Support Vector Machine Concept-Dependent Active Learning for Image Retrieval

Edward Y. Chang
VIMA Technologies
120 Cremona Drive
Santa Barbara, CA 93117
echang@vimatech.com

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

Kingshy Goh
VIMA Technologies
120 Cremona Drive
Santa Barbara, CA 93117
kingshy@engineering.ucsb.edu

Cheng-Wei Chang
VIMA Technologies
120 Cremona Drive
Santa Barbara, CA 93117
bchang@vimatech.com

Abstract

Relevance feedback is a critical component when designing image databases. With these databases it is difficult to specify queries directly and explicitly. Relevance feedback interactively *learns* a user's desired output or *query concept* by asking the user whether certain proposed images are relevant or not. For a learning algorithm to be effective, it must *learn* a user's query concept accurately and quickly, while also asking the user to label only a small number of images. In addition, the concept-learning algorithm should consider the *complexity* a concept in determining its learning strategies. In this paper, we present the use of support vector machines active learning in a concept-dependent way ($\mathsf{SVM}^{CD}_{Active}$) for conducting relevance feedback. We characterize a concept's complexity using three measures: *hit-rate*, *isolation* and *diversity*. To reduce concept complexity so as to improve concept learnability, we propose a multimodal learning approach that uses images' semantic labels to intelligently adjust the *sampling strategy* and the *sampling pool* of $\mathsf{SVM}^{CD}_{Active}$. Our empirical study on several datasets shows that active learning outperforms traditional passive learning, and concept-dependent learning is superior to the traditional concept-independent relevance-feedback schemes.

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 instances 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* [35, 39]. 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* ($\mathsf{SVM}^{CD}_{Active}$) to achieve these goals. $\mathsf{SVM}^{CD}_{Active}$ combines active learning with support vector machines (SVMs). SVMs [6, 60] have met with significant success in numerous real-world learning tasks. However, like most machine learning algorithms, SVMs 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, 53, 59]. In the first part of this paper, we use the theoretical motivation of [59] on active learning with SVMs to extend the use of support vector machines to the task of relevance feedback for image databases.

When conducting a pool-query, $\mathsf{SVM}^{CD}_{Active}$ should consider the *complexity* of the target concept to adjust its *sampling strategies* for choosing informative images within the pool to ask user to label. In the second part of this paper, we characterize *concept complexity* using three parameters: *scarcity*, *isolation* and *diversity*. Each of these parameters affects the learnability of a concept in a different way. The *hit-rate* (scarcity) is the percentage of instances matching the target concept in a database. When the *hit-rate* is low, it is difficult for active learning to find enough relevant instances to match

the query. When *isolation* of a concept is poor, or several concepts are not well isolated from one another in the input space[1], the learner might confuse the target concept with its neighbors. Finally, when a concept's *diversity* is high, the algorithm needs to be more explorative, rather than narrowly focused, so as to find the relevant instances that are scattered in the input space. We investigate how these complexity factors affect concept learnability, and then propose a multimodal learning approach that uses semantic labels (keywords) of images to guide our *concept-dependent active-learning* process. We adjust the *sample pool* and *sampling strategy* according to concept complexity: $\mathsf{SVM}^{CD}_{Active}$ carefully selects the sample pool to improve *hit-rate* and *isolation* of the query concept, and it enhances the concept's learnability by adapting its sampling strategy to the concept's *diversity*.

Intuitively, $\mathsf{SVM}^{CD}_{Active}$ works by combining the following four ideas:

1. $\mathsf{SVM}^{CD}_{Active}$ 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. $\mathsf{SVM}^{CD}_{Active}$ learns the classifier quickly via active learning. The active part of $\mathsf{SVM}^{CD}_{Active}$ 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. $\mathsf{SVM}^{CD}_{Active}$ learns the classifier in a concept-dependent way. With multimodal information from keywords and images' perceptual features, $\mathsf{SVM}^{CD}_{Active}$ improves concept learnability for effectively capturing the target concept.

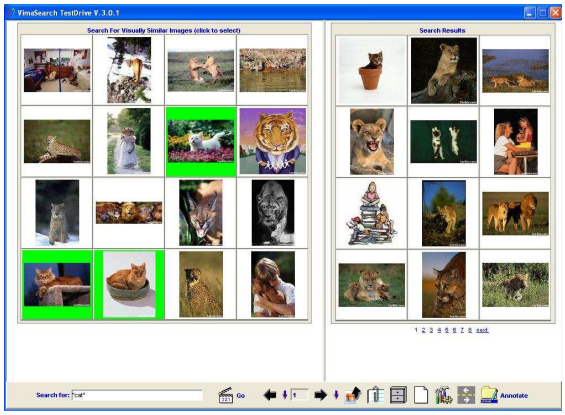
4. Once the classifier is trained, $\mathsf{SVM}^{CD}_{Active}$ returns the top-k most relevant images. These are the k images farthest from the hyperplane on the query concept side.

Example Queries

We present two example queries in Figure 1 to demonstrate how a query concept is learned in an iterative process to improve query results. The user interface shows two frames. The frame on the left-hand side is the feedback frame, on which the user marks images relevant to his or her query concept. On the right-hand side, the search engine returns what it considers matching the concept learned this far from the image database. The column on the left-hand side of Figure 1 shows three iterations of a "cat" query; the column on the right-hand side shows three iterations of an "eagle" query.

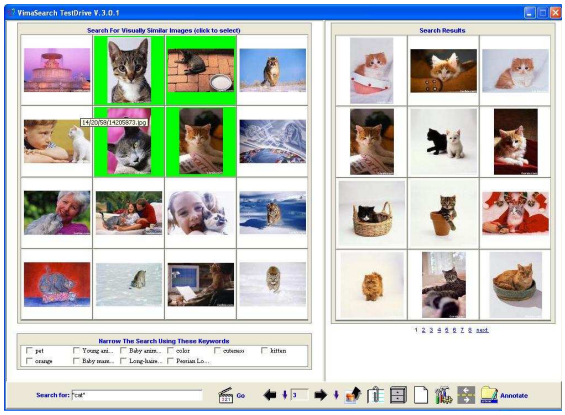
To query "cat," we first enter keyword *cat* to get the first screen of results in Figure 1(a). The right-hand-side frame shows a couple of images containing cats, but several images containing tigers or lions. This is because many tiger/lion images were annotated with "wild cat" or "cat." To disambiguate

[1]The *input space* (denoted as X in Machine Learning and Statistics literature) is defined as the original space in which the data vectors are located, and the *feature space* (denoted as F) is the space into which the data are projected, either linearly or non-linearly.

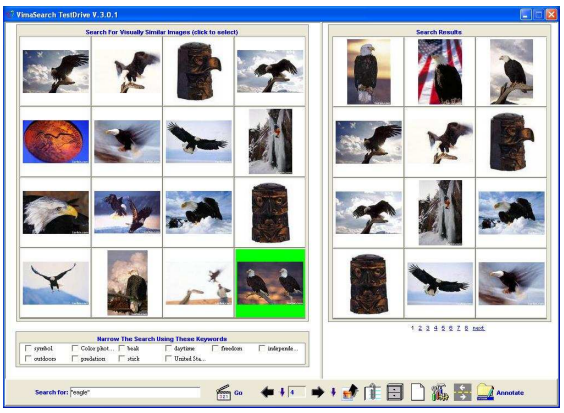


(a) Cat Iteration 2

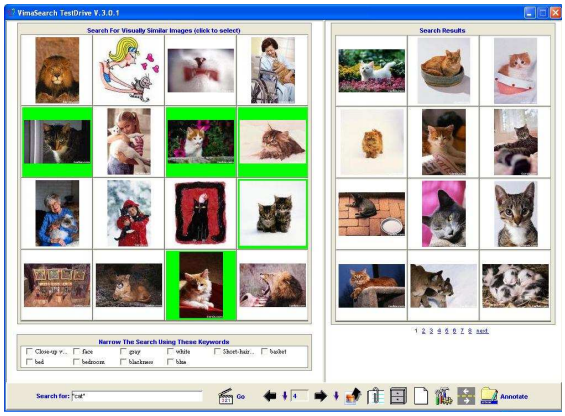
(b) Eagle Iteration 1



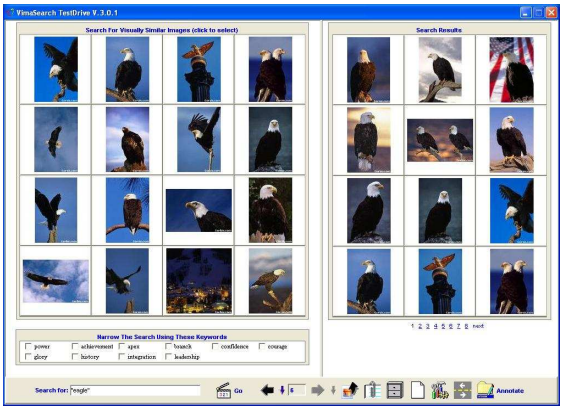
(c) Cat Iteration 4



(d) Eagle Iteration 2



(e) Cat Iteration 4



(f) Eagle Iteration 6

Figure 1: Two Example Queries, “Cat” on the left, and “Eagle” on the right.

the concept, we click on a couple of cat images on the feedback frame. $\mathsf{SVM}^{CD}_{Active}$ refines the class boundary, and then returns the second screen in Figure 1(c). In this figure, we can see that the results in the (right-hand-side) result frame have been much improved. All returned images contain a cat or two. After we perform another couple of rounds of feedback to make some slight refinement, the third screen in Figure 1(e) shows satisfactory results.

