Cogn Comput
DOI 10.1007/s12559-012-9137-4

Supervised Learning and Codebook Optimization for Bag-of-Words Models

Mingyuan Jiu · Christian Wolf · Christophe Garcia · Atilla Baskurt

Received: 28 July 2011 / Accepted: 3 April 2012

Abstract In this paper, we present a novel approach for supervised codebook learning and optimization for bag-of-words models. This type of models is frequently used in visual recognition tasks like object class recognition or human action recognition. An entity is represented as a histogram of codewords, which are traditionally clustered with unsupervised methods like *k*-means or random forests and then classified in a supervised way. We propose a new supervised method for joint codebook creation and class learning, which learns the cluster centers of the codebook in a goal-directed way using the class labels of the training set. As a result, the codebook is highly correlated to the recognition problem, leading to a more discriminative codebook. We propose two different learning algorithms, one based on error backpropagation and the other based on cluster label reassignment. We apply the proposed method to human action recognition from video sequences and evaluate it on the KTH data set, reporting very promising results. The proposed technique allows us to improve the discriminative power of an unsupervised learned codebook or to keep the discriminative power while decreasing the size of the learned codebook, thus decreasing the computational complexity due to the nearest neighbor search.

M. Jiu (✉) · C. Wolf · C. Garcia · A. Baskurt
Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, Villeurbanne 69621, France
e-mail: jmyhit@gmail.com

C. Wolf
e-mail: christian.wolf@liris.cnrs.fr

C. Garcia
e-mail: christophe.garcia@liris.cnrs.fr

A. Baskurt
e-mail: atilla.baskurt@liris.cnrs.fr

Keywords Bag-of-words models · Supervised learning · Neural networks · Action recognition

Introduction

The *bag-of-words* (BoW) model has been widely used in computer vision applications such as object class [1, 2] or human action recognition [3]. In these applications, the objective is to recognize high-level information (objects, actions) from a large quantity of low-level data, for example, image or video pixels. BoW has proved to be an efficient representation in this context. This popular model was first introduced in natural language processing, in which each document is expressed as a histogram of frequencies of orderless words. In order to employ the BoW model in computer vision applications, an image or a video is treated as a document, which can be considered as a collection of interesting local events often called visual concepts [1]. Information on the presence of these visual concepts, that is, whether each one of them is present or not, and with which frequency, serves as an indicator of the contents.

Visual concepts can be generated in different ways, usually through the extraction of discriminant and invariant descriptors (features) around local primitives like interest points, patches, regions, edges, followed by clustering in order to identify clusters in feature space of descriptors. The thus-obtained clusters are considered as visual concepts or visual codewords. A set of such visual codewords produces a visual codebook.

Traditionally, a visual codebook is learned by unsupervised clustering or vector quantization of feature vectors extracted from the local primitives in the image or video, often with algorithms such as *k*-means [1] or random

Published online: 24 April 2012

forests [4]. The BoW for a new image or video is calculated in a similar way: extraction of descriptors on local primitives, projection of the descriptors on the codebook precalculated on a training set, and calculation of a histogram of the occurrences of each codeword of the codebook. An image or a video is classified passing the BoW model to any learning machine, for instance, a support vector machine (SVM) or a neural network (NN).

The traditional method of codebook creation through unsupervised clustering ignores the class labels of the feature vectors in the training set. Of course, the labels are used for the classification of the BoW models, but not for the creation of the codebook. As a consequence, the visual codebook is less discriminative. In order to create a more discriminative codebook, several attempts have been made to reveal the semantic relations between the codewords of *k*-means. In the work of Liu [5], they iteratively obtain an optimal and compact codebook via maximal mutual information. At each iteration, they merge two clusters that have minimum loss in mutual information. The iterative procedure continues until a threshold of maximal mutual information or minimum cluster number is achieved. The optimal codebook size is found by unsupervised learning. In [6], Liu uses a diffusion map to embed a mid-level codebook in a semantic codebook. However, it is not appropriate to measure semantic distances using diffusion distances. In recent work, Saghafi [7] proposes a concept space to illustrate the semantic relations between the visual codewords. They apply generative models such as latent semantic analysis (LSA) and probabilistic latent semantic analysis (pLSA) to discover the latent semantic relations between the initial codewords. In contrast to the unsupervised pLSA learning framework in which the number of latent topics is equal to the number of classes [8], the number of topics is variable in this method. In all methods mentioned above, the codebooks are created by unsupervised methods, and the label information of feature vectors is ignored in the codebook creation. Intuitively, the discriminative power of the codewords could be increased by learning using the label information, as we propose in this paper.

In contrast to unsupervised codebook learning, we propose a supervised learning and codebook optimization framework. The whole sequence, codebook creation and class learning, is formulated as an artificial NN, and the error gradient information is used to update the codebook cluster centers as well as the classical multilayer perceptron (MLP) weights for class recognition.

The learning and optimization framework we present in this work are well suited for any application for which bag-of-words models can be successfully used. We restrict ourselves here to human action recognition in videos, for which BoW models and extensions are widely used [3, 8–11]. The improvements we propose make the codebook more discriminative and, therefore, are likely to improve many of the existing extensions of the basic BoW model, such as, for instance, correlograms [5], topic models [8], local grouping and compound features [9], spatial co-occurrences of pairs of features [12–14] and parts-based models [15].

To the best of our knowledge, this is the first attempt to combine codebook learning with action classification in a unified framework.

