A Closer Look at Boosted Image Retrieval

Nicholas R. Howe

Smith College, Northampton, Massachusetts

Abstract. Margin-maximizing techniques such as boosting have been generating excitement in machine learning circles for several years now. Although these techniques offer significant improvements over previous methods on classification tasks, little research has examined the application of techniques such as boosting to the problem of retrieval from image and video databases. This paper looks at boosting for image retrieval and classification, with a comparative evaluation of several top algorithms combined in two different ways with boosting. The results show that boosting improves retrieval precision and recall (as expected), but that variations in the way boosting is applied can significantly affect the degree of improvement observed. An analysis suggests guidelines for the best way to apply boosting for retrieval with a given image representation.

1 Introduction

The fields of machine learning and visual information retrieval have independently each seen gratifying research progress of late. Boosting [4], support vector machines [1] and other so-called *large-margin techniques* consistently demonstrate improved performance when applied on top of older, more established classification methods from machine learning. Simultaneously, researchers in the field of image and video retrieval have devised new representations that allow quick comparisons between images based upon multiple cues – color and texture distributions, for example. Retrieval techniques using automatically extracted feature vectors, such as color correlograms [8], redundant banks of texture filters [3], and others [7] have shown measurable improvements over earlier, more simplistic methods such as color histograms [11]. These two bodies of research combined have the potential to generate powerful image classification and retrieval algorithms, and video retrieval algorithms by extension. Unfortunately, with a few exceptions [12, 2], very little current research appeals to both fields by incorporating the best elements of each. The combination of the newer image analysis techniques with the concurrent advances in machine learning turns out to contain subtle complexities that have not been adequately addressed to date.

Incorporating boosting into retrieval algorithms necessarily implies moving away from the single-image query typical of many works on information retrieval [3, 7, 8, 11], since boosting requires a *set* of positive and negative examples to work. Fortunately, there is a movement afoot in the field as a whole in this direction, toward approaches that might automatically compare images in a

collection with a library of different models for classification and subsequent retrieval via keywords. A boosted classifier provides a promising candidate model for such a library. If single-image queries are desirable or necessary for some applications, then boosting may still prove useful, employing the top set of retrieved images (hand-classified online by the user) as the training set. In this mode, boosting represents a way to move from simply retrieving a few images related to the query, toward locating the entire set of images of a target class that are available in the database.

While this paper does not attempt to address all these possibilities at once, it does examine the fundamentals of combining boosting with a range of promising image comparison methods. It examines several candidate approaches (as detailed in Section 2), since the best way to incorporate boosting using these techniques has not yet been established. In particular, a novel approach is developed herein using conic decision boundaries that allows boosting to be combined with any of the leading image representations. Section 3 gives the experimental procedure, and a summary of the findings appears in Section 4. The experimental results show the conic-boundary method to work better with the high-dimensional image representations that are becoming more common today.

2 The Problem

Several characteristics of image comparison techniques make the straightforward application of boosting difficult. Image representations typically exhibit a high number of linearly nonseparable dimensions. This makes the use of machine learning mainstays such as C4.5 [9] both slower and less effective than they are in the sorts of problems typically looked at by the learning community. The comparison metrics developed for image retrieval, on the other hand, also form an incomplete foundation for boosting. Oriented towards retrieval rather than classification, they do not address the issue of establishing a classification threshold. More significantly, these techniques are designed to measure image similarities given a *single* target image; they do not necessarily handle a set of target images, possibly with weights indicating their importance. For boosting, incorporating such a weighted set of targets is essential.

The naïve approach to expanding from a single-target technique to a multiple-target technique would be to use some linear combination of the representations of the multiple targets, such as the mean. Unfortunately, this method does not work: typically the combined representation is significantly *worse* at picking out members of the class than many of the individual training examples alone [6]. This reflects the complexity of image classes: they tend to be only diffusely clustered in any given image representation, interspersed with non-members of the class, and rife with outliers. Linear combinations of positive feature vectors typically lie closer to negative examples than to the positive examples they are drawn from, and therefore serve as a poorer basis for classification.