For the "eagle" query, we would like to find "sitting eagles." Since keyword "sitting" was not used to annotate any of the eagle images in the database, we cannot rely on keywords to find matching images. Typing in "eagle," we do obtain mostly eagle images returned in Figure 1(b). However, only the eagles on the top row display a sitting posture. We provide feedback to the system by clicking on relevant images on the feedback frame. We are able to remove "flying eagle" images after two iterations, and "wooden sitting eagles" after another two iterations. All but one image in Figure 1(f) satisfy the "sitting eagles" query concept.

The rest of this paper is organized into five sections. Section 2 surveys related work. Section 3 introduces SVMs and the notion of a *version space* which in Section 4 provides theoretical motivation for sampling methods of performing active learning with SVMs. Section 5 presents the concept-dependent component of $\mathsf{SVM}^{CD}_{Active}$. In Section 6, we report experimental results on several datasets. Finally, we offer our conclusions in Section 7.

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 [2, 33, 41, 66, 67], and they require seeding a query with "good" examples [32, 48, 65, 46]. 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* [3], *arcing* [4], and *boosting* [24, 52, 64] have been proposed to improve classification 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 [18, 24, 30, 42].

To reduce the number of required samples, researchers have conducted several studies of active learning [12, 8, 29, 58] for classification. Active learning can be modeled formally as follows: Given a dataset S consisting of an unlabeled subset U and a labeled subset L, an active learner has two components: f and q. The f component is a classifier that is trained on the current set of labeled

data L. The second component q is the sampling function that, given a current labeled set L, decides which subset $\mathbf{u}$ 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 [22, 54] 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 [17, 39]) 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 [31, 20], 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 typically non-linear[2], our $\mathsf{SVM}^{CD}_{Active}$ with kernel mapping provides more flexible and accurate concept modeling.

Specifically for image retrieval, the PicHunter system [15, 14, 13, 16] 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 [11].

2.2 Relevance Feedback

The study of [49] 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* [28, 45, 44, 48, 51]. 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 [28, 65], 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* [49, 65]. The *query expansion* approach can be regarded as a multiple-instance 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 [49] shows that query expansion achieves only a slim margin of improvement (about 10% in preci-

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

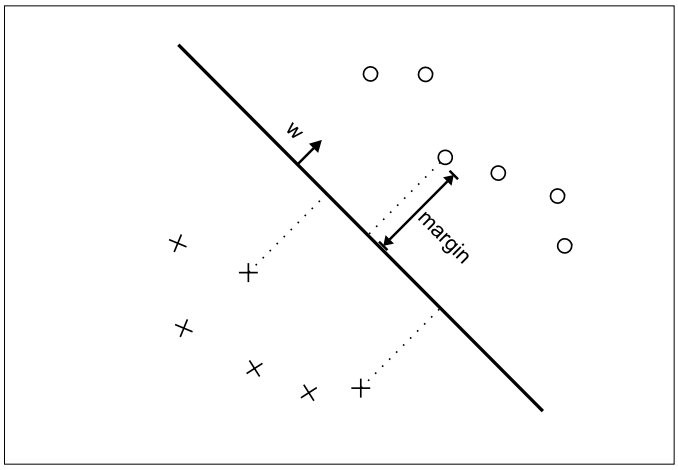


Figure 2: A simple linear Support Vector Machine

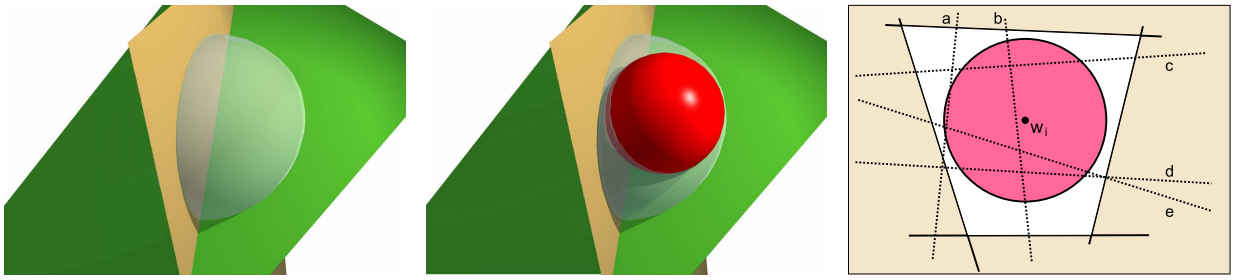
sion/recall) over query point movement.

Almost all traditional relevance feedback methods require seeding the methods with "good" positive examples [21, 25, 27, 38, 56, 57, 63], 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. $\mathsf{SVM}^{CD}_{Active}$ effectively uses negative-labeled instances to induce more negative instances, and thereby improves 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 disambiguate 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 and Version Space

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 [61], object recognition [47], and text classification [31].

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 $\mathsf{X} \subseteq \mathbb{R}^{\mathsf{d}}$. 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. 2). 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



(a) (b) (c)

Figure 3: (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.

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:

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

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 $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + \mathbf{1})^\mathbf{p}$, 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 $\mathbf{x_i}$, $\|\Phi(\mathbf{x_i})\| = \varphi$ for some fixed φ. 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 $\Phi(\mathbf{x_i})$ [58].

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* [40]. 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 3.1 *Our set of possible hypotheses is given as:*

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

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

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

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:*

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

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 [55], 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 [61, 26] 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 $y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) = \mathbf{w} \cdot y_i \mathbf{\Phi}(\mathbf{x_i}) > \mathbf{0}$ defines a half-space in W. Furthermore $\mathbf{w} \cdot y_i \mathbf{\Phi}(\mathbf{x_i}) = \mathbf{0}$ 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.3(a) for an example.

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

$$\begin{aligned} \text{maximize}_{\mathbf{w} \in \mathsf{F}} \quad & \min_i \{y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}))\} \\ \text{subject to:} \quad & \|\mathbf{w}\| = \mathbf{1} \\ & y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i})) > \mathbf{0} \;\; \mathbf{i} = \mathbf{1} \ldots \mathbf{n}. \end{aligned}$$

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 $\min_i\{\mathbf{w} \cdot y_i \mathbf{\Phi}(\mathbf{x_i})\}$. 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 3(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.

4 Active Learning and Batch Sampling Strategies

$\mathsf{SVM}^{CD}_{Active}$ performs the following two steps for each round of relevance feedback until the process is terminated by the user.

- *Sampling.* Select a batch of images and ask the user to label them as either "relevant" or "irrelevant" to the query concept, and
- *Learning.* Learn an SVM on data labeled thus far.

After the relevance feedback rounds have been performed, $\mathsf{SVM}^{CD}_{Active}$ retrieves the top-k most relevant images. The final SVM boundary separates "relevant" images from irrelevant ones. The top-k most "relevant" images are the k images farthest from the SVM boundary on the relevant side. The key step of $\mathsf{SVM}^{CD}_{Active}$ is its *sampling* step in which it selects most useful images to solicit user feedback. In the remainder of this section, we present the theoretical foundation of $\mathsf{SVM}^{CD}_{Active}$ (Section 4.1), and then four sampling strategies (Section 4.2).

4.1 Theoretical Foundation

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, L). The first component is a classifier, $f : \mathsf{L} \rightarrow \{-1, 1\}$, trained on the current set of labeled data L (and possibly unlabeled instances in U too). The second component $q(\mathsf{L})$ is the querying function that, given a current labeled set L, 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 4.1 *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 4.2 *(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,*

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

with strict inequality whenever there exists a pool-query $j \in \{1 \dots i\}$ *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 $\mathsf{V}_1 \supset \mathsf{V}_2 \supset \mathsf{V}_3 \dots$, where 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 [59]:

Simple **Method.** Recall from Section 3 that, given data $\{\mathbf{x_1} \dots \mathbf{x_i}\}$ and labels $\{y_1 \dots y_i\}$, 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 V_i. The position of $\mathbf{w_i}$ in the version space V_i clearly depends on the shape of the region 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 3(c) presents an illustration. In the stylized picture we have flattened out the surface of the unit weight vector hypersphere that appears in Figure 3(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 Figure3(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 $\mathbf{b}$.

Our $\mathsf{SVM}^{CD}_{Active}$ 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 [7] that do not require a constraint on the modulus.

4.2 Sampling Strategies

For the image retrieval domain, we 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, h) at each round of pool-querying. Thus, for each round, the active learner has to choose not just one image to be labeled but h. Theoretically it would be possible to consider the size of the resulting version spaces for each possible labeling of each possible set of h pool-queries, but clearly this would be impractical. Thus instead, for matters of computational efficiency, $\mathsf{SVM}^{CD}_{Active}$ employs heuristic methods for choosing unlabeled instances. We present and examine the following four sampling strategies: *speculative*, *batch-simple*, *angle-diversity*, and *error-reduction*. (For other relevant work, please consult [37, 23, 43].)

Speculative

Input: L, U, h; /* labeled set, unlabeled set, batch size

Output: S; /* Pool-queries or samples

Procedures:

- $\text{SVM}_{Train}()$ /* SVM training algorithm
- $f()$ /* Classifier learned by SVMs on L

Initialization:

- $S \leftarrow \emptyset$

BEGIN

1. if $(h = 0)$ then return $\emptyset$;
2. $f \leftarrow \text{SVM}_{Train}(\mathsf{L})$;
3. For each $\mathbf{x}_i \in \mathsf{U}$
$\mathbf{x}_i.distance \leftarrow \mid f(\mathbf{x}_i) \mid$;
4. $\mathbf{x}_s \leftarrow argmin_{\mathbf{x}_i \in \mathsf{U}}(\mathbf{x}_i.distance)$;
5. $\mathsf{U} \leftarrow \mathsf{U} - \{\mathbf{x_s}\}$; $\mathsf{L} \leftarrow \mathsf{L} \cup \{\mathbf{x_s}\}$;
6. $\mathbf{x}_s.label \leftarrow +$;
7. $S \leftarrow S \cup Speculative(\mathsf{L}, \mathsf{U}, (\mathsf{h}-1)/2)$;
8. $\mathbf{x}_s.label \leftarrow -$;
9. $S \leftarrow S \cup Speculative(\mathsf{L}, \mathsf{U}, \mathsf{h}-(\mathsf{h}-1)/2)$;
10. $S \leftarrow S \cup \{\mathbf{x}_s\}$;
11. return S;

END

Figure 4: Speculative Sampling.

