LEARNING AND MEASURING PERCEPTUAL SIMILARITY

Edward Chang

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

ABSTRACT

For almost a decade, Content-Based Information Retrieval has been an active research area, yet two fundamental problems remain largely unsolved: how best to learn users' query concepts, and how to measure perceptual similarity. To learn subjective query concepts, most systems use relevance feedback techniques. However, these traditional techniques often require a large number of training instances to converge to a concept, and a typical online user may be too impatient to provide much feedback. Thus traditional relevance feedback techniques are ineffective. To measure perceptual similarity, most researchers employ the Minkowski metric or the L-norm distance function. Our extensive data-mining experiments on visual data show that, unfortunately, the Minkowski-type metric is ineffective in modeling perceptual similarity. In this paper, we report the progress we have made recently in developing more effective methods for learning and measuring perceptual similarity, and our future research plans.

1. OVERVIEW

Research in multimdeia content-based information has gained steady momentum in recent years as a result of the dramatic increase in the volume of digital images, video, and audio. However, two fundamental problems remain largely unsolved: how best to learn users' query concepts, and how to measure perceptual similarity.

1. *Learning Query Concepts.* Perception is subjective, and so is similarity. Many learning and relevance feedback methods have been proposed to learn users' subjective query concepts. However, traditional techniques are ineffective for online query-concept learning for at least two reasons.

- *Time and sample constraints*. Traditional learning methods such as decision trees and neural networks require a large number of training instances (i.e., samples) and can take a long time (more than a few seconds) to learn a concept [7, 10]. But, online users are typically impatient and cannot be expected to wait around for results or to provide a great deal of feedback during the search.
- *Seeding constraint*. All traditional relevance feedback methods [5, 11, 14] require users to provide "good" examples to seed a query. However, finding good seeds is the job of the search engine itself, and this circular requirement leaves the core problem— learning users' query concepts—unsolved.

2. *Measuring Perceptual Similarity*. To achieve effective retrieval, an image system must be able to accurately characterize and quantify perceptual similarity. Various distance functions, such as Minkowski metric, earth mover distance, and fuzzy logic, have been used to measure similarity between feature vectors representing images. However, our experiments show that these functions can perform poorly in finding images that are obviously similar, and hence are not adequate for measuring perceptual similarity.

For learning subjective query concepts, we have developed two learning algorithms: MEGA [4] and SVM_{Active} [12]. For measuring image similarity according to human perception, we have discovered a perceptual distance function through mining a large set of visual data. We call the discovered function *dynamic partial distance function* (DPF). Section 2 describes MEGA and SVM_{Active}, and Section 3 presents our mining effort for finding DPF. Finally, we sketch our future research plans in Section 3.1.

2. LEARNING QUERY CONCEPTS

This example shows how our prototype [2, 3] that employs MEGA and SVM_{Active} works. Our prototype does not require seeding a query. It presents randomly selected images as the first round of examples. Even if all of the images generated in the first round are marked irrelevant by the user, our learning algorithm (i.e., MEGA) uses the irrelevant images to reduce the set of potentially relevant images substantially. This method increases the probability that a relevant image will be sampled and presented to the user for feedback in the next round.

Figure 1 shows two sample results after three rounds of relevance feedback from our prototype: one from a top-10