2.1 Boosting in Context

Boosting began as a technique for combining differently-trained classifiers with unique sets of strengths and weaknesses. Properly done, a weighted vote of each classifiers' predictions can reinforce the strengths and cancel out the weaknesses [10]. Thus a classification algorithm that displays marginal success (accuracy slightly better than chance) can be "boosted" into an algorithm with much higher accuracy. AdaBoost [4] first provided a widely known algorithmic approach to boosting. Since then, many variants have appeared that seek to address some of its shortcomings, such as intolerance to errors in the training data [5], but AdaBoost continues to be widely used.

AdaBoost and boosting algorithms in general require a base learning algorithm, often referred to as a *weak learner*, that can classify any set of weighted instances with better than 50% accuracy. With a two-class system, such weak learners are not hard to develop: nearly any division of the space of possible instances will do. Although the theoretical results place only weak requirements on the base algorithm, empirical experience suggests that more powerful base classifiers tend to work better when boosted [4]. The base classifier is trained in successive rounds on different subsets or weightings of the initial training data, producing the required set of differently-trained classifiers that can be combined to produce a final, more reliable classification. Those interested in retrieval applications should note that the algorithm actually produces a numeric score for each instance that when thresholded yields a classification, but which might as easily be used to rank the images for retrieval.

Boosting has been shown to exhibit a number of desirable properties, particularly a resistance to overfitting the training data. Yet in spite of the success of boosting in other areas, little work has been done to date in applying it to images. Tieu and Viola [12] use a feature-selection algorithm equivalent to simple boosting, but the focus of their work is elsewhere. Perhaps one reason that so little attention has been devoted to the topic is that researchers working with images have focused mainly on retrieval rather than classification. Only recently have algorithms developed that offer reasonable classification performance on any but the simplest of image categories.

2.2 Two Approaches

In order to apply boosting to most extant image representations designed for retrieval, one must first decide how to adapt a representation designed for pairwise determination of similarity so as to produce a class decision boundary. In doing so, one may decide to adopt an approach that has more of the flavor found in traditional machine learning, or one may opt instead for an approach that retains more of the flavor of the original image retrieval technique. This paper looks at representatives of both of these paths. The first approach uses simple single-dimension thresholded decision boundaries. Intuitively, it looks for dimensions (a.k.a. *features*) demonstrating exceptional values that happen to be highly correlated with membership in the class. The second approach uses

the entire vector for comparison. It achieves this by using a thresholded cosine metric (measuring the angle between two normalized vectors) to yield classifiers whose decision surfaces are hypercones. Intuitively, it looks for images that are similar enough to the exemplar image according to the cosine metric. One might expect either or both of these approaches to work better, depending upon the underlying image representation chosen.

Regardless of the method, any simple classifier used for boosting must conform to a few simple assumptions. As input it receives two collections of vectors, V_p and V_n, containing respectively the representations of positive and negative training examples of the class to be learned. In addition it receives a collection of weights on these vectors, W_p and W_n, indicating the importance placed upon learning to classify the corresponding training example. From these inputs, the classifier should generate a rule that classifies any image representation $\mathbf{v}$ as either a class member or not; this may be thought of as a function from the space of possible representations $\mathcal{V}$ onto $\{0, 1\}$. Section 3.1 describes several common image feature spaces $\mathcal{V}$ used in this paper.

Two features of most image representations make them somewhat different from many of the types of data typically used with boosting. They tend to be of very high dimension, with some schemes using tens of thousands of dimensions [6, 12]. The correlations between individual dimensions tend to be unknown and presumably highly complicated. Furthermore, any single individual dimension typically has low correlation with any interesting image class: there are few "smoking guns". These considerations have led to the development of the two techniques described in detail below, one which concentrates on individual features, and one which looks at the vector representation as a whole.