4.2.1 Speculative Sampling

We present a *speculative* procedure, which recursively generates samples by speculating user feedback. The *speculative* procedure may not be a practical method because it's computationally intensive. We use it as a *yardstick* to measure how well the other active-learning strategies perform. The algorithm starts by finding one most informative sample (the closest unlabeled instance to the hyperplane). It then speculates upon the two possible labels of that sample (positive and negative, respectively). Assuming the sample would be labeled as positive, *speculative* generates a hyperplane and finds a most informative sample with respect to that hyperplane. Assuming the sample would be labeled as negative, it again generates a hyperplane and finds a most informative sample with respect to that hyperplane. (At this point, it has generated three samples.) The algorithm speculates recursively, generating a binary tree of samples. Figure 4 presents the *speculative* algorithm. Steps 6 and 8 of the algorithm speculate the pool-query to be positive and negative, respectively, and recursively call the *speculative* procedure to select the next samples. The *speculative* procedure terminates after at least h samples have been generated.

4.2.2 Batch-Simple Sampling

The *batch-simple* strategy chooses h unlabeled instances closest to the separating hyperplane (between the relevant and the irrelevant instances in the feature space) to solicit user feedback. Figure 5 summarizes the algorithm. Based on the labeled pool L, the algorithm first trains a binary classifier f (step 1). The binary classifier f is then applied to the unlabeled pool U to compute each unlabeled instance's

Batch-Simple
Input: L, U, h; /* labeled set, unlabeled set, batch size
Output: S; /* Pool-queries or samples
Procedures:
• $\text{SVM}_{Train}()$ /* SVM training algorithm
• $f()$ /* Classifier learned by SVMs on L
Initialization:
• $S \leftarrow \emptyset$
BEGIN
1. $f \leftarrow \text{SVM}_{Train}(\mathsf{L})$;
2. For each $\mathbf{x}_i \in \mathsf{U}$
$\mathbf{x}_i.distance \leftarrow |f(\mathbf{x}_i)|$;
3. While ($|S| < h$)
$\mathbf{x}_s \leftarrow argmin_{\mathbf{x}_i \in \mathsf{U}}(\mathbf{x}_i.distance)$;
$S \leftarrow S \cup \{\mathbf{x}_s\}$; $\mathsf{U} \leftarrow \mathsf{U} - \{\mathbf{x}_s\}$;
4. return S;
END

Figure 5: Batch-Simple Sampling Algorithm.

distance to the separating hyperplane (step 2). The h unlabeled instances closest to the hyperplane and relatively apart from each other are chosen as the next batch of samples for conducting pool-queries.

4.2.3 Angle-Diversity Sampling

The main idea (described in Step 2 of Figure 6) of *angle-diversity* [5] is to select a collection of samples close to the classification hyperplane, and at the same time, maintain their diversity. The diversity of samples is measured by the angles between the samples. Given an example $\mathbf{x}_i$, its normal vector is equal to $\Phi(\mathbf{x}_i)$. The angle between two hyperplanes $\mathbf{h}_i$ and $\mathbf{h}_j$, corresponding to instances $\mathbf{x}_i$ and $\mathbf{x}_j$, can be written in terms of the kernel operator K:

$$|\cos(\angle(\mathbf{h}_i, \mathbf{h}_j))| = \frac{|\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)|}{\|\Phi(\mathbf{x}_i)\|\|\Phi(\mathbf{x}_j)\|} = \frac{|K(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)K(\mathbf{x}_j, \mathbf{x}_j)}}$$

The *angle-diversity* algorithm starts with an initial hyperplane $\mathbf{h}_i$ trained by the given labeled set L. Then, for each unlabeled instance $\mathbf{x}_j$, it computes its distance to the classification hyperplane $\mathbf{h}_i$. The angle between the unlabeled instance $\mathbf{x}_j$ and the current sample set S is defined as the maximal angle from instance $\mathbf{x}_i$ to any instance $\mathbf{x}_j$ in set S. This angle measures how diverse the resulting sample set S would be, if instance $\mathbf{x}_j$ were to be chosen as a sample.

Algorithm *angle-diversity* introduces parameter λ to balance two components: the distance to the classification hyperplane and the diversity of angles among samples. Incorporating the trade-off factor, the final score for the unlabeled instance $\mathbf{x}_i$ can be written as

$$\lambda * |f(\mathbf{x}_i)| + (1 - \lambda) * (\max_{\mathbf{x}_j \in S} \frac{|k(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)K(\mathbf{x}_j, \mathbf{x}_j)}}), \tag{2}$$

where function f computes the distance to the hyperplane, function K is the kernel operator, and S the training set. After that, the algorithm selects the unlabeled instance that enjoys the smallest score

Angle Diversity
Input: L, U, h, λ; /* Training set, unlabeled set, batch size, weighting parameter
Output: S; /* Sample set
Procedures:
• $\text{SVM}_{Train}()$ /* SVM training algorithm
• $f()$ /* Classifier learned by SVMs on L
• $K()$; /* SVM kernel function
Initialization:
• $S \leftarrow \emptyset$
BEGIN
1. $f \leftarrow \text{SVM}_{Train}(\mathsf{L})$;
2. While ($|S| < h$)
$\mathbf{x}_s \leftarrow argmin_{\mathbf{x}_i \in \mathsf{U}}((\lambda * |f(\mathbf{x}_i)| + (1-\lambda) * (\max_{\mathbf{x}_j \in S} \frac{|k(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i) K(\mathbf{x}_j, \mathbf{x}_j)}})$
$S \leftarrow S \cup \{\mathbf{x}_s\}$;
3. return S;
END

Figure 6: Angle Diversity Sampling.

in U as the sample. The algorithm repeats the above steps h times to select h samples. In practice, with trade-off parameter λ set at 0.5, [5] shows that the algorithm achieves good performance "on the average." We will show in Section 5.3.4 that λ can be adjusted in a concept-dependent way according to the *diversity* of the target concept.

4.2.4 Error-Reduction Sampling

Arriving from another perspective, Roy and McCallum [50] proposed an active learning algorithm that attempts to reduce the expected error on future test examples. In other words, their approach aims to reduce future generalization error. Since the class-prediction error on unseen examples cannot be known in advance, Roy and McCallum proposed a method for estimating future error. Suppose we are given a labeled set L and an unlabeled set $\mathsf{U} = \{\mathbf{x_1} \ldots \mathbf{x_n}\}$ where each $\mathbf{x}_i$ is a vector. Suppose the distribution $P(\mathbf{x})$ of the image vectors is assumed to be i.i.d.. In addition, each image $\mathbf{x}_i$ is associated with a label $y_i \in \{-1, 1\}$ according to some unknown conditional distribution $P(y|\mathbf{x})$. The classifier trained by the labeled set L can estimate an output distribution $\hat{P}_{\mathsf{L}}(y|\mathbf{x})$ for a given input $\mathbf{x}$. Then the expected error of the classifier can be written as

$$E[Error_{\hat{P}_{\mathsf{L}}}] = \int_{\mathbf{x}} Loss(P(y|\mathbf{x}), \hat{\mathbf{P}}_{\mathsf{L}}(y|\mathbf{x}))\mathbf{P}(\mathbf{x})\mathbf{dx}.$$

The function *Loss* is some loss function employed to measure the difference between the true distribution, $P(y|\mathbf{x})$, and its estimation, $\hat{P}_{\mathsf{L}}(y|\mathbf{x})$. One popular loss function is the log loss function which is defined as follows:

$$Loss(P(y|\mathbf{x}), \hat{\mathbf{P}}_{\mathsf{L}}(y|\mathbf{x})) = \sum_{y \in \{-\mathbf{1}, \mathbf{1}\}} \mathbf{P}(y|\mathbf{x})\mathbf{log}(\hat{\mathbf{P}}_{\mathsf{L}}(y|\mathbf{x})).$$

The algorithm proposed by Roy and McCallum selects a query, $\mathbf{x}^*$, that causes minimal error. The

Error Reduction

Input: L, U, h; /* Training set, unlabeled set, batch size

Output: S; /* Sample set

Procedures:

• $\text{SVM}_{Train}()$,

• $f()$, $g()$ /* Classifier learned by SVMs

• $Regression()$ /* Perform logic regression

Variables:

• Γ, Ψ; /* local training and test set

Initialization:

• $S \leftarrow \emptyset$

BEGIN

1. $f \leftarrow \text{SVM}_{Train}(\mathsf{L})$;

2. For each $\mathbf{x}_i \in \mathsf{U}$

3 $\hat{P}_{\mathsf{L}}(y|\mathbf{x}_i) \leftarrow Regression(|f(\mathbf{x}_i)|)$;

$\Psi \leftarrow \mathsf{U} - \{\mathbf{x}_\mathbf{i}\}$;

$\mathbf{x}_i.loss \leftarrow 0$;

$\Gamma \leftarrow \mathsf{L} \cup \{\mathbf{x}_\mathbf{i}, \mathbf{1}\}$; /* assign positive label

$g \leftarrow \text{SVM}_{Train}(\Gamma)$;

4. For each $\mathbf{x}_j \in \mathsf{U}$

$\hat{P}_{\Gamma}(y|\mathbf{x}) \leftarrow \mathbf{Regression}(|\mathbf{g}(\mathbf{x_j})|)$;

