Section 7.

SVM_{Active} Algorithm Summary

To summarize, our SVM_{Active} system performs the following for each round of relevance feedback:

- Learn an SVM on the current labeled data
- If this is the first feedback round, ask the user to label twenty randomly selected images. Otherwise, ask the user to label the twenty pool images closest to the SVM boundary.

After the relevance feedback rounds have been performed, SVM_{Active} retrieves the top-k most relevant images:

- Learn a final SVM on the labeled data.
- The final SVM boundary separates "relevant" images from irrelevant ones. Display the k "relevant" images that are farthest from the SVM boundary.

6 Indexing Image Features

We describe how our system characterizes images and how we index image features. Multi-resolution image characterization makes learning effective and efficient. Our high-dimensional indexing scheme makes image retrieval efficient.

6.1 Image Characterization

We believe that image characterization should follow human perception [27]. In particular, our perception works in a multi-resolution fashion. For some visual tasks, human eyes may select coarse filters to obtain coarse image features; for others, they select finer features. Similarly, for some image applications (e.g., for detecting image replicas), employing coarse features is sufficient; for other applications (e.g., for classifying images), employing finer features may be essential. An image search engine thus must have the flexibility to model subjective perceptions and to satisfy a variety of search tasks.

Filter Name	Resolution	Representation
Masks	Coarse	Appearance of culture colors
Spread	Coarse	Spatial concentration of a color
Elongation	Coarse	Shape of a color
Histograms	Medium	Distribution of colors
Average	Medium	Similarity comparison within the same culture color
Variance	Fine	Similarity comparison within the same culture color

Table 1: Multi-resolution Color Features.

Our image retrieval system employs a multi-resolution image representation scheme [13]. In this scheme, we characterize images by two main features: color and texture. We consider shape as an attribute of these main features.

6.1.1 Color

Although the wavelength of visible light ranges from 400 nanometers to 700 nanometers, research [27] shows that the colors that can be named by all cultures are generally limited to eleven. In addition to *black* and *white*, the discernible colors are *red*, *yellow*, *green*, *blue*, *brown*, *purple*, *pink*, *orange* and *gray*.

We first divide color into 12 color bins including 11 bins for culture colors and one bin for outliers [31]. At the coarsest resolution, we characterize color using a color mask of 12 bits. To record color information at finer resolutions, we record eight additional features for each color. These eight features are color histograms, color means (in H, S and V channels), color variances (in H, S and V channel), and two shape characteristics: elongation and spreadness. Color elongation characterizes the shape of a color, and spreadness characterizes how that color scatters within the image [38]. Table 1 summarizes color features in coarse, medium and fine resolutions.

6.1.2 Texture

Texture is an important cue for image analysis. Studies [41, 58, 61, 40] have shown that characterizing texture features in terms of structuredness, orientation, and scale (coarseness) fits well with models of human perception. A wide variety of texture analysis methods have been proposed in the past. We choose a discrete wavelet transformation (DWT) using quadrature mirror filters [58] because of its computational efficiency.

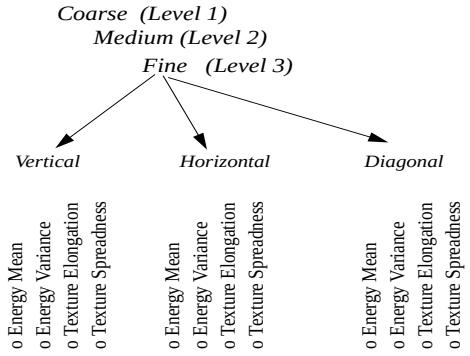


Figure 3: Multi-resolution Texture Features.

Each wavelet decomposition on a 2-D image yields four subimages: a $\frac{1}{2} \times \frac{1}{2}$ scaled-down image of the input image and its wavelets in three orientations: horizontal, vertical and diagonal. Decomposing the scaled-down image further, we obtain the tree-structured or wavelet packet decomposition. The wavelet image decomposition provides a representation that is easy to interpret. Every subimage contains information of a specific scale and orientation and also retains spatial information. We obtain nine texture combinations from subimages of three scales and three orientations. Since each subimage retains the spatial information of texture, we also compute elongation and spreadness for each texture channel. Figure 3 summarizes texture features.

Now, given an image, we can extract the above color and texture information to produce a 144 dimensional vector of numbers. We use this vector to represent the image. Thus, the space X for our SVMs is a 144 dimensional space, and each image in our database corresponds to a point in this space.

6.2 High-dimensional Indexer

An indexer is essential to make both query-concept learning and image retrieval efficient and scalable. Our indexing method is a statistical approach that works in two steps. It first performs non-supervised clustering using Tree-Structured Vector Quantization (TSVQ) [24] to group similar objects. To maximize IO efficiency, each cluster is stored in a sequential file. We determine which cluster an image belongs to by treating the tasks as a classification problem. Our hypothesis is that if a image's cluster prediction yields C probable clusters, then the probability is high that its nearest neighbors can be found in these C clusters. By searching for the most probable clusters into which the query object might belong, we can obtain most of the similar objects. To achieve accurate cluster predictions, we use classification error-reduction schemes including bagging and simulated annealing on similarity search results. For details, please consult [25, 26].

The indexing structure is helpful for both selecting samples and retrieving images given the class boundary.

- Informative example. Each image cluster is represented by its centroid. Initially we use just the centroids as unlabeled images in our pool. Since the number of cluster-centroids is substantially less than the number of data objects[2], they can be cached in main memory to avoid IOs. When $\mathsf{SVM}_{\mathit{Active}}$ considers images from the pool, it is simply accessing unlabeled centroids in main memory. The time required for selecting images can thus be drastically reduced. After using just the centroids in the pool, $\mathsf{SVM}_{\mathit{Active}}$ then adds the images in the clusters close to the current decision boundary.
- Image retrieval. Given an SVM hyperplane, our search engine tests the centroids that fall on the relevant side of the boundary. Each cluster is ranked by its centroid's distance to the SVM hyperplane: the farther away a cluster is from the hyperplane on the relevant side, the higher the priority. The search engine then reads the "farthest" cluster into memory and performs a sequential scan on the cluster to find the top-k nearest neighbors. If higher recall is desired, we read the next farthest cluster.