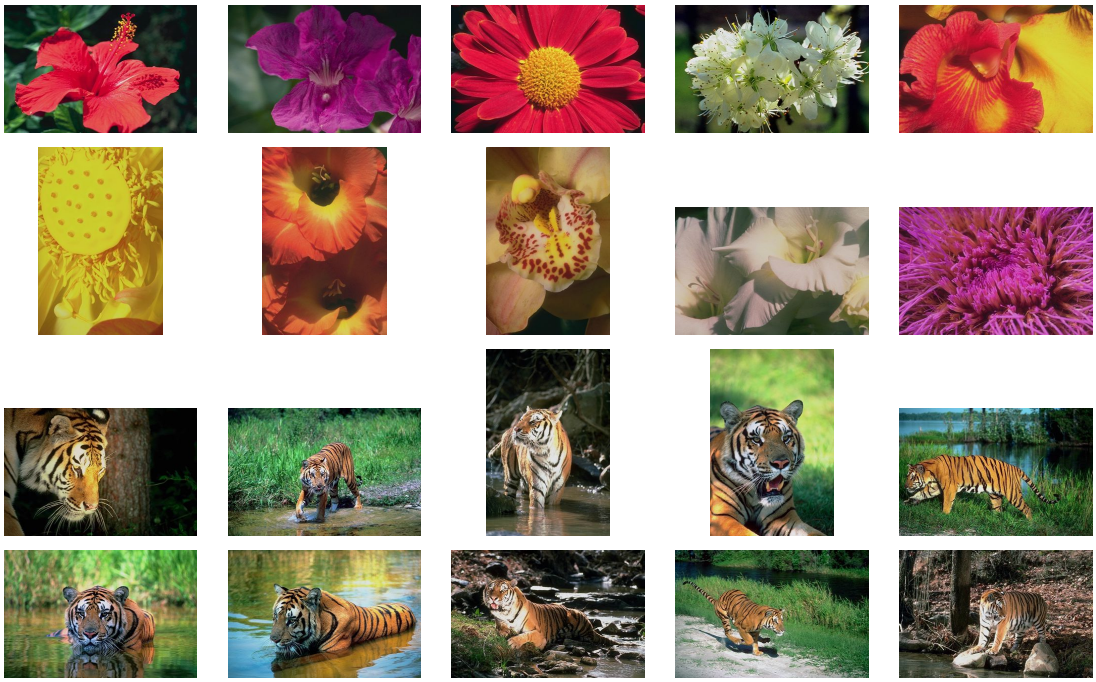

Figure 1: Flowers and Tigers Sample Query Results.

flowers query, and one from a top-10 tigers query. The returned images vary somewhat in color, shape, and background. The returned flowers have colors of red, purple, white, and yellow, with or without leaves. The returned tiger images have tigers of different postures on different backgrounds. Our concept learner works entirely in a very high dimensional feature space and does not have a semantical layer to assist searches.

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

- *Eliminating the seeding requirement*. One is not required to initiate a query with "good" seed images.
- *Supporting personalized search*. One can search for a general flower concept, or a specific kind of flower. One can search for a general animal concept, or a specific kind of animal (e.g., tigers).
- *Accomplishing the above tasks quickly and accurately*.

2.1. MEGA

The Maximizing Expected Generalization Algorithm (MEGA) models query concepts in k-CNF [6], which can formulate virtually all practical query concepts[1]. MEGA uses k-DNF to bound the sampling space from which to select the most informative samples to solicit user feedback.

Definition 1: k-CNF: For constant k, the representation class k-CNF consists of Boolean formulae of the form $c_1 \wedge \cdots \wedge c_\theta$, where each c_i is a disjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$. (A Boolean variable represents an image feature.) No prior bound is placed on θ.

Definition 2: k-DNF: For constant k, the representation class k-DNF consists of Boolean formulae of the form $d_1 \vee \cdots \vee d_\theta$, where each d_i is a conjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$. No prior bound is placed on θ.

MEGA initializes the query concept-space (QCS) as a k-CNF and the candidate concept-space (CCS) as a k-DNF. The QCS starts as the most specific concept and the CCS as the most general concept. The target concept that the learner learns is more general than the initial QCS and more specific than the initial CCS. The learner learns the QCS, while at the same time refining the CCS to delimit the boundary of the sampling space. (The shaded area in Figure 2 shows the sampling space that is between the QCS and the CCS.)

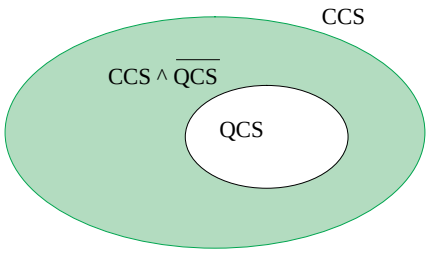

Figure 2: MEGA's Sampling Space: $CCS \wedge \overline{QCS}$.