$\mathbf{x}_i.loss \leftarrow \mathbf{x}_i.loss + \hat{P}_{\mathsf{L}}(y|\mathbf{x}_i) * \sum_{y \in \{-1,1\}} \hat{P}_{\Gamma}(y|\mathbf{x})\mathbf{log}(\hat{\mathbf{P}}_{\mathbf{\Gamma}}(y|\mathbf{x}))$;

$\Gamma \leftarrow \mathsf{L} \cup \{\mathbf{x}_\mathbf{i}, \mathbf{-1}\}$; /* assign negative label

$g \leftarrow \text{SVM}_{Train}(\Gamma)$;

5. For each $\mathbf{x}_j \in \mathsf{U}$

$\hat{P}_{\Gamma}(y|\mathbf{x}) \leftarrow \mathbf{Regression}(|\mathbf{g}(\mathbf{x_j})|)$;

$\mathbf{x}_i.loss \leftarrow \mathbf{x}_i.loss + (1 - \hat{P}_{\mathsf{L}}(y|\mathbf{x}_i)) * \sum_{y \in \{-1,1\}} \hat{P}_{\Gamma}(y|\mathbf{x})\mathbf{log}(\hat{\mathbf{P}}_{\mathbf{\Gamma}}(y|\mathbf{x}))$;

6. While ($|S| < h$)

$\mathbf{x}_s \leftarrow argmin_{\mathbf{x}_i \in \mathsf{U}}(\mathbf{x}_i.loss)$; $S \leftarrow S \cup \{\mathbf{x}_s\}$;

7. return S;

END

Figure 7: Error Reduction Sampling.

algorithm includes $\mathbf{x}^*$ in its sample set if $E[Error_{\hat{P}_{\mathsf{L}\cup\{\mathbf{x}^*\}}}]$ is smaller than $E[Error_{\hat{P}_{\mathsf{L}\cup\{\mathbf{x}\}}}]$ for any other instance $\mathbf{x}$. Figure 7 summarizes the algorithm. Given the training pool L, the algorithm first computes the current classifier's posterior $\hat{P}_{\mathsf{L}}(y|\mathbf{x})$ in step 3. For each unlabeled image $\mathbf{x}$ with each possible label y, the algorithm then adds the pair $(\mathbf{x}, y)$ to the training set, re-trains the classifier with the enlarged training set, and computes the expected log loss. Steps 4 and 5 of the algorithm compute the expected log loss by adding $(\mathbf{x}, y)$ to the training data.

5 Concept-Dependent Learning

Ideally, concept learning should be done in a concept-dependent manner. For a simple concept, we can employ, e.g., algorithm *angle diversity* or *batch simple* to learn the concept. For a complex concept, we must make proper adjustments in the learning algorithm. We define concept complexity as the level of difficulty in learning a target concept. To model concept complexity, we propose three quantitative measures [34]: *scarcity*, *isolation*, and *diversity*.

In the remainder of this section, we will first define our measures for quantifying query complexity. We will then discuss major limitations of current active learning algorithms. Finally, we propose detailed algorithms of the concept-dependent component of $\mathsf{SVM}^{CD}_{Active}$, which uses keywords to guide learning in the perceptual space, to alleviate the limitations.

5.1 Concept Complexity

Before we can compute the concept complexity, we need to know the user's target concept, which is unavailable beforehand. Fortunately, we have a rough description of an image's semantic content from its keyword-annotation (even though the annotation quality might not be perfect). We treat each keyword as a concept, and precompute *scarcity*, *isolation*, and *diversity* to characterize concept complexity in advance.

5.1.1 Scarcity

Scarcity measures how well-represented a concept is in the retrieval system. We use *hit-rate*, defined as the percentage of data matching the concept, to indicate scarcity. As we assume that each keyword is equivalent to a concept, the hit-rate of a keyword is the number of images being annotated with that keyword. This parameter is dataset-dependent; while a concept such as *photography* is very general and may produce a high hit-rate, other general concepts such as *landmark* (209 images in our 300K-dataset) may be scarce simply because the system does not contain many matching images. Similarly, a very specific concept such as *laboratory coat* (1,497 images) may have a high hit-rate solely because the system has many such images.

5.1.2 Isolation

Isolation characterizes a concept's degree of separation from the other concepts. We measure two types of isolation: *input space* isolation and *keyword* isolation. The input space isolation is considered low (or poor) when the concept is commingled with others in the perceptual space. When the keywords used to describe a concept has several meanings or senses, the keyword isolation is considered poor; very precise keywords provide good isolation. An example of a poorly-isolated keyword is "feline," which is often used to annotate images of tigers, lions and domestic cats. If the user seeds a query with "feline," it is difficult to tell which sense of the word the user has in mind. A well-isolated seed keyword like "Eiffel Tower" has less ambiguity; we can be quite sure that the user is thinking about the famous Parisian landmark. We estimate isolation as follows:

1. *Input Space Isolation.* We characterize the isolation of a concept in the perceptual input space with:

$$I_s(T,\sigma) = \frac{1}{N_T} \sum_{\mathbf{x_i} \in \mathbf{T}} \frac{N(T)}{N(\sigma)}. \tag{3}$$

For each instance $\mathbf{x_i}$ that belongs to the target concept T, $N(\sigma)$ is the number of instances that are within an L_1 distance of σ from $\mathbf{x_i}$, and $N(T)$ is the number such instances that are from T. N_T is the total number of instances from T in the entire dataset. In essence, $I_s(T, \sigma)$ gives the percentage of nearest neighbors (NNs) within a distance of σ that are from the target concept. The higher the value of I_s, the more isolated is the target concept. A low value indicates that the target concept is commingled with others in the input space. Since I_s is only an estimation, we use σ values ranging from 1 to 9 in increments of one.

2. *Mining Keyword Associations for Keyword Isolation.* To measure keyword isolation, we employ association-rules mining [1] to model the co-occurrences of keywords. Given a query keyword w_q, we find the set of images I that are annotated with w_q. Let W be the set of keywords, besides w_q, that are used to annotate the images in I. For each 2-itemset $\{w_q, w_i\}$, where $w_i \in W$, we compute the confidence (C_{qi}) of the rule $w_q \Rightarrow w_i$. We define isolation of w_q with respect to w_i as

$$I_k(w_q, w_i) = C_{qi} \times (1 - C_{qi}). \tag{4}$$

When C_{qi} is close to 0.5, I_k is at its maximum, which implies that the rule is highly ambiguous, and that w_q is poorly isolated from w_i. Let us consider two query keywords "fruit" and "apple." Suppose the confidence value for the rule $fruit \Rightarrow apple$ is 0.5, and for $apple \Rightarrow fruit$ it is 0.7. Using Equation 4, we get $I_k = 0.25$ for the first rule, and $I_k = 0.21$ for the second rule. We say that the keyword "fruit" is poorly isolated from "apple", whereas "apple" is well isolated from "fruit." Although "fruit" is a more general word than "apple" according to rules of linguistics, this is not necessarily true in the annotations of the image-set. Therefore, employing a general thesaurus such as Wordnet may be counter-productive for determining keyword isolation. The isolation value must be computed in a dataset-dependent fashion. The causes of poor keyword isolation could be due to the words being synonyms, or that some images contain semantics or objects described by the words. To quantify isolation, we need not discern the reason of co-occurrences. When some words often appear together with the query keyword in I, it indicates that the query keyword is ambiguous. For instance, the word "apple" in our dataset is often associated with either "computer" or "fruit." Therefore, $\mathsf{SVM}^{CD}_{Active}$ needs to determine whether it is "computer" or "fruit" that is more relevant to an "apple" query. With association-rules mining, the derived keyword isolation is dateset-dependent. If our dataset only contains images of "fruit apple", the concept "apple" will have good semantic isolation. Thus, we only need to determine the semantic of the word when it appears in more than one 2-itemsets.

We preview some empirical results to show that concept isolation can affect retrieval accuracy. Our dataset consists of 300K images that are annotated with up to 40K annotation keywords altogether. We choose 6K representative keywords—words that are not too common or too rare—to represent the semantic classes present in this dataset. Figure 8 plots the average isolation value (I_s in Equation 3) at $\sigma = 1 \ldots 9$ for the 6K concepts. Based on the I_s values, we divide the concepts into three categories of isolation: well-, moderately- and poorly-isolated. We notice that well-isolated concepts generally have higher I_s values at smaller σ, implying that similar NNs are nearby; poorly-isolated concepts generally have very low I_s values.

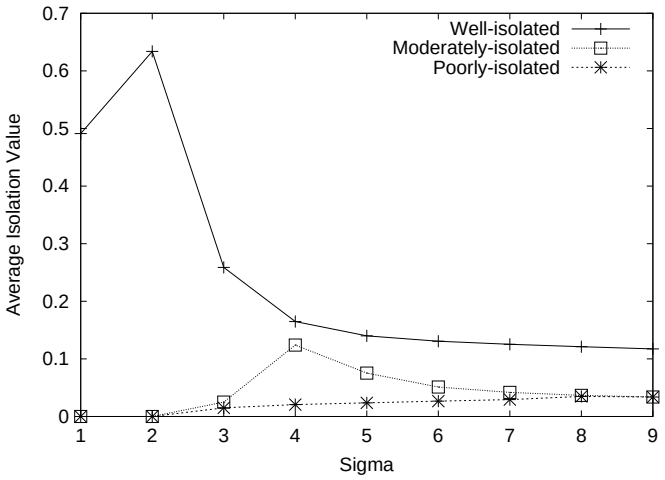


Figure 8: Average Isolation Values.