7 Seeding

$\mathsf{SVM}_{\mathit{Active}}$ requires at least one relevant and one irrelevant example to start its learning. Though irrelevant examples abound, relevant examples can be difficult to find. In particular, if the number of objects relevant to the query concept

[2]The ratio of the number of clusters over the total number of objects is data-set dependent and is adjustable. A typical ratio is less than 0.5%

is substantially less than that of the irrelevant ones, we need an effective strategy to find relevant starting images.

We call the process of finding initial relevant images *seeding*, and we present three seeding options: *seeding by* MEGA, *seeding by keywords*, and *seeding by images*. Although SVM_{Active} can be seeded by any of these methods, we believe that the first two options are more effective. The seeding-by-images paradigm, which relies upon users to start a search with "good" images, may not be realistic. After all, if users can always find good images to seed a search, why would they need an image search engine? In the remainder of this section, we first present our seeding-by-MEGA method, which uses another learning algorithm MEGA [11] to find relevant images. We then discuss a seeding-by-keywords alternative.

7.1 Seeding by MEGA

We propose using MEGA to seed SVM_{Active}. MEGA uses a boolean formula to describe query concepts. For the task of searching for an initial relevant image, MEGA uses a k-DNF formula to bound the set of permitted formulae. Thus, any image with features that satisfy the k-DNF is regarded as potentially relevant. MEGA attempts to choose images that either will be relevant, or will make the k-DNF as specific as possible. The detailed MEGA algorithm is documented in [11].

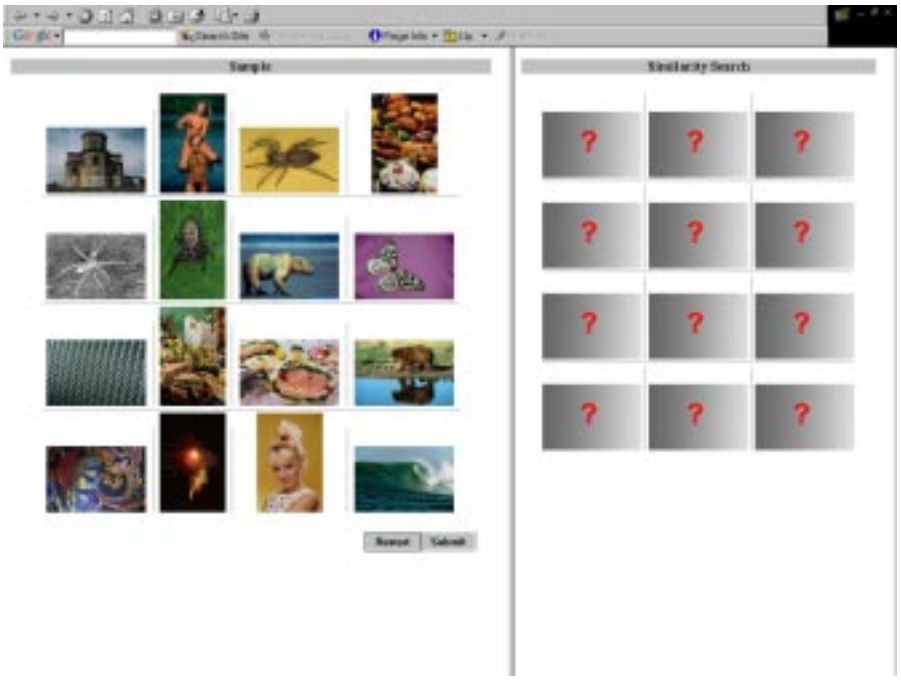
Seeding by MEGA Example

We present an example produced using a Corel image dataset to show the learning steps of SVM_{Active} seeded by MEGA for grasping a query concept "tigers." Our prototype works directly with the low-level image features described in Section 6.1. This interactive query session involves four screens that are illustrated in four figures.

Screen 1. Initial Screen. The system presents the initial screen to the user as depicted in Figure 4(a). The screen is split into two frames vertically. On the left-hand side of the screen is the learner frame; on the right-hand side is the retrieval frame. In the learner frame, the system displays informative images so as to learn what the user wants via an active learning process. The retrieval frame displays images that the system believes are most relevant, i.e., the images that the user wants. In this first screen, the system displays a random selection of images from the database. Notice that there is no tiger image present, so we press the submit button without highlighting any images.

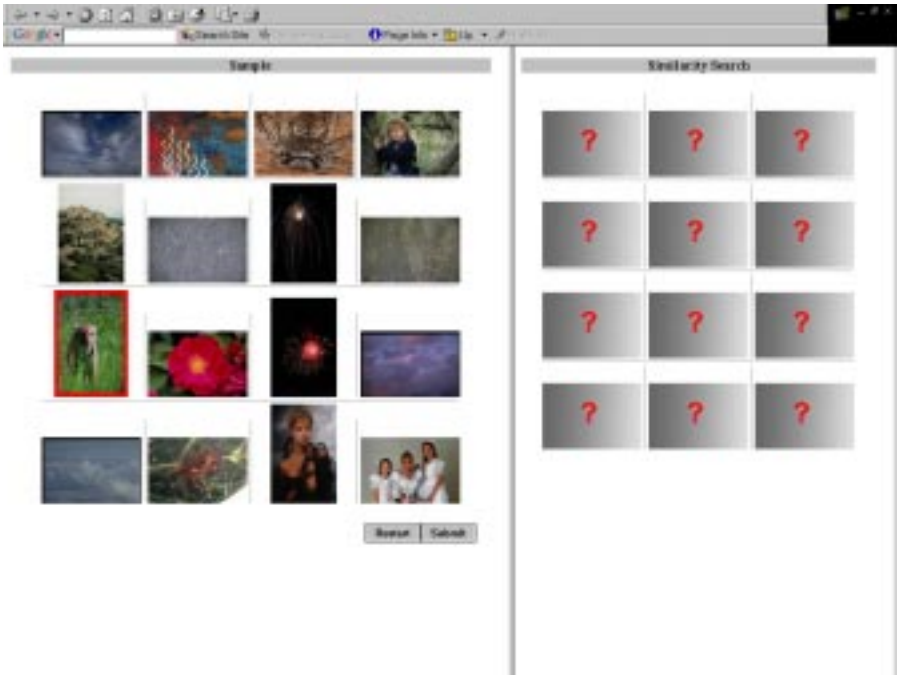
Screen 2. Since SVM_{Active} did not receive a relevant image in the initial round of random images, it uses MEGA to search for a single relevant seed image. Thus, MEGA now decides which images should be displayed for the user to label next. It chooses images that are rather different from the initial random selection. Notice that it presents a flower, clouds and tiger images. We are interested in tigers, so we highlight the tiger image as relevant, and the rest of the unmarked images are considered irrelevant. We indicate our selection by clicking on the submit button in the learner screen. This action brings up the next screen.