The amount of literature on action recognition skyrocketed in the last years, and it is not possible anymore to give an exhaustive account in this restricted space. We refer the interested reader to some very recently published surveys [16–18]. While early work on modeling human activities focused on articulated motion (e.g., [19]), most recent work on activity and event recognition does not explicitly model the human body. Instead, the current state of the art focuses on sparse local features like interest points and space-time interest points, as the work cited above, or on motion segmentation through background subtraction [20–22], dense optical flow [23] or other holistic features [24, 25], with possible hybrid methods [26–28] and classification through dense matching [29, 30]. Fully taking into account spatial relationships through graph matching has recently been proposed [31], but this requires matching against several graph models per action class.

Pure statistical and unstructured machine learning without feature extraction is difficult in this context due to several reasons as follows: (1) the non-rigid nature of the relevant information, (2) the mixture of relevant motion information and irrelevant texture information in the signal and (3) the high dimension of the feature space in which the data are embedded. For these reasons, machine learning of human actions has been dominated by methods learning the temporal evolution of features like HMMs, semi-Markov models and dynamic belief networks [32–39]. Typically, a vectorial description is created frame per frame, and its temporal evolution is modeled and learned. Other learning-based methods include biologically inspired ones [40], convolutional deep learning [41, 42], methods based on topic models [8], boosting low-level features [43], trajectory matching [44], statistics calculated on the results of tracking [45], learning of spatiotemporal predicates and grammars [13, 46, 47] and other probabilistic graphical models [48].

The paper is organized as follows. Our formulation of the BoW model and the subsequent classification phase as a unique global neural model is presented in Sect. "The Neural Model". Section "Joint Codebook and Class Label Learning" describes the integrated and joint learning algorithm that updates the cluster centers as well as the MLP weights discriminating between the targeted classes. Two different learning algorithms are presented, a classical

backpropagation algorithm and cluster reassignment algorithm specific to the BoW model. In Sect. "Experimental Results," we evaluate the proposed approach on the public KTH human action data set [11]. Section "Conclusion" gives a conclusion.

The Neural Model

In our application, space-time interest points are calculated on each video, and discriminant and invariant features are calculated on a space-time cuboid around each interest point location. Initially, a video is therefore described as a collection of feature vectors. In traditional ways to translate this description into a BoW model, codebook creation and learning of the BoW models of the training set are treated as two different phases addressed with two different methods. Here, we present a novel formulation as a single artificial NN.

In classical NNs, each entity is classified separately by the learned NN after a stimulation phase. In our proposed model, for each video, multiple feature vectors (one per interest point) are presented sequentially while the NN integrates this information internally. Classification is done after all feature vectors have been presented. The scheme in Fig. 1 illustrates this concept. We first give an overview of its purpose before explaining each layer in detail.

The NN consists of two parts: an initial part at the left that processes feature vectors and projects them to a codebook. The cluster centers of this codebook are stored as "weights" of this part of the NN. While passing through the left part of the network, the feature vector is translated into a binary vector indicating which cluster center it activates. When several feature vectors are presented, this information is integrated into a BoW model. The second part of the NN is a classical MLP that takes a decision on the action class for each BoW model.

The Layers of the Proposed NN Model

The input layer consists of a set $\boldsymbol{a}$ of M input nodes $\boldsymbol{a} = [a_1, \ldots, a_M]^T$ corresponding to the feature values assigned to a single local primitive, that is, an interest point.

The N nodes of the second layer $\boldsymbol{b} = [b_1, \ldots, b_N]^T$ correspond to the distances of the input feature vectors to each of N cluster centers. To each node i and each distance b_i is thus assigned a cluster center $\boldsymbol{w}_i^{cc}$, that is, a vector of dimension M, which is involved in the distance computation:

$$b_i = ||\boldsymbol{a} - \boldsymbol{w}_i^{cc}|| \tag{1}$$

The N nodes of the third layer compute an indicator of the nearest cluster center. The nearest corresponding node will be assigned 1, the other nodes 0. The minimum distance will result in the largest value. This is approximated through a softmin function, similar to the classical softmax:

$$c_i = g^b(b_i) = \frac{\exp(-b_i/T)}{\sum_j \exp(-b_j/T)} \tag{2}$$

where T is a parameter controlling the stability of the softmin function.

The network layers described above propagate the stimulation of a single feature vector corresponding to a single local primitive. As mentioned before, in our stimulation strategy, multiple feature vectors of the same entity (a video in our case) are presented iteratively, resulting in different values for different feature vector p, which we will denote as c_i^p. The nodes of the next layer integrate the responses for a single video over all P points:

$$d_i = \sum_{p=1}^{P} c_i^p \tag{3}$$

The next two layers, $\boldsymbol{e}$ and $\boldsymbol{f}$, are a classical MLP with weights $\boldsymbol{w}^e$ and $\boldsymbol{w}^f$ and activation function $g(x)$:

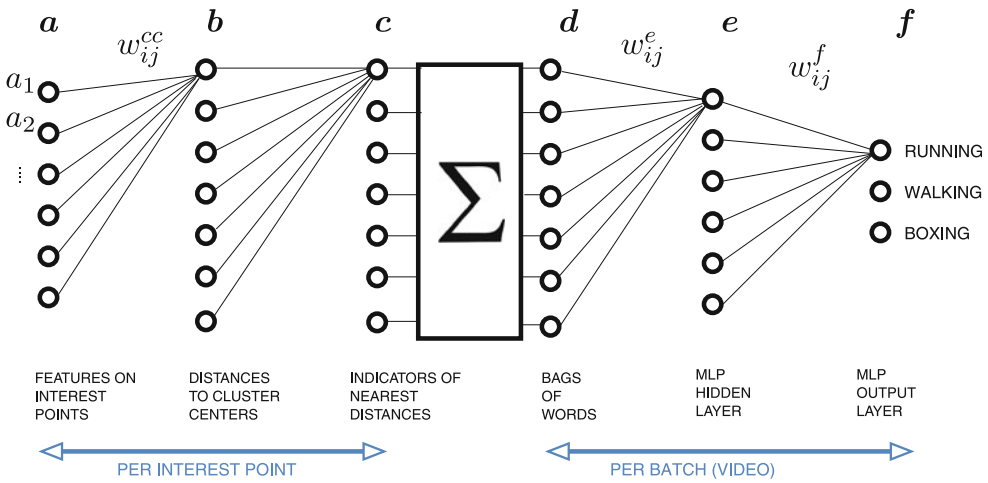

Fig. 1 A scheme of the different layers of the neural network. The *left part* is stimulated per interest point. The *right part* is a classical MLP taking decisions for each video