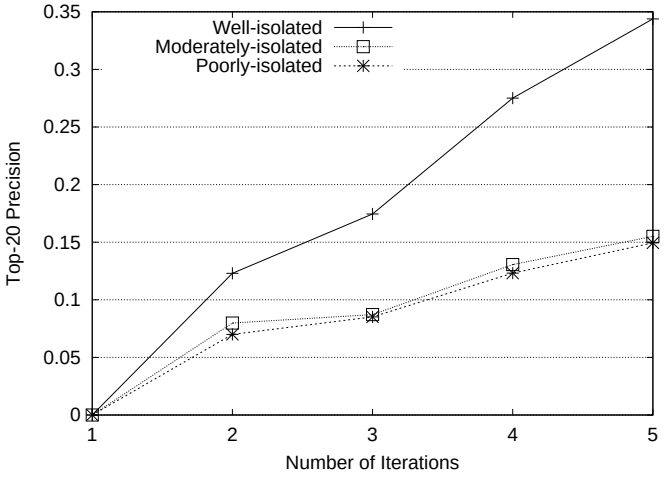


Figure 9: Precision for Various Input-space Isolation Types.

Next, we use the *angle diversity* algorithm with $\lambda = 0.5$ to study the effect of isolation on retrieval accuracy. To circumvent the scarcity problem, we use the keywords to seed each query with a positive and negative example. Relevance feedback is then conducted purely on perceptual features. We are not able to use keywords to bound the search as we are using keywords to indicate the query concept. This experimental setup is different from that in Section 6 where real users interact with the retrieval system to conduct relevance feedback. In Figure 9, we plot the average top-20 precision rates using 5 iterations of relevance feedback for the three types of isolation. As expected, the plot shows that concepts with good isolation achieve higher precision rates.

5.1.3 Diversity

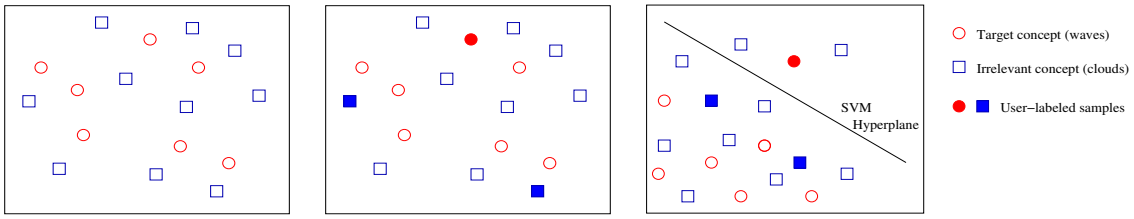
The diversity of a concept is characterized by the way relevant images are distributed in the input space; it is an indication on how learnable a concept is. A diverse concept has relevant images scattered all over the input space. For example, the *flowers* concept, which encompasses flowers of different colors and types, is more diverse than the *red roses* concept. Instead of concentrating the sampling effort on a few small subspaces, we want to explore more subregions of the input space and select diverse samples to present to the user. As diversity is related to the spread of the relevant images in the input space, we can characterize concept diversity using the spatial variance of images belonging to a concept in the perceptual input space [19]. Alternatively, we can make use of the TSVQ clusters from the indexer of our dataset [36]. Since clustering will group instances based on how they are scattered in the input space, we characterize diversity using the distance between centroids of the clusters that contain instances from the target concept.

5.2 Limitations of Active Learning

When the target concept instances are scarce and not well isolated, active learning can be ineffective for locating relevant images.

The first limitation is related to scarcity—the availability of images relevant to the concept to be learned. Most active learning algorithms require at least one positive (relevant) and one negative (irrelevant) instance to begin the learning process. A common approach is to randomly sample images from the input space. If none of the samples are labeled as positive, they are all treated as negatives, which will be used to prune the sample space by applying an algorithm like [10]. The seed-image sampling continues until we obtain a positive image. While this process is viable, it may not be effective if the matching images are scarce. When the percentage of matching images is low, even if we can winnow irrelevant images from the unlabeled pool, the probability of finding a positive image can still be quite low. For instance, if a concept has 5% matching images, the probability is about 68% that at least one of the 20 randomly selected images is positive. But when the concept is as scarce as 1%, the probability declines to 20%. One can easily find scenarios where the number of concept-matching images is much lower than 1%. Under such conditions, active learning can take many iterations just to find one relevant image.

The second limitation—concept isolation—further complicates the problem. Figure 10 illustrates this problem. The squares and circles represent images from two different concepts, which overlap each other in the input space (Figure 10(a)). Suppose three images presented to the user have been labeled as follows: two squares are labeled as "irrelevant" (filled squares), and one circle is labeled "relevant" (filled circle in Figure 10(b)). The circles around the filled squared can easily be mistakenly inferred as squares, and the squares around the filled circles as circles. Figure 10(c) shows that after using the kernel trick to project the instances into the feature space, several instances lie on the wrong side of



(a) Input Space (b) User-provided Labels (c) Projected Feature Space

Figure 10: Example illustrating how poor concept isolation can hurt target concept learning.

the separating hyperplane. This is because the labeled images influence their neighboring images in the input space to be classified into the same category and we have two classes mix together. As a result, it is difficult to separate these two classes apart through any learning algorithm (unless we have substantially larger amount of labeled instances, which defeats the purpose of active learning).

5.3 Concept-Dependent Active Learning Algorithms

$\mathsf{SVM}^{CD}_{Active}$ consists of a state-transition table and three algorithms, *disambiguate keywords* (Figure 12) *disambiguate input-space* (Figure 13), and *angle diversity* (Figure 6). First, $\mathsf{SVM}^{CD}_{Active}$ addresses the scarcity problem by using keywords to seed a query. Thus, we eliminate the need to search the entire dataset for a positive image. The user can type in a keyword (or keywords) to describe the target concept. Images that are annotated with that keyword are added to the initial unlabeled pool U. If the number of images with matching keywords is small, we can perform query expansion[3] using a thesaurus to obtain related words (synonyms) that have matching images in our dataset. For each related semantic, we select matching images and add them to U.

Using U to conduct learning, the *hit-rate* is much improved compared to using the entire dataset. At this juncture, the state of learnability can be in one of the four states depicted in Figure 11. The rows show that the specified keyword(s) may enjoy good or suffer from poor isolation. The columns show that the query concept may be well or poorly isolated in the input space formed by the perceptual features. The goal of $\mathsf{SVM}^{CD}_{Active}$ is to move the learnability state of the target concept to the ideal state A, where both keywords and perceptual features enjoy good isolation.

5.3.1 State C - Keyword Disambiguation

In state C, the keyword isolation is poor (due to aliasing). Thus, we first need to disambiguate the keywords by presenting images of different semantics to the user. Once the semantic is understood, the learnability makes a transition to state A.

[3]Query expansion is a vital component of any retrieval systems and it remains a challenging research area. This component is not deployed in our system yet and is part of our ongoing research.

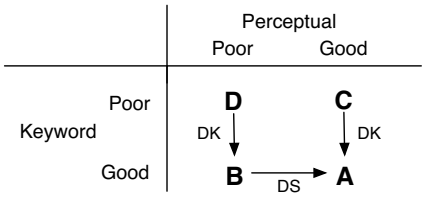


Figure 11: Four Learnability States.

Disambiguate Keywords
Input: w, U, n; /* query word, unlabeled pool, sample #
H; /* keyword association table
Output: U; /* unlabebed pool
Procedures:
• $findAssociations(w, H)$ /* find w's 2-itemsets in H
• $relevanceFeedback(S)$ /* user labels S
Initialization:
• $S \leftarrow \emptyset$
BEGIN
1. $A \leftarrow findAssociations(w, H)$;
2. **while** $(n \neq 0)$
$\mathbf{s} \leftarrow Random(U)$;
if $(\mathbf{s}.W \cap A \neq \emptyset)$ /* Check annotation set W of $\mathbf{s}$
$S \leftarrow S \cup \{\mathbf{s}\}$;
$n \leftarrow n - 1$;
3. $relevanceFeedback(S)$;
4. **for each** $\mathbf{s}^- \in S$
for each $\mathbf{u} \in U$
if $(\mathbf{s}^-.W \cap u.W \neq \emptyset)$
$U \leftarrow U - \{\mathbf{u}\}$;
5. **return** U;
END

Figure 12: Disambiguate Keywords Algorithm.

Algorithm *disambiguate keywords* (DK) depicted in Figure 12 helps state C to arrive at A. The algorithm first performs a lookup in the association table H for the query keyword w. If 2-itemsets are found, the algorithm randomly selects in U images that are annotated by the co-occurred word set A. The resulting sample set S is presented to the user to solicit feedback. Once feedback has been received, $\mathsf{SVM}^{CD}_{Active}$ removes images labeled with the words associated with the irrelevant images from U.

5.3.2 State B - Input-space Disambiguation

State B has poor perceptual isolation but good keyword isolation. If the number of matching images yielded by the good keyword is small, we can perform query expansion to enlarge the unlabeled sample pool. For these poorly-isolated concepts, we attempt to improve the perceptual isolation of the sample pool through input-space disambiguation. $\mathsf{SVM}^{CD}_{Active}$ employs *disambiguate input-space* (DS) in Figure 13 to achieve this improvement. The algorithm removes from U the instances that can interfere

Disambiguate Input-Space
Input: L, U w; /* labeled, unlabeled sample pool, keyword
Output: U;
Procedures:
• $query\_expansion(w)$;
• $generateWordlist(L, type)$;
Initialization:
• $P, N \leftarrow \emptyset$
BEGIN
1. $U \leftarrow U + query\_expansion(w)$;
2. $P \leftarrow generateWordlist(L, \text{Positive})$;
3. $N \leftarrow generateWordlist(L, \text{Negative})$;
4. **for each u** $\in U$
  if $(|\mathbf{u}.W \cap N| > |\mathbf{u}.W \cap P|)$
    $U \leftarrow U - \{\mathbf{u}\}$;
5. **return** U;
END

Figure 13: Dismbiguate Input-Space Algorithm.

with the learnability of the target concept. Let us revisit Figure 10. Rather than learning the labels of both the square and circle classes, DS removes the instances of the non-target class from the sample pool.