Screen 3. Sampling and relevance feedback continues. Now that MEGA has successfully found a relevant image, SVM_{Active} then learns an SVM using the one relevant image and 31 irrelevant images. SVM_{Active} then displays Figure 5(a). The right-hand frame displays images that the system expects will match the user's query concept at this time. As the figure indicates, with nine out of the top fifteen images already showing tigers, the system is beginning to learn which types of images we are seeking. In the left-hand frame, the system displays the images that it expects would be most informative for the user to label. The user can ask the system to further refine the target concept by selecting relevant images in this learner frame. After the user clicks on the submit button in the



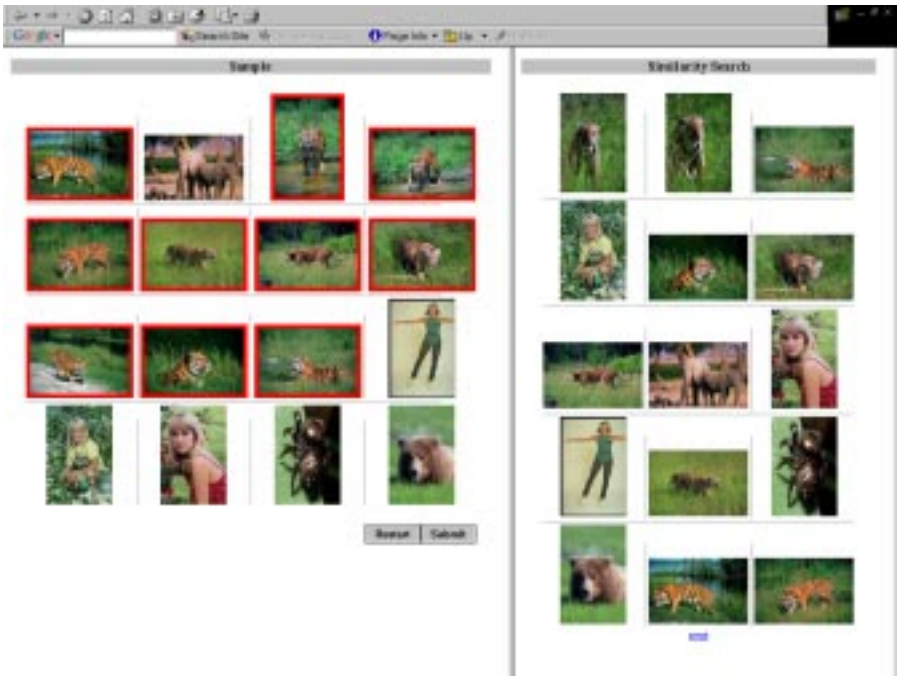
Learner Screen Retrieval Screen Screen

(a) Screen #1

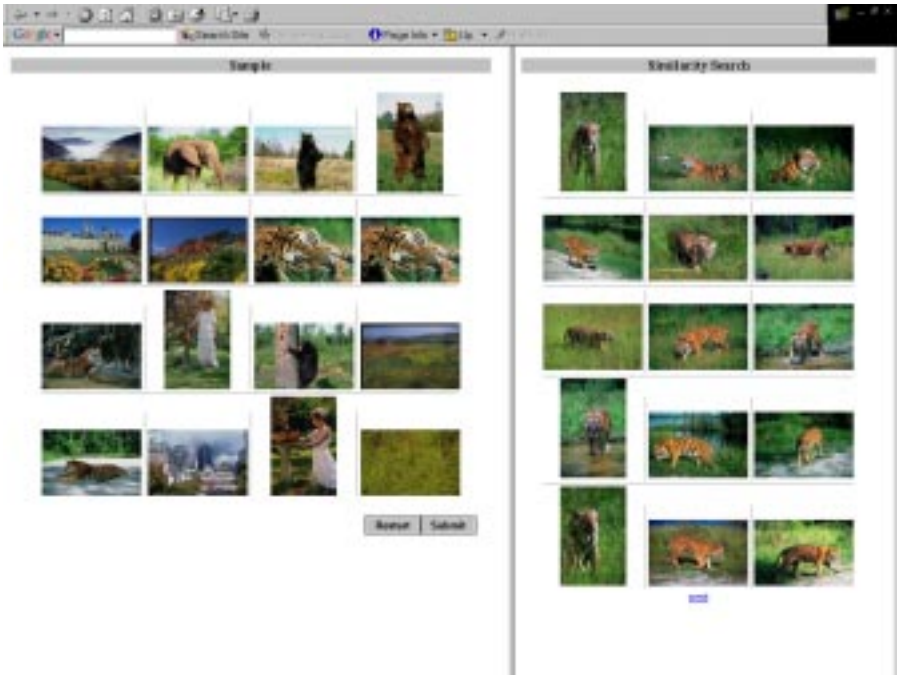


(b) Screen #2

Figure 4: Tiger Query Screens #1 and #2.



(a) Screen #3



(b) Screen #4

Figure 5: Tiger Query Screens #3 and #4.

learner frame, the fourth screen is displayed.

Screen 4. Sampling and relevance feedback ends. Figure 5(b) shows that all returned images in the similarity search frame fit the query concept (tiger).

This example shows that by using SVM_{Active} seeded with MEGA, a query concept can be learned in very few user iterations from no initial starting knowledge. The learner is able to match high-level concepts to low-level features directly through an active learning process.

