Adaptively Learning the Crowd Kernel

Abstract

We introduce an algorithm that, given n objects, learns a similarity matrix over all n^2 pairs, from crowdsourced data alone. The algorithm samples responses to *adaptively chosen* triplet-based relative-similarity queries. Each query has the form "is object a more similar to b or to c?" and is chosen to be maximally informative given the preceding responses. The output is an embedding of the objects into Euclidean space (like MDS); we refer to this as the "crowd kernel."

The runtime (empirically observed to be linear) and cost (about \$0.15 per object) of the algorithm are small enough to permit its application to databases of thousands of objects. The distance matrix provided by the algorithm allows for the development of an intuitive and powerful sequential, interactive search algorithm which we demonstrate for a variety of visual stimuli. We present quantitative results that demonstrate the benefit in cost and time of our approach compared to a nonadaptive approach. We also show the ability of our approach to capture different aspects of perceptual similarity by demonstrating a variety of binary attribute classifiers ("is striped," "vowel vs. consonant,") trained using the learned kernel.

We construct an end-to-end system that, given a set of objects, automatically crowdsources the kernel acquisition. It then uses the kernel to build an interactive visual search tool.

1. Introduction

The problem of capturing and extrapolating a human notion of perceptual similarity has received increasing attention in recent years including areas such as vision (Agarwal et al., 2007), audition (McFee & Lanckriet, 2009), information retrieval (Schultz & Joachims, 2003) and a variety of others represented in the UCI Datasets (Xing et al., 2003; Huang et al., 2010). Concretely, the goal of these approaches is to estimate a similarity matrix K over all pairs of n objects given a (potentially exhaustive) subset of human perceptual measurements on tuples of objects. In some cases the set of human measurements represents 'side information' to computed descriptors (MFCC, SIFT, etc.), while in other cases – the present work included – one proceeds exclusively with human reported data. When K is a positive semidefinite matrix induced purely from distributed human measurements, we refer to it as the *crowd kernel* for the set of objects.

Given such a Kernel, one can exploit it for a variety of purposes including exploratory data analysis or embedding visualization (as in Multidimensional Scaling) and relevance-feedback based interactive search. As discussed in the above works and (Kendall & Gibbons, 1990), using a *triplet based* representation of relative similarity, in which a subject is asked "is object a more similar to b or to c," has a number of desirable properties over the classical approach employed in Multi-Dimensional Scaling (MDS), i.e., asking for a numerical estimate of "how similar is object a to b." These advantages include reducing fatigue on human subjects and alleviating the need to reconcile individuals' scales of similarity. The obvious drawback with the triplet based method, however, is the potential $O(n^3)$ complexity. It is therefore expedient to seek methods of obtaining high quality approximations of K from as small a subset of human measurements as possible. Accordingly, the primary contribution of this paper is an efficient method for estimating K via an information theoretic adaptive sampling approach.

At the heart of our approach is a new scale-invariant Kernel approximation model. The choice of Kernel approximation model is shown to be crucial in terms of the adaptive triples that are produced, and the new model produces effective triples to label. Although this model is nonconvex, we prove that it can be optimized under certain assumptions. We compare our model to a convex logistic model.

We construct an end-to-end system for interactive vi-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Figure 1. A sample top-level of a similarity search system that enables a user to search for objects by similarity. In this case, since the user clicked on the middle-left tile, she will "zoom-in" and be presented with similar tiles.

sual search and browsing using our Kernel acquisition algorithm. The input to this system is a set of images of objects, such as products available in an online store. The system automatically crowdsources the kernel acquisition and then uses this kernel to produce a visual interface for searching or browsing the set of products. Figure 1 shows this interface for a database of 433 floor tiles available at amazon.com.

1.1. Human kernels versus machine kernels

The bulk of work in Machine Learning focuses on "Machine Kernels" that are computed by computer from the raw data (e.g., pixels) themselves. Additional work employs human experiments to try to learn kernels based upon machine features, i.e., to approximate the human similarity assessments based upon features that can be derived by machine. In contrast, when a kernel is learned from human subjects alone (whether it be data from an individual or a crowd)[1] one may call it a *human kernel*. When learning human kernels, we consider no machine features whatsoever. To the computer, the objects are recognized by ID's only – the images themselves are hidden from our system and are only presented to humans.