Given the current labeled pool of images, DS first generates two word lists: P from positive-labeled images and N from negative-labeled images. In the third step, DS checks the annotation of each image in the unlabeled pool for matches to words in P and N. If there are more negative keyword matches, $\mathsf{SVM}^{CD}_{Active}$ removes that image from U. Once perceptual isolation is enhanced, state B makes the transition to A.

5.3.3 State D — Keyword & Space Disambiguation

State D suffers from both poor semantic and input-space isolation. To remedy the problem, $\mathsf{SVM}^{CD}_{Active}$ first improves the keyword isolation using the same DK strategy as state C. Thereafter, the state is moved to state B, and the perceptual isolation is improved by applying state B's method for disambiguation in the input space. Figure 11 shows the path that D takes to reach the ideal state A.

5.3.4 State A — Adapt to Diversity

Knowledge of the concept diversity can guide the learner during its exploration of the input space for potential positive samples. For example, if a concept is very diverse, like *flowers*, the learner may need to be more explorative and search for flowers of all colors. For such concepts, we make our learner more explorative by changing the parameter λ used in our classification score function in the *angle diversity* algorithm:

$$\lambda * |f(\mathbf{x}_i)| + (1 - \lambda) * (\max_{\mathbf{x}_j \in S} \frac{|k(\mathbf{x}_i, \mathbf{x}_j)|}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i) K(\mathbf{x}_j, \mathbf{x}_j)}}).$$

If the concept diversity is low, we keep λ at its original value of 0.5. For diverse concepts, we reduce λ, resulting in more weight being assigned to the angle diversity during sample selections.

6 Experiments and Discussion

We conducted experiments to answer the following five questions:

1. How does active learning perform, compared to passive learning in terms of retrieval accuracy? (Section 6.2)
2. Can active learning outperform traditional relevance feedback schemes? (Section 6.3)
3. Which sampling strategies, *batch simple*, *speculative*, *angle diversity*, or *error correction*, perform the best for $\mathsf{SVM}^{CD}_{Active}$, and why? (Section 6.4)
4. For concepts with differing degrees of isolation, can the multimodal approach of $\mathsf{SVM}^{CD}_{Active}$ improve the retrieval accuracy? (Section 6.5)
5. Can the *angle diversity* algorithm be tuned to accommodate highly diverse concepts? (Section 6.6)

6.1 Testbed and Setup

For empirical evaluation of $\mathsf{SVM}^{CD}_{Active}$, we used five image datasets: a four-category, a ten-category, a fifteen-category, a 107-category, and a 300K real-world image dataset.

- *Four-category* set. The 602 Corel-CD images in this dataset belong to four categories — *architecture*, *flowers*, *landscape*, and *people*. Each category consisting of 100 to 150 images.
- *Ten-category* set. The $1,277$ 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 (i.e., to reduce hit rate and decrease isolation). 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 $1,920$ 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.
- 107-*category* set. This set consists of nearly $50,000$ images that we collected from Corel Image CDs. The categories are documented in [9].

- *Large* set. A 300K-image dataset with images from a stock-photo company.

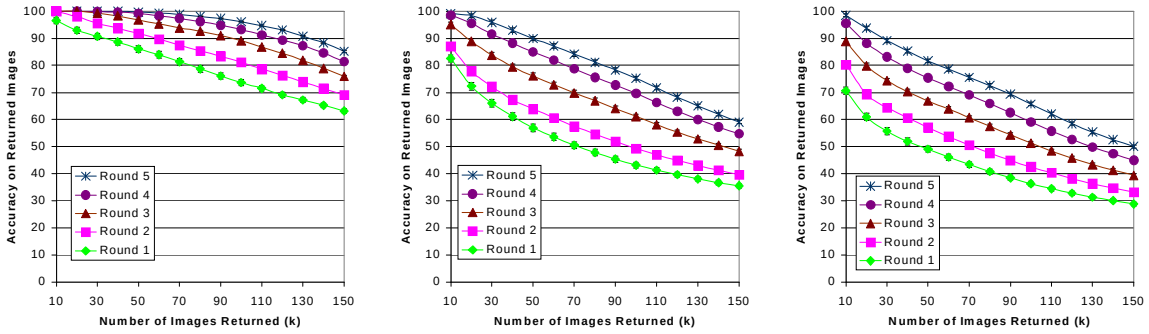
Each image is described by 144 perceptual features (108 from color and 36 from texture). (These features have been thoroughly tested and verified to be quite effective. Since the focus of this paper is not on feature extraction, please consult [58] for details.) In the large image dataset, each image is annotated with up to 50 keywords; 40,000 keywords are used in the entire dataset. For $\mathsf{SVM}^{CD}_{Active}$, we used a Laplacian RBF kernel $K(\mathbf{u}, \mathbf{v}) = (\mathbf{e}^{-\gamma \sum_{\mathbf{i}} |\mathbf{u_i} - \mathbf{v_i}|})$, with the γ set at 0.001. For the parameter λ used in the *angle diversity* algorithm, we set its default value at 0.5.

To obtain an objective measure of performance, we assumed that a query concept was an image category. The $\mathsf{SVM}^{CD}_{Active}$ learner has no prior knowledge about image categories[4]. The goal of $\mathsf{SVM}^{CD}_{Active}$ is to learn a given concept through a relevance feedback process. In this process, at each feedback round $\mathsf{SVM}^{CD}_{Active}$ selects sixteen or 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, $\mathsf{SVM}^{CD}_{Active}$ then retrieves the top-k most relevant images 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 few screens 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.

6.2 Active Learning vs. Passive

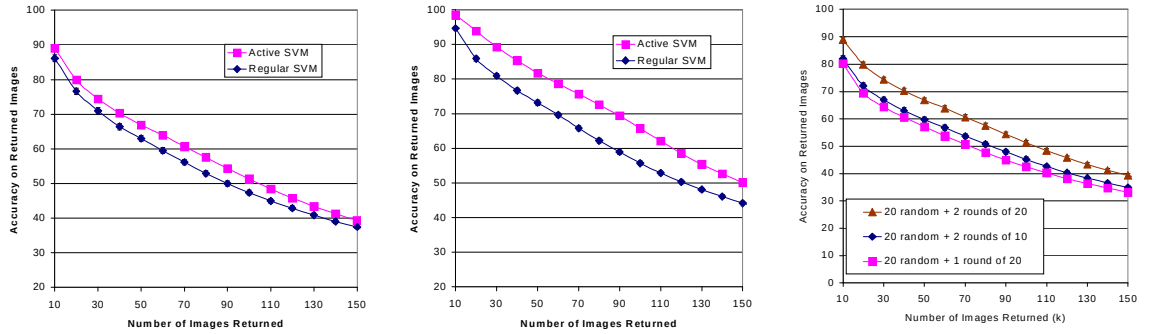
In this first experiment, we examined the gain of active learning over passive learning, and did not activate the concept-dependent component of $\mathsf{SVM}^{CD}_{Active}$. Figures 14(a-c) show the average top-k accuracy for the three small datasets. We considered the performance of $\mathsf{SVM}^{CD}_{Active}$ after each round of relevance feedback. The graphs indicate that performance clearly increases after each round. Also, the $\mathsf{SVM}^{CD}_{Active}$ algorithm's performance degrades gracefully when the concept complexity 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 $\mathsf{SVM}^{CD}_{Active}$ 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. (For these three datasets, all sampling strategies

[4]Unlike some recently developed systems [62] that contain a semantic layer between image features and queries to assist query refinement, our system does not have an explicit semantic layer. We argue that having a hard-coded semantic 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 14: (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.



(a) (b) (c)

Figure 15: (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.

work equally well. When the concept complexity further increases, we will see shortly in Section 6.4 that the *angle-diversity* method outperforms the others.)

We examined the effect that the active querying method had on performance. Figures 15(a) and (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 (a 5% to 10% gain after five iterations) by using the active method.

We used 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 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 15(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 not surprising since the active learner has more control and freedom to adapt when asking 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 often negligible since users can pick out relevant images easily. Furthermore, there is virtually no additional computational cost in calculating the 20 images to query over the 10 images to query. (The size of the training data is very small, under or around 100 in five iterations.) Thus, for this particular task, we believe that it is worthwhile to display around 20 images per screen and limit the number of querying rounds, rather than display fewer images per screen and use many more querying rounds.

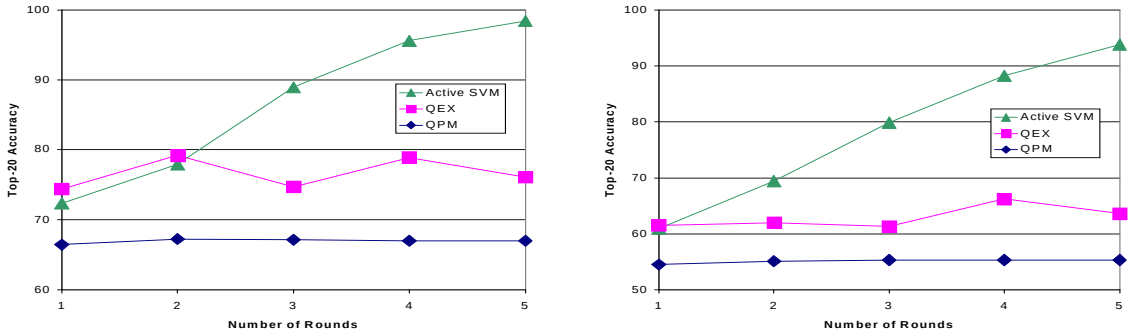
6.3 Against Traditional Relevance Feedback Schemes

We compared $\mathsf{SVM}^{CD}_{Active}$ 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 $\mathsf{SVM}^{CD}_{Active}$ 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 16(a) shows that $\mathsf{SVM}^{CD}_{Active}$ 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 16(b) shows that $\mathsf{SVM}^{CD}_{Active}$ outperforms the others by even wider margins. $\mathsf{SVM}^{CD}_{Active}$ 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 $\mathsf{SVM}^{CD}_{Active}$ 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 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.