Although much work is needed to improve our learning algorithms, this prototype demonstrates a potential to facilitate information retrieval in three respects:

- *Eliminating the seeding requirement*. A user is not required to initiate a query with "good" seed images. He or she does not need to provide an example image to start the search. Even if all initial examples presented are negative, the negative-labeled instances can be used to shrink the pool of the next sampling round substantially. Therefore, the probability that a relevant example will be presented in the next iteration will be substantially higher.
- *Supporting flexible concept formulation*. One can search for a general animal concept or a specific kind of animal. In the above example, if we were to mark all wild animal images as relevant, the final result would show other wild animals as well as tigers. Also note in the third screen that the system presents tigers of different kinds and on different backgrounds, to ask for feedback. The system can intelligently explore more possibilities to refine the query concept. (The readers are encouraged to experiment with our online demo [9].)
- *Accomplishing the above tasks quickly and accurately*. Our multi-resolution image-feature extractor and high-dimensional indexer (presented in Section 6) make both query-concept learning and image retrieval efficient and effective.

7.2 Seeding by Keyword

A text-based image retrieval engine finds images based on keywords. The keywords entered by users may have multiple senses and hence may not precisely characterize the users' query concept. There are a number of systems that perform image retrieval by keyword. For example, Figure 6 shows the search results with the keyword "baby" using the Google image search engine available at images.google.com. As observed, the returned images range from baby photos to baby furniture, statues, and cartoons.

Although keyword searches may not achieve high precision, a low-accuracy keyword search can be used instead of MEGA to seed SVM_{Active} with a single relevant image. As pointed out by [1, 2, 10], automatic annotation may not attain extremely high accuracy at the present state of computer vision and image processing. However, providing images with some reliable semantical labels and then refining these *unconfirmed* labels via relevance feedback is deemed an effective approach [68].

8 Experiments

For empirical evaluation of our learning methods, we used three real-world image datasets: a four-category, a ten-category, and a fifteen-category image dataset, each category consisting of 100 to 150 images. These image datasets were collected from Corel Image CDs and the Internet.

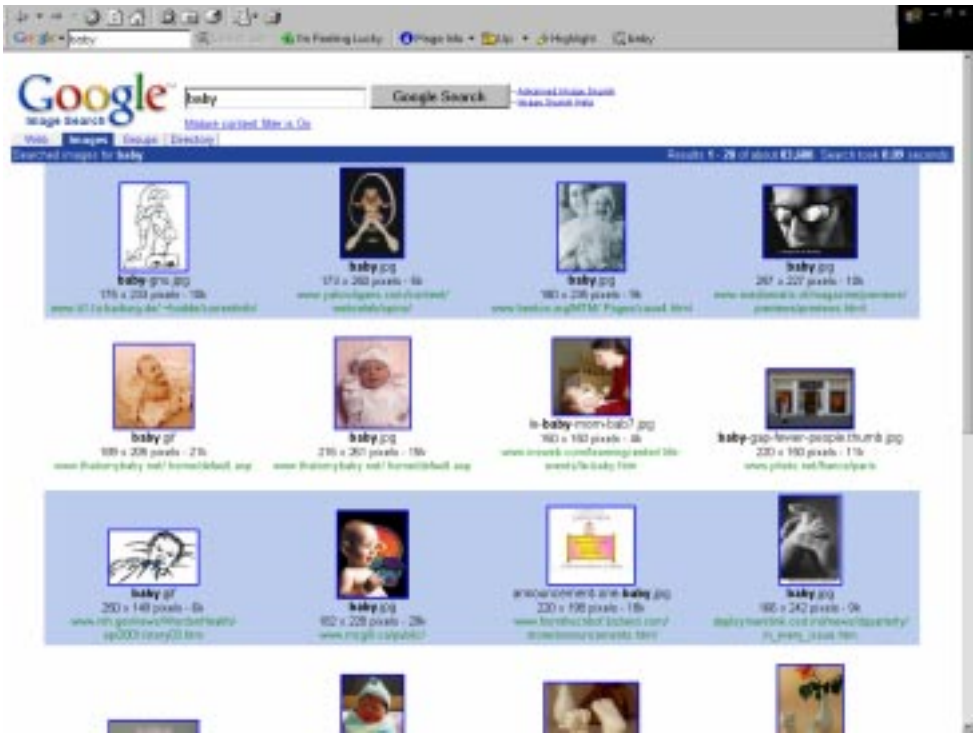
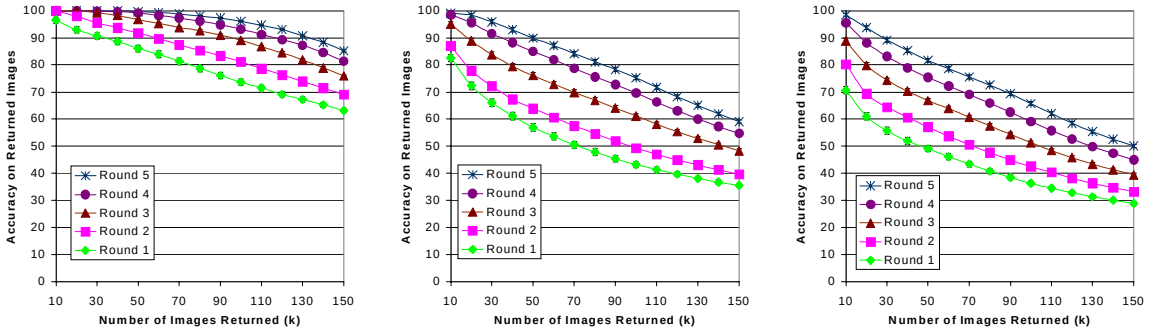


Figure 6: Google Image Search Results with Keyword "baby".

- *Four-category* set. The 602 images in this dataset belong to four categories — *architecture*, *flowers*, *landscape*, and *people*.
- *Ten-category* set. The 1277 images in this dataset belong to ten categories — *architecture*, *bears*, *clouds*, *flowers*, *landscape*, *people*, *objectionable images*, *tigers*, *tools*, and *waves*. In this set, a few categories were added to increase learning difficulty. The tiger category contains images of tigers with landscape and water backgrounds to complicate landscape category. The objectionable (pronographic) images can be confused with people wearing little clothing (beach wear). Clouds and waves have substantial color similarity.
- *Fifteen-category* set. In addition to the ten categories in the above dataset, the total of 1920 images in this dataset includes *elephants*, *fabrics*, *fireworks*, *food*, and *texture*. We added elephants with landscape and water backgrounds to increase learning difficulty in distinguishing landscape, tigers and elephants. We added colorful fabrics and food to interfere with flowers. Various texture images (e.g., skin, brick, grass, water, etc.) were added to raise learning difficulty for all categories.