$$\bar{e}_i = \sum_j w_{ij}^e d_j, \quad e_i = g(\bar{e}_i) \tag{4}$$

$$\bar{f}_i = \sum_j w_{ij}^f e_j, \quad f_i = g(\bar{f}_i) \tag{5}$$

The last layer is thus the output layer with the set of nodes $\boldsymbol{f} = [f_1, \ldots, f_C]^T$, one for each of the C actions.

Joint Codebook and Class Label Learning

Assuming a set of video files with interest points and their corresponding feature vectors, as well as a groundtruth action label per video, backpropagation propagates the error between the stimulated response and the groundtruth back to the input layers, adjusting weights during the process. In our case, this means adjusting two different types of parameters, namely the weights $\boldsymbol{w}^{e,f}$ of the MLP and the cluster centers $\boldsymbol{w}^{cc}$. The weights $\boldsymbol{w}^{e,f}$ of the MLP are updated with a classical error backpropagation scheme, which is recalled in Sect. "MLP Learning". The cluster centers can be updated by two algorithms, which are, respectively, addressed in Sects. "Supervised Codebook Learning with Error Backpropagation" and "Supervised Codebook Learning Through Cluster Reassignment". These two different types of weights are sequentially and iteratively learned in our framework. At any time, the proposed system only learns one type of weights. The former is used to learn an optimal MLP model, and the latter makes use of the backward errors of the optimal MLP. The pseudocode of the proposed framework is shown in Algorithm 1.

MLP Learning

In the proposed framework, we adopt a MLP network with one output unit for each class. Sigmoid and softmax activation functions are, respectively, employed in the hidden layer and output layer. The errors are computed by the cross-entropy function [49]. An 1-of-c coding scheme is used to describe the target output. The weights $\boldsymbol{w}^{e,f}$ are adjusted in the classical way:

1. The input vector $\boldsymbol{a}$ forward propagates through the network, until all the input vectors over the whole video are summed in the integration layer, and then the activations of all hidden and output layer units are computed.
2. Given a desired output t_j from the groundtruth, compute the error in the output layer:
$$\delta_j^f = (f_j - t_j)$$
3. Backpropagate the error into the hidden layer:
$$\delta_j^e = g'(\bar{e}_j) \sum_k w_{jk}^f \delta_k^f$$
4. Backpropagate the error to the input layer:
$$\delta_j^d = \sum_k w_{jk}^e \delta_k^e$$
5. Compute the increments for all the weights:
$$\Delta w_{ij}^f = \eta \delta_j^f \bar{e}_i \quad \Delta w_{ij}^e = \eta \delta_j^e \bar{d}_i,$$

where η is a learning parameter.

Algorithm 1 The iterative codebook learning framework

Input: F_{tr} (training features), F_{val} (validation features)
Output: $\boldsymbol{w}^{cc}$ (optimal codebook)

```
w^cc ← k-means;
repeat
  B_tr ← Bags-of-words computation (w^cc, F_tr)
  B_val ← Bags-of-words computation (w^cc, F_val)
  w^{e,f} ← random;
  w^{e,f} ← MLP learning (w^{e,f}, w^cc, B_tr, L_tr)
  w^cc ← Cluster center learning (w^{e,f}, B_tr, L_tr) (Sect. "Supervised Codebook Learning with Error Backpropagation" or "Supervised Codebook Learning Through Cluster Reassignment");
  E ← Validation error (w^{e,f}, w^cc, B_val, L_val);
Until convergence (E)
```

Supervised Codebook Learning with Error Backpropagation

The classical error backpropagation algorithm can be adapted to our novel formulation. In particular, the errors in layer $\boldsymbol{d}$ are continued backpropagating into layers $\boldsymbol{b}$, which can directly act on the cluster centers. A particularity of our model is the integrator between the *per-feature-vector* part and the *per-video* part. The errors in layer $\boldsymbol{c}$ (nearest distance indicator for each feature point) can be approximated by taking the error of layer $\boldsymbol{d}$ (bag-of-words) and equally distributing it over the corresponding feature points:

$$\delta_i^c = \frac{\delta_i^d}{d_i}. \tag{6}$$

Subsequently, the error can be further backpropagated to the previous layer and the cluster centers $\boldsymbol{w}^{cc}$ can be updated in the same manner as the weights of the MLP. In the above, the distance of each node i of layer $\boldsymbol{b}$ can be computed as follows:

$$b_i = ||\boldsymbol{a} - \boldsymbol{w}_i^{cc}||^2 = \sum_j (a_j - w_{ij}^{cc})^2. \tag{7}$$

In traditional MLP, the discriminant function consists of a nonlinear combination (activation function) that acts on the

linear sum of the input variables, and the coefficients are the parameters of the model. In this case, our discriminant function is a nonlinear activation function which acts on a sum of a two-order polynomial of input variables. Since b_i is differentiable with respect to w_{ij}^{cc} and the activation function (softmin function) is also differentiable, we thus can resort to gradient descent to adjust the cluster centers.

