Learning Non-Redundant Codebooks for Classifying Complex Objects

Wei Zhang* WEI.ZHANG22@HP.COM

Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, California 94304, United States

Akshat Surve SURVEA@EECS.OREGONSTATE.EDU

Xiaoli Fern XFERN@EECS.OREGONSTATE.EDU

Thomas Dietterich TGD@EECS.OREGONSTATE.EDU

Oregon State University, 1148 Kelley Engineering Center, Corvallis, Oregon 97331, United States

Abstract

Codebook-based representations are widely employed in the classification of complex objects such as images and documents. Most previous codebook-based methods construct a single codebook via clustering that maps a bag of low-level features into a fixed-length histogram that describes the distribution of these features. This paper describes a simple yet effective framework for learning multiple non-redundant codebooks that produces surprisingly good results. In this framework, each codebook is learned in sequence to extract discriminative information that was not captured by preceding codebooks and their corresponding classifiers. We apply this framework to two application domains: visual object categorization and document classification. Experiments on large classification tasks show substantial improvements in performance compared to a single codebook or codebooks learned in a bagging style.

1. Introduction

Codebook based methods have been widely applied in many application domains for representing complex objects. In visual object categorization, local interest region detectors (Mikolajczyk et al., 2005) and descriptors (Lowe, 2004) are computed to robustly represent images that are subject to noise and deformations. Each image is thus represented as a bag of interest region descriptors. In such tasks, the visual object categorization problem reduces to the problem of classifying a bag of low-level features (interest region descriptors) into one or a subset of the possible object classes. This is an example of the Multiple-Instance problem first proposed by Dietterich, Lathrop & Lozano-Perez (1997).

Codebooks provide a way of mapping a bag of low-level features (descriptors) to a fixed-length attribute vector, to which standard classifiers can be directly applied. Many previous methods construct a codebook by pooling all low-level features from the training images and applying an unsupervised clustering algorithm such as *K*-means in Csurka et al. (2004), Gaussian mixture modeling in Dorko & Schmid (2005) or on-line clustering with mean-shift in Jurie & Triggs (2005). Each cluster is treated as a codeword in the codebook. Once a codebook is constructed, the bag of low-level features is mapped into a fixed-length attribute vector according to the codebook using strategies such as the histogram of occurrences method (e.g., Csurka et al., 2004; Jurie & Triggs, 2005). Standard learning algorithms, including Naïve Bayes (Csurka et al., 2004) and SVMs (Csurka et al., 2004; Jurie & Triggs, 2005; Perronnin et al., 2006), have been applied to these attribute vectors to train the final classifier. Such codebook based methods have shown good performance on various visual object categorization tasks.

Similarly, document classification has also seen successful applications of codebook-based methods. In document classification, each document is represented as a bag of words (BOW). The BOW representation typically involves a very large dictionary, leading to computational and over-fitting issues in the classification stage. Codebook-based methods are applied to address this issue by clustering the words into word clusters, where each word cluster can be viewed as a composite *codeword* (Baker & McCallum, 1998; Slonim & Tishby, 2001; Bekkerman et al., 2003; Dhillon et al., 2003). The representation of a given document using a codebook (i.e., a set of word clusters) is created by introducing one attribute for each word cluster as the number of times the words of that cluster appear in the document. This produces a more compact representation of the documents, which has been shown to achieve better classification accuracy with various stan-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

* The work was performed when the first author was graduate student at Oregon State University.

dard classifiers such as Naïve Bayes and SVMs (Baker & McCallum, 1998; Slonim & Tishby, 2001; Dhillon et al., 2003), especially when the classification task has high complexity (Bekkerman et al., 2003).

All the codebook learning methods introduced above build one codebook by performing a single clustering on the low-level features. But in practice, data can be represented in many different ways; and oftentimes a single codebook is not enough to fully describe the different structures of the data. In this paper, we propose a simple framework for learning multiple non-redundant codebooks that produces surprisingly good results. The basic idea is to wrap the codebook construction process *inside* a boosting procedure. Each iteration of boosting begins by learning a codebook according to the weights assigned by the previous boosting iteration. The resulting codebook is then applied to encode the training examples; a new classifier is learned; and new weights are computed. We apply the proposed non-redundant codebook learning framework on both visual object categorization and document classification tasks. The resulting methods give very substantial performance gains over the baseline of learning a single codebook of equivalent size. For image classification, we obtain a 77% reduction in error on a challenging 9-class object recognition task. For document classification, we obtain error reductions of 18% and 38% on two difficult document classification tasks.