To obtain an objective measure of performance, we assumed that a query concept was an image category. The SVM_{Active} learner has no prior knowledge about image categories[3]. It treats each image as a 144-dimension vector described in Section 6.1. The goal of SVM_{Active} is to learn a given concept through a relevance feedback process. In this process, at each feedback round SVM_{Active} selects twenty images to ask the user to label as "relevant" or "not relevant" with respect to the query concept. It then uses the labeled instances to successively refine the concept boundary. After finishing the relevance feedback rounds, SVM_{Active} then retrieves the top-k most relevant images

[3] Unlike some recently developed systems [66] that contain a semantical layer between image features and queries to assist query refinement, our system does not have an explicit semantical layer. We argue that having a hard-coded semantical layer can make a retrieval system restrictive. Rather, dynamically learning the semantics of a query concept is more flexible and hence makes the system more useful.



(a) (b) (c)

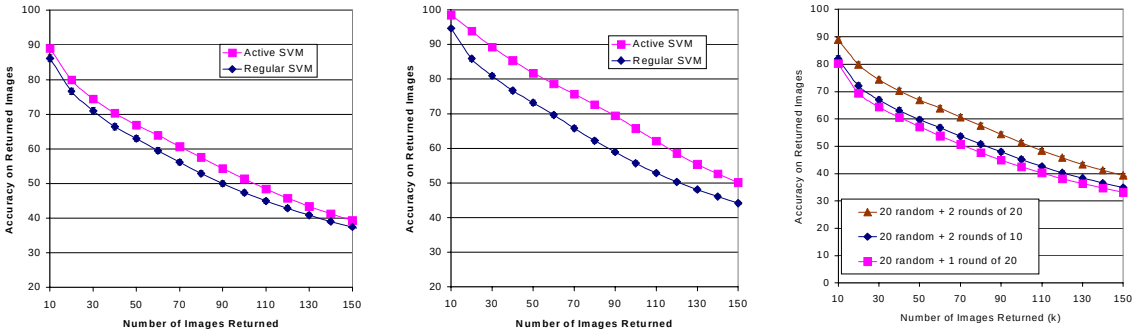
Figure 7: (a) Average top-k accuracy over the four-category dataset. (b) Average top-k accuracy over the ten-category dataset. (c) Average top-k accuracy over the fifteen-category dataset. Standard error bars are smaller than the curves' symbol size. Legend order reflects order of curves.

from the dataset, based on the final concept it has learned. Accuracy is then computed by looking at the fraction of the k returned result that belongs to the target image category. We note that this computation is equivalent to computing the precision on the top-k images. This measure of performance appears to be the most appropriate for the image retrieval task — particularly since, in most cases, not all of the relevant images can be displayed to the user on one screen. As in the case of web searching, we typically wish the first screen of returned images to contain a high proportion of relevant images. We are less concerned that not every single instance that satisfies the query concept is displayed. As with all SVM algorithms, SVM$_{Active}$ requires at least one relevant and one irrelevant image to function. In practice a single relevant image could be seeded into SVM$_{Active}$ by a keyword search or by MEGA, as we discussed in Section 7. In either case our experiments assume that we start off with one randomly selected relevant image and one randomly selected irrelevant image.

8.1 SVM$_{Active}$ Experiments

Figures 7(a-c) show the average top-k accuracy for the three different sizes of data sets. We considered the performance of SVM$_{Active}$ after each round of relevance feedback. The graphs indicate that performance clearly increases after each round. Also, the SVM$_{Active}$ algorithm's performance degrades gracefully when the size and complexity of the database is increased – for example, after four rounds of relevance feedback, it achieves an average of 100%, 95%, and 88% accuracy on the top-20 results for the three different sizes of data sets, respectively. It is also interesting to note that SVM$_{Active}$ is not only good at retrieving just the top few images with high precision, but it also manages to sustain fairly high accuracy even when asked to return larger numbers of images. For example, after five rounds of querying it attains 99%, 84% and 76% accuracy on the top-70 results for the three different sizes of data sets respectively. SVM$_{Active}$ uses the active querying method outlined in Section 5.

We examined the effect that the active querying method had on performance. Figures 8(a) and 8(b) compare the active querying method with the regular passive method of sampling. The passive method chooses images randomly from the pool to be labeled. This method is typically used with SVMs since it creates a randomly selected data set. It is clear that the use of active learning is beneficial in the image retrieval domain. We gain a significant increase in performance by using the active method. SVM$_{Active}$ displays 20 images per pool-querying round. There is a tradeoff between the number of images to be displayed in one round, and the number of querying rounds. The fewer images



(a) (b) (c)

Figure 8: (a) Active and regular passive learning on the fifteen-category dataset after three rounds of querying. (b) Active and regular passive learning on the fifteen-category dataset after five rounds of querying. (c) Comparison between asking ten images per pool-query round and twenty images per pool-querying round on the fifteen-category dataset. Standard error bars are smaller than the curves' symbol size. Legend order reflects order of curves.

displayed per round, the lower the performance. However, with fewer images per round we may be able to conduct more rounds of querying and thus increase our performance. Figure 8(c) considers the effect of displaying only ten images per round. In this experiment our first round consisted of displaying twenty random images and then, on the second and subsequent rounds of querying, active learning with 10 or 20 images is invoked. We notice that there is indeed a benefit to asking (20 random + two rounds of 10 images) over asking (20 random + one round of 20 images). This is unsurprising since the active learner has more control and freedom to adapt when asking for two rounds of 10 images rather than one round of 20. What is interesting is that asking (20 random + two rounds of 20 images) is far, far better than asking (20 random + two rounds of 10 images). The increase in the cost to users of asking 20 images per round is generally negligible since users can pick out relevant images easily. Furthermore, there is virtually no additional computational cost in calculating with the 20 images to query instead of the 10 images. Thus, for this particular task, we think it worthwhile to display around 20 images per screen and limit the number of querying rounds, rather than display fewer images per screen and require many more querying rounds.