The primary advantage of machine kernels is that they can generalize immediately to new data, whereas each additional object needs to be added to our system, for a cost of approximately \$0.15. On the other hand, working with a human kernel has two primary advantages. First, it does not require any domain expertise. While for any particular domain, such as music or images of faces, cars, or sofas, decades of research may have provided high-quality features, one does not have to find, implement, and tune these sophisticated feature detectors. This is of value to consumers of such a system, such as store vendors who may not have the necessary expertise.

Second, human kernels may contain features that are simply not available with state-of-the-art feature detectors, because of knowledge and experience that humans possess. For example, from images of celebrities, human similarity may be partly based on whether the two celebrities are both from the same profession, such as politicians, actors, and so forth. Until the long-standing goal of bridging the semantic gap is achieved, humans will be far better than machines at interpreting a number of features, such as "does a couch look comfortable," "can a shoe be worn to an informal occasion," or "is a joke funny."

We give a simple demonstration of external knowledge through experiments on 26 images of the lower-case Roman alphabet. Here, the learned Kernel is shown to capture features such as "is a letter short or tall" (acemnorsuvwxz vs. bfhkl), which could be determined from pixels alone. This is shown by using the Kernel in an SVM and achieving 0% error rate in leave-one-out cross validation. However, it also exhibits the feature "vowel versus consonant," which uses external knowledge beyond the pixels. Note that this experiment is interesting in itself because it is not at first clear if people can meaningfully answer the question: "is the letter e more similar to i or p." One person may feel that the question is ill-posed, another may feel that e is more similar to i because the are both vowels, while a third person may feel that e is more similar to p because the letter names rhyme. Our experiments show statistically significant consistency with 58% ($\pm 2\%$, with 95% confidence) agreement between users on a random triple of letters. (For random image triplets from an online tie store, 68% agreement is observed, and for floor tile images 65%).

2. Benefits of adaptation

We first give high-level intuition for why adaptively choosing triples may yield better kernel approximations than randomly choosing triples. First consider a dataset of n objects that naturally partitions into

[1] It would be possible run our system using a single user, but that would be slower due to the massive parallelism enabled by the crowd.

$k \ll n$ disjoint equal-sized clusters, such that between clusters objects are completely dissimilar but within clusters they have varied similarities. For example, our images from an online tie store cluster into ties, tie clips, and scarves. Say that, within any specific cluster, one can locate the object using q queries by comparing it to other objects in the same cluster. On the other hand, suppose comparisons with objects in two different classes simply yield 50/50 random results if the three objects are in different classes but that the crowd will select an object of the same class if one exists in the comparison pair. The number of adaptive queries to learn in such a setting is $\Theta(nk + nq)$: $\Theta(k)$ comparisons are required to determine which class each object is in (with high probability) and then an additional q queries are required. With random queries, one would require $\Theta(nk^2q)$ queries, because only a $1/k^2$ fraction of the random queries will count towards the q necessary queries within objects of the same class.

Next, consider data representing an underlying rooted tree with $k \ll n$ leaves, inspired by, say, phylogenic trees involving animal species.[2] Say the similarity between objects is decreasing in their distance in the tree graph and, furthermore, that objects are drawn uniformly at random from the classes represented by the leaves of the tree. Ignoring the details of how one would identify that two objects are in the same leaf or subtree, it is clear that a nonadaptive method would have to ask $\Omega(nk)$ questions to determine the leaves to which n objects belong (or at least to determine which objects are in the same tree). On the other hand, in an ideal setting, an adaptive approach might determine such matters using $O(n \log k)$ queries in a balanced binary tree, assuming a constant number of comparisons can determine to which subtree of a node an object belongs, hence an exponential savings.