2. Related Work

Codebook learning has been a very active research area in the vision community. Recently, some researchers have begun to introduce discriminative mechanisms into the codebook learning process. Winn et al. (2005) first apply unsupervised learning to construct a very large codebook and then apply the information bottleneck principle to merge the codebook entries. Similarly, Perronnin et al. (2006) first learn a universal codebook and then employ MAP estimation to adapt the universal codebook to class-specific data. The framework proposed in this paper seeks to advance the state-of-the-art of codebook learning in an orthogonal direction in the sense that any new advanced codebook learning algorithm can be employed in this framework to further improve its performance.

Our framework is inspired by the successful application of multi-view clustering algorithms (Cui et al, 2007; Jain et al., 2008) for exploratory data analysis. Non-redundant clustering is motivated by the fact that real-world data are complex and may contain more than one interesting form of structure. The goal of non-redundant clustering is to extract multiple structures from data that are different from one another; each is then presented to the user as a possible view of the underlying structure of the data. In principle, the non-redundant clustering techniques developed by Cui et al. (2007) and Jain et al. (2008) can be directly applied to learn multiple clusterings of the low-level features and create codebooks that are non-redundant to each other. However, such an approach is fundamentally unsupervised and will not necessarily lead to more accurate classifiers. In contrast, our framework learns codebooks that are non-redundant in the sense that they complement each other in their discriminative power.

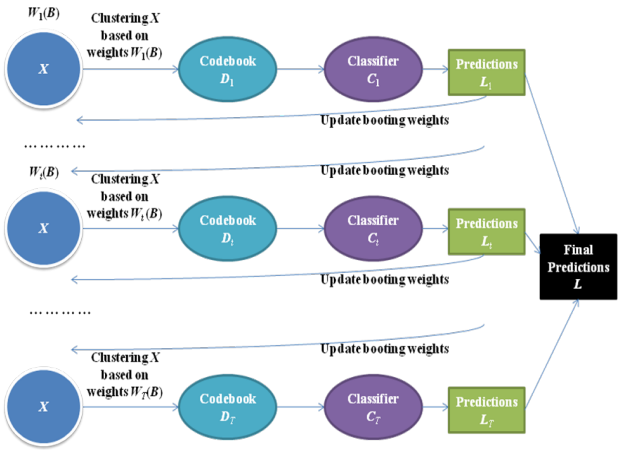

Figure 1. Illustration of iterative non-redundant codebook and classifier learning framework.

There are a few recent developments in the vision community that are related to our framework. Moosmann et al. (2007) learn multiple, independent randomized decision trees to partition the low-level feature space. The leaves of the decision trees define the codewords. Note that this work can also be used to generate multiple codebooks, each from a single decision tree (in their work the multiple codebooks are concatenated to form a single large one). However, they do not produce codebooks of complementary discriminative power. More recently, Yang et al. (2008) proposed a codebook learning framework that is integrated with classifier learning. In their work, the visual codebooks have a restricted form as a sequence of visual bits. Each visual bit is a linear classifier that maps the low-level features to a binary bit for classification. The learning is performed in a sequential manner, where the performance of the classifier using previous visual bits is used to extract the next set of visual bits. In comparison, the method proposed in this paper shares the same basic boosting principle of sequential learning of codebooks; yet our framework is much more general and can be applied with any form of visual codebooks and classifiers.

3. Methodology

The overall framework of our non-redundant codebook learning method is illustrated in Figure 1.

Given a base codebook learner and a classifier, we iteratively learn one codebook at a time and stop when reach-

ing the pre-defined T iterations. Each iteration consists of following steps:

1. Codebook learning: The inputs to the base codebook learner at iteration t are the training examples $\{B_i\}_{i=1}^N$ (where $B_i = \{x_1^i, \ldots, x_{M_i}^i\}$ is the bag of features for training example i) and a set of weights $W^t = \{w_i^t\}_{i=1}^N$ specifying the importance of each example. The output is a codebook $D_t = (d_1^t, \ldots, d_K^t)$ where d_k^t is the k-th code word (cluster). In the first iteration, the weights are initialized to be uniform over the training examples.