Texture features	Top-50 Accuracy
None	80.6 ± 2.3
Fine	85.9 ± 1.7
Medium	84.7 ± 1.6
Coarse	85.8 ± 1.3
All	86.3 ± 1.8

Table 2: Average top-50 accuracy over the four-category data set using a regular SVM trained on 30 images. Texture spatial features were omitted.

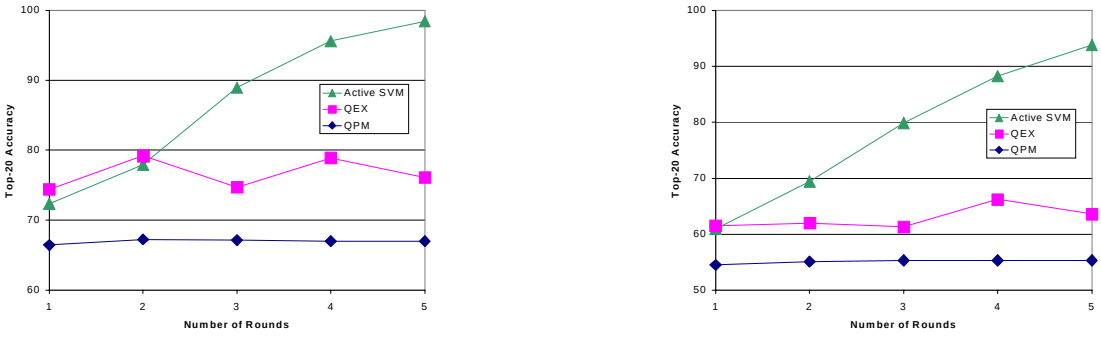
We also investigated how performance altered when various aspects of the algorithm were changed. Table 2 shows that all three of the texture resolutions are important. Also, the performance of the SVM appears to be greatest

	Top-50	Top-100	Top-150
Degree 2 Polynomial	95.9 ± 0.4	86.1 ± 0.5	72.8 ± 0.4
Degree 4 Polynomial	92.7 ± 0.6	82.8 ± 0.6	69.0 ± 0.5
Radial Basis	$\mathbf{96.8 \pm 0.3}$	$\mathbf{89.1 \pm 0.4}$	$\mathbf{76.0 \pm 0.4}$

Table 3: Accuracy on four-category data set after three querying rounds using various kernels. Bold type indicates statistically significant results.

Dataset	Dataset Size	round of 20 queries (secs)	Computing final SVM	Retrieving top 150 images
4 Cat	602	0.34 ± 0.00	0.5 ± 0.01	0.43 ± 0.02
10 Cat	1277	0.71 ± 0.01	1.03 ± 0.03	0.93 ± 0.03
15 Cat	1920	1.09 ± 0.02	1.74 ± 0.05	1.37 ± 0.04

Table 4: Average run times in seconds



(a) (b)

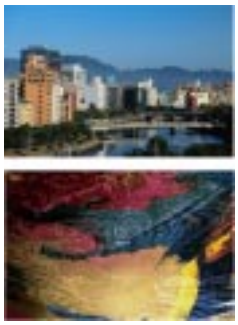
Figure 9: (a) Average top-k accuracy over the ten-category dataset. (b) Average top-k accuracy over the fifteen-category dataset.

when all of the texture resolutions are included (although in this case the difference is not statistically significant). Table 3 indicates how other SVM kernel functions perform on the image retrieval task compared to the radial basis function kernel. It appears that the radial basis function kernel is the most suitable for this feature space. One other important aspect of any relevance feedback algorithm is the wall clock time that it takes to generate the next pool-queries. Relevance feedback is an interactive task, and if the algorithm takes too long then the user is likely to lose patience and be less satisfied with the experience. Table 4 shows that SVM_{Active} averages about a second on a Sun Workstation to determine the 20 most informative images for the users to label. Retrieval of the 150 most relevant images takes a similar amount of time, and computing the final SVM model never exceeds two seconds.

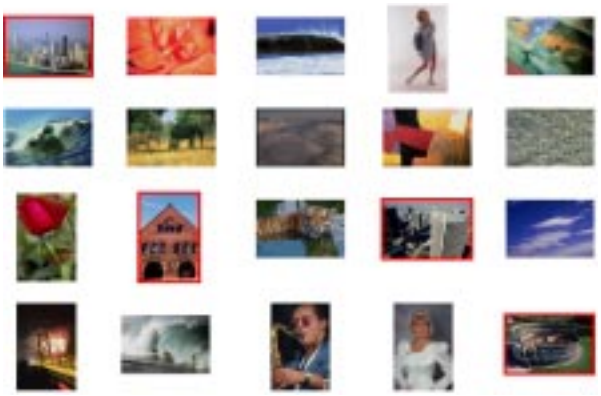
8.2 Scheme Comparison

We also compared SVM_{Active} with two traditional query refinement methods: *query point movement* (QPM) and *query expansion* (QEX). In this experiment, each scheme returned the 20 most relevant images after up to five rounds of relevance feedback. To ensure that the comparison to SVM_{Active} was fair, we seeded both schemes with one randomly selected relevant image to generate the first round of images. On the ten-category image dataset, Figure 9(a) shows that SVM_{Active} achieves nearly 90% accuracy on the top-20 results after three rounds of relevance feedback, whereas the accuracies of both QPM and QEX never reach 80%. On the fifteen-image category dataset, Figure 9(b) shows that SVM_{Active} outperforms the others by even wider margins. SVM_{Active} reaches 80% top-20 accuracy after three rounds and 94% after five rounds, whereas QPM and QEX cannot achieve 65% accuracy.

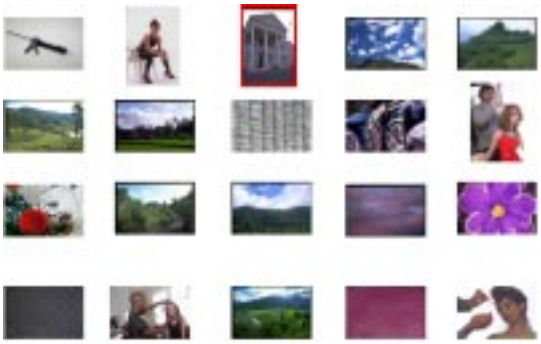
These results hardly surprise us. Traditional information retrieval schemes require a large number of image instances to achieve any substantial refinement. By just refining around current relevant instances, both QPM and QEX tend to be fairly localized in their exploration of the image space and hence rather slow in exploring the entire space. In contrast, during the relevance feedback phase SVM_{Active} takes both the relevant and irrelevant images into account when choosing the next pool-queries. Furthermore, it chooses to ask the user to label images that it regards



Initializing



Feedback Round 1



Feedback Round 2

Feedback Round 3

Figure 10: Searching for architecture images. SVM_{Active} Feedback phase.

as most informative for learning the query concept, rather than those that have the most likelihood of being relevant. Thus it tends to explore the feature space more aggressively.

