Learning Weights for Codebook in Image Classification and Retrieval

Hongping Cai[1,2] Fei Yan[2] Krystian Mikolajczyk[2]
[1]National University of Defense Technology, China [2]University of Surrey, UK
{hongping.cai,f.yan,k.mikolajczyk}@surrey.ac.uk

Abstract

This paper presents a codebook learning approach for image classification and retrieval. It corresponds to learning a weighted similarity metric to satisfy that the weighted similarity between the same labeled images is larger than that between the differently labeled images with largest margin. We formulate the learning problem as a convex quadratic programming and adopt alternating optimization to solve it efficiently. Experiments on both synthetic and real datasets validate the approach. The codebook learning improves the performance, in particular in the case where the number of training examples is not sufficient for large size codebook.

1. Introduction

Bag-of-words model [3] has been widely and successfully used in visual recognition and retrieval, although it was first proposed for text retrieval. In the bag-of-words model, local features extracted from an image are mapped to a codebook, which is typically produced with a clustering method. An image is then represented as a histogram of codeword occurrences. Among all attempts to improve the bag-of-words model, much attention has been focused on generating a discriminative codebook [6, 9, 13, 14]. In this paper our aim is to learn weights for an existing codebook. Some codewords can be more informative than others thus more discriminative for classification. This assumption holds in particular for dense sampling of features, which tends to produce better performance than interest points in many classification tasks [7, 9, 13]. Such densely sampled descriptors result in codewords that should not be attributed equal importance. Our weight learning is based on the intuition that the weighted similarity between same labeled images should always be larger than that between differently labeled images. Inspired by the recent work on learning local feature weights for image classification [4], in which the problem is modeled as maximization of the margin between within-class distance and between-class distances [12], we learn the codewords weights in a framework similar to distance metric learning. A constrained quadratic programming approach is proposed to learn the weights for codebook, where the sum of the empirical loss and a regularization term is minimized.

In order to capture fine differences between features, large-size codebooks have been widely used in image classification and retrieval [11]. However, large-size codebooks suffer from insufficient training examples. We demonstrate that in such a case our learning approach can improve the performance. Moreover, the large size of codebook also poses challenges for weight learning, since it requires large memory and intensive computation when using standard optimization software. We address this issue at two stages. First, we select only a subset of constraints in the optimization problem and update the subset iteratively in a bootstrapping manner. Secondly, we employ the idea of Alternating Optimization (AO) [1] for solving the optimization problem. In each iteration of the AO, the objective function is optimized with respect only to a subset of variables. This AO has been shown experimentally to converge to the global solution quickly. We combine these two iterations in a nested manner, and show that the approach offers significant efficiency gains in terms of both time and space complexity.

In the following we first review the existing work in both codebook weighting and supervised distance metric learning. In Section 3, we propose our codebook weight learning approach. Section 4 discusses two techniques for solving the optimization problem efficiently. We then evaluate the learning approach on both synthetic dataset in Section 5 and four image benchmarks in Section 6. Finally, conclusions end this article.

2. Related Work

Codebook weighting Codebook plays the key role in the bag-of-words model, and it is a collection of vector quantized features. The most popular way of creating codebook is by using k-means clustering or its variant, i.e., hierarchical k-means [5]. However, it is argued that k-means does not select the most informative descriptors as it tends to concentrate the cluster centers in high density areas of the feature space and starves lower density ones [6, 14]. In [6, 14], radius-based clustering is used for codebook generation. However, radius-based or agglomerative clustering results in an unbalanced codebook. Most features are assigned to very few codewords, while majority of codewords are not statistically significant. Therefore, with overall consideration, the k-means clustering is still most widely used and also adopted in our work. Our weighting strategy for codebook can be used as a method for improving the discriminability of the k-means clustered codebook.

A simple codebook weighting approach, often used in retrieval, is the tf-idf (term frequency-inverse document frequency) weighting scheme, where commonly occurring codewords are down-weighted. Instead of assigning each feature to only one codeword, soft assignment for a few closest codewords is investigated in [14], which can also be considered as a codebook weighting. The reported improvements with tf-idf and soft assignment suggest that the discriminability of codebooks can be improved.

Codewords selection can also be viewed as assigning a binary weight. In [16], a large k-means clustered codebook was reduced by merging codewords to achieve a trade-off between within-class compactness and between-class discriminability. In [6], the codewords were selected according to mutual information between optimally thresholded codeword frequency and category, while [13] performed codewords selection according to the relative frequency.

In all these codeword selection and weighting approaches, the weights are assigned according to some criteria. However, little effort has been made to accurately measure the contribution of each codeword and to incorporate this information into a machine learning based approach. In this context, our contribution is to learn the weights for the codebook in a similarity learning framework with the goal of distinguishing same labeled images from differently labeled images.

Supervised distance metric learning Although our optimization problem is learning codebook weights, it is closely related to learning supervised distance metric [2]. This has been successfully used in many machine learning problems (see [17] for a comprehensive discussion). The motivation of supervised distance metric learning is to induce a distance metric with labeled data so as to preserve the class discriminatory information from labeled points.

Among supervised distance metric learning techniques, large margin methods have been a particularly promising direction. A weighted Euclidean distance was learnt from relative constraints in [12]. The solution was to solve a convex quadratic problem similar to SVM by minimizing ℓ_2 regularization and the hinge loss. This idea was successfully used in local feature weight learning in object categorization [4]. Our work is inspired by these two approaches, but instead of learning millions of weights for all local features, we learn the weights for codewords. Thus, we avoid the problem of overfitting that occurs in large dimensional models from limited training samples. Another successful work in this area is Large Margin Nearest Neighbor (LMNN) [15], which formulates distance metric learning as a constrained semi-definite programming problem and maximizes the margin between within-class pairs and between-class pairs. However, optimization is inefficient for a full $M \times M$ matrix from an M-dimensional codebook and it is easy to overfit. Furthermore, experiments in [12] have shown that the Mahalanobis matrix solution does not improve the results over the weight vector. Therefore, a weight vector, instead of a matrix, is learnt for similarity measure in our approach.

3. Learning Weighted Codebook

We first briefly introduce the notations used in this section. Let $\mathcal{T}$ be a triplet index set of training images: $\mathcal{T} = \{(i,j,k)|y_i = y_j, y_i \neq y_k\}$, where y_i denotes the class label for image I_i. For image I_i, the local features are assigned to the predefined codebook. Suppose $h_i(m)$ ($m = 1, ..., M$) is the frequency of the m-th codeword in image I_i . The intersection of the m-th codeword occurrence frequency between image I_i and image I_j is denoted by $s_{ij}(m) = \min(h_i(m), h_j(m))$ in the bag-of-words model (or level weighted intersection in SPMK, refer to [7] for more details). Accordingly, $\mathbf{s}_{ij}$ represents the intersection vector between image I_i and image I_j: $\mathbf{s}_{ij} = (s_{ij}(1), s_{ij}(2), ..., s_{ij}(M))^T$.

In the typical bag-of-words model or SPMK [7], the similarity between two images is the sum of the equally weighted intersections: $s(I_i, I_j) = \sum_m s_{ij}(m)$. In contrast, we assign different weights for codewords, resulting in a weighted similarity defined as $s_w(I_i, I_j) = \mathbf{w}^T\mathbf{s}_{ij} = \sum_m w(m)s_{ij}(m)$, which is a similarity metric according to the definition in [2].

We aim to make the weighted similarity between same labeled images larger than that between differently labeled images. Ideally, the learnt weight vector $\mathbf{w} \in \mathbb{R}^M$ satisfies the constraint

$$\mathbf{w}^T\mathbf{s}_{ij} > \mathbf{w}^T\mathbf{s}_{ik}, \forall(i,j,k) \in \mathcal{T} \tag{1}$$

It is impossible to fulfil these constraints for all triplets simultaneously. Hence, a soft margin method is used by inducing slack variables $\xi_{ijk} \geq 0$: $\mathbf{w}^T\mathbf{s}_{ij} - \mathbf{w}^T\mathbf{s}_{ik} \geq 1 - \xi_{ijk}$. This means that $\mathbf{w}^T\mathbf{s}_{ij} > \mathbf{w}^T\mathbf{s}_{ik}$ can be violated but this violation (or empirical loss) should be minimized: $\min \sum_{ijk} \xi_{ijk}$. However, focusing only on the empirical loss may result in over-fitting. Therefore, an ℓ_2 regularization term $\|\mathbf{w}\|^2$, similar to SVM, is imposed. Hence, the similarity metric learning corresponds to a constrained convex quadratic programming:

$$
\begin{aligned}
\min_{\mathbf{w},\xi_{ijk}} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\textstyle\sum_{ijk}\xi_{ijk} \\
s.t. \quad & \mathbf{w}^T\mathbf{s}_{ij} - \mathbf{w}^T\mathbf{s}_{ik} \geq 1 - \xi_{ijk}, \forall (i,j,k) \in \mathcal{T} \\
& \mathbf{w} \geq 0, \xi_{ijk} \geq 0, \forall (i,j,k) \in \mathcal{T}
\end{aligned} \tag{2}
$$

where C is a trade-off constant between the empirical loss and regularization.

4. Implementation

To solve (2) efficiently with some standard optimization software, especially when the codebook size is large, two issues have to be addressed. One is how to choose the triplets when the training set $\mathcal{T}$ is large. The other is how to optimize the speed and memory usage for a large codebook case.

We tackle these two issues with two nested recursive procedures. The outside loop, with subscript q, is for triplets selection where $\mathbf{w}$ is estimated on a growing set of triplets. The inside loop, with subscript r, is for estimating $\mathbf{w}$ in an efficient way. Instead of optimizing (2) directly (we call it Global Optimization, GO), Alternating Optimization (AO), which optimizes sub-vectors of $\mathbf{w}$ in a recursive way, is adopted.

4.1. Triplets Selection

There exist a large amount of image triplets even in the case of a small-size classification problem. For example, a two-class classification problem with 150 training samples in each class has more than 3×10^6 triplets $(150\times149\times150)$. To overcome this complexity problem, a subset of triplets was selected in [12] and showed to achieve considerable performance. In [4], half of the triplets were pruned according to their feature-to-set distances. In our approach, we adopt a bootstrapping strategy for triplets selection. The weight vector is estimated iteratively on a growing set of triplets. Except for the first iteration which randomly samples triplets, only triplets that do not fulfil $\mathbf{w}^T\mathbf{s}_{ij} > \mathbf{w}^T\mathbf{s}_{ik}$, are added in each iteration.

Suppose there are N triplets from the whole index set $\mathcal{T}$. We estimate the weight vector only on a subset $\mathcal{T}^{(q)}$ in the q-th iteration. The weight vector is refined using the stages discussed below and illustrated in Figure 1 (a). Note that in this process the estimation set $\mathcal{T}^{(q)}$ is different from the candidate set $\mathcal{T}_q$ and the adding set $\mathcal{T}_q^+$.

- **Initialization** The weight vector is initialized with equal weights $\mathbf{w}_0 = \mathbf{1}$. We first randomly pick a subset of triplets $\mathcal{T}_1 \subset \mathcal{T}$ with cardinality N_1 for estimating $\mathbf{w}_1$ in (2): $\mathcal{T}^{(1)} = \mathcal{T}_1$.

- **Update** In the q-th iteration ($q = 2, 3, ...$), $\mathbf{w}_q$ is estimated with alternating optimization (see Section 4.2) on $\mathcal{T}^{(q)}$ with cardinality N_q, with $N_{q-1} < N_q < N$. $\mathcal{T}^{(q)} = \mathcal{T}^{(q-1)} \cup \mathcal{T}_q^+$ with $\mathcal{T}_q^+$ the adding set of triplets that violates condition (1):

  $$\mathcal{T}_q^+ = \{(i,j,k) \in \mathcal{T}_q | \mathbf{w}_{q-1}^T\mathbf{s}_{ij} < \mathbf{w}_{q-1}^T\mathbf{s}_{ik}\} \tag{3}$$

  here $\mathcal{T}_q$ is a subset randomly selected from the remaining triplets: $\mathcal{T}_q \subset \mathcal{T}/\cup_{l=1}^{q-1}\mathcal{T}_l$. The error is estimated on the triplets subset $\mathcal{T}_{q+1}$: $\epsilon^{(q)} = |\mathcal{T}_{q+1}^+|/|\mathcal{T}_{q+1}|$ where $|\cdot|$ represents the cardinality of the set.

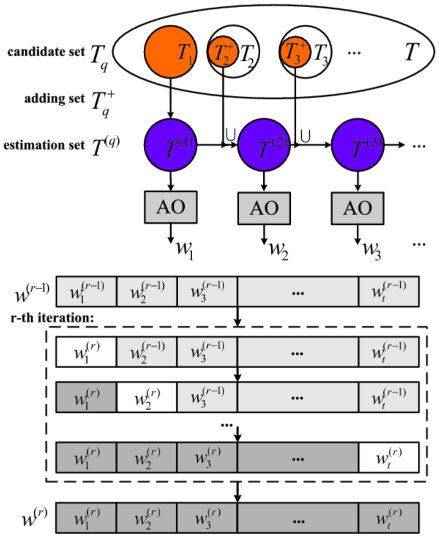
candidate set T_q
adding set T_q^+
estimation set $T^{(q)}$
T_1
T_2^+
T_2
T_3^+
T_3
...
T
$T^{(1)}$
$T^{(2)}$
$T^{(3)}$
∪
∪
...
AO
AO
AO
w_1
w_2
w_3
...
$w^{(r-1)}$
$w_1^{(r-1)}$ $w_2^{(r-1)}$ $w_3^{(r-1)}$ ... $w_t^{(r-1)}$
r-th iteration:
$w_1^{(r)}$ $w_2^{(r-1)}$ $w_3^{(r-1)}$ ... $w_t^{(r-1)}$
$w_1^{(r)}$ $w_2^{(r)}$ $w_3^{(r-1)}$ ... $w_t^{(r-1)}$
...
$w_1^{(r)}$ $w_2^{(r)}$ $w_3^{(r)}$ ... $w_t^{(r)}$
$w^{(r)}$
$w_1^{(r)}$ $w_2^{(r)}$ $w_3^{(r)}$ ... $w_t^{(r)}$

Figure 1. Top: diagrams for triplets selection. In each iteration, $\mathbf{w}_q$ is estimated on $\mathcal{T}^{(q)}$ with Alternating Optimization (AO), shown in the bottom figure. Bottom: one iteration in AO. Inside r-th iteration, the sub-vector in the white box is the item being optimized with all the remaining sub-vectors (in shadow boxes) fixed.

- **Stop criterion** The iteration stops if $\epsilon^{(q)} > \epsilon^{(q-1)}$ or $q > L_q$, where L_q is a maximal number of iterations.

Intuitively, adding the triplets that do not satisfy condition (1) to constraints will reduce the empirical error as it enables the learning process to focus on the triplets lying close to the separation hyperplane. However, adding too many such triplets may lead to overfitting. Hence the process stops when the error starts to increase or the process reaches the maximum number of iterations. Figure 2 (Left) shows the classification performance decreases after $q = 3$.

4.2. Alternating Optimization

Although with the proposed bootstrapping strategy, the number of constraints is significantly reduced, it is still computationally and memory intensive to estimate the high dimensional vector $\mathbf{w}_q$ using standard optimization software, such as Mosek[1]. We adopt the alternating optimization [1] to solve this bottleneck.

---

[1] http://www.mosek.com/

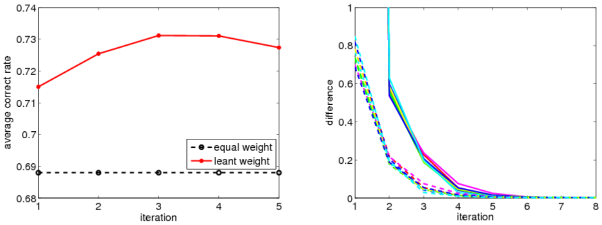

Figure 2. Left: the average correct rate on Scene-15 [7] in consecutive iterations of triplets selection. Right: The convergence of alternating optimization. The dash lines show the difference between the AO solutions and the global solution: $\|\mathbf{w}^{(r)} - \mathbf{w}^*\|/\|\mathbf{w}^*\|$, while the solid lines represent $(\|\mathbf{w}^{(r)} - \mathbf{w}^{(r-1)}\|/\|\mathbf{w}^{(r-1)}\|)$ (when $r = 1$, the average difference is 17.34). Experiments are performed 10 times with randomly selected training and test sets, visualized with different colors.

Formulation Alternating Optimization (AO) is an iterative procedure for minimizing a function by optimizing a subset of variables recursively. In our approach, we partition the weight vector into t parts: $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_t)^T$, $\mathbf{w}_i \cap \mathbf{w}_j = \emptyset, i \neq j$. In contrast to Global Optimization (GO) in (2), which optimizes the whole vector, the AO optimizes a subvector $\mathbf{w}_l$ $(l = 1, 2, ..., t)$ one by one in the iterative procedure with the objective function as follows.

$$\min_{\mathbf{w}_l, \xi_{ijk}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{ijk} \xi_{ijk} \tag{4}$$

The only difference between (4) and (2) is the optimized variable, $\mathbf{w}_l$ and $\mathbf{w}$ respectively. To distinguish from solution $\mathbf{w}^*$ obtained with GO (2), the solution with AO (4) is referred to as $\tilde{\mathbf{w}}^*$. To simplify notation, the sub-index q is omitted in this section, i.e., $\mathbf{w}_q^{(r)}$ and $\mathbf{w}_{ql}^{(r)}$ are referred to as $\mathbf{w}^{(r)}$ and $\mathbf{w}_l^{(r)}$, respectively. Note that the alternating optimization is performed in each triplets selection iteration, which is visualized in Figure 1 (Top).

In the r-th iteration, $\mathbf{w}^{(r)} = (\mathbf{w}_1^{(r)}, \mathbf{w}_2^{(r)}, ..., \mathbf{w}_t^{(r)})^T$. As visualized in Figure 1 (Bottom), sub-vectors $\mathbf{w}_l^{(r)}$, $l = 1, ..., t$, are optimized one by one by the following quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w}_l^{(r)}, \xi_p} \quad & \tfrac{1}{2}\|\mathbf{w}_l^{(r)}\|^2 + C\textstyle\sum_p \xi_p \\ s.t. \quad & \mathbf{w}_l^{(r)T}\boldsymbol{\Delta}_{pl} \geq 1 - C_l^{(r)} - \xi_p, \forall p \in \mathcal{T} \\ & \mathbf{w}_l^{(r)} \geq 0, \xi_p \geq 0, \forall p \in \mathcal{T} \end{aligned} \tag{5}$$

where $\boldsymbol{\Delta}_p = \mathbf{s}_{ij} - \mathbf{s}_{ik}$ with ijk simplified with p. $\boldsymbol{\Delta}_p$ is accordingly partitioned as $\boldsymbol{\Delta}_p = (\boldsymbol{\Delta}_{p1}, \boldsymbol{\Delta}_{p2}, ..., \boldsymbol{\Delta}_{pt})^T$. $C_l^{(r)} = \sum_{j=1}^{l-1} \mathbf{w}_j^{(r)T}\boldsymbol{\Delta}_{pj} + \sum_{j=l+1}^{t} \mathbf{w}_j^{(r-1)T}\boldsymbol{\Delta}_{pj}$ is a constant since $\mathbf{w}_j, j \neq l$ are fixed.

The iteration terminates when the stopping criterion $\|\mathbf{w}^{(r)} - \mathbf{w}^{(r-1)}\|/\|\mathbf{w}^{(r-1)}\| < \epsilon$ or $r > L_r$ is satisfied.

Convergence of AO The convergence of AO has been investigated in [1]. We compare GO with AO on "bike" category in GRAZ02 dataset [10]. $\mathbf{w} \in \mathbb{R}^{2000}$ is partitioned into 20 parts evenly. The experiments are carried out 10 times with different randomly selected training samples. Consistent convergence behavior is observed, as shown in Figure 2 (Right). The AO converges very fast and it does converge to the GO solution $\mathbf{w}^*$. In our experiments, we use the Mosek toolbox to solve the quadratic problems (2) and (4). The GO needs 36 hours in average with 30,000 triplets. By contrast, AO takes one hour and 15 minutes for the same setting.

5. Experiments on Synthetic Data

We first perform experiments on a two-category synthetic dataset. In this simulation, neither actual image nor codebook is generated. We start the simulation at the intermediate step of bag-of-words approach, namely, at codebook occurrence histogram. 8-dimensional codebook occurrence histograms $\mathbf{h} = (h_1, h_2, ..., h_M), M = 8$ are generated to simulate the normalized histograms that are typically computed for every image. Due to the normalization, these histograms are located on a hyperplane defined by $\sum_i h_i = 1$. The distributions of these codeword occurrences in the examples from the two categories are displayed in Figure 3 (Left). Each histogram dimension is distributed according to Gaussian or uniform distributions whose detailed coefficients are given on the top of the distribution in Figure 3. It is easy to see that the most discriminative codeword is h_1 followed by h_2, since their distributions differ most for the two categories. While the least discriminative dimensions are h_5, h_6 and h_7.

We perform a classification experiment in which the histograms are simulated according to distributions from Figure 3 (Left). The Equal Error Rate (EER) of the Receiver-Operating Characteristic (ROC) is adopted for evaluation on 150 testing histograms per category. Readers are referred to section 6.1 for how the ROC curve is computed. We generate several sets of training data with increasing number of training histograms per category. Each experiment is run 50 times for a given number of training samples. The average EER and its standard deviation are shown in Figure 4. Note that we do not use AO, since GO can be done efficiently for this small size codebook. Intuitively, the performance is lower for a small number of training examples due to insufficient training data. However, the codebook with learnt weights gives much higher performance than that with equal weights. An interesting observation is that the learnt weights help much when there is few training examples compared to the codebook size, while only slight improvement is observed in the case of large number of training examples. This significant improvement for insufficient training examples is exactly what we expect in recognition of real image data. In most recognition bench-

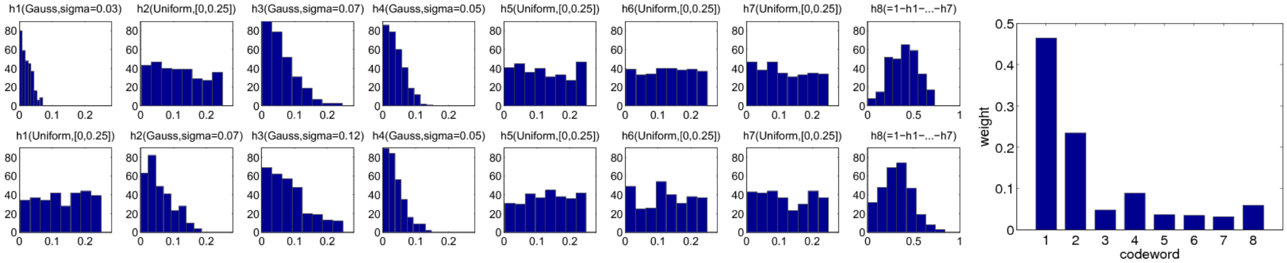

Figure 3. Left 16 figures: distribution for each dimension of the simulated data. Top row: category 1. Bottom row: category 2. (x axis: the i-th codeword occurrence: h_i, y axis: occurrence). Right: learnt weights averaged in 50 runs with 15 training samples per category.

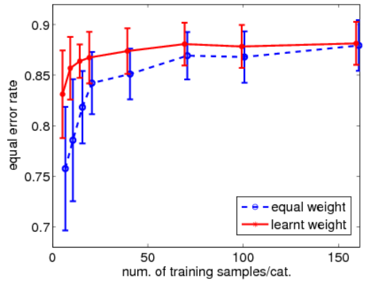

Figure 4. Means and standard deviation of ROC equal error rates with varying training samples on 8-dim simulated histograms for 50 runs.

marks, there are hundreds of training examples, while the codebook consists of a few thousands of codewords, which leads to underestimated high dimensional occurrence histograms.

The difference of performance for various numbers of training examples comes from the fact that the non-discriminative entries (eg. h_5, h_6, h_7) do not help in classification, neither do they decrease the score in the case of sufficient training samples. However, they strongly affect the performance in the case of insufficient training samples. Hence, the learnt weights can reduce this effect by assigning small weights to the non-discriminative codewords. The learnt weights for 15 examples per category are shown in Figure 3 (Right), which is consistent with what was expected from the distributions in Figure 3 (Left). Higher weights are assigned to the first two codewords, lower weights to the others.

6. Experiments on Real Data

In this section we conduct experiments on standard image classification and retrieval data. The classification experiments are performed with Spatial Pyramid Match Kernel (SPMK) [7] and k-nearest neighbor classifier, in which 2000 codewords with level $L = 1$, i.e., 1×1 and 2×2 cells, are used. Densely sampled regions combined with SIFT [8] are used. All the results reported in this section are averaged over 10 runs with randomly selected different training and test images.

6.1. Binary Classification on GRAZ02 & GRAZ01

We carry out object vs. background classification on two datasets with large intra-class variations, namely, GRAZ02 [10] and GRAZ01 [10]. GRAZ02 and GRAZ01 contain 3 and 2 object classes respectively as well as the background class. We follow the evaluation protocol from [10]. For GRAZ02, we train on 150 object images and 150 background images and test on 150 images, half from the object and half from the background class. For GRAZ01, these three numbers equal 100.

Figure 5 (b) illustrates the learnt weights for the 2000-dimensional “person” codebook. Sample patches from the three top-weighted codewords are shown in Figure 5(c) and highlighted in Figure 6 with solid red rectangles. The most distinctive codewords for “person” represent the edges of the legs and the body. The corresponding codeword occurrence distributions on “person” and “no person”, similar to Figure 3, are shown to the right of the patches, in which the distribution differ for the two categories. According to the distributions, much fewer such codewords occur in “no person” images than in “person” images, which shows the codeword’s discriminability. In contrast, both categories have very similar distributions of the zero-weighted codewords, as shown in Figure 5 (f). The first two zero-weighted codewords corresponds to the ground, wall and tree texture, while the last one is a vertical edge which is not helpful for distinguishing “person” and “no person” either. These three codewords are highlighted in Figure 6 with solid blue rectangles.

Patches randomly picked from the 6 highest-weighted codewords for “bike” and “car” are displayed in Figure 7 (Top). The patches mainly come from different parts of the bike wheels, bottom of the car and corners of car windows. It clearly shows that the codewords learning is able to find the most distinctive features for specific categories. The 6 highest-weighted and 10 lowest-weighted (zeros) ones are also indicated by red and blue in bike and car images in Figure 7 (Bottom). Most of the highest-weighted codewords are located on the object and the low-weighted patches are on the background. This indicates that the learnt high-weighted codewords can be used for object localization.

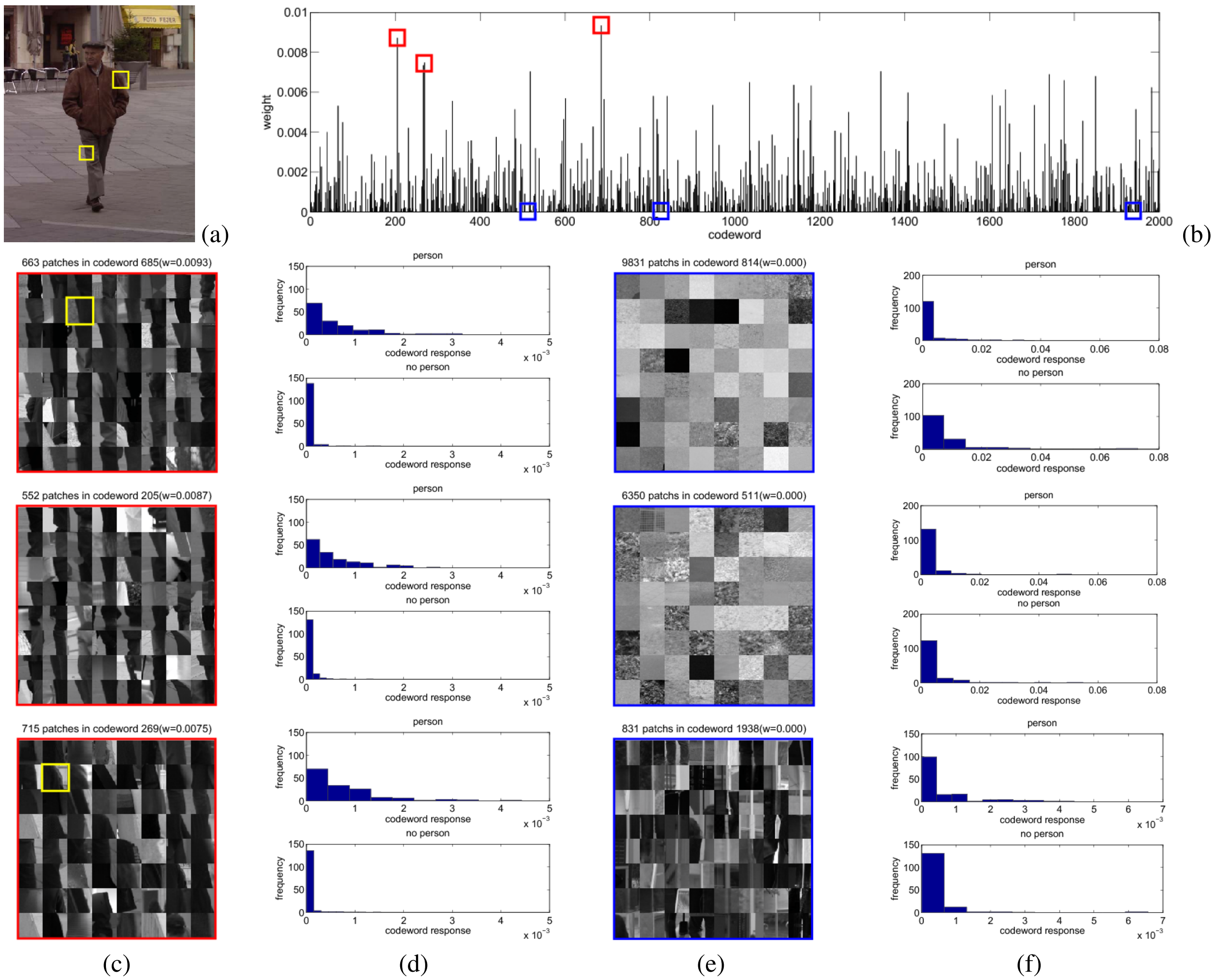

Figure 5. (a) One "person" image shown with two high-weighted regions. (b) the learnt weights for codebook for "person/no person". (c) sample patches in the three highest-weighted codewords highlighted in (b) with red rectangles (e) sample patches in the three zero-weighted codewords highlighted in (b) with blue rectangles (d)(f) the codeword occurrence distributions in the two categories for the left codewords.

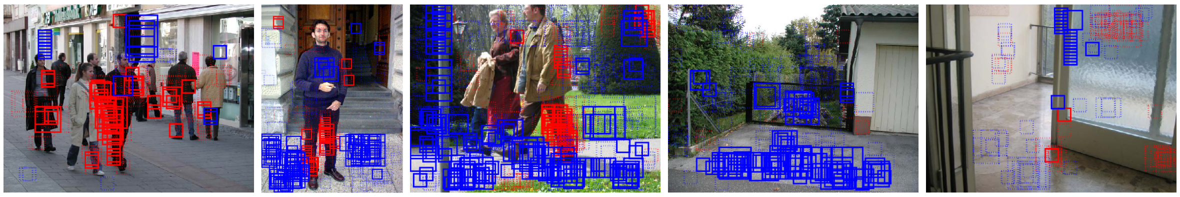

Figure 6. The 10 highest-weighted codewords are highlighted with red rectangles in "person" and "no person" images, while the 10 zero-weighted codewords are denoted with blue rectangles. The solid rectangles are the six codewords shown in Figure 5.

We adopt ROC curve for evaluation as in [10]. For a test example a, all the training samples are ranked according to the similarity with this test example in a decreasing order. Let $y_i \in \{0, 1\}$ be the label of the i-ranked training sample b_i: $y_i = 1$ if b_i belongs to positive class. The confidence of the positive category with is defined as: $p^{(I)}(a) = \sum_{i=1}^{I} \frac{\sum_{j=1}^{i} y_j}{i} y_i$. The ROC curve is generated by thresholding this confidence for each test example.

Table 1 (Top) summarizes the ROC equal error rates of our approach as well as the state-of-the-art results on GRAZ02. Compared with the equally weighted codebook, the learnt codebook improves the performance significantly, especially for "car" and "person", where the improvement of nearly 10% is achieved. Our approach outperforms [10] significantly and also exceeds the recent work in [9], in which randomized clustering forests are used for generat-

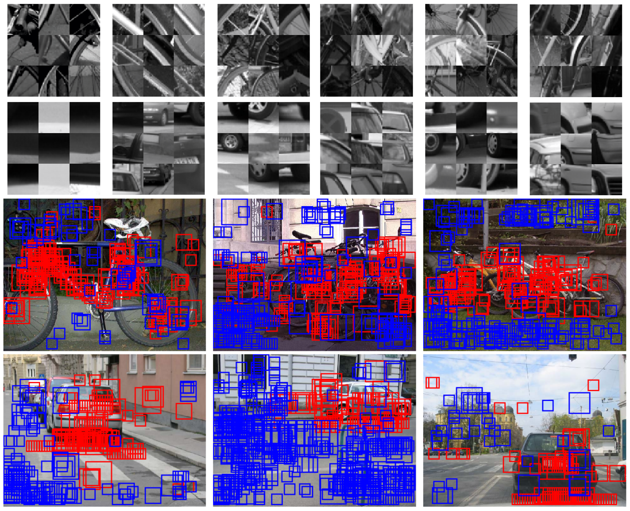

Figure 7. Top two rows: The 6 highest-weighted codewords learnt in "bike" and "car". Bottom two rows: Highest and zero-weighted codewords are highlighted with red and blue rectangles.

GRAZ02, SPMK, M=2000, L=1

	[10]	[13]	[9]	equal_w	our approach
bike	76.5	**89.5**	84.4	81.3±1.3	**84.9±2.1**
car	70.7	**80.2**	79.9	72.9±0.8	**82.2±2.0**
person	81.0	**85.2**	–	75.6±0.8	**84.9±2.0**

GRAZ01, SPMK, M=2000, L=1

	[10]	[7]	[18]	equal_w	our approach
bikes	86.5	86.3	**92.0**	83.3±1.2	**86.7±4.6**
person	80.8	82.3	**88.0**	80.7±2.3	**84.0±2.0**

Table 1. The ROC equal error rate (%) on GRAZ02 and GRAZ01

ing codebook. Compared with the best performance on this dataset so far [13], we obtain lower performance on "bike", but better on "car" and comparable on "person". In [13], SVM classifier is used on a selected 5000 or 10,000-size codebook, which is computationally more expensive. Our approach uses kNN classifier with only 2000-size codebook.

Table 1 (Bottom) shows consistent improvement compared to the equal-weighted codebook on GRAZ01. Our approach achieves similar results to [10] and [7] for "bikes", but better for "person". The performance is lower than the best performance in [18] where two types of feature detectors (Harris-Laplace and Laplace), two descriptors (SPIN and SIFT) and SVM were used.

6.2. Multi-Class Classification on Scene-15

We also test our approach on a scene classification dataset from [7] which consists of 15 categories. 100 images randomly selected in each category are for training and the rest for testing. We pick $N_0 = 90,000$ initial triplets as constraints, $N_q = 90,000$ candidate triplets in each iteration. We perform experiments for two codebook sizes with k-nearest neighbor classifier on this dataset. The results shown in Table 2 indicate that the learnt weights can hardly help for small size codebook where the training examples is sufficient. This is consistent with the observation on the synthetic dataset in Section 5. In contrast, 4.6% improvement is achieved for large-size codebook ($M = 2000$, $L = 1$). Surprisingly, the 2000-size equal-weighted codebook performs even worse than 200-size codebook if the level difference is ignored. We argue that it is due to insufficient number of training examples for the large codebook. Despite this, the weighted codebook given by the proposed approach can compensate for insufficient data. The state-of-the-art performance in this dataset is 81.4% reported by [7]. However, strong SVM classifier has been used there as opposed to our approach based on kNN.

	equal_w	our approach
$M = 200$, $L = 2$	70.3±1.1	70.8±0.7
$M = 2000$, $L = 1$	68.8±1.4	**73.4±1.0**

Table 2. The average correct rate (%) on Scene-15 with K-nearest neighbor classifier ($K = 15$)

Discussion We also compare the equal weight scheme with our learning scheme by replacing kNN with SVM. 83.5% classification rate is achieved with equal weight scheme and the learnt weights have marginal effect on the performance if SVM is used. This is consistent with the observations in [13] and shows that codeword weighting is more suitable for kNN classifiers. These are still more applicable in the example based retrieval scenario than SVM. Note that we learn global weights for all categories. Since each codeword may have different contribution for different categories, the performance can be boosted if the weights are class-specific, in a similar way to [4, 15]. Our approach can also be extended to class-specific case although the number of parameters to optimize significantly increases.

6.3. Image Retrieval on Oxford5K

To validate our learnt codebook weighting scheme, we compare to the commonly used tf-idf weighting strategy in the context of image retrieval in the *Oxford* Building dataset [11]. It contains 5,062 images for 11 Oxford landmarks with manually annotated ground truth and distracters. We follow the setup in [11], where 5 images per landmark are used for query. The Average Precision (AP) is taken as the performance metric for measuring the retrieval accuracy.

We learn the codebook weights for each landmark with randomly selected 7 positive images and 500 negative images. The 4,555 (=5,062-7-500) remaining images are used for testing. Note that the 7 positive images do not include the 5 queries. This process is repeated 5 times and the mean performance is reported. This offline supervised learning for retrieval is applicable in situations such as searching for

cat.	equal_w	tf_idf	our approach
all souls	49.31	50.47	**63.94**
ashmolean	55.61	57.92	**60.31**
bodleian	55.73	53.06	**72.70**
christ church	58.85	59.14	**62.51**
hertford	70.53	**71.09**	66.40
magdalen	12.22	13.27	**20.47**
radcliffe camera	65.42	**66.01**	60.20
MAP	52.53	53.01	58.08

Table 3. Average precisions (%) on Oxford5K

a logo, a person or a specific landmark, where many training images are available in advance. We test the learning strategy on 7 landmarks out of 11 as there are too few training examples for the remaining 4.

To make our results comparable, we directly use the codebook assignments mapped from a 1M-size codebook downloaded from Oxford website, where the local features are generated with SIFT descriptors on multi-scale Hessian interest points. The similarity between two images is the weighted histogram intersection with standard bag-of-words model ($M = 10^6$, $L = 0$). For the implementation of the weight learning, all 21,000 constraints ($= 7\times6\times500$) are used without triplets selection and the long weight vector is partitioned into 300 subsets for AO.

As shown in Table 3, the tf-idf only slightly outperforms equal weights. The performance of our learnt weights exceeds that of the tf-idf scheme for most categories. In terms of the mean AP on the 7 landmarks, we observe a significant gain of 5% over tf-idf weights. This experiment confirms the superiority of the proposed learning method over the commonly-used tf-idf scheme.

7. Conclusions

We have proposed a machine learning based approach for estimating codebook weights. The complexity problem has been addressed by iterative constraint selection and alternating optimization which has been shown to converge to the global optimum. Experiments on both synthetic and real datasets demonstrate that our learning approach can improve the classification and retrieval performance over the commonly used equally weighting and tf-idf weighting schemes, especially when the training examples are insufficient for a large size codebook. The approach can also be used for codeword selection to reduce the dimensionality of codeword occurrence histograms in large scale image classification and retrieval problems, as more than half of the codewords are zero weighted in our experiments. Moreover, a potential research direction can be object localization with the help of our learnt codebook.

Acknowledgments

This research was supported by UK EPSRC EP/F0034 20/1 grant and the BBC R&D.

References

[1] J. C. Bezdek and R. J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11(4):351–368, 2003.

[2] S. Chen, B. Ma, and K. Zhang. On the similarity metric and the distance metric. *Theoretical Computer Science*, 410:2365–2376, 2009.

[3] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop of ECCV*, pages 1–22, 2004.

[4] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, pages 1–8, 2007.

[5] K. Grauman and T. Darrell. The pyramid match kernel: discriminative classification with sets of image features. In *ICCV*, volume 2, pages 1458–1465, 2005.

[6] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, volume 1, pages 604–610, 2005.

[7] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[9] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *PAMI*, 30(9):1632–1646, 2008.

[10] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *PAMI*, 28(3):416–431, 2006.

[11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR*, 0:1–8, 2007.

[12] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2004.

[13] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *ICCV*, pages 1–8, 2007.

[14] J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008.

[15] K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *ICML*, pages 1160–1167, 2008.

[16] J. M. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, volume 2, pages 1800–1807, 2005.

[17] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering. Michigan State University, 2006.

[18] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73(2):213–238, 2007.