Intuitively, at a given stage, we have a QCS and a CCS, representing the boundary of the candidate concept-space. To make sure that an example is most useful, we have tested

[1] k-CNF is more expressive than k-term DNF, and it has both polynomial sample complexity and time complexity [7, 9].

two strategies.

1. Bounding the sample space: Avoid choosing useless unlabeled instances by using the CCS and QCS to delimit the sampling boundary.

2. Maximizing the usefulness of a sample: Choose an example that shall remove the maximum expected number of disjunctive terms from QCS. In other words, we choose an example that can maximize the expected generalization of the concept. Even if the example is labeled negative by the user, it can be useful to remove conjunctive terms in the CCS.

It may appear that if we pick an example that has more dissimilar disjunctions (compared to the QCS), we would have a better chance of eliminating more disjunctive terms. This is, however, not true. An example must be labeled by the user as positive to be useful to refine QCS. Unfortunately, an example is less likely to be labeled positive when it has more disjunctions that are dissimilar to the target concept. Therefore, there is a tradeoff between choosing an example that has more contradictory terms and choosing one that is more likely to be labeled positive.

Let Ψ denote the number of disjunctions remaining in the concept. The number of disjunctions that can be eliminated in the current round of sampling, denoted as ψ, is between zero and Ψ. We can write the probability of eliminating ψ terms as $P_e(\psi)$. $P_e(\psi)$ is a monotonically decreasing function of ψ. We attempt to find the ψ that can eliminate the maximum expected number of disjunctive terms given an example. Our objective function can be written as

$$\psi^* = arg_\psi \max E(\psi) = \psi \times P_e(\psi). \qquad (1)$$

To solve ψ^*, we must know $P_e(\psi)$, which can be estimated by the two methods below: probabilistic estimation and empirical estimation. (Please refer to [4] for detailed discussion.) We have compared MEGA with the five traditional sampling schemes: *random*, *bounded random*, *nearest neighbor*, *query expansion* and *aggressive*. Our preliminary experimental results show that MEGA substantially outperforms traditional sampling schemes in all query scenarios when the number of labeled instances is limited. Our prototype [2, 3] employing MEGA can grasp a query concept in three to five rounds of feedback in a 144-feature dimensional space.

2.2. SVM*Active*

SVM*Active* is another method that we have developed for supporting online concept learning. SVM*Active* combines active learning with support vector machines (SVMs). For the purpose of query-concept learning, we 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 $\mathcal{X} \subseteq \mathbb{R}^\Gamma$. 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. 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 $\mathcal{X}$ to a higher dimensional feature space $\mathcal{F}$ via a Mercer kernel operator K. In other words, we consider the set of classifiers of the form: $f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_\mathbf{i} \mathbf{K}(\mathbf{x_i}, \mathbf{x})$. When $f(\mathbf{x}) \geq \mathbf{0}$ we classify $\mathbf{x}$ as $+1$, otherwise we classify $\mathbf{x}$ as -1.

When K satisfies Mercer's condition [1] we can write: $K(\mathbf{u}, \mathbf{v}) = \mathbf{\Phi}(\mathbf{u}) \cdot \mathbf{\Phi}(\mathbf{v})$ where $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ and "$\cdot$" denotes an inner product. We can then rewrite f as:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}), \text{ where } \mathbf{w} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_\mathbf{i} \mathbf{\Phi}(\mathbf{x_i}). \qquad (2)$$

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

Intuitively, SVM*Active* works by combining the following three ideas:

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

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

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

Let V denote the uncertain concept space. The current version of SVM*Active* employs Tong & Koller's lemma [13], which chooses a pool-query that halves the uncertain space. In reality, however, finding the pool-query that can divide the uncertain space in half can be difficult and computationally intensive. Given an unlabeled instance $\mathbf{x}$ from the pool, it is not practical to explicitly compute the sizes of the new uncertain spaces $\mathcal{V}^-$ and $\mathcal{V}^+$ (i.e., the uncertain spaces obtained when $\mathbf{x}$ is labeled as -1 and $+1$ respectively). We have tried a simple method that works as follows: Learn an

SVM with the existing labeled data, and choose as the next batch of samples, those closest to the hyperplane in feature space $\mathcal{F}$.