Figures 10 and 11 show an experimental run of the SVM_{Active} system. For this run, we are interested in obtaining architecture images. In Figure 10 we initialize the search by giving SVM_{Active} one relevant and one irrelevant image. We then have three feedback rounds. The images that SVM_{Active} asks us to label in these three feedback rounds are images that SVM_{Active} will judge to be most informative. For example, we see that it asks us to label a number of landscape images and other images with a blue or gray background with something in the foreground. The feedback rounds allow SVM_{Active} to narrow down the range of images that we like. When it comes to the retrieval phase (Figure 10) SVM_{Active} returns, with high precision, a large variety of different architecture images, ranging from old buildings to modern cityscapes.

9 Conclusions and Future Work

We have demonstrated that active learning with support vector machines can provide a powerful tool for searching image databases, outperforming a number of traditional query refinement schemes. SVM_{Active} not only achieves consistently high accuracy on a wide variety of desired returned results, but also does it quickly and maintains high precision when asked to deliver large quantities of images. Also, unlike recent systems such as SIMPLIcity [66],

First Screen of Results

Second Screen of Results

Third Screen of Results

Fourth Screen of Results

Fifth Screen of Results

Sixth Screen of Results

Figure 11: Searching for architecture images. SVM_{Active} Retrieval phase.

it does not require an explicit semantical layer to perform well. Our system takes advantage of the intuition that there can be considerable differences between the set of images that we are already confident a user wishes to see, and the set of images that would most informative for the user to label. By decoupling the notions of feedback and retrieval, and by using a powerful classifier with active learning, we have demonstrated that SVM_{Active} can provide considerable gains over other systems.

We have also presented our image characterization and feature indexing methods, which make concept-learning and image retrieval effective and efficient. We have built prototypes [9, 12], which use MEGA or keywords to seed SVM_{Active} by finding the first image(s) relevant to the query concept, and then switches to use SVM_{Active} for the subsequent rounds of feedback. We are currently investigating methods to improve SVM_{Active}'s sample selection method. More specifically, we observed that when the training instances of the target class (i.e., relevant images with respect to the query concept) are heavily outnumbered by non-target training instances (irrelevant images), SVMs can be ineffective in determining the class boundary. To remedy this problem, we have proposed an adaptive conformal transformation (ACT) algorithm [69]. ACT considers feature-space distance and the class-imbalance ratio when it performs conformal transformation on a kernel function. We will validate ACT's effectiveness with SVM_{Active}.

10 Acknowledgements

The first author is supported by NSF grants IIS-0133802 (CAREER) and IIS-0219885 (ITR).

References

[1] K. Barnard and D. Forsyth. Learning the semantics of words and pictures. In *International Conference on Computer Vision*, volume 2, pages 408–415, 2000.
[2] K. Barnard and D. Forsyth. Clustering art. In *CVPR*, 2001.
[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1998.
[4] L. Breiman. Bagging predicators. *Machine Learning*, pages 123–140, 1996.
[5] L. Breiman. Arcing classifiers. *The Annals of Statistics*, pages 801–849, 1998.
[6] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.
[7] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.
[8] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of ACM*, 44(3):427–485, 1997.
[9] E. Chang, K.-T. Cheng, W.-C. Lai, C.-T. Wu, C.-W. Chang, and Y.-L. Wu. PBIR — a system that learns subjective image query concepts. *Proceedings of ACM Multimedia, http://www.mmdb.ece.ucsb.edu/˜ demo/corelacm/*, pages 611–614, October 2001.
[10] E. Chang, K. Goh, G. Sychay, and G. Wu. Content-based soft annotation for multimodal image retrieval using bayes point machines. *IEEE Transactions on Circuits and Systems for Video Technology Special Issue on Conceptual and Dynamical Aspects of Multimedia Content Description*, 13(1):26–38, 2003.
[11] E. Chang and B. Li. Mega — the maximizing expected generalization algorithm for learning complex query concepts (extended version). *Technical Report http://www-db.stanford.edu/∼echang/mega.pdf*, November 2000.
[12] E. Chang, B. Li, W.-C. Lai, C. Chang, K.-T. Cheng, and M. Crandell. A multimodal image database system (demo). *Proceedings of IEEE CVPR*, 2003.
[13] E. Chang, B. Li, and C. Li. Towards perception-based image retrieval. *IEEE Content-Based Access of Image and Video Libraries*, pages 101–105, June 2000.
[14] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4, 1996.
[15] I. Cox, J. Ghosn, M. Miller, T. Papathomas, and P. Yianilos. Introduces pichunter, the bayesian framework, and describes a working system including measured user performance. hidden annotation in content based image retrieval. In *IEEE Workshop on Content-Based Access of Image & Video Libraries*, pages 76–81, June 1997.