2. Classifier learning: the training examples are mapped to fixed-length attribute vectors $F_t = \{f_i^t\}_{i=1}^N$ based on the codebook D_t such that the j-th attribute, $f_i^t(j)$, is related to the number of features in B_i mapped to code word d_j^t: $|\{x_m \in B_i : x_m \mapsto d_j^t\}|$. A classifier C_t is then learned from the attribute vectors F_t. The output of C_t is the class label predictions L_t.

3. Weight updating: the predictions L_t are used to update the training example weights as in AdaBoost (Freund & Schapire, 1996) − the weights of the incorrectly-classified examples are increased and the weights of the correctly-classified examples are decreased. The updated weights, W^{t+1}, are provided to the base codebook learner for the next iteration.

To classify a new example B, it will be mapped into T fixed-length attribute vectors $\{f^t\}_{t=1}^T$, and each f^t will then be classified by C_t. The outputs of the T classifiers are combined to give the final class label prediction L via the weighted voting scheme of AdaBoost.

We want to emphasize that the above framework can be applied with any base codebook learner and any classifier. Some codebook learners can be easily modified to take the weights of the examples into consideration (as exemplified in Section 5). Incorporating the weights into others may not be straightforward. For the latter case, as we will show in Section 4, sampling can be employed to learn codebooks from weighted examples effectively. In the following sections, we will present two instantiations of our framework for different application domains to demonstrate the general applicability of our framework using different codebook learning algorithms and classifiers.

4. Non-Redundant Codebook Learning for Continuous Low-Level Features

4.1 The Visual Object Categorization Problem

Recognizing objects in images is a fundamental problem in computer vision. It is challenging for computers because of significant intra-class variations of the objects and the variations in imaging conditions. Recently, significant advances have been achieved through the application of interest region detectors that can reliably find salient regions sparsely distributed in images despite these sources of variation. A comprehensive evaluation of the state-of-the-art detectors is given in Mikolajczyk et al. (2005). Each detected region is represented by a local region descriptor. The most famous descriptor is Lowe's 128-element SIFT descriptor (Lowe, 2004). So the original image is represented as a bag of region descriptors, which are the low-level features for codebook learning.

When learning with large scale visual object categorization data sets, such as the Caltech data set (Dorko & Schmid, 2005; Opelt et al., 2006), PASCAL dataset (Yang et al., 2008) and Stoneflies dataset (Larios et al., 2008), a practical challenge is computational efficiency. In particular, we often need to learn from thousands of images, and each image often contains hundreds or even thousands of low-level interest region descriptors in order to capture sufficient visual information for classification. The huge number of low-level features requires our non-redundant codebook learning algorithm to be highly efficient. Below we present how we apply the general framework described in Section 3 to learn a set of non-redundant visual codebooks efficiently.

4.2 Non-Redundant Codebooks Learned by Boost-Resampling

Below we describe the design of the critical components of our framework.

Base Codebook learner. There are many options for base codebook learners for visual object categorization, including both unsupervised and supervised clustering methods. In order to reduce the risk of overfitting the training data, we choose to use unsupervised K-means clustering as our base codebook learner (Csurka et al., 2004). K-means clustering is also preferable because it is very efficient. The more complex Gaussian Mixture Modeling algorithm (Dorko & Schmid, 2005) was also tested on the Stonefly recognition task and failed to outperform the simpler K-means algorithm in our framework. The size of each codebook, K, is set to 100 empirically. According to our experiments, different values of K had little effect on the performance of the algorithm.

Feature mapping based on codebook. Previous work (Csurka et al., 2004; Jurie & Triggs, 2005; Moosmann et al., 2007; Larios et al., 2008) usually assign each of the low-level features extracted from a new image to one of the codewords (clusters); then the example is represented by a normalized histogram $\boldsymbol{f}$ of occurrence counts for the codewords. But this mapping method treats the codewords as equally important without considering their distribution over different examples. In this paper, we use the *tf–idf* weight (*term frequency–inverse document frequency*) (Salton & Buckley, 1988) developed for information retrieval and text mining. For image B_i, the j-th *tf–idf* attribute $f_i(j)$ is given by:

$$f_i(j) = \left(\frac{|\{x_m \in B_i : x_m \mapsto d_j\}|}{M_i} \right) \times \log\left(\frac{N}{|\{B_l : \exists_{x \in B_l} x \mapsto d_j\}|} \right) \quad (1)$$

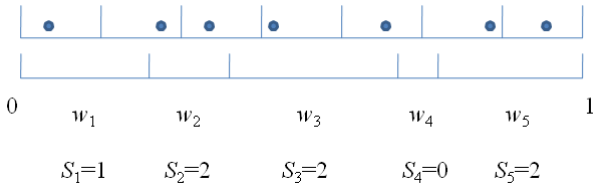

Figure 2. Illustration of Quasi-Random Weighted Sampling (QWS) technique.