3. Related work

4. Preliminaries

The set of n objects is denoted by $[n] = \{1, 2, \ldots, n\}$. For $a, b, c \in [n]$, a comparison or *triple* of the form, "is a more similar to b or to c." We refer to a as the *head* of the triple. We write p^a_{bc} for the probability that a *random* crowd member rates a as more similar to b, so $p^a_{bc} + p^a_{cb} = 1$. The n objects are assumed to have d-dimensional Euclidean representation, and hence the data can be viewed as a matrix $M \in \mathbb{R}^{n\times d}$, and the *similarity matrix* $K \in \mathbb{R}^{n\times n}$ is defined by $K_{ab} = M_a \cdot M_b$, or equivalently $K = MM^T$. Note that K is necessarily positive semidefinite (PSD), and for any PSD matrix K, one can efficiently find an embedding in $\mathbb{R}^d$ (unique up to change of basis), for some $d \le n$. Also equivalent is the representation in terms of distances, $d^2(a, b) = K_{aa} - 2K_{ab} + K_{bb}$.

In our setting, an *MDS algorithm* takes as input m comparisons $(a_1b_1c_1, y_1) \ldots (a_mb_mc_m, y_m)$ on n items, where $y_i \in \{0, 1\}$ indicates whether a_i is more like b_i than c_i. Unless explicitly stated, we will often omit y_i and assume that the b_i and c_i have been permuted, if necessary, so that a_i was rated as more similar to b_i than c_i. The MDS algorithm outputs an embedding $M \in \mathbb{R}^{n\times d}$ for some $d \ge 1$. A probabilistic MDS model outputs predicts $\hat{p}^a_{bc}$ based on M_a, M_b, and M_c. The *empirical log-loss* of a model that predicts $\hat{p}^{a_i}_{b_ic_i}$ is $\sum_i \log 1/\hat{p}^{a_i}_{b_ic_i}$. Our probabilistic MDS model attempts to minimize empirical log loss subject to some regularization constraint. We choose a probabilistic model due to its suitability for use in combination with our information-gain criteria for selecting adaptive triples, described later.

An *active* MDS algorithm chooses each triple, $a_ib_ic_i$, adaptively based upon $(a_1b_1c_1, y_1), \ldots, (a_{i-1}b_{i-1}c_{i-1}, y_{i-1})$. We denote by M^T the transpose of matrix M and $\|M\|_F = \sqrt{\sum_{ij} M^2_{ij}}$ denotes the Frobenius norm. For compact convex set W, let $\Pi_W(K) = \arg\min_{T\in W} \|K - T\|^2_F$ is the closest matrix in W to K. Also define the set of symmetric unit-length PSD matrices,

$$B = \{K \succeq 0 \mid S_{11} = S_{22} = \ldots = S_{nn} = 1\}.$$

Projection to the closest element of B is a quadratic program which can be solved via a number of existing techniques – see (Srebro & Shraibman, 2005; Lee et al., 2010).

5. Our algorithm

Our algorithm proceeds in phases. In the first phase, it queries a certain number of random triples comparing each object $a \in [n]$ to random pairs of distinct b, c. (Note that we never present a triple where $a = b$ or $a = c$ except for quality control purposes.) Subsequently, it fits the results to a matrix $M \in \mathbb{R}^{n\times d}$ using the

[2] This example is based upon a tree metric rather than a Euclidean one. However, note that any tree with k leaves can be embedded in k-dimensional Euclidean space so that the squared distance between any pair of embedded points is equal to the number of edges in their shortest path on the tree. Moreover, the rich study of Embeddings (see, e.g., Indyk & Matousek, 2004) has shown that many types of metrics can be embedded (to varying degrees of approximation) within Euclidean space.

relative probabilistic similarity model described below. Then it uses our adaptive selection algorithm to select further random triples. This iterates: in each phase all previous data is refit to the relative model, and then the adaptive selection algorithm generates more triples.

- For each item $a \in [n]$, crowdsource labels for R random triples with head a.
- For $t = 1, 2, \ldots, T$:
  - Fit S^t to the labeled data gathered thus far, using the method described in Section 5.1 (with d dimensions).
  - For each $a \in [n]$, crowdsource a label for the maximally informative triple with head a, using the method described in Section 6.