Let us first note that sum of error is used at layer $\boldsymbol{c}$:

$$E = \sum_m E^m = \sum_m \sum_i \frac{1}{2}(c_i^m - \widetilde{c_i^m})^2 \tag{8}$$

where m corresponds to the index of the input feature vector and c_i^m is the forward response of the network for feature m, $\widetilde{c_i^m}$ is the optimal value for the best cluster center according to the groundtruth, which can be approximated using Eq. (6). According to gradient descent:

$$\frac{\partial E}{\partial w_{ij}^{cc}} = \sum_m \frac{\partial E^m}{\partial w_{ij}^{cc}} = \sum_m \frac{\partial E^m}{\partial b_i^m}\frac{\partial b_i^m}{\partial w_{ij}^{cc}} \tag{9}$$

In the following, only the values for a single feature m will be considered. For clarity we therefore remove the superscript m from the notation, in particular E^m will be noted as E.

In order to evaluate the derivative of the softmin function, we should consider the input to all the outputs units (layer $\boldsymbol{c}$). So we have

$$\frac{\partial E}{\partial b_i} = \sum_{i'} \frac{\partial E}{\partial c_{i'}}\frac{\partial c_{i'}}{\partial b_i}. \tag{10}$$

From Eq. (6), we can get

$$\frac{\partial E}{\partial c_{i'}} = \delta_{i'}^c. \tag{11}$$

Applying the quotient rule to Eq. (2), we get

$$\frac{\partial c_{i'}}{\partial b_i} = -\frac{1}{T}(c_{i'}\delta_{ii'} - c_{i'}c_i) \tag{12}$$

where $\delta_{ii'}$ is Kronecker delta function with $\delta_{ii'} = 1$ if $i = i'$ and 0 else. It can be seen that the computations of the partial derivatives of $c_{i'}$ with respect to b_i are not complex, which only need the current values of layer $\boldsymbol{c}$.

Substituting Eqs. (11), (12) into Eq. (10) we obtain

$$\frac{\partial E}{\partial b_i} = -\sum_{i'} \frac{\delta_{i'}^c}{T}(c_{i'}\delta_{ii'} - c_{i'}c_i). \tag{13}$$

From Eq. (7), we can get the derivative of b_i with respect to w_{ij}^{cc}:

$$\frac{\partial b_i}{\partial w_{ij}^{cc}} = -2(a_j - w_{ij}^{cc}). \tag{14}$$

Now Eqs. (13), (14) are substituted into Eq. (9), which gives us the derivatives of the backward error of layer $\boldsymbol{c}$ with respect to the cluster centers w_{ij}^{cc}:

$$\frac{\partial E}{\partial w_{ij}^{cc}} = \frac{2}{T}(a_j - w_{ij}^{cc})\sum_{i'} \frac{\delta_{i'}^c}{T}(c_{i'}\delta_{ii'} - c_{i'}c_i). \tag{15}$$

So far we have given the evaluation of the derivative of the backward error of layer $\boldsymbol{c}$ with respect to the weights (cluster centers) in the network. In order to update the weights, the gradient descent algorithm is applied as follows:

$$\Delta w_{ij}^{cc}(t) = \alpha \Delta w_{ij}^{cc}(t-1) - \eta_b \sum_m \frac{\partial E^m}{\partial w_{ij}^{cc}} \tag{16}$$

where α and η_b are learning parameters and the feature vector index m has been used again to distinguish the batch entries. The cluster centers are adjusted after stimulation for each input feature vector.

Supervised Codebook Learning Through Cluster Reassignment

In the previous subsection, we have presented a learning algorithm which adopted gradient descent after classical error backpropagation to the particular functional form of our NN architecture. The BoW in layer $\boldsymbol{d}$ was interpreted as a general numerical vector without any special structure.

In this subsection, we propose another algorithm that uses our prior knowledge that the information stored in layer $\boldsymbol{d}$ is a BoW, that is, a histogram. Instead of simply backprojecting an error of this layer through the softmin function which, after all, is an approximation of the required minimum function, we change it by moving input feature vectors from one histogram bin to another one.

This strategy is illustrated in Fig. 2. Supposing the error for cluster A is positive (too many feature vectors), and the error for cluster center B is negative (not enough feature

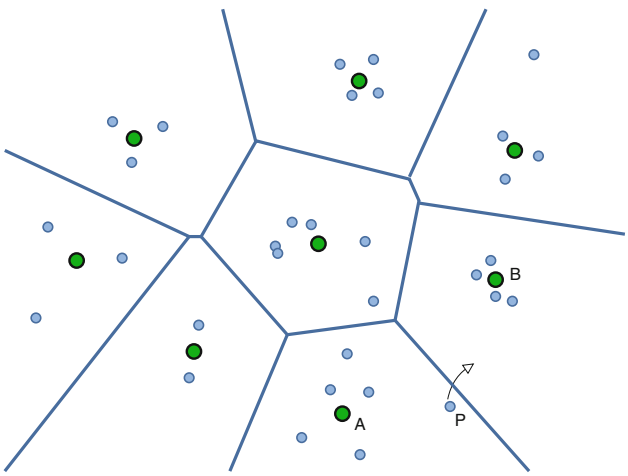

Fig. 2 Codebook learning through cluster reassignment: a Voronoi diagram of the feature space, for simplicity in 2D. Cluster centers are *green* and *large*; training points are *blue* and *small* (Color figure online)

vectors), at each weight update a single feature vector is moved from the Voronoi cell of A into the Voronoi cell of B, followed by an update of the cluster centers as a calculation of the mean of the assigned training points.