6.4 Sampling Method Evaluation

This experiment was conducted to examine the performance of five sampling algorithms: *random*, *simple active*, *speculative*, *angle diversity*, and *error reduction*. As we mentioned in Section 6.2, when the

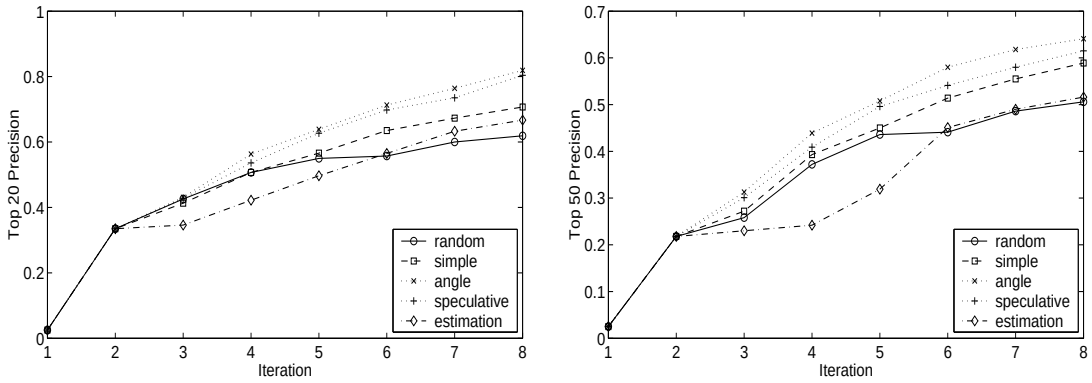


(a) (b)

Figure 16: (a) Average top-k accuracy over the ten-category dataset. (b) Average top-k accuracy over the fifteen-category dataset.

concept complexity is low, all sampling methods perform about the same. To conduct this experiment, we used the 107-category dataset.

The first strategy selects random images from the dataset as samples for user feedback. The rest of the sample algorithms have been described in Section 4.2. We conducted experiments to compare these five sampling strategies in terms of two factors: the top-k retrieval accuracy and execution times. In the experiments, we tested the sampling algorithms by fixing the number of sample images per-round at sixteen.



(a) Top-20 accuracy (b) Top-50 accuracy

Figure 17: Comparison of Sampling Strategies.

Figures 17(a) and 17(b) report the results for the 107-category dataset. For this dataset, it is too expensive to scan the entire dataset to perform sample selection and retrieval. Therefore, we conducted sampling and retrieval *approximately* through an indexing structure [36]. The figures show that the *angle diversity* algorithm performs the best among all active-learning algorithms. The *angle diversity* algorithm performs as well, and even better in some interactions, as the *speculative* algorithm, which is supposed to achieve nearly optimal performance. This result confirms that the samples should be

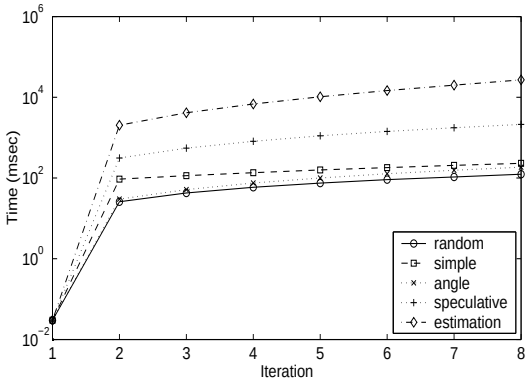


Figure 18: Execution time on large dataset.

diverse, as well as semantically uncertain (near the hyperplane).

Next, Figure 18 evaluates the execution time (in milli-seconds) taken by the five algorithms. The figure shows that the fastest sampling algorithm is the *random* method, followed by *angle diversity*, *simple active*, *speculative*, and then *error-reduction*. Thus, *angle diversity* algorithm is the ideal choice in terms of both effectiveness and efficiency.

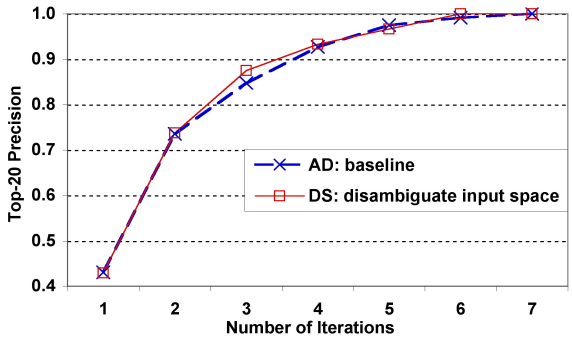
6.5 Concept-Dependent Learning

This experiment was designed to assess the effectiveness of the concept-dependent approach to image retrieval. The experiments were conducted on a dataset comprising 300K images of a stock-photo company's.

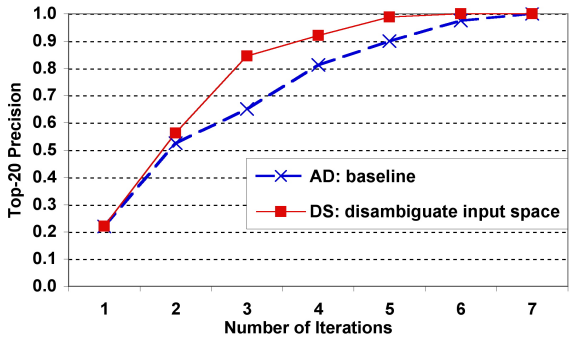
We evaluated retrieval accuracy (in terms of precision) for concepts belonging to one of the four possible learnability states depicted in Figure 11. The representative concepts we queried, based on their learnability states, are shown in Table 1. The *angle diversity* (AD) sampling strategy is used as the baseline for comparing the performance of the other algorithms used to modify the sample pool, namely *disambiguate input-space* (DS) and *disambiguate keywords* (DK).

State	A	B	C	D
	castle	frog	apple	bug
	crabs	lion	bird	city
	crayfish	Paris	cat	dessert
	fireworks	river	china	fruit
	outer space	tiger	fish	mountain

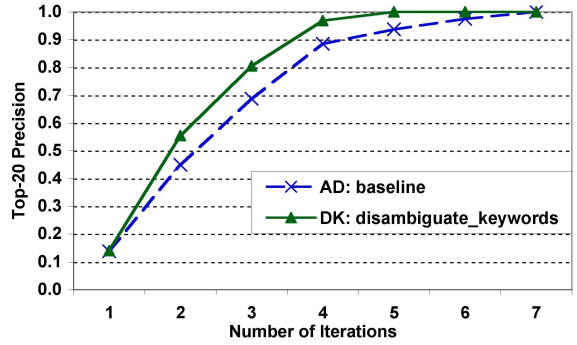
Table 1: Queries for various states of learnability.



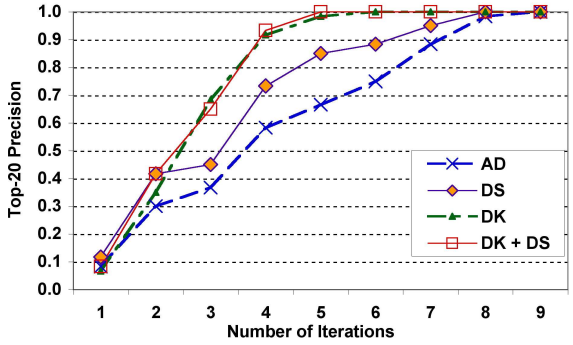
(a) Good keyword and perceptual



(b) Good keyword, poor perceptual



(c) Poor keyword, good perceptual



(d) Poor keyword and perceptual

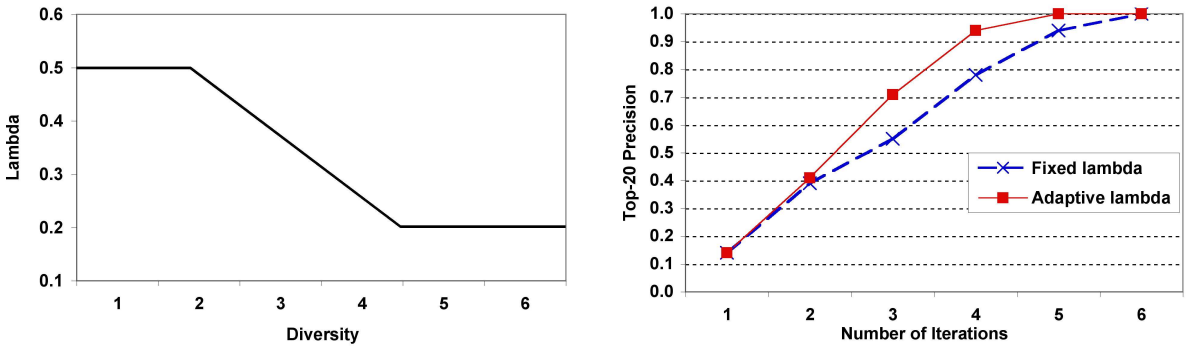
Figure 19: Results for the four states of keyword and perceptual isolation.