[16] I. Cox, M. Miller, S. Omohundo, and P. Yianilos. Pichunter: Bayesian relevance feedback for image retrieval. In *13th International Conference on Pattern Recognition*, pages 361–369, August 1996.
[17] I. Cox, M. Miller, S. Omohundo, and P. Yianilos. Target testing and the pichunter bayesian multimedia retrieval system. In *Proceedings of the Forum on Research & Technology Advances in Digital Libraries*, pages 66–75, 1996.
[18] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The bayesian image retrieval system, pichunter: Theory, implementation and psychological experiments. *IEEE Transaction on Image Processing (to appear)*, 2000.
[19] I. Dagan and S. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. Morgan Kaufmann, 1995.
[20] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artifical Intelligence Research*, 2, 1995.
[21] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*. ACM Press, 1998.
[22] M. Flickner, H. Sawhney, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the QBIC system. *IEEE Computer*, 28(9):23–32, 1995.
[23] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the Query by Committee algorithm. *Machine Learning*, 28:133–168, 1997.
[24] A. Gersho and R. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic, 1991.
[25] K. Goh and E. Chang. Indexing multimedia data in high-dimensional and weighted feature spaces (invited). *The 6^{th} Visual Database Conference*, 2002.
[26] K. Goh, B. Li, and E. Chang. Dyndex: A dynamic and non-metric spaceindexer. *Proceedings of ACM International Conference on Multimedia*, December 2002.
[27] E. B. Goldstein. *Sensation and Perception (5^{th} Edition)*. Brooks/Cole, 1999.
[28] A. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proc. 15th National Conference on Artificial Intelligence (AAAI)*, 1998.
[29] A. Gupta and R. Jain. Visual information retrieval. *Comm. of the ACM*, 40(5):69–79, 1997.
[30] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines: Estimating the bayes point in kernel space. In *International Joint Conference on Artificial Intelligence Workshop on Support Vector Machines*, pages 23–27, 1999.
[31] K. A. Hua, K. Vu, and J.-H. Oh. Sammatch: A flexible and efficient sampling-based image retrieval technique for image databases. *Proceedings of ACM Multimedia*, November 1999.
[32] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. *VLDB*, 1998.
[33] T. Jaakkola and H. Siegelmann. Active information retrieval. *Advances in Neural Information processing systems 14*, 2001.
[34] G. James and T. Hastie. *Error Coding and PaCT's*. 1997.
[35] T. Joachims. Text categorization with support vector machines. In *Proceedings of the European Conference on Machine Learning*. Springer-Verlag, 1998.
[36] K. S. Jones and P. W. (Editors). *Readings in Information Retrieval*. Morgan Kaufman, July 1997.
[37] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
[38] J.-G. Leu. Computing a shape's moments from its boundary. *Pattern Recognition*, pages Vol.24, No.10,pp.949–957, 1991.
[39] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag, 1994.
[40] W. Y. Ma and H. Zhang. Benchmarking of image features for content-based retrieval. *Proceedings of Asilomar Conference on Signal, Systems & Computers*, 1998.
[41] B. Manjunath, P. Wu, S. Newsam, and H. Shin. A texture descriptor for browsing and similarity retrieval. *Signal Processing Image Communication*, 2001.
[42] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.
[43] A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998.
[44] T. Mitchell. Generalization as search. *Artificial Intelligence*, 28:203–226, 1982.
[45] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
[46] M. Moreira and E. Mayoraz. Improving pairwise coupling classification with error correcting classifiers. *Proceedings of the Tenth European Conference on Machine Learning*, April 1998.
[47] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang. Supporting similarity queries in mars. *Proc. of ACM Conf. on Multimedia*, 1997.
[48] M. Ortega, Y. Rui, K. Chakrabarti, A. Warshavsky, S. Mehrotra, and T. S. Huang. Supporting ranked boolean similarity queries in mars. *IEEE Transaction on Knowledge and Data Engineering*, 10(6):905–925, December 1999.
[49] M. Ortega-Binderberger and S. Mehrotra. *Relevance Feedback in Multimedia Databases*. in Handbook of Video Databases: Design and Applications, Borko Furht and Oge Marquez eds., 2003.
[50] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the International Conference on Computer Vision*, 1998.

[51] K. Porkaew, K. Chakrabarti, and S. Mehrotra. Query refinement for multimedia similarity retrieval in mars. *Proceedings of ACM Multimedia*, November 1999.
[52] K. Porkaew, S. Mehrota, and M. Ortega. Query reformulation for content based multimedia retrieval in mars. *ICMCS*, pages 747–751, 1999.
[53] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool in interactive content-based image retrieval. *IEEE Tran on Circuits and Systems for Video Technology*, 8(5), Sept 1998.
[54] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceeding of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann, 1997.
[55] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.
[56] H. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, pages 287–294. Morgan Kaufmann, 1992.
[57] J. Shawe-Taylor and N. Cristianini. Further results on the margin distribution. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 278–285, 1999.
[58] J. Smith and S.-F. Chang. Automated image retrieval using color and texture. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, November 1996.
[59] J. Smith and S.-F. Chang. Tools and techniques for color image retrieval. *Volume 2670 of SPIE Proceedings*, pages 1630–1639, 1996.
[60] J. R. Smith and S.-F. Chang. Visualseek: A fully automated content-based image query system. *ACM Multimedia Conference*, 1996.
[61] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *IEEE Transaction on Systems Man Cybernet (SMC)*, 1978.
[62] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *Proceedings of ACM International Conference on Multimedia*, pages 107–118, October 2001.
[63] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Proceedings of the 17th International Conference on Machine Learning*, pages 401–412, June 2000.
[64] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag, 1982.
[65] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
[66] J. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *ACM Multimedia Conference*, 2000.
[67] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei. Wavelet-based image indexing techniques with partial sketch retrieval capability. *Proceedings of the 4th ADL*, May 1997.
[68] L. Wenyin, S. Dumais, Y. Sun, H. Zhang, M. Czerwinski, and B. Field. Semi-automatic image annotation. In *Proc. of Interact 2001: Conference on Human-Computer Interaction*, pages 326–333, July 2001.
[69] G. Wu and E. Chang. Adaptive feature-space conformal transformation for learning imbala nced data. *International Conference on Machine Learning*, August 2003.
[70] H. Wu, H. Lu, and S. Ma. A practical svm-based algorithm for ordinal regression in image retrieval. *ACM International Conference on Multimedia*, 2003.
[71] L. Wu, C. Faloutsos, K. Sycara, and T. R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. *The* 26^{th} *VLDB Conference*, September 2000.
[72] C. Zhang and T. Chen. An active learning framework for content-based information retrieval. *IEEE Transactions on Multimedia*, 4(2):260–268, 2002.
[73] X. S. Zhou and T. S. Huang. Comparing discriminating transformations and svm for learning during multimedia retrieval. *Proc. of ACM Conf. on Multimedia*, pages 137–146, 2001.
[74] X. S. Zhou and T. S. Huang. Relevance feedback for image retrieval: a comprehensive review. *ACM Multimedia Systems Journal, special issue on CBIR (to appear)*, 2003.