As in the document domain, the first term (*tf*) measures the number of occurrences of the j^{th} codeword in the example divided by the total number of low-level features in the example. This term is exactly the histogram of occurrences used in previous work. We will call it the *tf* mapping. The second term, *idf*, measures the (lack of) distinctiveness of the j^{th} codeword over different examples. A codeword that appears in more examples has lower *idf* value; while a codeword that is only found in one example has the highest *idf* value. The *tf–idf* mapping method improves the robustness of learning algorithms, especially when the distribution of codewords is significantly unbalanced over different examples, which is the case in many object categorization problems. In our initial experiments, the *tf–idf* mapping systematically outperforms the *tf* mapping. Hence, we adopt *tf–idf* as the feature mapping method for the visual object categorization tasks.

Classifier learning. We employ an ensemble of 50 unpruned C4.5 decision trees (Quinlan, 1993) in each boosting iteration. The trees are generated via bagging (Breiman, 1996). Learning more than 50 trees did not provide superior performance in our experiments.

Weighted Sampling. Note that instead of directly using the weights with K-means clustering, we adopt the Quasi-Random Weighted Sampling (QWS) (Kalal et al., 2008) approach to achieve improved efficiency. QWS creates a smaller "active set" of the training images based on the weights assigned to the examples such that the examples with larger weights have higher probability of being selected. As a result, the algorithm selects the data pool that has not been well-represented and classified by previous codebooks. Thus, the codebook learned on this pool is encouraged to be different to the previous codebooks.

The principle of QWS is illustrated in Figure 2. The weights are represented as intervals and arranged on the unit line segment. The line segment is split into N equal intervals, where N is the total number of training images. Within each interval, a number (shown as a dot in the figure) is generated uniformly at random, whose position determines the index of the selected sample. S_i represents the number of occurrences of the i^{th} example in the sampled active set. As can be seen from the figure, an example with larger weight (e.g., w_3) is more likely to be sampled; and an example may be selected multiple times (e.g., S_3=2). More details of QWS method is described in Kalal (2008). QWS reduces the variance of weighted random sampling and has worked well in previous work (Moosmann et al., 2007; Kalal, et al., 2008).

Figure 3. Example images from the nine categories of stoneflies (top to bottom, left to right): *Cal*, *Dor*, *Hes*, *Iso*, *Mos*, *Pte*, *Swe*, *Yor* and *Zap*.

The total number of codewords learned is: $T \times K$ (the number of boosting iterations × the size of each codebook). This number scales up to several thousands in our experiments. But at each iteration, we only sample 20% of the training data to form the active set. Therefore the learning of the codebooks is memory and time efficient because each clustering operation is performed on a small subset of data. The *Boost-Resampling* algorithm can be directly applied to problems with continuous high-dimensional low-level features. Below we empirically evaluate it using a large scale real-world object categorization dataset.

4.3 Experiments on Stonefly Recognition

In order to test the performance of our *Boost-Resampling* algorithm on complex object categorization problem, we evaluate it on the stonefly larvae dataset supplied by the Oregon State University insect ID group (Larios et al., 2008). This dataset contains 3826 stonefly images belonging to nine different species. Examples of stonefly images are shown in Figure 3. Due to the biological nature of stoneflies and the imaging process, these images exhibit large intra-class variations and small inter-class differences. This poses a very challenging object categorization problem even for expert biologists. As described by Larios et al. (2008), humans who have been trained to recognize images of two species of stoneflies – *Cal* and *Dor* – achieved only 78.6% classification accuracy. We refer this binary classification problem as STONEFLY2 (*Cal*, *Dor*). A generative codebook learning algorithm (Larios, 2008) has previously been applied to this STONEFLY2 dataset and achieved performance slightly superior to

Table 1. Classification accuracies (%) of *Boost-Resampling* algorithm and previously reported results on STONEFLY2 and STONEFLY4 datasets.