6.5.1 State B (Figure 19(b))

The concepts belonging to state B have good keyword isolation but poor perceptual isolation. We used the DS algorithm to weed the sample pool of images annotated by negative keywords. The figure shows the retrieval performance of the DS algorithm and the baseline's. In the 5^{th} iteration, DS gives a 99% precision rate compared to 90% for the baseline. On average, we can improve the precision rate by 10%. The results show that DS is effective for increasing the concept's perceptual isolation.

6.5.2 State C (Figure 19(c))

In order to resolve the aliasing problem for concepts with poor keyword isolation, we applied the DK algorithm and compared its retrieval performance with the baseline AD's. We notice that the difference in precision rate after the first round of relevance feedback is 30% for AD and 40% for DK. The higher precision rate for DK persists through subsequent iterations as well. By disambiguating keywords in the initial sample pool, DK is able to provide better retrieval results.



(a) Adaptive Lambda

(b) Results for Diverse Concepts

Figure 20: Concept diversity.

6.5.3 State D (Figure 19(d))

Concepts falling into this state are the most problematic. They suffer from both poor keyword and poor perceptual isolation. The figure plots the retrieval results for four algorithms: the baseline, DS, DK, and $DK + DS$. Using only DS to improve perceptual isolation, the precision rate is on average 10% higher than the baseline. Using only DK to improve the keyword isolation, the average precision increase is 27%. When both DS and DK are used, the precision rate improves a couple of more percentage points over the case when only DK is used. These results suggest that DK not only helps to disambiguate keywords, but improves the perceptual isolation as well. This is not surprising, because the high-frequency keywords that cause the aliasing problem often also describe concepts visually similar to the target concept. Hence by labeling images from those other concepts as "negative", the user can simultaneously perform disambiguation in the perceptual space.

6.5.4 State A (Figure 19(a))

Since the keyword isolation is good in this state, we evaluated the effectiveness of DS for further improving the already-good perceptual isolation. The plot shows the top-20 precision rate for up to 7 iterations of relevance feedback. We can observe that using negative keywords to remove samples makes little difference to the retrieval results. We conclude that such concepts are ideal for active learning and hence require no further assistance from keywords besides seeding the query.

6.6 Concept Diversity Evaluation

For highly-diverse concepts, we reduced the value of λ in Equation 2 so more weight is given to the angle diversity during sample selections. Figure 20(a) shows how we adapt λ to the concept diversity

on the 300K large dataset. For concepts with low diversity values (≤ 1), we set $\lambda = 0.5$. For more diverse concepts, we reduce λ as diversity increases. However, we cannot reduce λ indefinitely, and ignore usefulness of samples near the hyperplane. Therefore, when the diversity value is greater than 3.5, we set $\lambda = 0.2$. We plotted the precision results for queries with diverse concepts in Figure 20(b). The dashed curve plots the precision results using λ fixed at 0.5. The solid curve is for the case where λ adapts to the diversity.

At higher iterations, using an adaptive λ improves the precision rate by about 10%. The improvement is not significant for the 2^{nd} iteration since the learner still has insufficient labeled images to construct a good classifier. A qualitative observation reported by users is that when λ is adaptive, the variety of images is significantly better, in both the sample pool and the results set. Thus, our approach of adjusting λ according to the concept's diversity proves to be both simple and effective.

6.7 Evaluation Summary

Our experiments have provided us information to answer the five questions posed in the beginning of the section.

1. Active learning outperforms passive learning methods. The gain of active learning becomes more significant when complexity increases.

2. Active learning outperforms traditional relevance feedback schemes by significant margins. This is hardly a surprise, since active learning maximizes information gain when it selects a sample to perform pool-query.

3. The *angle-diversity* sampling method, which strikes a good balance between uncertainty and diversity, works most effectively and efficiently among the sampling methods.

4. When concept complexity increases beyond certain level (e.g., when a target concept is scarce or poorly isolated), the effectiveness of active learning can suffer from severe degradation. We have shown that by taking advantage of keyword profiling (similar to the way that all relational databases perform query optimization based on some profiling techniques), $\mathsf{SVM}^{CD}_{Active}$ can perform concept-dependent active learning by disambiguating either keyword semantics or perceptual features. Improving concept learnability assures the algorithm's scalability with respect to concept complexity.

5. Concept-dependent active learning can also adapt its *diversity* parameter to behave exploratively when the target concept is scattered in the feature space.

7 Conclusions

We have demonstrated that active learning with support vector machines can provide a powerful tool for searching image databases, outperforming key traditional query refinement schemes. $\mathsf{SVM}^{CD}_{Active}$ 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 [62], it does not require an explicit semantic 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 $\mathsf{SVM}^{CD}_{Active}$ can provide considerable gains over other systems.

We have also proposed a multimodal, concept-dependent active learning scheme, which combines keywords with images' perceptual features in a synergistic way to perform image retrieval. In contrast to traditional active learning methods, $\mathsf{SVM}^{CD}_{Active}$ adjusts its learning process based on concept complexity: it meticulously constructs the sample pool in order to ameliorate the query concept's hit-rate and isolation, and enhances the learnability of the query concept by adapting the sampling strategy to the concept's diversity.

8 Acknowledgements

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

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD*, 1993.

[2] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1998.

[3] L. Breiman. Bagging predicators. *Machine Learning*, pages 123–140, 1996.

[4] L. Breiman. Arcing classifiers. *The Annals of Statistics*, pages 801–849, 1998.

[5] K. Brinker. Incorporating diversity in active learning with support vector machines. *Proceedings of the Twentieth International Conference on Machine Learning*, pages 59–66, August 2003.

[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. 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.

[10] E. Chang and B. Li. Mega — the maximizing expected generalization algorithm for learning complex query concepts. *ACM Transaction on Information Systems*, December 2003.

[11] E. Y. Chang, B. Li, G. Wu, and K.-S. Goh. Statistical learning for effective visual information retrieval (invited paper). *IEEE International Conference on Image Processing (ICIP)*, pages 609–612, 2003.

[12] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4, 1996.

[13] 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.

[14] 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.

[15] 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.

[16] 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*, 2000.

[17] 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.

[18] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artifical Intelligence Research*, 2, 1995.

[19] R. Duda, P. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2 edition, 2001.

[20] 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.

[21] 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.

[22] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the Query by Committee algorithm. *Machine Learning*, 28:133–168, 1997.

[23] P. Gosselin and M. Cord. Retin al: An active learning strategy for image category retrieval. *IEEE ICIP*, 2004.

[24] 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.

[25] A. Gupta and R. Jain. Visual information retrieval. *Comm. of the ACM*, 40(5):69–79, 1997.

[26] 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.

[27] 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.

[28] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. *VLDB*, 1998.

[29] T. Jaakkola and H. Siegelmann. Active information retrieval. *Advances in Neural Information processing systems 14*, 2001.

[30] G. James and T. Hastie. *Error Coding and PaCT's*. 1997.

[31] T. Joachims. Text categorization with support vector machines. In *Proceedings of the European Conference on Machine Learning*. Springer-Verlag, 1998.

[32] K. S. Jones and P. W. (Editors). *Readings in Information Retrieval*. Morgan Kaufman, July 1997.

[33] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[34] W.-C. Lai, K. Goh, and E. Y. Chang. On scalability of active learning for formulating query concepts. *Workshop on Computer Vision Meets Databases (CVDB) in cooperation with ACM International Conference on Management of Data (SIGMOD)*, pages 11–18, 2004.

[35] 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.

[36] C. Li, E. Chang, H. Garcia-Molina, and G. Wilderhold. Clindex: Approximate similarity queries in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(4), July 2002.

[37] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Mach. Learn.*, 54(2):125–152, 2004.

[38] 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.

[39] 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.
[40] T. Mitchell. Generalization as search. *Artificial Intelligence*, 28:203–226, 1982.
[41] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
[42] M. Moreira and E. Mayoraz. Improving pairwise coupling classification with error correcting classifiers. *Proceedings of the Tenth European Conference on Machine Learning*, April 1998.
[43] H. Nguyen and A. W. Smeulders. Active learning using pre-clustering. *Proc. of the 21th Inter. Conf. on Machine Learning*, 2004.
[44] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang. Supporting similarity queries in mars. *Proc. of ACM Conf. on Multimedia*, 1997.
[45] 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.
[46] 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.
[47] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the International Conference on Computer Vision*, 1998.
[48] K. Porkaew, K. Chakrabarti, and S. Mehrotra. Query refinement for multimedia similarity retrieval in mars. *Proceedings of ACM Multimedia*, November 1999.
[49] K. Porkaew, S. Mehrota, and M. Ortega. Query reformulation for content based multimedia retrieval in mars. *ICMCS*, pages 747–751, 1999.
[50] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 441–448, August 2001.
[51] 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.
[52] 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.
[53] 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.
[54] 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.
[55] 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.
[56] J. Smith and S.-F. Chang. Tools and techniques for color image retrieval. *Volume 2670 of SPIE Proceedings*, pages 1630–1639, 1996.
[57] J. R. Smith and S.-F. Chang. Visualseek: A fully automated content-based image query system. *ACM Multimedia Conference*, 1996.
[58] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *Proc. of ACM Int. Conf. on Multimedia*, pages 107–118, October 2001.
[59] 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.
[60] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag, 1982.
[61] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
[62] J. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *ACM Multimedia Conference*, 2000.
[63] 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.
[64] H. Wu, H. Lu, and S. Ma. A practical svm-based algorithm for ordinal regression in image retri eval. *ACM International Conference on Multimedia*, 2003.
[65] 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.
[66] 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.
[67] X. S. Zhou and T. S. Huang. Relevance feedback for image retrieval: a comprehensive review. *ACM Multimedia Systems Journal, special issue on CBIR*, 2003.