It is generally required to move several feature vectors in order to significantly change the errors of the BoW layer $\boldsymbol{d}$, denoted as $\delta^d = \{\delta_0^d, \delta_1^d, \ldots, \delta_N^d\}$. A positive error of a value (histogram bin) of the BoW, for example, $\delta_i^d > 0$, indicates that at least one feature vector has been assigned to the cluster center corresponding to this bin which should been assigned to a different cluster according to the groundtruth. In the same sense, a negative error $\delta_j^d < 0$ indicates that at least one feature vector should be added to this histogram bin of the BoW. In the following, we suppose that the solution to this problem is specified as a multiset $\boldsymbol{x} = \{x_1, x_2, \ldots, x_D\}$ of indices indicating from where a vector is moved, and a multiset $\boldsymbol{y} = \{y_1, y_2, \ldots, y_D\}$ of indices indicating to which cluster a vector is moved. For instance, $x_1 = 5$ and $y_1 = 7$ indicate that the first move will go from cluster 5 to cluster 7.

A good solution should minimize two criteria. First, the error should be low; that is, we should minimize

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left[\sum_k \delta_k^d - |\{x_s : x_s = k\}| + |\{y_s : y_s = k\}| \right] \tag{17}$$

where $|\{x_s{:}x_s = k\}|$ is the number of source indices equal to k, and the second expression can be understood in a similar way. Secondly, the feature vector movements performed by the solution should be minimal; that is, we should minimize

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left[\sum_{s=1}^{D} |b_{x_s} - b_{y_s}| \right] \tag{18}$$

One possibility would be to minimize an energy function consisting of a weighted linear combination of (17) and (18). Instead, we opted for an iterative greedy solution, where cluster pairs (i, j) are chosen decreasing (17), and then for each pair of clusters, a feature vector is chosen such that its move from cluster i to cluster j minimizes (18). We added an additional constraint requiring that the chosen feature vector to move—which is (naturally) closest to cluster center A—also be second closest to cluster center B. The details of the update algorithm are given as follows:

1. Randomly choose a pair (i, j) of histogram bins (thus of cluster centers), where the error of one bin is positive and the other is negative, i.e. $\delta_i^d > 0 \wedge \delta_j^d < 0$.
2. Calculate the Voronoi diagram of the cluster centers in feature space and determine all the feature vectors of the training set falling into the sets of $\boldsymbol{w}_i^{cc}$ and $\boldsymbol{w}_j^{cc}$, respectively.
3. Pick a single feature vector f such that:
   - it falls into the Voronoi cell i (distance between f and cluster center $\boldsymbol{w}_i^{cc}$) is minimum, i.e. nearest.
   - its distance to cluster center $\boldsymbol{w}_j^{cc}$ is second nearest.
   - if several vectors satisfy the above two criteria, choose the one minimizing the distance to the border of the two Voronoi cells, i.e. the one minimizing $|b_i - b_j|$.
4. The chosen feature vector f is reassigned from histogram bin i to histogram bin j with the following consequences:
   - the two centers $\boldsymbol{w}_i^d$ and $\boldsymbol{w}_j^{cc}$ are recalculated as the means of the feature vectors of their respective Voronoi cells.
   - the errors of the BoW layer of the corresponding bins are updated:
     $$\delta_i^{d\,[t+1]} = \delta_i^{d\,[t]} - 1$$
     $$\delta_j^{d\,[t+1]} = \delta_j^{d\,[t]} + 1$$
5. The reassignments are continued (back to step 1) until the error of layer $\boldsymbol{d}$ is zero or none of the feature vectors satisfy the above conditions.

Experimental Results

The proposed model and learning algorithms have been evaluated on the publicly available KTH action data set [11]. It is one of the largest available published data sets and contains 6 actions—boxing, hand clapping, hand waving, jogging, running and walking—performed by 25 subjects in four different scenarios such as indoors, outdoors, outdoors with scale variation and clothes changing. It contains 2391 video sequences, and each sequence lasts four seconds in average. Representative frames of the data set are shown in Fig. 3. For video representation, space-time interest points were detected by the 3D Harris-corner detector proposed by Laptev [50]. A space-time cuboid was extracted around each interest point and described by features of type histogram of gradient (HOG) and histogram of oriented flow (HOF) descriptors, which were obtained from the software supplied by Laptev [50].

As usual, a cross-validation scheme and early stopping strategy are employed to control the learning phase. The data set is divided into three independent sets: training (12 people), validation (4 people) and test (9 people), as in [11]. The MLP is trained on the training set and evaluated on the validation set for stopping to avoid over-fitting. Unless said otherwise, all errors are reported on the 863 test instances.

Fig. 3 The KTH data set [11]

Different MLP architectures and learning hyper-parameters were tested, and the best ones were chosen from the performances on the test set. Values are given below. Executions times are given for a C++ implementation running on an Intel Core i7 640 PC under Linux.

Classical Unsupervised Codebook Learning

To demonstrate the discriminative power of classical codebook and compare with our methods, we created a baseline with classical unsupervised k-means clustering and MLP learning of the BoW descriptors. With respect to Fig. 1, this corresponds to a scheme where the weights $\boldsymbol{w}^{cc}$ are set without taking into account cluster labels of the groundtruth and with supervised MLP learning of the action class decisions.