Feature-Based Boosting The first method, denoted hereafter as *feature-based boosting*, or *FBoost* for brevity, creates a simple classifier as follows. For each dimension in $\mathcal{V}$, it sorts the values found in V_p and V_n, removing duplicates, to determine a complete set of candidate decision thresholds. It then scores each of these decision thresholds in terms of the weighted error rate it would generate if used as a classifier on the training set. The best threshold is computed for each individual dimension, and the best of these becomes the rule used to classify unknown instances.

Feature-based boosting represents a fairly traditional way to apply boosting. Tieu and Viola [12] adopt this approach in their work. From a machine-learning viewpoint, FBoost is equivalent to using decision trees with a single branch (also called *decision stumps*) as the base classifier. Friedman, Hastie, & Tibshirani [5] present evidence that the simplicity of the decision stumps as compared to full decision trees is unimportant, as it may be counteracted by performing a sufficient number of boosting steps.

Vector-Based Boosting The second method, denoted hereafter as *vector-based boosting*, or *VBoost* for brevity, represents a non-traditional application of boosting concepts with no close analogues known to the author. Hypercones

form the decision surface of the base classifier. Class membership may be quickly determined by thresholding the dot product of the normalized candidate feature vector with the unit vector along the axis of the hypercone. (Although this is equivalent to selecting a single feature under some arbitrary basis transformation, the transformation changes with each round of boosting.)

Implementing boosting effectively with this type of classifier turns out to require some creativity. If only the positive training instances are used as cone axes in creating the weak classifiers, then the resulting set of decision boundaries lacks enough variety for effective boosting. (The algorithm quickly reaches a point where none of the available decision boundaries are of high quality.) On the other hand, allowing any axis at all leaves an infinite number of possible decision boundaries to check, with no guide towards finding the best one. A compromise heuristic is therefore used, with reasonable results produced in practice. The algorithm described below consistently generates individual classifiers with greater than 50% accuracy even after many successive rounds of boosting.

Let $\mathbf{v}_p$ and $\mathbf{v}_n$ be the sum of the vectors in V_p and V_n respectively, as weighted by the weights in W_p and W_n. Consider the hyperplane that bisects the angle between $\mathbf{v}_p$ and $\mathbf{v}_n$. Experience shows that the majority of the weight of positive examples will tend to lie on one side of the hyperplane, while the majority of the weight of negative examples will tend to lie on the other side. (Usually the positive examples are clustered on the $\mathbf{v}_p$ side, but if this is not the case the algorithm simply exchanges the classification labels for that round of boosting.) The dot product with a vector orthogonal to the bisecting hyperplane therefore proves useful in discriminating between positive and negative examples (see Figure 1). The heuristic algorithm calculates the orthogonal vector $\mathbf{v}_\perp$ according to Equation 1 and then computes its dot product with all the training vectors.

$$\mathbf{v}_\perp = \mathbf{v}_p - \frac{\mathbf{v}_n \cdot (\mathbf{v}_p + \mathbf{v}_n)}{\|\mathbf{v}_p + \mathbf{v}_n\|} \tag{1}$$

As was the case for the previous classifier, the range of dot products between $\mathbf{v}_\perp$ and the elements of the training set offers a finite choice of decision thresholds. The best can be chosen simply by computing the weighted training error for each possibility. VBoost runs relatively faster as the number of dimensions rises, because threshold selection happens only once, as opposed to once per dimension for FBoost.

3 Experimental procedure

The experimental procedure described below has three axes of variation: the image representation used, the type of base classfier used for boosting, and the image category used. Of these, the comparisons between image representations and between base classifiers are most interesting. All experiments are performed on the same set, comprising 20,100 images from the Corel photo library. Corel images have been used in many works on image retrieval, and more details on

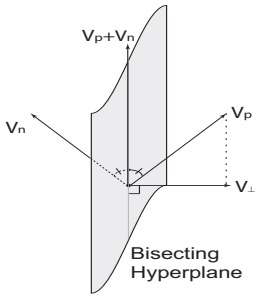

Fig. 1. Schematic illustration of $\mathbf{v}_\perp$

this set of images are available elsewhere [6]. The Corel collection exhibits strong correlations in image composition within many image categories.