Once we have performed a number of rounds of querying, we will have amassed a small set of labeled instances. We learn a final SVM using these labeled instances and use this SVM as our final classifier. In the retrieval task, we wish to return instances in order of decreasing relevance. This is easy to achieve with an SVM. Recall that an SVM is a hyperplane in the feature space such that any feature vector lying on one side of the hyperplane is labeled as "relevant" and any feature vector lying on the opposite side of the hyperplane is labeled as "not relevant." We have greatest confidence in those labels when the vectors are farthest away from the hyperplane. Hence, we can rank all of the instances in the database in order of relevance, simply by computing their signed distance from the hyperplane.

3. MEASURING SIMILARITY

Quantifying perceptual similarity is a difficult problem. Indeed, fully understanding how human perception works may still be decades away. We mine visual data extensively to *discover* a good perceptual distance function for measuring image similarity. Our mining hypothesis is the following: Suppose most similar images can be clustered in a feature space. We can then claim with high confidence that 1) the feature space can adequately capture visual perception, and 2) the distance function used for clustering images in that feature space can accurately model perceptual similarity.

To ensure that sound inferences can be drawn from our mining results, we carefully construct the dataset. First, we prepare a dataset that is comprehensive to cover a diversified set of images. To achieve this goal, we collect $60,000$ images from Corel CDs and from the Internet. Second, we define "similarity" in a slightly restrictive way so that individuals' subjectivity can be safely excluded. For each image in the $60,000$-image set, we perform 24 transformations including rotation, scaling, cropping, and downsampling, and hence form $60,000$ similar-image sets. Our mining work is then to discover in what feature space using what distance function, the members of a similar set can be kept in the nearest neighborhood of each other.

We perform our mining in two stages. In the first stage, we isolate the distance function factor (we use the Euclidean distance) to find a reasonable feature set. In the second stage, we freeze the features and discover a perceptual distance function. We call the discovered perceptual function *dynamic partial distance function* (DPF) [8]. When we empirically compare DPF to Minkowski-type distance functions, DPF performs remarkably better. Equally encouraging is the fact that DPF can be interpreted by simple psychological principles.

3.1. Future Work

Our future work can be divided into two thrusts.

1. *Accuracy improvement thrust*. We will improve the effectiveness of MEGA and SVM_{Active}, and possibly lead to the development of new online learning algorithms. We plan to investigate algorithms to perform *adaptive sampling*, to explore *co-training* using MEGA and SVM_{Active}, and to detect and deal with *concept drift*. We will conduct extensive multimedia data mining to discover better multimedia data representations, and to continue improving DPF.

2. *Efficiency improvement thrust*. We will ensure that the learning algorithms are scalable in feature dimension, dataset size, and concept complexity. In this regard, we plan to investigate *multiresolution learning* (for feature-dimension scalability), *image classification* (for concept-complexity scalability), and *high-dimensional indexing* (for dataset-size scalability) methods.

4. REFERENCES

[1] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[2] E. Chang, K.-T. Cheng, and L. Chang. PBIR — perception-based image retrieval. *ACM Sigmod (Demo)*, May 2001.

[3] E. Chang and et al. PBIR — a system that learns subjective image query concepts. *ACM Multimedia (Demo)*, October 2001.

[4] E. Chang and B. Li. Mega — the maximizing expected generalization algorithm for learning complex query concepts (extended version). *Technical Report http://www-db.stanford.edu/∼echang/mega.pdf*, November 2000.

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

[6] M. Kearns, M. Li, and L. Valiant. Learning boolean formulae. *Journal of ACM*, 41(6):1298–1328, 1994.

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

[8] B. Li, E. Chang, and C.-T. Wu. Discovery of perceptual distance functions for measuring image similarity. *Submitted to SIAM Data Mining Conference*, September 2001.

[9] T. Michell. *Machine Learning*. McGraw Hill, 1997.

[10] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[11] K. Porkaew, K. Chakrabarti, and S. Mehrotra. Query refinement for multimedia similarity retrieval in mars. *Proceedings of ACM Multimedia*, November 1999.

[12] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *ACM International Conference on Multimedia*, October 2001.

[13] 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.

[14] 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.