In our baseline experiments, the cluster centers were learned by k-means, and then, the MLP part was trained given the BoW models. To cope for random initialization of the MLP weights, we repeated our baseline experiments in order to obtain statistically sound results: First a codebook is created using k-means clustering. Then, for each run, the cluster centers were kept fixed and the MLP weights were randomly initialized between -0.5 and 0.5 and learned. We ran 100 runs for each codebook in our experiments. Different MLP architectures were explored with numbers of hidden units of 25, 75, 100 for 50, 150, 300 codewords. Figure 4 shows an example using 150 codewords, where learning stops at the 54th iteration.

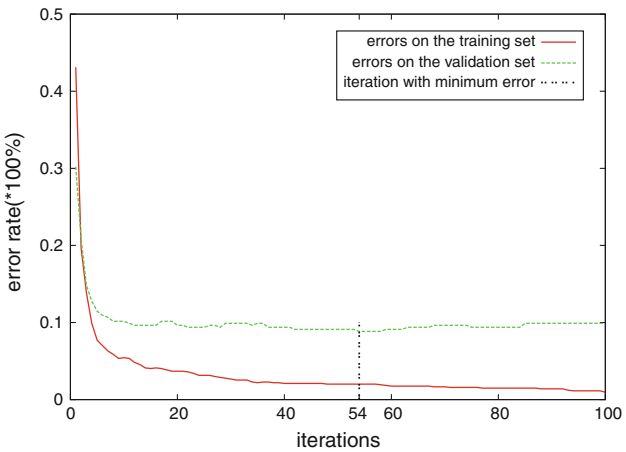

Fig. 4 A schematic illustration of the early stopping strategy during MLP learning with 150 codewords

Table 1 shows error rates (on the test set) of the learned MLP with different codebooks. A local running mean filter was applied to the results. From the table, we can see that the MLP learning is robust. On the other hand, a larger codebook is more discriminative, resulting in lower error.

Supervised Codebook Learning with Error Backpropagation

Results with supervised learning through the backpropagation method (Sect. "Supervised Codebook Learning with Error Backpropagation") are shown in Fig. 5. We repeated

Table 1 Errors in (%) on the test with classical unsupervised learning of the MLP: mean and standard deviation over 100 independent runs

Codebook size	50	150	300
Error rate	20.86 (±0.06)	18.34 (±0.02)	16.86 (±0.05)

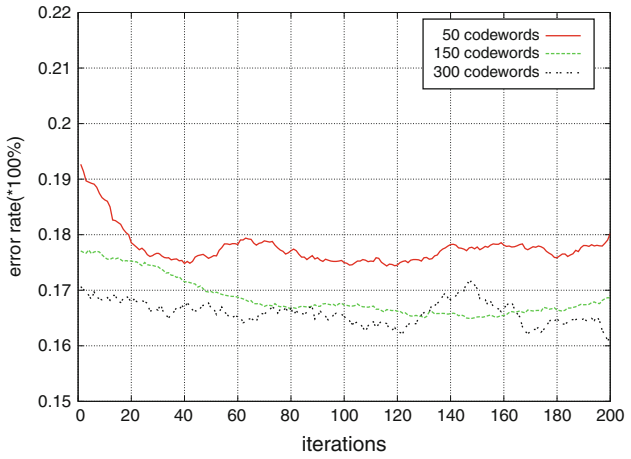

Fig. 5 Supervised learning with error backpropagation (Sect. "Supervised Codebook Learning with Error Backpropagation"): errors on the test set over different iterations

the above experiments with the same architecture, except that the cluster centers were adjusted by using a gradient descent algorithm according to the backpropagated errors of the optimal MLP in each iteration and the BoW entities of the videos were recomputed. We tried several values and selected the best parameters for gradient descent. α was set to 0.00001 for all the codebooks and η_b varied for different codebooks. The MLP was retrained, and the error rates are depicted in Fig. 5, again after applying a local running mean to smooth the data. It can be seen that the error decreases at the beginning and converges for 50 codewords and for 150 codewords. However, for 300 codewords, the error oscillates after 120 iterations due to the non-adaptive characteristics of gradient descent. Comparative results are presented in Table 2. We can see that supervised learning codebook through error backpropagation approximately gains 2.1 % for 50 codewords and 0.8 % for 300 codewords with respect to the baseline codebooks obtained with *k*-means clustering.

Supervised Codebook Learning with Cluster Reassignment

Results with supervised learning through the cluster reassignment method (Sect. "Supervised Codebook Learning Through Cluster Reassignment") are shown in Fig. 6. We again repeated the above experiments with the same neural architecture. At each iteration, the cluster centers were adjusted using the Voronoi cell updates, and then the MLP is retrained. Figure 6 shows the results, which are obtained by applying a local running mean. As we can see, the classification accuracy on the test set increases as the cluster centers are adjusted. The improvement is higher with smaller codebook, which is also observed from Fig. 5. The learned codebooks through cluster reassignment therefore improve by 1.7 % for 50 codewords and 0.8 % for 300 codewords with respect to the baseline codebooks obtained with *k*-means clustering.

From Table 2, we can see that both methods clearly improve the discriminative quality of the codebook when the codebook size is small. This is an important advantage since larger codebooks significantly increase the computational complexity of the classification algorithm due to the nearest neighbor search necessary when the feature vectors are projected on the codebook. Indexed data structures like KD-trees are not always helpful in these situations since visual data are generally embedded in feature spaces of very high dimensions—162 dimensions for the HoG/HoF features we employed in our experiments.