DATA SET	*BOOST*	*LARIOS08*	*OPELT06*
STONEFLY2	**97.85**	79.37	70.10
STONEFLY4	**98.21**	82.42	N/A

humans. A similar 4-class dataset, STONEFLY4 (*Cal*, *Dor*, *Hes*, *Yor*), has also be studied under the multi-way classification setting. In this paper, we evaluate our *Boost-Resampling* algorithm on the STONEFLY2 and STONEFLY4 datasets using the same 3-fold cross validation experimental setting as in Larios et al. (2008). For a fair comparison we also employ the same interest region detectors—the Hessian affine regions (Mikolajczyk & Schmid, 2004), the Kadir's salient regions (Kadir et al., 2004) and the PCBR regions (Deng et al., 2007)—described by SIFT descriptors (Lowe, 2004). Each detector generates approximately 300 SIFT vectors from each image. A separate codebook (of size K = 100) is built for each detector. For a new image, its low-level features from each interest region detector are mapped according to their corresponding codebooks to generate an attribute vector for that particular region detector; and the three attribute vectors are concatenated to form the 300-dimensional attribute vector for classification. The number of boosting iterations T is set to 30 in our experiments. The results are summarized in Table 1.

From Table 1, we can see that our non-redundant codebook learning algorithm performs much better that the previous work (Larios et al., 2008; Opelt et al., 2006). All the improvements are statistically significant at a 95% level using an unpaired test for the difference between two proportions (Dietterich, 1998).

Figure 4 shows the performance of our *Boost-Resampling* algorithm on the stonefly datasets versus the number of boosting iterations. We can see that comparing to the starting points of the curves (using a single codebook of size *K=100*), the addition of non-redundant codebooks significantly improves the discriminative power of the recognition system. For all three tasks, the learning converged within 25 iterations and showed no sign of overfitting.

In order to test the value of building non-redundant codebooks and the value of weighted sampling, we compare our *Boost-Resampling* algorithm with two baseline algorithms. The first algorithm (referred as *Single*) learns only a single codebook at each channel to represent the data. This codebook is built by *K*-means clustering on the pool of all training features from the channel. The size of the codebook is set to $T \times K$ (the number of boosting iterations × the size of each codebook) for fair comparison

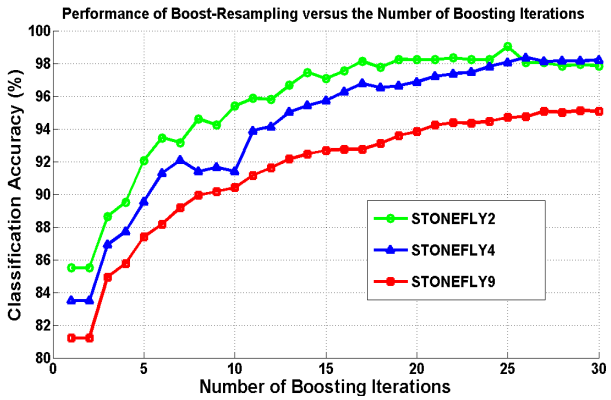

Figure 4. The performance of *Boost-Resampling* algorithm versus the number of boosting iterations.

Table 2. Classification accuracies (%) of *Boost-Resampling* algorithm and two baselines on STONEFLY2, STONEFLY4 and STONEFLY9 datasets.

DATA SET	*BOOST*	*SINGLE*	*RANDOM*
STONEFLY2	**97.85**	85.84	89.16
STONEFLY4	**98.21**	67.20	90.42
STONEFLY9	**95.09**	78.33	89.07

with non-redundant codebooks. The second baseline (called *Random*), replaces QWS sampling with uniform random sampling that neglects the boosting weights. This comparison experiment is performed on STONEFLY2, STONEFLY4 datasets and the complete STONEFLY9 dataset, which contains all the nine categories of stoneflies as shown in Figure 3. The comparison results are summarized in Table 2. *Boost-Resampling* outperforms the two baselines on all the datasets, and the differences are statistically significant at a 95% level. Comparing to a single codebook of equivalent size, the proposed method was able to achieve error reductions of 94.5%, 84.8% and 77.3% respectively.

5. Non-Redundant Codebook Learning for Discrete Low-Level Features

5.1 Document Classification Problem

In document classification, each document is represented as a bag of words (BOW). In many document classification tasks, the number of words in the dictionary can be very large (tens of thousands or more), resulting in a very high-dimensional representation of the documents. This leads to both computational and over-fitting issues in the classification stage. One way to address this issue is to cluster the words into word clusters, where each word cluster can be viewed as a composite *codeword*. To

represent a given document using a set of word clusters, we produce one attribute for each word cluster by counting the number of times that the words of that cluster appeared in the document. This produces a more compact representation of the documents and has been shown to achieve better classification accuracy (Baker and McCallum, 1998; Slonim and Tishby, 2001; Dhillon et al., 2003; Bekkerman et al., 2003). By applying non-redundant codebook learning to the document classification task, we seek to generate multiple non-redundant clusterings of the words to further improve document classification accuracy.

5.2 Non-Redundant Codebook Learned by Boosting-Reweighing

Base codebook learner. In document classification, the most successful word clustering algorithms to date are discriminative in nature. They aim to produce word-clusters that preserve information about the class labels. Representative examples include the *Information Bottleneck* (IB) approach (Bekkerman et al., 2003) and the *Information-theoretic Divisive Clustering (IDC)* approach (Dhillon et al., 2003). In our work, we choose to use the *Information Bottleneck* approach as our base learner for generating the base codewords (i.e., word clusters). Note that the *IDC* approach achieves very similar results. Below we briefly highlight the crux of the IB method and how it is applied within our framework.

Consider two categorical random variables X and Y, whose co-occurrence distribution is made known through a set of paired i.i.d observations (X,Y). IB seeks to extract X', a compact representation of X, with minimal loss of information regarding the relevant variable Y. This goal is formalized as minimizing the following objective function:

$$L = I(X;X') - \beta I(X';Y) \quad (2)$$

where β determines the tradeoff between compressing X and extracting information regarding Y and $I(\cdot\,;\,\cdot)$ denotes the mutual information.

In the document classification task, X represents the words and Y represents the class labels. In iteration t, the relation between X and Y is summarized by the empirical joint distribution table $P_t(X, Y)$. Given K, the desired number of clusters, IB learns $X_t'=\{wc^t_1, wc^t_2, \ldots, wc^t_K\}$, a set of K word clusters, that compactly represent X and preserve as much information as possible about the class label Y. In our implementation, we used the sequential Information Bottleneck (sIB) algorithm as described by Slonim et al. (2002).

As in the visual object categorization task, we set K, the size of each codebook to be 100. The parameter β is empirically set to 100. (Note that a large β is essential in order to balance the tradeoff between the two terms in the objective function. This is because these two terms are upper-bounded by the entropy of X and Y respectively; and the entropy of X is significantly larger than the entropy of Y).

Table 3. The characteristics of the document data sets

DATA SET	*#classes*	*#documents*	*#words*
NG10	10	5000	24246
ENRON10	10	6188	24812

Feature mapping based on the codebook. Once the K word clusters are extracted, we follow the basic approach of Dhillon et al. (2003) and Bekkerman et al. (2003) and create one attribute $f^t(j)$ for each word cluster j by counting the number of times the words of cluster j appear in each document. This produces a fixed-length attribute vector representation for each document.

Classifier learning. In each iteration, we build a Naïve Bayes classifier C_t using the attribute vector induced with the word clusters learned by IB. We choose Naïve Bayes for its computational efficiency. SVMs and bagged decision trees have also been tested and yielded similar results in terms of the effect of the proposed method.

Reweighing. The learned Naïve Bayes classifier C_t is then applied to predict the class labels of the training documents. We then follow the scheme of AdaBoost and update the training example weights according to their prediction success: the weights of correct examples are decreased and the weights of incorrect examples are increased. The weights are then used to update the empirical joint distribution table $P_{t+1}(X, Y)$, which becomes the input to the IB method for learning the next codebook.

Given a new document, it will be mapped to T different attribute vectors $f^t, t = 1,\ldots,T.$, each is assigned a prediction by its corresponding classifier C_t. The final prediction is obtained by a weighted vote of all T classifiers.

5.3 Experimental Results

In this section, we evaluate the boosting-reweighting approach on two benchmark document classification data sets: the Newsgroup data set and the Enron email classification data set. For the Newsgroup dataset, we reduced the dataset size by randomly selecting 10 of the classes and for each class 500 documents are randomly selected. We will refer to this as NG10. For the Enron data set, we choose the ten largest folders among all users to be our ten classes (containing 6188 emails). Below we refer to this data set as Enron10.

We employ the Mallet package (McCallum, 2002) to preprocess the documents. Stop words and the words appearing in no more than three documents were removed. The characteristics of the resulting data sets are summarized in Table 3.

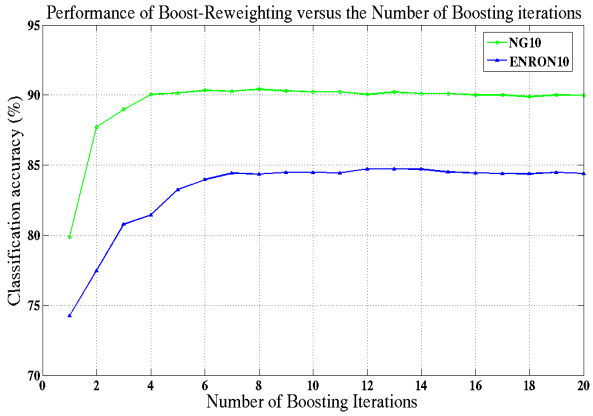

Figure 5. The performance of *Boost-Reweighting* algorithm versus the number of boosting iterations.

Interestingly, we observed that the performance of our approach converges very quickly as we increase the number of iterations, within the first five or six iterations. This can be seen from Figure 5, which plots the accuracy of our method on both data sets as we increase T from 1 to 20. This suggests that the complexity of the low-level features may be significantly lower for the document classification task than for the visual object categorization task. Hence, fewer non-redundant codebooks are needed to capture all of those features' discriminative power.

Based on this convergence behavior, in our following comparisons with the baseline methods, we fix T to be ten. We compare our approach to the same set of baseline methods as in the previous section. The first baseline is *Single*, in which only a single codebook is learned to represent the data. This codebook is built by applying IB to the original (un-weighted) empirical joint distribution $P_1(X, Y)$. We examine two different codebook sizes: the first is 100 (which is the same as the size of our individual codebooks, referred to as *S100*), and the second is 1000 (which is the number of total codewords used by our non-redundant codebooks, referred to as *S1000*). The second baseline is called *Random*, in which ten bagged samples of the original training corpus are created and each is used to estimate an empirical joint distribution $P_t(X, Y)$ and learn a codebook using IB. Note that all training examples are used to learn the classifiers and the bagged samples are only used to produce the codebooks.

Table 4 reports the accuracy of our method and the baselines on NG10 and Enron10. For all experiments, we randomly sample two thirds of the data for training and test on the remaining examples. The testing examples are not used in any part of the training process. Each of the reported numbers is averaged across five random runs. The standard deviations are also shown in the parentheses.

As shown by Table 4, the proposed boost-reweighing method was able to significantly improve over both *Single* and *Random*. The differences are all statistically significant at a 95% level. This supports the conclusion that by forcing the codebook learner to focus on the input space that has not been well classified, our framework was able to effectively capture additional discriminative information. Note that we also applied boosted Naïve Bayes to a single codebook (i.e., boosted *Single*). The results were comparable to *Random* and inferior to the proposed method.

Table 4. Classification accuracies (%) of the *Boost-Reweighing* algorithm and three baselines on document datasets.

DATASET	*BOOST*	*RANDOM*	*S1000*	*S100*
NG10	**90.24** (.80)	85.43 (.53)	84.31 (.97)	79.88 (1.1)
ENRON10	**84.44** (.22)	81.09 (.52)	80.90 (.61)	74.23 (1.1)

6. Conclusions and Future Work

This paper proposes a framework for learning non-redundant codebooks for the categorization of complex objects based on the bag-of-features representation. The proposed framework is simplistic and highly general: it can be easily applied with any codebook learner and classifier of choice. It is surprisingly effective: we evaluated the proposed framework on both visual object categorization and document classification domains, and obtained performance substantially superior to existing work and to the baseline methods

For future work, we will explore the application of stronger non-redundancy constraints such as side information (Chechik & Tishby, 2002). We will also work to further improve the efficiency of our learning algorithm for large-scale datasets.

Acknowledgments

We thank the colleagues in the Oregon State University insect ID group for help on the stoneflies experiments. We thank anonymous reviewers for useful comments on this paper. The authors gratefully acknowledge the support of the NSF under grant number IIS-0705765.