Typical parameter values which worked quickly and well across a number of medium-sized data sets of (hundreds of objects) were $R = 10$, $T = 25$, and $d = 3$. These settings were also used to generate Figure **??**. We first describe the probabilistic MDS model and then the adaptive selection procedure. Further details are given in Section 7.

5.1. Relative similarity model

The *relative* probabilistic model is motivated by the scale-invariance observed in many perceptual systems (see, e.g., Chater & Brown). Let $\delta_{ab} = \|M_a - M_b\|^2 = K_{aa} + K_{bb} - K_{ab}$. A simple scale-invariant proposal takes $\hat{p}^a_{bc} = \frac{\delta_{ac}}{\delta_{ab}+\delta_{ac}}$. Such a model must also be regularized or else it would have $\Theta(n^2)$ degrees of freedom. One may regularize by the rank of K or by setting $K_{ii} = 1$. Due to the scale-invariance of the model, however, this latter constraint does not have reduce complexity. In particular, note that halving or doubling the matrix M doesn't change any probabilities. Hence, descent algorithms may lead to very small, large, or numerically unstable solutions. To address this, we modify the model as follows, for distinct a, b, c:

$$\hat{p}^a_{bc} = \frac{\mu + \delta_{ac}}{2\mu + \delta_{ab} + \delta_{ac}} \quad \text{and} \quad K_{ii} = 1, \tag{1}$$

for some parameter $\mu > 0$. Alternatively, this change may be viewed as an additional assumption imposed on the previous model – we suppose each object possesses a minimal amount of "uniqueness," $\mu > 0$, such that $K = \mu I + T$, where $T \succeq 0$. We fit the model by local optimization performed directly on M (with random initialization), and produces high-quality adaptive triples even for low dimensions, such as 3.[3] Here μ serves a purpose similar to a margin constraint.

There are two interesting points to make about our choice of model. First, the loss is not convex in K, so there is a concern that local optimization may be susceptible to local minima. In Section 6.1, we state a theorem which explains why this does not seem to be a significant problem. Second, in Section 6.2, we discuss a simple convex alternative based on logistic regression, and we explain why this model, in combination with our adaptive selection criterion, gives rise to poor adaptively-selected triples.

6. Adaptive selection algorithm

We describe the adaptive selection algorithm with respect to the relative model above, but it can equally well be applied to the exponential model. The idea is to capture the uncertainty about the location of an object through a probability distribution over points in $\mathbb{R}^d$, and then to ask the question that maximizes information gain.

Given a set of previous comparisons of n objects, we generate, for each object $a = 1, 2, \ldots, n$, a new triple to compare a to, as follows. First, we embed the objects into $\mathbb{R}^d$ as described above, using the available comparisons. Initially, we use a seed of randomly selected triples for this purpose. Later, we use all available comparisons - the initial random ones and those acquired adaptively.

Now, say the crowd has previously rated a as more similar to b_i than c_i, for $i = 1, 2, \ldots, j-1$, and we want to generate the jth query, $\frac{a}{b_j, c_j}$ (this is a slight abuse of notation because we don't know which of b_j or c_j will be rated as closer to a). These observations imply a posterior distribution of $\rho(x) \propto \pi(x) \prod_i \hat{p}^x_{b_i c_i}$ over $x \in \mathbb{R}^d$, where x is the embedding of a, and $\pi(x)$ is a prior distribution, to be described shortly.

Given any candidate query for objects in the database b and c, the model predicts that the crowd will rate a as more similar to b than c with probability $p \propto \int_x \frac{\delta(x,c)}{\delta(x,b)+\delta(x,c)} \rho(x) dx$.[4] If it rates a more simi-

[3] For high-dimensional problems, we perform a gradient projection descent on K. In particular, starting with $K^0 = \lambda I$, we compute $K^{t+1} = \Pi_B(S^t - \eta \nabla \mathcal{L}(K))$ for step-size η (see Preliminaries for the definition of Π_B).

[4] Like other active learning models, e.g. (**?**), it is tempting to choose b and c so as to make this probability close to 1/2. In our case, this is not sufficient because b and c could be known to give p close to half without being a useful query, e.g., b and c are very close to one another but far from a. However, we wouldn't want to compare a to b

lar to b than c then x has a posterior distribution of $\rho_b(x) \propto \rho(x)\frac{\delta(x,c)}{\delta(x,b)+\delta(x,c)}$, and $\rho_c(x)$ (of similar form) otherwise. The *information gain* of this query is defined to be $H(\rho)-pH(\rho_b)-(1-p)H(\rho_a)$, where $H(\cdot)$ is the entropy of a distribution. This is equal to the mutual information between the crowd's selection and x. The algorithm greedily selects a query, among all pairs $b, c \neq a$, which maximizes information gain. This computation can be somewhat computationally intensive (seconds per object in our datasets), so for efficiency we take the best pair from a sample of random pairs.

It remains to explain how we generate the prior π. We take π to be the uniform distribution over the set of points in M. Hence, the process can be viewed as follows. For the purpose of generating a new triple, we pretend the coordinates of all other objects are perfectly known, and we pretend that the object in question, a, is an unknown one of these other objects. The chosen pair is designed to maximize the information we receive about which object it is, given the observations we already have about a. The hope is that, for sufficiently large data sets, such a data-driven prior is a reasonable approximation to the actual distribution over data. Another natural alternative prior would be a multinormal distribution fit to the data in M.

6.1. Optimization guarantee

This model is appealing in that it fits the data well, suggests good triples, and also represents interesting features on the data. Unfortunately, the model itself is not convex. We now give some justification for why gradient descent should not get trapped in local minima. As is sometimes the case in learning, it is easier to analyze an online version of the algorithm, i.e., a stochastic gradient descent. Here, we suppose that the sequence of triplets is presented in order: the learner predicts S^{t+1} based on $(a_1, b_1, c_1, y_1), \ldots, (a_t, b_t, c_t, y_t)$. The loss on iteration t is $\ell_t(S^t) = \log 1/p$ where p is the probability that the relative model with S^t assigned to the correct outcome.

We state the following theorem about stochastic gradient descent.

Theorem 1 *Let $W = \{K \succeq 0 \mid K_{ii} = 1\}$ and let $a_t, b_t, c_t \in [n]$ be arbitrary, for $t = 1, 2, \ldots$. Suppose there is a matrix $S^* \in W$ such that $\Pr[y_t = 1] = \frac{\mu+2-2K_{ac}}{2\mu+4-2K_{ab}-2K_{ac}}$. For any $\epsilon > 0$, there exists an T_0 such that for any $T > T_0$ and $\eta = 1/sqrtT$,*

$$\frac{1}{T}\sum_{t=1}^{T} \ell_t(S^t) - \ell_t(S^*) \leq \epsilon.$$

Due to space limitations, the proof is omitted.[5]

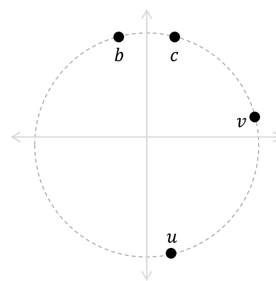

Figure 2. When unsure whether a point is at location b or c, the logistic model would strangely prefer comparing it to u and v over b and c themselves. The exponential model makes this prediction because $(b-c)\cdot(u-v) > (b-c)\cdot(b-c)$.

6.2. The logistic model: A convex alternative

As a small digression, we explain why the choice of probabilistic model is especially important for adaptive learning. To this end, consider the following *logistic* model.

$$\hat{p}^a_{bc} = \frac{e^{K_{ab}}}{e^{K_{ab}} + e^{K_{ac}}} = \frac{1}{1 + e^{K_{ac}-K_{ab}}}. \qquad (2)$$

Note that $\log 1 + e^{K_{ac}-K_{ab}}$ is a convex function of $K \in \mathbb{R}^{n \times n}$. Hence, the problem of minimizing its empirical log loss over a convex set is a convex optimization problem.

Experiments indicate that the logistic model fits data well and reproduces interesting features, such as vowel/consonant or stripedness. However, empirically it performs poorly in terms of deciding which triples to ask. Figure 2 gives a simple example illustrating where the exponential model chooses a poor question.

This criterion for evaluating a model, namely the quality of triples it gives rise to, is perhaps an interesting one.

7. System parameters and quality control

We've described abstractly how our system is implemented. This section describes parameters and specifics of our optimization algorithms and experiments.

and c repeatedly in this case.

[5]We have included the proof in the supplementary materials.

7.1. Mechanical Turk

Experiments were performed using Amazon's Mechanical Turk web service, where we defined 'Human Intelligence Tasks' to be performed by one or more users. Each task consists of 50 comparisons and the interface is optimized to be performed with 50 mouse clicks (and no scrolling). The mean completion time was approximately 2 minutes, for which workers were paid 15 cents (US). This price was determined based upon worker feedback. At 10 cents per task, though workers actively performed the tasks, some complained about low wages and several suggested that they be paid 15 cents per task. At 15 cents per task, feedback was extremely positive – the users reported that the tasks were enjoyable and requested more. Initial experiments revealed a high percentage of seemingly random responses, but after closer inspection the vast majority of these poor results came from a small number of individuals. To improve quality control, we imposed a limit on the maximum number of tasks a single user could perform on any one day, we selected users who had completed at least 48 tasks with a 95% approval rate, and each task included 20% triples for which there was tremendous agreement between users. These "gold standard" triples were also automatically generated and proved to be an effective manner to recognize and significantly reduce cheating. The system is implemented using Python, Matlab, and C, and runs completely automatically in Windows and Unix.

7.2. Question phrasing and crowd alignment

One interesting issue is how to frame similarity questions. On the one hand, it seems purest in form to give the users carte blanche and ask only, "is a more similar to b than c." On the other hand, in feedback users complained about these tasks and often asked what we meant by similarity. Moreover, different users will inevitably weigh different features differently when performing comparisons. For example, consider a comparisons of face images, where a is a white male, b is a black male, and c is a white female. Some users will consider gender more important in determining skin color, and others may feel the opposite is true. Others may feel that the question is impossible to answer. Consider phrasing the question as follows, "At a *distance*, who would you be more likely to mistake for a: b or c?" For any two people, there is presumably some distance at which one might be mistaken for the other, so the question may seem more possible to answer for some people. Second, users may more often agree that skin color is more important than gender, because both are easily identified close up by skin color may be identifiable even at a great distance. While we

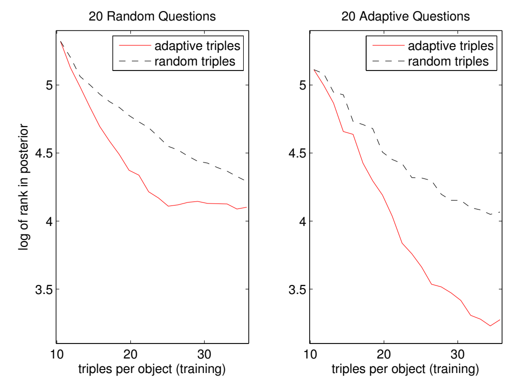

Figure 3. The 20 Questions log-rank metric plots comparing training based on adaptively selected triples to randomly selected training triples. The left plot shows their mean predicted log-ranks of randomly chosen objects after 20 randomly chosen questions. The right plot shows their mean predicted log-ranks of randomly chosen objects after 20 adaptive queries.

haven't done experiments to determine the importance of question phrasing, anecdotal evidence suggests that users enjoy the tasks more when more specific definitions of similarity are given.

Two natural goals of question phrasing might be: (1) to align users in their ranking of the importance of different features and (2) to align user similarity notions with the goals of the task at hand. For example, if the task is to find a certain person, the question, "which two people are most likely to be (genealogically) related to one another," may be poor because users may overlook features such as gender and age. In our experiments on neckties, for example, the task was titled "Which ties are most similar?" and the complete instructions were:

> Someone went shopping at a tie store and wanted to buy the item on top, but it was not available. Click on item (a) or (b) below that would be the **best substitute**.