The performance improvement can be explained by the nature of the features it creates. The *k*-means algorithm clusters features based on the appearance of the cuboids only. When the codebook is small, the intra-cluster

Table 2 Error in (%) on the test with different methods, different classifiers and different codebook sizes: mean and standard deviation over 3 independent runs

Classifier	Codebook size	50	150	300
MLP	*k*-means	19.31 (±0.26)	17.44 (±0.31)	16.98 (±0.64)
	Error backpropagation	**17.19** (±0.51)	16.56 (±**0.07**)	**16.13** (±0.12)
	Cluster reassignment	17.57 (±**0.11**)	**16.29** (±0.53)	16.18 (±**0.07**)
	Recognition time per video (*ms*)	1.679	4.696	9.294
SVM	*k*-means	17.16 (±0.66)	15.47 (±1.06)	15.7 (±0.24)
	Error backpropagation	**15.94** (±0.25)	15.30 (±0.33)	**14.54** (±0.41)
	Cluster reassignment	16.80 (±**0.16**)	**14.89** (±**0.24**)	14.89 (±**0.24**)
	Recognition time per video (*ms*)	1.797	4.905	10.488

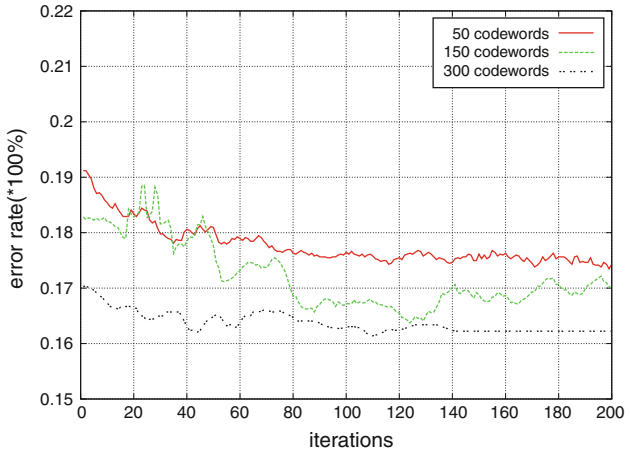

Fig. 6 Supervised learning with cluster reassignment (Sect. "Supervised Codebook Learning Through Cluster Reassignment"): errors on the test set over different iterations

variance is large, which lowers discriminative power. Our methods regroup the feature vectors into different clusters based on class labels of the groundtruth, thus choosing optimal codewords.

Retraining with SVM Classifiers

The experiments above show that the combined codebook and MLP learning outperforms the classical sequential steps k-means clustering followed by MLP learning. However, the question arises whether the improvement is due to a better codebook or to the integration of the two methods. We therefore performed an additional experiment with a two-step learning process:

1. Codebook learning according to one of the three methods (k-means; supervised codebook learning with error backpropagation; supervised codebook learning with cluster reassignment).
2. Class label retraining with SVMs on the learned codebook.

We trained an SVM with a radial basis function kernel on the training set and validation set, which were the same with the ones used in the above experiments [52]. The errors are shown in the lower block of Table 2. Classification time includes the projection of feature vectors on the codebook and the classification with MLP and SVM classifiers. The SVM outperforms the MLP by up to 2 % with different codebooks. However, the recognition of the MLP is faster due to its simple calculations, which do not need inner-products of high-dimensional vectors. As shown in Table 3, the MLP gains more benefits from computational complexity with respect to the costs from error with two codebooks when compared to the SVM. We can also see that the classification performance of our two joint supervised methods is maintained after retraining with a different classifier, indicating a real gain in discriminative power of the codebooks.

Figure 7 shows the recognition results for different learning methods on 150 codewords and after retraining with the SVM classifier. Not surprisingly, the largest error is between the classes *running* and *jogging*, which are indeed very similar in behavior. The supervised codebook learning methods can achieve significant gains on some of these classes, as the recognition rate jumps from 65 to 78 for *jogging* with error backpropagation—confirmed by a z test as described in [51].

In this paper we proposed two different joint supervised learning algorithms. The reformulated backpropagation algorithm adjusts the cluster centers directly through gradient descent algorithm. In contrast to cluster reassignment, two more parameters besides the learning rate η in MLP learning need to be set: the momentum coefficient α and the learning rate η_b. The drawback of a gradient descent algorithm applied to a nonlinear system is well-known: it is difficult to learn a set of optimal parameters, the algorithms mostly converge to local minima and sometimes even diverge. As shown in Fig. 5, the error with 300 codewords began to converge after 60 iterations, but it begin to diverge from 120 iterations.

In comparison, the cluster reassignment algorithm adjusts the cluster centers indirectly by rearranging the cluster labels for all the feature vectors. It does not need any more learning parameters except η in MLP learning and is easier to control, but needs more iterations to converge, fortunately often to a better solution. From Fig. 6 we can see that the errors converge after 100 iterations. This can also be observed for the errors on 300 codewords—it becomes constant after 140 iterations compared with the errors on 300 codewords in Fig. 5.

Table 3 Cost–benefit table (in %) of the MLP compared to the SVM with the results of cluster reassignment method

Codebook size	50		150		300	
Classifier	SVM	MLP	SVM	MLP	SVM	MLP
Error	100	107	100	109	100	105
Time	100	88	100	96	100	93

	Box	Clap	Wave	Jog	Run	Walk
Box	**95**	0	0	0	0	5
Clap	12	**88**	0	0	0	0
Wave	2	7	**91**	0	0	0
Jog	0	0	0	**65**	26	9
Run	0	0	0	23	**73**	4
Walk	0	0	0	7	2	**91**

(a)

	Box	Clap	Wave	Jog	Run	Walk
Box	**94**	0	0	1	0	5
Clap	10	**90**	0	0	0	0
Wave	3	7	**90**	0	0	0
Jog	0	0	0	**78**	14	8
Run	0	0	0	30	**66**	4
Walk	1	0	0	5	1	**93**