All experiments use 5×2-fold cross-validation: For each of five replications, the image set is split in half, with half of the positive instances in each fold. Each fold is used to train a classifier, and its performance is tested on the opposite fold. Comparing results across the five replications provides an estimate of the deviation.

3.1 Image Representation

Three image representations (correlograms, Stairs, T-V) come from relatively recent work on ways of describing images that preserve multiple primitive image cues: color, texture, relative location, etc. Correlograms [8] assemble statistical information about color co-occurrences on the pixel level. Correlogram features are of the form "the probability that a pixel B at distance x from pixel A has the same color as A." Stairs [7] explicitly records the patches of color and texture found in different locations within the image. Thus each feature in Stairs represents the presence or absence of a patch with one discrete combination of color, texture, and location. An unnamed technique introduced by Tieu and Viola [12], henceforth referred to as T-V, registers the output of banks of layered color and texture filters. A feature in this representation corresponds to the output of a set of three successively applied filters summed over the entire image. Finally, color histograms [11] are a well-established representation, included here as a control. Each feature in a color histogram represents the percentage of the image that is of a particular discrete color.

The T-V representation is altered somewhat here in order to achieve a practical algorithm. The original representation stores about 50,000 numbers per image. With a collection of 20,100 images, therefore, the entire data set occupies roughly 8 GB of memory. While this may easily fit on a disk, it will not fit into memory for efficient processing. An approximation yields the necessary reduction in required memory: the values are normalized by subtracting the mean for each feature and dividing by the standard deviation, after which values differ-

ing from the mean by fewer than 2.5 standard deviations are set to zero. This allows the data to be stored as a sparse matrix with about 10^7 elements. Tieu and Viola hypothesize that the success of their method stems from extreme feature values in the range of 4 or more standard deviations, so we posit that this approximation should not unduly affect performance. On the other hand, it is problematic for the images that show no features at all with values more than 2.5 standard deviations from the mean (roughly one third of the test collection). Tieu and Viola do not address the memory issue since they run tests with only 3000 images.

3.2 Boosting Application

Strictly speaking, only one boosting algorithm, AdaBoost [4], is used in all experiments. Rather, the variations presented come from changing the base classifier used, as described in the previous section. Although the base classifier used for boosting makes no difference in theory, common wisdom holds that some bases make for better boosting than others. Because the two methods used here work quite differently, any disparity in their performance should be instructive. As a control, the experiment includes a third, unboosted classifier. In order to give a more meaningful comparison to the boosted algorithms, this is not simply a single application of the base classifier (which does quite poorly). Rather, the control is a nearest-neighbor classifier using the best exemplars of the class as selected by a greedy additive approach [6], a method which outperforms retrieval using any single exemplar image.

3.3 Image Categories

Five hand-defined categories reflect a moderate range of difficulty and subject category: *Sun* (226 images of sunrises and sunsets), *Wolf* (110 images of wolves), *Church* (101 images of churches), *Tiger* (100 images of tigers), and *Car* (100 images of Formula 1 race cars). Although the number of categories is small, the choices include both natural and manufactured objects, full scenes and specific entities within scenes, and wide variation in ease of retrieval.

4 Results

Table 1 shows the mean area under a recall-precision curve computed for each of the 60 ($4 \times 3 \times 5$) experimental conditions, as a percent of total possible area. The data are grouped by image category, since the most interesting variations in performance show up when the category is held constant. The deviations shown come from the variation observed between the five folds of the experiment.

The results display some interesting trends. First of all, boosting improves recall and precision over the corresponding control in nearly all cases (36 out of 40 comparisons). This result is not automatic, since the control case is not a

Table 1. Percent area under the recall-precision curve for two boosting methods and an unboosted control. The best method in each group of three is boldface, as are methods with overlapping standard deviations. The best method in each group of twelve is underlined, as are methods with overlapping standard deviations.

Method	Hist	Corr	Stairs	T-V	
Control	5.0 ± 0.7	19.3 ± 1.7	17.8 ± 1.9	0.7 ± 0.1	Sun
VBoost	27.2 ± 4.0	**57.2 ± 4.1**	26.4 ± 1.8	**19.8 ± 2.3**	Sun
FBoost	**39.3 ± 4.8**	41.4 ± 8.9	**31.4 ± 2.2**	**21.6 ± 3.2**	Sun
Control	1.7 ± 0.3	2.0 ± 0.6	1.4 ± 0.1	0.5 ± 0.0	Church
VBoost	**2.4 ± 0.5**	**5.9 ± 3.0**	**3.7 ± 1.4**	**1.8 ± 0.5**	Church
FBoost	**2.5 ± 0.4**	**4.7 ± 1.3**	**6.0 ± 1.3**	**1.8 ± 0.5**	Church
Control	37.2 ± 3.0	12.9 ± 3.9	2.7 ± 0.2	0.4 ± 0.0	Car
VBoost	14.6 ± 1.5	50.0 ± 3.8	**52.9 ± 4.4**	**2.3 ± 0.4**	Car
FBoost	**56.3 ± 5.4**	**61.1 ± 5.0**	**49.8 ± 3.6**	**2.0 ± 0.6**	Car
Control	43.2 ± 3.8	30.1 ± 3.3	2.5 ± 0.2	0.4 ± 0.0	Tiger
VBoost	11.8 ± 0.7	35.9 ± 3.0	13.7 ± 2.2	**0.8 ± 0.2**	Tiger
FBoost	**49.6 ± 5.8**	**46.7 ± 4.8**	**22.9 ± 3.6**	**0.6 ± 0.1**	Tiger
Control	**9.2 ± 1.5**	12.3 ± 3.1	4.1 ± 0.6	0.5 ± 0.0	Wolf
VBoost	6.6 ± 0.9	**13.5 ± 1.9**	**14.4 ± 3.9**	**1.6 ± 0.3**	Wolf
FBoost	**9.8 ± 1.6**	10.5 ± 1.1	**16.8 ± 4.8**	**2.2 ± 0.7**	Wolf

(straw-man) single application of the boosted classifier, but an effective nearest-neighbor classifier using the best exemplars of the class. A single application of the unboosted base classifier in fact does little better than chance (Table 2).

4.1 Comparing boosting types

Comparisons between the two types of boosting reveal trends according to the underlying image representation. For the histograms, feature-based boosting consistently performs better, achieving significantly higher performance on four of the five image classes. (Bold face type indicates the best boosting method in each category, along with any other method whose range of standard deviation overlaps with that of the best.) Correlograms give mixed results, with each boosting method doing better on two categories, and the fifth a statistical toss-up. For the Stairs and T-V, most of the results are statistically close, although FBoost does distinguishably better on two of the categories with Stairs.

Interestingly, the two methods appear most similar when the number of dimensions in the feature space grows large. The representations are listed in the table from left to right in order of increasing vector length: histograms (128 dimensions), correlograms (512), Stairs (19,200) and T-V (46,875). The last two columns show the largest number of statistical ties, and use the largest feature spaces. This result is interesting in that VBoost is much faster than FBoost for large dimensions, suggesting that it may be a better choice in these cases. (An-

Table 2. Percent area under the recall-precision curve for a single iteration of the base classifier. Without boosting, results are scarcely better than chance.