8. Experiments and Applications

We experiment on four datasets: (1) images of twenty-six lowercase letters, (2) 223 flag images, (3) 433 tile images from Amazon, and (4) 300 product images from an online tie store. Surprisingly, it seems that for these datasets about 30 random triples per object suffice to learn the Crowd Kernel well. Roughly, we find that adaptive queries can achieve the same performance as uniformly random queries using a seed of 10 triples per object and an additional 10 adaptive queries. We expect the saving to increase for larger and more diverse

Figure 4. Below each of the six objects, we show the adaptive pairs to which that object was compared along with the crowd's selections (in red). The first pair below each large object was chosen adaptively after observing the results of ten random comparisons. Then, proceeding down, the pairs were chosen using the ten random comparisons plus the results of the earlier comparisons above. It appears that early questions are aimed at learning the object's general type, while later questions are aimed at recovering finer details.

datasets.

For ease of implementation, we assume all users are identical. This is a natural starting point, especially given that our main focus is on active learning.

Figure 3 show the adaptive triples selected on an illustrative dataset composed of a mixture of flags, ties and tiles.

8.1. 20Q Metric

It is not clear how to judge the predictions that a particular embedding implies. One application of such systems is search, i.e., searching for an item that a user knows what it looks like (we assume that the user can answer queries as if she even knows what the store image looks like). Therefore, it is natural to ask how well we have "honed in" on the desired object after a certain number of questions. For our metric, we suppose that the user has selected a secret random object in the database and the system is allowed to make query 20 triples, adaptively (as in the game "20 questions"), after which it produces a ranking of items in the database. The metric is the average position of the random target item in this list. This metric is meant to roughly capture performance, but of course in a real system users may not have the patience to click on twenty pairs of images and may prefer to choose from larger sets. (Our system has the user select one of 8 or 9 images, which could potentially convey the same information as 3 binary choices.)

8.2. Using the Kernel for classification

The learned Kernels may be used in a linear classifier such as a support vector machine. This helps elucidate which features were used by humans in labeling the data. In the experiments below, images were labeled with binary $\pm$ classes and ? indicating don't care. For example, if the classification task is identifying whether a tie is striped or not, labeling the tie clips seems irrelevant and we ignore them. The LIBSVM (Chang & Lin, 2001) package was used with default parameters. For all learning tasks but letters, results are shown on 30? held-out examples while the rest were used for training. For the letters, we show results based on leave-one-out classification. The results are shown by sorting the held-out images from left to right in order of their inner product with the learned direction. In addition, the following table summarizes accuracy on a variety of tasks.

8.3. Nearest neighbors and PCA

Below we show the nearest neighbors for some objects in the ties dataset.

The reference image is on the left and the 14 nearest neighbors are displayed from left to right.

Below, the flag images are displayed according to their projection on the top two principal components of a PCA. (The principal component is the horizontal axis.)

More nearest neighbor and PCA charts are available in the supplementary material.

8.4. Optimization

What we find here is that a low-rank constraint, which can be interpreted as fixing the dimensionality d of $M \in \mathbb{R}^{n \times d}$, provides better regularization when the size of the learning set is small, but does not capture features of interest as well as a high dimensional representation, for large sample sizes. Fixing the diagonal to $K_{ii} = 1$ gives a high-quality fit but does not generate quite as meaningful questions when we have little data.

Hence, when we have little data, we use the low-dimensional model in conjunction with our selection algorithm, for generating triples. When we analyze the data which we have, we generally use the fixed-diagonal constraint without a rank bound. Interestingly, the trace bound performed poorly in this setting. In fact, a fixed-diagonal setting of $K_{ii} = r$ outperformed a trace bound of nr, even on training data. This is counterintuitive because a fixed-diagonal setting of $K_{ii} = r$ directly implies a trace equal to nr. The reason the optimization with the trace bound was failing is because the optimization problem is not convex, and hence gradient descent may reach local minima. It seems that the trace bound and fixed-diagonal settings have different optimization landscapes, and the fixed-diagonal optimization performs better.

8.5. Visual Search

Our primary application is a visual search tool, depicted in Figure 1. Given n images, their embedding into $\mathbb{R}^d$ and the related probabilistic model for triples, we would like to help a user find either a particular object she has in mind, or a similar one.

We do this by playing a "20 questions game" of sorts with the user. We assume the user has one of the objects in mind, and choose an initial prior to quantify our uncertainty regarding this object. We choose a uniform prior, but this can be replaced by empirical priors when available. We then pick 9 objects $\{b_1, \ldots, b_9\}$ to show the user, and expect her to click on object b_i with probability $\propto \delta(a, b_i)$ is her object is a. Our choice of these objects is one that maximizes the information we gain from her click.

Using the same probabilistic model we can now update our distribution of the user's object and show another 9 objects, using the same method.

9. Conclusion and Discussion

In this work, we capture the crowd kernel using no machine attributes whatsoever. Machine attributes are of course desirable when it is possible to approximate the crowd kernel automatically. In this case our work could be used as a component of such a hybrid system. However, approximating the crowd kernel automatically requires, in general, extremely good domain-specific features. One of the biggest challenges in machine learning is selecting good features for a data sets. Learning the crowd kernel without machine features sidesteps this issue, to some extent. This may be feasible at least for applications such as online stores where a small price per object is reasonable and the number of objects is not prohibitively large. Learning the crowd kernel is a natural first step in bridging the semantic gap between computer and humans, and active learning should be a key part of this process.

There is room to improve the adaptive component of our system. First, one may make it online in the sense that it could add objects to the database one at a time or in batches, rather than having all the objects present up-front. Second, it may be desirable to have personalized user-specific models as in (?), or group specific models. For example, it may be interesting to contrast the crowd kernels of men and women on various domains. Third, in the case where our model is not perfectly accurate, our algorithm suffers from the fact that the training distribution on queries is different from the test distribution. Techniques such as importance weighting have been shown to be one practical solution to this problem for active learning (Beygelzimer et al., 2009), and one might try to apply them to the problem at hand.

References

Agarwal, Sameer, Wills, Josh, Cayton, Lawrence, Lanckriet, Gert, Kriegman, David, and Belongie, Serge. Generalized non-metric multidimensional scaling. In *AISTATS*, San Juan, Puerto Rico, 2007.

Beygelzimer, Alina, Dasgupta, Sanjoy, and Langford, John. Importance weighted active learning. In Danyluk, Andrea Pohoreckyj, Bottou, Léon, and Littman, Michael L. (eds.), *ICML*, volume 382 of *ACM International Conference Proceeding Series*, pp. 7. ACM, 2009. ISBN 978-1-60558-516-1.

Chang, Chih-Chung and Lin, Chih-Jen. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chater, N and Brown, G D.

Huang, Kaizhu, Ying, Yiming, and Campbell, Colin. Generalized sparse metric learning with relative comparisons. *Knowledge and Information Systems*, pp. 1–21, 2010. ISSN 0219-1377. URL http://dx.doi.org/10.1007/s10115-010-0313-0. 10.1007/s10115-010-0313-0.

Indyk, Piotr and Matousek, Jiri. *Low-Distortion Embeddings of Finite Metric Spaces*. CRC Press, 2004.

Kendall, Maurice and Gibbons, Jean D. *Rank Correlation Methods*. A Charles Griffin Title, 5 edition, September 1990. ISBN 0195208374. URL http://www.worldcat.org/isbn/0195208374.

Lee, Jason, Recht, Ben, Salakhutdinov, Ruslan, Srebro, Nathan, and Tropp, Joel. Practical large-scale optimization for max-norm regularization. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R.S., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 23*, pp. 1297–1305. 2010.

McFee, B. and Lanckriet, G. R. G. Heterogeneous embedding for subjective artist similarity. In *Tenth International Symposium for Music Information Retrieval (ISMIR2009))*, October 2009.

Schultz, Matthew and Joachims, Thorsten. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2003.

Srebro, Nathan and Shraibman, Adi. Rank, trace-norm and max-norm. In Auer, Peter and Meir, Ron (eds.), *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pp. 545–560. Springer, 2005. ISBN 3-540-26556-2.

Xing, Eric P., Ng, Andrew Y., Jordan, Michael I., and Russell, Stuart. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pp. 505–512. MIT Press, 2003.