References

Baker, L. D. & McCallum, A. K. (1998). Distributional clustering of words for text classification. In *Proc. SIGIR conf. Resear. and develo. infor. retriev.*, pp 96-103.

Bekkerman, R., El-yaniv, R., Tishby, N., Winter, Y., Guyon, I., & Elisseeff, A. (2003). Distributional word clusters vs. words for text categorization, *J. of Machine Learning Research*, Vol 3, pp 1183-1208.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24 (2), pp 123-140.

Chechik, G. & Tishby, N. (2002). Extracting relevant structures with side information. *Proc. Advances in Neural Information Processing Systems*, pp 857-864.

Csurka, G., Dance, C. R., Fan L., Willamowski, J., & Bray, C. (2004). Visual categorization with bags of keypoints. *Euro. Conf. Comput. Vision Workshop*, pp 59-74.

Cui, Y., Fern, X. Z., & Dy, J. G. (2007). Non-redundant Multi-view Clustering via Orthogonalization. *IEEE Int'l Conf. on Data Mining*, pp 133-142.

Deng, H., Zhang W., Mortensen, E., Dietterich, T. & Shapiro, L. (2007). Principal curvature-based region detector for object recognition. *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp 1-8.

Dhillon, I., Mallela, S. & Kumar, R. (2003). A divisive information-theoretic feature clustering algorithm for text classification. *J. of Machine Learning Research*, Vol 3, pp 1265-1287.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, Vol 89, pp 31-71.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, pp 1895-1923.

Dorko, G. & Schmid, C. (2005). Object class recognition using discriminative local features. *Technical Report RR-5497, INRIA-Rhone-Alpes.*

Freund, Y. & Schapire, R. (1996). Experiments with a new boosting algorithm. *Proc. Int'l Conf. Machine Learning*, pp 148-156.

Jain, P., Meka R., & Dhillon I. S. (2008). Simultaneous Unsupervised Learning of Disparate Clusterings. *Statistical Analysis and Data Mining***.** Vol 1**,** pp 195-210.

Jurie, F. & Triggs, B. (2005). Creating efficient codebooks for visual recognition. *Proc. IEEE Int'l Conf. Comput. Vision*, Vol 1, pp 604-610.

Kadir, T., Zisserman A., & Brady, M. (2004). An affine invariant salient region detector. *Proc. Euro. Conf. Comput. Vision*, pp 228-241.

Kalal, Z., Matas, J., & Mikolajczyk K. (2008). Weighted sampling for large-scale boosting. *Proc. Brit. Machine Vision Conf.*

Larios, N. et al. (2008). Automated insect identification through concatenated histograms of local appearance features. *Machine Vis. and App.*, 19(2), pp 105-123.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*., 2(60), pp 91-110.

McCallum, A. K. (2002). MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Mikolajczyk, K., & Schmid, C. (2004). Scale and affine invariant interest point detectors. *Int. J. Comput. Vision*., Vol 60, pp 63-86.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & Van Gool, L. (2005). A comparison of affine region detectors. *Int. J. Comput. Vision.*, Vol 65, pp 43-72.

Moosmann, F., Triggs, B. & Jurie, F. (2007). Fast discriminative visual codebooks using randomized clustering forests. *Proc. Advances in Neural Information Processing Systems*, pp 985-992.

Opelt, A, Pinz A, Fussenegger. M, & Auer P. (2006). Generic Object Recognition with Boosting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(3), pp 416-431.

Perronnin, F., Dance, C., Csurka, G. & Bressan, M. (2006). Adapted vocabularies for generic visual categorization. *Proc. Euro. Conf. Comput. Vision*, pp 464-475.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), pp 513–523.

Slonim, N. & Tishby, N. (2001). The power of word clusters for text classification. In *Proc. Euro. Colloq. Information Retrieval Research*.

Slonim, N., Friedman, N., & Tishby, N. (2002). Unsupervised document classification using sequential information maximization. *Proc. SIGIR conf. Resear. and develo. infor. retriev.*, pp 129-136.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp 511-518.

Winn, J., Criminisi, A. & Minka, T. (2005). Object categorization by learned universal visual dictionary. *Proc. IEEE Int'l Conf. Comput. Vision*, Vol 2, pp 1800-1807.

Yang, L., Jin, R., Sukthankar R., & Jurie, F. (2008). Unifying discriminative visual codebook generation with classifier training for object category recognition. *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp 1-8.