(b)

	Box	Clap	Wave	Jog	Run	Walk
Box	**98**	0	0	0	0	2
Clap	13	**87**	0	0	0	0
Wave	2	6	**92**	0	0	0
Jog	0	0	1	**72**	19	8
Run	0	0	0	24	**72**	4
Walk	0	0	1	6	2	**91**

(c)

Fig. 7 Confusion matrix for a codebook with 150 codewords according to different learning algorithms and after retraining with SVMs. The codebook has been learned with **a** *k*-means, **b** error backpropagation (Sect. "Supervised Codebook Learning with Error Backpropagation"), **c** cluster reassignment (Sect. "Supervised Codebook Learning Through Cluster Reassignment")

Conclusion

In this paper, we proposed a joint supervised codebook learning and optimization framework, which can be applied to any method based on bag-of-words models such as object class recognition or human action recognition. The codebook learning process and the recognition phase are integrated into a uniform framework. The codebook therefore is created in a goal-directed way and is more discriminative than classical ones. We have presented two algorithms to update the cluster centers (codewords) through the back-propagated errors: one is based on classical error back-propagation, in which the codewords are adjusted using a gradient descent algorithm. The other is based on cluster reassignments, in which we reassign the cluster labels for all the feature vectors based on the errors. Our framework has been tested on the public KTH action data set, and we have obtained very promising and close results for both methods. At the same time, they demonstrated that error backpropagation learned the optimal codebook faster than cluster reassignment. However, it may suffer more from over-fitting, while cluster reassignment is easier to control. The experiments on the KTH human action data set have confirmed that our framework is able to optimize the codebooks and that it makes them more discriminative. It is also able to increase the speed of a method by decreasing the codebook size while keeping its discriminative power.

References

1. Csurka G, Dance C, Fan LX, Willamowski J, Bray C. Visual categorization with bags of keypoints. In: Proceedings of ECCV international workshop on statistical learning in computer vision. 2004. p. 1–22.
2. Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vis (IJCV). 2004;60(2):91–110.
3. Dollar P, Rabaud V, Cottrell G, Belongie S. Behavior recognition via sparse spatio-temporal features. In: ICCV workshop on visual surveillance and performance evaluation of tracking and surveillance. 2005. p. 65–72.
4. Moosmann F, Triggs B, Jurie F. Fast discriminative visual codebooks using randomized clustering forests. In: NIPS. 2007. p. 985–92.
5. Liu J, Shah M. Learning human actions via information maximization. In: CVPR. 2008. p. 1–8.
6. Liu J, Yang Y, Shah M. Learning semantic visual vocabularies using diffusion distance. In: CVPR. 2009. p. 461-68.
7. Saghafi B, Farahzadeh E, Rajan D, Sluzek A. Embedding visual words into concept space for action and scene recognition. In: BMVC. 2010. p. 1–11.
8. Niebles JC, Wang H, Fei-Fei L. Unsupervised learning of human action categories using spatial-temporal words. Int J Comput Vis (IJCV). 2008;79(3):299–318.
9. Gilbert A, Illingworth J, Bowden R. Action recognition using mined hierarchical compound features. IEEE Trans Pattern Anal Mach Intell (PAMI). 2011;33(5):883–97.
10. Laptev I. On space-time interest points. Int J Comput Vis (IJCV). 2005;64(2/3):107–23.
11. Schuldt C, Laptev I, Caputo B. Recognizing human actions: a local svm approach. In: ICPR. 2004. p. 32–6.
12. Oikonomopoulos A, Patras I, Pantic M. An implicit spatiotemporal shape model for human activity localization and recognition. In: CVPR. 2009. p. 27–33.
13. Ryoo MS, Aggarwal JK. Spatio-temporal relationship match: video structure comparison for recognition of complex human activities. In: ICCV. 2009. p. 1593–600.
14. Ta A-P, Wolf C, Lavoué G, Baskurt A, Jolion JM. Pairwise features for human action recognition. In: ICPR. 2010. p. 3224–7.
15. Mikolajczyk K, Uemura H. Action recognition with appearance-emotion features and fast search trees. Comput Vis Image Underst (CVIU). 2011;115(3):426–38.
16. Aggarwal JK, Ryoo MS. Human activity analysis: a review. ACM Comput Surv (inpress).
17. Turaga P, Chellappa R, Subrahmanian VS, Udrea O. Machine recognition of human activities: a survey. IEEE Trans Circuits Syst Video Technol. 2008;18(11):1473–88.
18. Weinland D, Ronfard R, Boyer E. A survey of vision-based methods for action representation, segmentation and recognition. Comput Vis Image Underst (CVIU). 2011;115:224–41.
19. Song Y, Concalves L, Perona P. Unsupervised learning of human motion. IEEE Trans Pattern Anal Mach Intell (PAMI). 2003;25(7):814–27.
20. Gorelick L, Blank M, Shechtman E, Irani M, Basri R. Actions as space-time shapes. IEEE Trans Pattern Anal Mach Intell (PAMI). 2007;29(12):2247–53.
21. Wang L, Geng X, Leckie C, Ramamohanarao K. Moving shape dynamics: a signal processing perspective. In: CVPR. 2008. p. 1–8.
22. Weinland D, Boyer E, Ronfard R. Action recognition from arbitrary views using 3D exemplars. In: ICCV. 2007. p. 1–7.
23. Ke Y, Sukthankar R, Hebert M. Efficient visual event detection using volumetric features. In: ICCV. 2005. p. 166–73.

24. Mikolajczyk K, Uemura H. Action recognition with motion-appearance vocabulary forest. In: CVPR. 2008. p. 1–8.
25. