The Power of Comparative Reasoning

Jay Yagnik, Dennis Strelow, David A. Ross, Ruei-sung Lin
{jyagnik, strelow, dross, rslin}@google.com

Abstract

Rank correlation measures are known for their resilience to perturbations in numeric values and are widely used in many evaluation metrics. Such ordinal measures have rarely been applied in treatment of numeric features as a representational transformation. We emphasize the benefits of ordinal representations of input features both theoretically and empirically. We present a family of algorithms for computing ordinal embeddings based on partial order statistics. Apart from having the stability benefits of ordinal measures, these embeddings are highly nonlinear, giving rise to sparse feature spaces highly favored by several machine learning methods. These embeddings are deterministic, data independent and by virtue of being based on partial order statistics, add another degree of resilience to noise. These machine-learning-free methods when applied to the task of fast similarity search outperform state-of-the-art machine learning methods with complex optimization setups. For solving classification problems, the embeddings provide a nonlinear transformation resulting in sparse binary codes that are well-suited for a large class of machine learning algorithms. These methods show significant improvement on VOC 2010 using simple linear classifiers which can be trained quickly. Our method can be extended to the case of polynomial kernels, while permitting very efficient computation. Further, since the popular MinHash algorithm is a special case of our method, we demonstrate an efficient scheme for computing MinHash on conjunctions of binary features. The actual method can be implemented in about 10 lines of code in most languages (2 lines in MATLAB), and does not require any data-driven optimization.

1. Introduction

Rank correlation measures have been well regarded as robust measures in many performance evaluation schemes. Since these methods rely on relative ordering of elements, they are very resilient to noise and variations that do not affect the implicit order. We make the case of applying this thought to feature representation. High-dimensional feature sets are quite common across all disciplines of signal analysis and many other domains. Precise values of each feature dimension in such high-dimensional spaces are often not important. We argue for creating representations that are based solely on the relative rank ordering of feature dimensions. Such representations would enjoy all the stability benefits of rank correlation measures while being useful to generate discriminative features. Further, we base our embeddings on multiple partial order statistics rather than total orderings, giving us another degree of resilience to noise and giving rise to representations that have local support on feature dimensions (useful for learning algorithms to distinguish "useless" dimensions if needed).

The main contribution of our paper is the Winner Take All (WTA) hash, a sparse embedding method that transforms the input feature space into binary codes such that Hamming distance in the resulting space closely correlates with rank similarity measures. Algorithm 1 gives the WTA hash and Figure 1 shows it operating on four example input vectors. In short, for each hash we permute the input feature vector with Θ, take the first K components from the permuted vector, and output the index of the maximum component. Many hashes corresponding to different Θ can be combined into an output hash vector.

This thought and simple methods that result from it are very widely applicable. Our embedding method requires no data-driven optimization. While being training-free and easy to compute it outperforms state-of-the-art methods on several tasks. Our method gives rise to a space of sparse binary vectors. This makes it readily amenable for many problems including:

- Using sparse binary vectors as tokens/hashes in Locality Sensitive Hashing schemes for fast similarity search. Here we show that our method outperforms several state-of-the-art machine learning methods [20, 19] for learning hash codes that are optimized for the specific problem of similarity search on LabelMe. The performance gap is significant, particularly when doing sub-linear (approximate) nearest neighbor search.

- Using our embeddings to induce a ranking metric on well known descriptors for similarity search. Here we show that our method improves on SIFT [21] and DAISY [26, 27] by about 11-12% (error rate for 95% recall) on standard benchmarks.
- Our embeddings act as a nonlinear feature-space transformation. When applied with linear classifiers they outperform linear and chi-square kernel classifiers on vocabulary histogram features.

Furthermore we make the following algorithmic contributions :

- We can compute rank embeddings of polynomial spaces of degree p in $O(p)$.
- Our method can bias the rank embeddings to be more sensitive to elements at the head of the rank list. This

is often argued as a very desirable property in rank-correlation [5, 6, 4] and makes intuitive sense in feature representation too (e.g. histogram bins with high counts might be more relevant than ones with lower values).

- We show that the well known MinHash [11, 12] is a special case of our method when applied to binary features. This means our polynomial rank embedding scheme is effectively a way to compute MinHash on p^{th}-order conjunctions of binary variables in $O(p)$ time (as opposed to $O(n^p)$), which makes for a very strong combinatorial result.
- Our embeddings are based on partial orders, in contrast to well known rank correlation methods, these give machine learning systems an opportunity to set preferences on partial orders that are important for a given problem, a fact that is quite visible in our empirical results on classification problems compared to baseline methods [30].
- Our method is based completely on random permutations requiring no data-driven learning (except tuning 2 hyperparameters).

We begin with a review of related work that employs ordinal measures. Next we discuss rank correlation metrics, showing how one of the simplest metrics motivates our proposed feature transform, the "Winner Take All" hash family. Algorithmic details of the transform appear in the following section. Finally we conclude with experimental results using WTA features for image retrieval, feature descriptor matching, and image classification, followed by a brief discussion.

2. Related Work

Ordinal measures have been studied to some extent in specific problems. [1] uses a permutation-distance-driven ordinal measure for pixel correspondence in stereo matching. Several papers talk about using pairwise pixel comparisons or related quantities for pattern matching [7, 13, 14, 15, 16, 17]. [10] builds an approximation to cosine distance based on concomitant rank orders. The basic strength of these methods comes from using pixel pair representations as features. Although there have been several papers using these features, they are often not emphasized as the core part of the system or there hasn't been any theoretical justification on why they should be used. Recent work relating to large scale nearest neighbor search has also examined the idea of approximation using comparisons rather than metric distances. However, their study is restricted to approximating nearest neighbors given a distance metric, which they relax to approximating top elements in rank order that should be returned from such large scale lookups. [9, 8] study properties of such algorithms that utilize a comparison oracle. Results from these and related works apply directly to kernel versions of our formulation with the added advantage that our method allows a specification of decay curves with respect to rank in the retrieval space that allows a) packing of multiple constraints in a code (see Section 4) and b) preference for selecting top scoring neighbors in similarity (see Figure 2).

3. Rank Correlation Spaces

As mentioned, when dealing with high-dimensional features, the precise value of each feature dimension is often not required to solve the given problem (classification, matching etc.). [2] argues that the "curse of dimensionality" leads to sparse sampling of high-dimensional spaces and that even strong norms (in terms of volume growth properties) like L_∞ exhibit dynamic range compression in the distances (which implies reduced variance on distribution of pair-wise distances and higher susceptibility to noise in each dimension). More commonly used distance metrics like Euclidean suffer much more in this scenario. This also implies that the contribution of any individual dimension to the final distance is small in well-normalized feature vectors under metrics like $L2$ and noise in each dimension can effectively add up to degrade the metric. Rank correlation measures are known for their stability to perturbations in numeric values while giving a good indication of inherent similarity / agreement between items / vectors being considered.

Our goal is a feature space transformation that results in a space that is not sensitive to the absolute values of the feature dimensions but rather on the implicit ordering defined by those values. In effect the similarity between two points is defined by the degree to which their feature dimension rankings agree. The simplest of these pairwise-order measures can be defined as below.

$$PO(X,Y) = \sum_i \sum_{j<i} T\left((x_i - x_j)\left(y_i - y_j\right)\right) \qquad (1)$$

where x_i and y_i are the i^{th} feature dimensions in $X, Y \in R^n$ and T is simply a threshold function

$$T(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Equation 1 simply measures the number of pairs of feature dimensions in X and Y that agree in ordering. Pairwise-order is a similarity measure and can easily be converted to a measure of distance. [3] presents a good analysis of various ranking based measures of disarray (distance) and shows that they satisfy metric properties. In particular [3] shows that variance of pairwise-disarray estimators grows as $O(n^3)$ where n is the dimensionality of the feature vector over which rank agreement is computed. Analysis presented in [2] implies that the metric space resulting from this transformation does not accentuate the effects of "curse of dimensionality" compared to the original vector space.

Another observation that is often made [4, 5, 6] in context of ranking is that agreement among high ranking co-

efficients in a list is often more important than those further down the list. We propose a framework that allows for such weight decay effects to be incorporated. Finally the main contribution of our paper is a sparse embedding method that transforms the input feature space into binary codes such that Hamming distance in the resulting space closely correlates with rank similarity measures. Such an embedding can be generated without any machine learning and shows superior performance on a wide variety of tasks. The embedding is highly nonlinear by construction and further allows learning algorithms (e.g. for classification tasks) to learn parameters that are local to different parts of the ordinal space. The sparse codes being binary, can also be used directly for indexing purposes for large scale nearest neighbor lookups, we show that this simple technique outperforms state-of-the art machine learning based methods on this task. Specifically we show later that our method is a generalization of the well-known MinHash [11, 12] and should enjoy the theoretical benefits of LSH schemes discussed in [12, 11, 9, 18].

If we regroup the pairwise summations with respect to ranks, then our simple pairwise-order function PO can be rewritten in the following form:

$$\begin{aligned} PO(X,Y) &= \sum_i \sum_{j<i} T\left((x_i - x_j)\left(y_i - y_j\right)\right) \\ &= \sum_i R_i\left(X,Y\right) \end{aligned} \tag{2}$$

where

$$R_i\left(X,Y\right) = \mid L\left(X,i\right) \cap L(Y,i) \mid \tag{3}$$

$$L\left(X,i\right) = \{j \mid X(i) > X(j)\} \tag{4}$$

Equation 2 groups pairwise agreement terms by one of the indices in the pair. Here $R_i(X,Y)$ basically measures the ranking agreement for index i with all the indices that rank below i. To do so we denote with $L\left(X,i\right)$ the indices of elements in X that are ranked below index i. The rank agreement at index i is the cardinality of the intersection of the corresponding L sets from X and Y. For the example given in Figure 1, we want to compute $PO(X,Y)$ between vectors specified in (a) and (b). The term $R_0(X,Y)$ will measure the size of the intersection set of indices smaller than index 0. $L(X,0) = \{1,2,3,5\}$ are the set of indices in (a) that have values smaller than that at index 0, similarly for Y, $L(Y,0) = \{3,4,5\}$ which gives $L\left(X,i\right) \cap L(Y,i) = \{3,5\}$leading to $R_0(X,Y) = 2$. Eq 2 rearranges all unique pairwise summations into intersections of these "less than" lists. The inner summation instead of covering all $j < i$ now covers all j s.t. $X(j) < X(i)$ and the result is the same since in both cases we do cover all unique pairs (i,j).

We now discuss a generalization of pairwise-ordering agreement that puts a steep nonlinearity on $R_i\left(X,Y\right)$ such that larger intersection sets are given more weight.

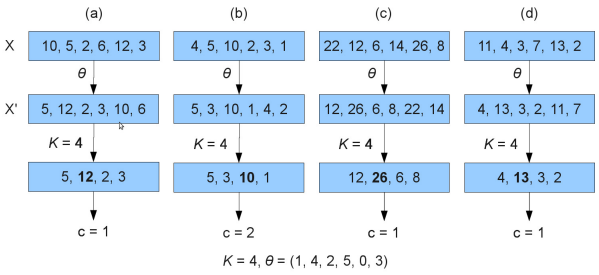


Figure 1: An example with 6-dimensional input vectors, K = 4, and θ = (1, 4, 2, 5, 0, 3). X in (a) and (b) are unrelated and result in different output codes, 1 and 2 respectively. X in (c) is a scaled and offset version of (a) and results in the same code as (a). X in (d) has each element perturbed by 1 which results in a different ranking of the elements, but the maximum of the first K elements is the same, again resulting in the same code.

Algorithm 1 WTA Hash

Input: A set of m Permutations Θ, window size K, input vector X.
Output: Sparse vector of codes C_X.

1. For each permutation θ_i in Θ.
   (a) Permute elements of X according to θ_i to get $X^{'}$.
   (b) Initialize i^{th} sparse code c_{x_i} to 0.
   (c) Set c_{x_i} to the index of the maximum value in $X'(1...K)$
      i. For $j = 0$ to $K-1$
         A. If $X^{'}(j) > X^{'}(c_{x_i})$ then $c_{x_i} = j$.
2. $C_X = \left[c_{x_0}, c_{x_1}, ..., c_{x_{m-1}}\right]$, C contains m codes, each taking a value between 0 and $K-1$.

4. WTA Hash family

4.1. Intuitive Explanation

We'll now discuss the details of the transformation technique. Figure 1 illustrates the basic transformation. Algorithm 1 outlines the transformation method where the permutations in the algorithm are generated randomly and stored for use by all data points. The method relies on coding multiple partial orderings of the data point as a way to lower bound the ranking agreement in case of a match. Essentially the method picks K dimensions at random from the sample and codes the dimension with the highest value in the subset. Note that the randomization is without replacement and has to be consistent across all samples and hence the notion of permutations. Thus equality in the codes c_i implies that we increase our estimate of Eq. 1 by $K-1$. If two vectors X and Y have the same value for

output code c_0, lets say $c_0 = a$, for a window size of K, this means that both X and Y match in $K-1$ inequalities that state $X'(a) > X'(i)$ and $Y'(a) > Y'(i)$ for all $i : 0 \leq i < K, i \neq a$. So $T\left(\left(x_a^{'} - x_i^{'}\right)\left(y_a^{'} - y_i^{'}\right)\right) = 1$ for all $i : 0 \leq i < K, i \neq a$ which gives us a progressively better lower bound on pairwise-order agreement (Eq. 1) between X and Y as more codes match. Since we are only encoding the "winner" in each subset according to a given criterion (max in this case) we call the resulting family of hash functions "Winner Take All" (WTA). Other families encoding multiple points from a partial order are possible, however for the scope of this work we only analyze the WTA family.

First we look at the simple case of $K = 2$. Algorithm 1 will pick two indices at random and generate a code c_i which is 0 if $\theta_i(0) \geq \theta_i(1)$ and 1 otherwise. So for $K = 2$ the algorithm codes pairwise inequalities as bits. When we compute Hamming similarity (number of matching bits) between two vectors X and Y that are coded through this method, the result is number of pairwise-order agreements between X and Y (which is the same as Eq. 2). Now for $K > 2$, algorithm 1 codes the index of the max element in a K-sized subset of feature dimensions. To see why this is meaningful, lets look at the example in Figure 1. Here $K = 4$, vectors (a), (c) and (d) are related while (b) is random. When we apply the permutation θ we get vectors $X^{'}$ given in the second row. We then look at the first K elements in the permuted vectors (given in row 3). The code is the index of the max element in this permuted form. Now $c = 1$ for (a) and (c) implies that index 1 was the max among the 4 indices in both the vectors. Which means that both (a) and (c) satisfy 3 inequalities, namely $X'(1) > X^{'}(0)$, $X'(1) > X^{'}(2)$ and $X'(1) > X^{'}(3)$. When we compute pairwise-order agreement between (a) and (c) (Eq 1), these 3 terms will add positive contribution. This implies that equality in code c_i adds $K-1$ to our estimate of PO in Eq. 1, effectively acting as a lower bound. If we generate a large number of codes, then the bound becomes tighter as all possible pair combinations are considered.

Another aspect of this method merits a discussion. Our choice of K leads to different emphasis on pair-wise agreements for indices at the head of the list. For example, consider the degenerate case in which $K = N$, where n is the dimensionality of the feature vector. Every permutation simply encodes the global max element, so $n-1$ inequalities that relate the max element to all the others would be captured. (In general each permutation encodes $K-1$ inequalities relating to the max within first K elements of that permutation.) Hence $K = n$ puts complete emphasis on the head of the list. On the other hand we saw that $K = 2$ does not put any bias on the head as all pairs are encoded. Values of K between 2 and n lead to a progressively steeper bias to the head elements. We formally analyze the dependence below.

4.2. Mathematical Analysis

Note that window size K has an influence in the distribution of which indices from the original vector end up in the code. It is easy to see that for $K = n$ where n is the dimensionality of the feature vector, we get only the global max of the vector in the code, and $K = 1$ follows a uniform distribution since there is no comparison involved. While both these codes are not useful in practice, they give us an idea of the distributions being induced. Lets analyze the more interesting case of $1 < K < n$.

First let us define an index j that represents the positions in the sorted vector of dimensions. For a window size K, the probability that max $\theta(c_i)$ over K elements lies at an index on or after j is given by

$$P_K\left(Rank\left(\theta\left(c_i\right)\right) > j\right) = \frac{\binom{n-j}{k}}{\binom{n}{k}}$$

So to derive $P_K\left(Rank\left(\theta\left(c_i\right)\right) = j\right)$ for $0 \leq j < n-K$ we difference the cumulative distribution followed by some simplifications that are easy to work out. Also $P_K\left(Rank\left(\theta\left(c_i\right)\right) = j\right) = 0$ for $j > n-K$.

$$\begin{aligned} P_K\left(Rank\left(\theta\left(c_i\right)\right) = j\right) &= P_K\left(Rank\left(\theta\left(c_i\right)\right) > j\right) - \\ &\quad P_K\left(Rank\left(\theta\left(c_i\right)\right) > (j+1)\right) \\ &= \frac{\binom{n-j}{K}}{\binom{n}{K}} - \frac{\binom{n-j-1}{K}}{\binom{n}{K}} \\ &= \frac{\binom{n-j}{K}}{\binom{n}{K}} . \frac{K}{n} \\ &= \frac{K}{n} \prod_{l=0}^{(K-1)} \left(1 - \frac{j}{n-l}\right) \end{aligned} \quad (5)$$

Figure 2 plots $P_K\left(Rank\left(\theta\left(c_i\right)\right) = j\right)$ for different values of K. The value of K gives us an easy knob to tune for giving higher weight to top elements in the vectors vs. the others. Assuming that lots of permutations are generated the final similarity function induced in the limit would be given by

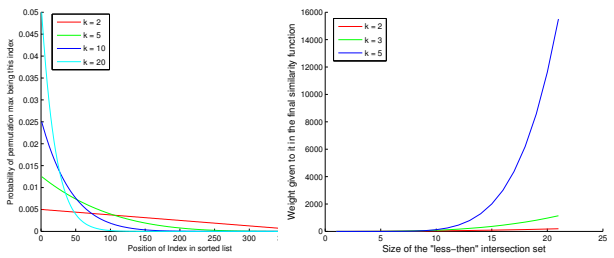
$$S_K\left(X, Y\right) = \frac{\sum_{i=0}^{n-1} \binom{R_i(X,Y)}{K-1}}{\binom{n}{K}} \quad (6)$$

where $R_i\left(X, Y\right)$ is given by Eq. (3) and (4).

Let us denote by i indices into vectors X and Y while $\theta(i)$ will denote indices into permuted vectors $X^{'}$ and $Y^{'}$. If index i is chosen as the hash code for for a given random permutation it implies

$$X(i) > X(j)\, \forall j\, 0 < \theta(j) < K, j \neq i$$

which means that all the other indices in the permutation are members of the set $L(X, i)$. If i is chosen as the hash code for both X and Y (for a fixed random permutation θ) it means that all the indices $0 < \theta(j) < K$ belong to both $L(X, i)$ and $L(Y, i)$. By implication the number of ways codes for a permutation θ on vectors X and Y can collide on value i is given by



(a) (b)

Figure 2: (a) Rate of sampling a given index from the sorted list. The curves show different decay factors depending on the value of K. (b) Contribution to similarity for a given value of $R_i(X,Y)$ for different values of K.

$$N(X,Y,i) = \binom{L(X,i) \cap L(Y,i)}{K-1} = \binom{R_i(X,Y)}{K-1}$$

The total number of ways codes for a permutation θ on vectors X and Y can collide on any code is given by

$$N(X,Y) = \sum_i N(X,Y,i) = \sum_i \binom{R_i(X,Y)}{K-1}$$

K leading indices of a permutation θ can be drawn from n dimensions in $\binom{n}{K}$ ways. So the probability of collision of codes given by a random permutation θ on vectors X and Y is given by

$$S_K(X,Y) = \frac{\sum_{i=0}^{n-1} \binom{R_i(X,Y)}{K-1}}{\binom{n}{K}}$$

This also proves the LSH property for our codes where the probability of collision induces a family of distance functions that are approximated.

An intuitive interpretation of $S_K(X,Y)$ is as follows. The number of ways in which index i can be the max over a K-sized permutation is given by the number of ways in which one can pick $K-1$ elements that are smaller than the element at i and common to both X and Y. Notice that for $K=2$ we get a normalized version of Eq. 2 back. However for larger values of K this family of hash functions strongly prefers intersection sets with large cardinality (with the corresponding indices following distribution given by Eq. 5). Figure 2 illustrates the nonlinear effect of K on amplifying $R_i(X,Y)$ for different values of K.

4.3. Polynomial kernel extension

Apart from allowing preferential treatment to top elements, the simple structure of the algorithm also allows us to perform the same operations in kernel spaces. Simply transforming the original feature vector to a vector that reflects kernelized distances over data samples allows us to use the same framework. WTA coding in such transformed

Algorithm 2 WTA Hash on polynomial kernels.

Input: A set of m Permutations Θ, window size K, Polynomial kernel degree p, input vector X.
Output: Sparse vector of codes C.

1. For each set of p permutations $\theta_{(i,l)}$ $1 \leq l < p$ in Θ.
   (a) Permute elements of X according to each of $\theta_{(i,l)}$ to get $X_l^{'}$.
   (b) Initialize i^{th} sparse code $c_i$to 0.
   (c) For $j = 1$ to $K-1$
      i. Set $X^{'}(j) = \prod_{l=1}^{p} \left(X_l^{'}(j)\right)$.
      ii. If $X^{'}(j) > X^{'}(c_i)$ then $c_i = j$.

spaces has properties related to many algorithms proposed in the combinatorial framework presented in [8], particularly if one has knowledge of the disorder constant D, it can be use to pick the decay curve parameter K in our algorithm. Small disorder constants would imply more stability of top ranking neighbors leading to high value of K, while large values of D would imply the opposite. Kernel-WTA allows us to exploit these rank distances in simple LSH style retrieval setup and by virtue of being based on a comparison oracle enjoy the theoretical benefits described in [8, 9]. Algorithms described in[9] are special cases of this framework where $K = 2$.

However this framework can support another special class of kernels very efficiently, namely rank correlation kernels over polynomial expansion of the feature space. We can compute WTA code on a degree p polynomial space of a feature vector in $O(p)$ time, basically requiring p permutations per code rather than 1 in the previous case. Algorithm [2] illustrates this process. Its easy to show that this is equivalent to computing the expanded polynomial space and then computing WTA codes on it, since every p-degree product is equally likely at step 1.(c).i in algorithm 2. This means we can directly compute permutations over polynomial spaces by taking products of permutations in the original space. The extension opens up our framework to working with arbitrarily large polynomial spaces while maintaining the efficiency of the coding process. Note that we have dropped the coefficients relevant to the polynomial expansion here to make feature-conjunctions more explicit, one can easily add the polynomial kernel coefficients by keeping an auxiliary hash map that counts the number of times each variable is seen without violating the $O(p)$ claim.

4.4. The connection to MinHash

It is also important to note that the popular MinHash [11, 12] is a special case of WTAHash when applied to binary vectors. A simple view of MinHash is that it encodes the index of the first 1 under random permutations of binary

vectors. It has been shown that the hash collision rate corresponds to the Jaccard similarity between binary vectors. We now show that this is a special case of our method.

If the input vector X in Algorithm 1 is binary, then step 1.(c) will retain the index of the first 1 that it sees in the given random permutation. This happens because we use a strict $>$ operator in step $1.(c).i$ in Algorithm 1, so when the first 1 is encountered c_i is set to that index, following 1s don't change c_i because of the strict inequality. For the equivalence to hold we must set $K = n$ so that we avoid the case of having all 0 values in the permutation (although one can think of a "clipping" version of MinHash where this is allowed with a small probability and is encoded by code 0). Note that $K = n$ for binary vectors has a different effect than that on numeric vectors because binary vectors have several global max indices with value 1.

In this context Algorithm 2 contributes an efficient method to compute MinHash on p^{th}- order conjunctions of binary vectors. The product terms in 1.(c).i are conjunctions when dealing with binary vectors. This is a very powerful result, since it allows us to compute MinHash over a combinatorial space of binary hypercubes in logarithmic time (where the conjunction order p is the log of the size of resulting space).

4.5. Implementation

Given a permutation vector theta. Each individual code can be computed in MATLAB with the 1-liner below:

theta = randperm(n);

[max_val, c(i)] = max(X(i,theta(1:K)));

One can as easily compute multiple codes by packing multiple permutations in a matrix Theta and computing max over all of them simultaneously. In C++ this would imply two for loops, with retaining the maximum index, which can also be done in less than 10 lines.

5. Similarity Search in High Dimensions

5.1. Image retrieval on LabelMe

We begin by demonstrating the performance of WTA codes on the exact experiment described in [20, 19]. As in [19], ground truth similarity is obtained by thresholding the L2 distance between these Gist vectors. Although the full rank ordering produced by Euclidean distance on Gist is unlikely to agree with ground truth semantic similarity, the top ranked vectors do represent a labeled set in which we can be reasonably confident. The experimental dataset, used in [19], consists of approximately 13,500 image thumbnails from the LabelMe dataset. Each image is represented using a 512-dimensional Gist feature vector. The dataset was divided into a training set containing 80% of the samples, and a test set containing the remainder. For each test sample, the nearest neighbors (based on Hamming distance between codes) were found from amongst the training samples, and performance was evaluated by measuring the precision and recall. Figure 3 compares WTA codes generated from Algorithm 2 with $p = 4$ and $K = 4$ (values for p and K were picked by line search along K followed by p in log scale on a hold out set). Even for small codes starting from 64 bits, WTA codes outperform Spectral Hashing (and by transitivity, Restricted Boltzmann Machines and Boosting) [19] as well as PCA Hash codes[22, 23] but are slightly worse than SPEC codes from [20]. With more bits WTA clearly performs better than the competitors.

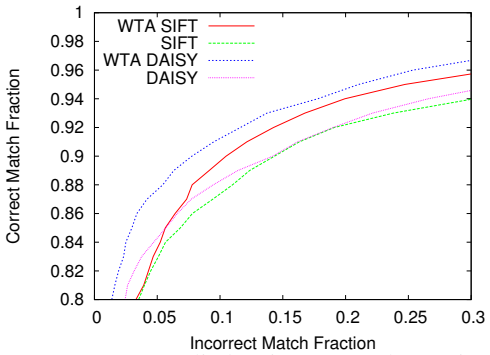


Figure 4: WTA applied to image patches at interest points for similarity search.

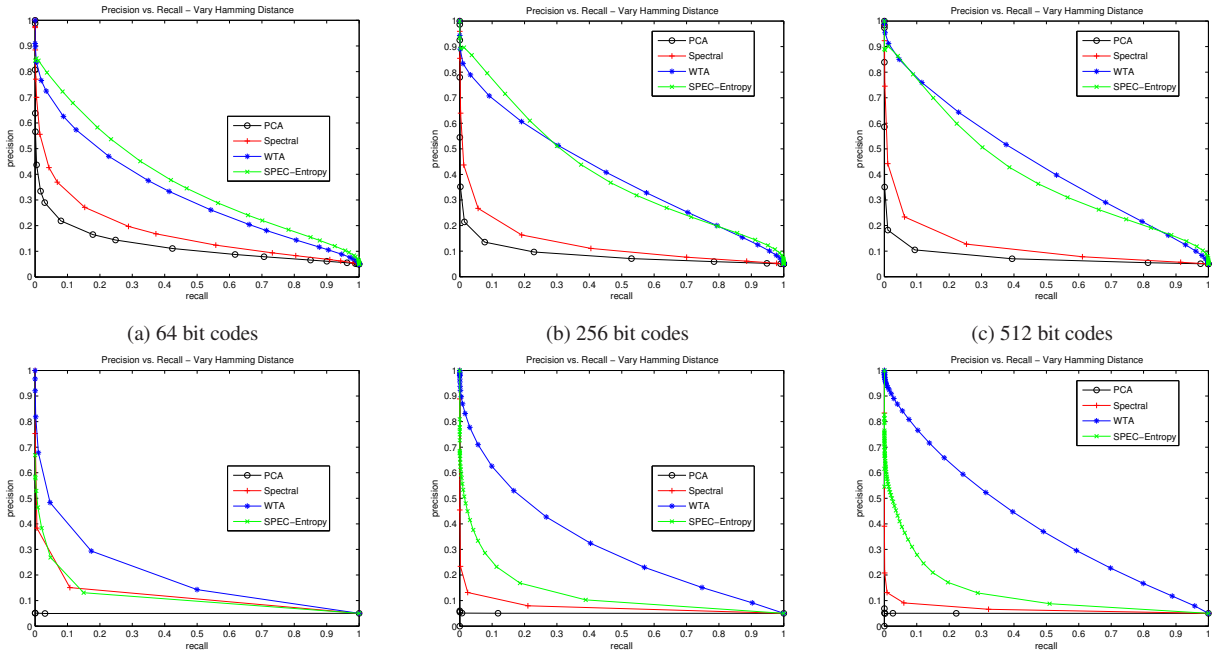
Real large scale systems can be required to index billions of items. This necessitates an increase in the code length, to preserve precision, as well as a matching strategy such as LSH that is more efficient than pairwise comparison. To estimate performance on a LSH retrieval task we use each byte of the hash code as an inverted index table. This allows us to reduce database search by $\frac{1}{256}$ in the best case. Typically each inverted index table would have a key consisting of several bytes, but since the number of images here has a small number of true near neighbors the performance will be representative. Each probe is queried in multiple tables and the resulting lists are "joined" to count the number of band hits for scoring each retrieved pair. Figure 3 illustrates the performance of the above methods on this task.

5.2. Matching local feature descriptors

We consider WTA's performance on the descriptor matching task described in [24]. In this task, we are given 5,000 pairs of matching image patches and 5,000 pairs of mismatched image patches, where each patch is 64 x 64 pixels. The goal is to compute an image descriptor from the patches that gives the best false match rate at 95% recall. Specifically, we look at [24]'s Liberty dataset, which is the harder of the two datasets they present (Liberty and Notre Dame).

We use the raw image pixels ($64^2 = 4096$ dimensions), SIFT [21] (128 dimensions) and DAISY [26, 27] as baseline descriptors, each with the Euclidean distance. These give 68.0% , 36.2% and 32.4% false matches at 95% true matches, respectively. As WTA descriptors, we use 10,000 codes computed from the raw image pixel or SIFT and DAISY descriptors[1], with the Hamming distance, we set

[1]Since the DAISY descriptors integrate over log-polar sampling of the



(a) 64 bit codes (b) 256 bit codes (c) 512 bit codes

(d) 64 bit codes - 8 1B tables (e) 256 bit codes - 32 1B tables (f) 512 bit codes - 64 1B tables

Figure 3: (a)-(c) show pairwise comparison performance of WTA codes on LabelMe dataset with $p = 4$ and $K = 4$. (d)-(f) show LSH lookup performance of WTA codes on LabelMe dataset with $p = 4$ and $K = 16$.

$K = 2$. The raw image pixel WTA descriptors give 56.4% false matches at 95% true matches, beating the raw image descriptor by 11.6%. [24] reports 35.1 for SIFT on Liberty in Table 1; our WTA hash applied to SIFT descriptors gives 24.7% false matches at 95% true matches, beating SIFT alone by 10.4%. When applied to the DAISY descriptor we get 21% leading to a gain of 11.2% from baseline. Figure 4 is similar Figure 6 in [24] (which reports for Notre Dame rather than Liberty), and shows the performance of SIFT, DAISY, WTA SIFT and WTA DAISY over a false positive, true positive range where their performance curves are visually distinguishable. Note that higher order polynomials were not tried for this experiment.

6. Nonlinear transformation for classification

We consider WTA codes for classifying PASCAL VOC 2010 images. VOC 2010 contains images from 20 classes, including people, animals (e.g. bird), vehicles (e.g. aeroplane), and indoor objects (e.g. chair), and are considered natural, difficult images for classification.

In the VOC 2007 and 2008 competitions, [30] achieved the state-of-the-art using multiple histogram types (e.g., SIFT bags-of-words quantized using hierarchical k-means, color histograms), an SPMK-like spatial breakdown of the image, and a nonlinear chi-squared SVM. In VOC 2009, [29] won with more sophisticated coding of SIFT descriptors (e.g., local coordinate coding) and a simpler linear SVM. We don't compare directly to those methods here. Instead, we apply our hash to a baseline method (a whole-image bag-of-words of local descriptors passed to a linear SVM), showing that it can give a strong relative improvement in some classification applications.

From each of these bag-of-words, we create WTA feature vectors including $10^{0.5}, 10^{1}, 10^{1.5}, ..., 10^{5}$ codes. We used $K = 4$, which we found to work well for in some initial experiments. To create the bags-of-words we extract local descriptors on a dense grid over the image and quantize them using hierarchical k-means [25]. Our local descriptors measure Gabor wavelet responses at 4 orientations and 27 positions relative to the descriptor center, for a total of 108 dimensions. The 27 positions (including scale and spatial offset) are a log-polar configuration similar to the DAISY descriptor's [26, 27]. We consider bags-of-words with $36, 100, 1000$ and 10000 bins.

For each bag-of-words and WTA feature vector we train a linear classifier for each of the 20 VOC classes using liblinear [28] and the VOC 2010 training set. Prior to training on the full training set we find a best SVM C parameter using 10-fold cross-validation. We evaluate the resulting classifiers on the VOC 2010 validation set. This approach is consistent with the "best practices" on the VOC website, which suggest cross-validation over the training set for VOC 2010 dataset when reporting results for a range of parameter choices.

image patch, it has scale bias across its dimensions. We pre-process the descriptor by substracting the mean and dividing by the variance along each direction to make the scales more uniform before applying WTA.

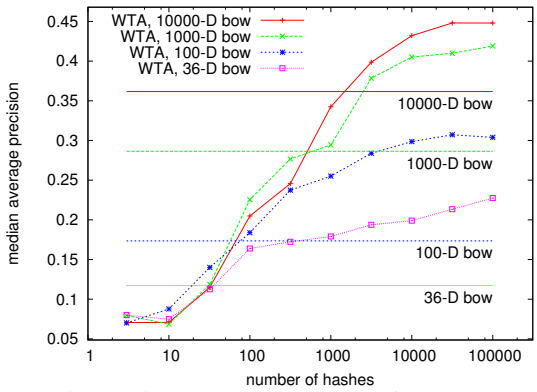


Figure 5: Median Average Precision for VOC 2010. The four horizontal lines from low to high are baselines for each of the 36-10000 bag of words features respectively.

The results are summarized in Figure 5, which shows the median average precision over the 20 classes for each of the feature vectors. The median average precision for the four bags-of-words are shown by the horizontal lines. The median average precision for each WTA feature vector are shown by the curves. Performance increases both as a function of the original bag-of-words dimension and the number of codes, with WTA significantly beating the bag-of-words in each case. Further, WTA from bags-of-words with X bins beats the bag-of-words with 10X bins for a sufficiently large number of hashes. Note that we gain 8-13% absolute over the baseline with a simple feature space transformation without any change to the underlying machine learning setup. An SVM with a chi-squared kernel performs at approximately 40.1% median average precision, for a 1000 bag-of-words model. In comparison, using WTA plus a simple linear classifier, the performance of our system is approximately 2% higher.

7. Conclusion

We make the case for feature representations that encode relative orderings of feature dimensions. We showed that these embeddings enjoy all the theoretical benefits of comparative (rank correlation) methods, are stable to perturbations and outperform several state-of-the art machine learning methods while being training-free. We emphasized that these representations are highly nonlinear and can be combined with simple linear classifiers to get state-of-the-art results. On similarity search tasks WTA outperforms well-tuned machine learning systems and hand crafted descriptors. We believe that several other advancements are possible within this framework, which essentially operates on comparisons rather than absolute values.

In future work we hope to analyze other classes of hash functions arising out of the partial ordering framework.

References

[1] D. Bhat, S. Nayar. Ordinal Measures for Visual Correspondence. In CVPR'96. 2

[2] J. Friedman. An Overview of Predictive Learning and Function Approximation. In From Statistics to Neural Networks'94. 2

[3] P. Diaconis, R. Graham. Spearman's footrule as a measure of disarray. In J. Roy. Statistical Society'77. 2

[4] A Maturi, E. Abdelfattah. A New Weighted Rank Correlation. In J. Mathematics and Statistics'08. 2

[5] J. Pinto da Costa, C. Soare. A Weighted Rank Measure of Correlation. In ANZJS'05. 2

[6] G. Shieh. A weighted Kendall's tau statistic. In Statistics & Probability Letters'98. 2

[7] M. Ozuysal, P. Fua, V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In CVPR'07. 2

[8] Y. Lifshits, S. Zhang. Combinatorial Algorithms for Nearest Neighbors, Near-Duplicates and Small World Design. In SODA'09. 2, 5

[9] D. Tschopp, S. Diggavi. Approximate nearest neighbor search through comparisons. In ArXiv preprint'09. 2, 3, 5

[10] K. Eshghi and S. Rajaram. Locality sensitive hash functions based on concomitant rank order statistics. In SIGKDD'08. 2

[11] A. Broder. On the resemblance and containment of documents. In SEQUENCES'97. 2, 3, 5

[12] A. Broder, M. Charikar, A. Frieze, M. Mitzenmacher. Min-wise independent permutations. In STOC '98. 2, 3, 5

[13] S. Baluja, H. Rowley. Boosting Sex Identification Performance. In IJCV'07. 2

[14] M. Ozuysal, V. Lepetit, F. Fleuret, P. Fua. Feature harvesting for tracking-by-detection. In ECCV'06. 2

[15] O. Pele, M. Werman. Robust real time pattern matching using bayesian sequential hypothesis testing. In PAMI'08. 2

[16] R. Zabih, J. Woodfill. Non-parametric local transforms for computing visual correspondence. In ECCV'94. 2

[17] V. Lepetit. Keypoint recognition using randomized trees. In PAMI'06. 2

[18] P. Indyk, R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In STOC'98. 3

[19] Y. Weiss, A. Torralba, R. Fergus. Spectral Hashing. In NIPS'09. 1, 6

[20] R. Lin, D. Ross, J. Yagnik. SPEC Hashing: Similarity Preserving algorithm for Entropy-based Coding. In CVPR'10. 1, 6

[21] D. Lowe. Object recognition from local scale-invariant features. In IJCV'99. 1, 6

[22] B. Wang, Z. Li, M. Li, W.Y. Ma. Large-scale duplicate detection for web image search. In ICME'06 6

[23] X.J. Wang, L. Zhang, F. Jing, and W.Y. Ma. Annosearch: Image auto-annotation by search. In CVPR'06. 6

[24] S. Winder, G. Hua, M. Brown. Picking the best DAISY. In CVPR'09. 6, 7

[25] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In CVPR'06. 7

[26] E. Tola, V. Lepetit, P. Fua. A fast local descriptor for dense matching. CVPR'08. 1, 6, 7

[27] E. Tola, V. Lepetit, P. Fua. Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo. In PAMI'10. 1, 6, 7

[28] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, C.J. Lin. LIBLINEAR: A Library for Large Linear Classification. In JMLR'08. 7

[29] K. Yu, T. Zhang, Y. Gong. Nonlinear Learning using Local Coordinate Coding. In NIPS'09. 7

[30] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. In IJCV'07.

2, 7