Method	Hist	Corr	Stairs	T-V	Chance	
base VBoost	1.1 ± 0.1	19.7 ± 3.7	3.8 ± 1.1	1.7 ± 0.7	1.1	Sun
base FBoost	2.0 ± 0.9	5.9 ± 3.5	4.5 ± 1.3	7.3 ± 1.7		
base VBoost	0.5 ± 0.1	0.5 ± 0.1	1.4 ± 1.0	0.5 ± 0.0	0.5	Chu.
base FBoost	0.5 ± 0.1	0.5 ± 0.1	0.6 ± 0.1	0.5 ± 0.1		
base VBoost	0.6 ± 0.1	7.8 ± 2.1	0.5 ± 0.0	0.5 ± 0.0	0.5	Car
base FBoost	0.7 ± 0.3	0.5 ± 0.0	0.8 ± 0.4	0.5 ± 0.1		
base VBoost	0.5 ± 0.1	1.2 ± 0.5	0.5 ± 0.0	0.5 ± 0.1	0.5	Tig.
base FBoost	0.5 ± 0.1	0.5 ± 0.1	0.8 ± 0.5	0.5 ± 0.0		
base VBoost	0.6 ± 0.1	1.1 ± 0.5	2.2 ± 0.6	0.6 ± 0.1	0.5	Wolf
base FBoost	0.6 ± 0.0	0.6 ± 0.1	2.6 ± 1.4	0.6 ± 0.1		

other possibility is that some other factor may unite the two right-hand columns, such as a higher intercorrelation of the individual dimensions.)

4.2 Comparing image representations

Several other trends reveal themselves in Table 1. First, although the T-V representation improves considerably under boosting, the ultimate performance using it does not match that of the other three representations. This may be in part because of the representational restrictions imposed, as described in Section 3.1. However, without some such approximation, it is difficult to see how to apply the technique to larger data sets.

Of the other three representations, histograms display the best overall score on one category (*Tiger*), correlograms on two (*Sun* and *Car*), and Stairs on two (*Church* and *Wolf*). The latter two categories proved hardest overall, implying that boosted Stairs may do well on more difficult visual concepts. However, at least one of the boosted correlogram results was statistically close to the best on every image category, so this may be the best method of those surveyed to choose when little is known about the target concept.

5 Conclusion

Boosting becomes relevant for image retrieval in two contexts: in systems where the user can provide a set of positive and negative exemplars (perhaps hand-labeled from the results of an initial retrieval attempt), and in systems designed to automatically annotate large collections with class tags according to previously learned concepts, for subsequent keyword retrieval. The results presented here may be viewed as indications of the potential of such systems, as well as (from a machine-learning perspective) raw classification ability. As expected,

boosting improves the retrieval of image classes virtually across the board. The two different methods described herein for constructing boostable classifiers from a base image representation both yield better results than a competitive unboosted control. Of these, the method based upon individual features shows better results when the number of dimensions in the image representation is small, but runs slowly when the number of dimensions becomes large. For large feature spaces, the method based upon the entire vector runs more quickly while showing comparable performance. The former represents traditional applications of boosting, while the latter uses a novel type of decision boundary for the base classifier. This work shows that choices must be made about how to combine advanced image representations with boosting, and that the best approach may vary depending upon the image representation chosen and the classes to be learned.

References

1. C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
2. O. Chapelle, P. Haffner, and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Neural Networks*, 10(5):1055–1064, 1999.
3. J. S. De Bonet and P. Viola. Structure driven image database retrieval. *Advances in Neural Information Processing*, 10, 1997.
4. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.
5. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998.
6. N. R. Howe. *Analysis and Representations for Automatic Comparison, Classification and Retrieval of Digital Images.* PhD thesis, Cornell University, May 2001.
7. N. R. Howe and D. P. Huttenlocher. Integrating color, texture, and geometry for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages "II:239–246", Los Alamitos, CA, 2000. IEEE Computer Society.
8. J. Huang, S. Ravi Kumar, M. Mitra, W. J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 762–768, Los Alamitos, CA, 1997. IEEE Computer Society.
9. J. R. Quinlan. *Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.
10. R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
11. M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
12. K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume I, pages 228–235, 2000.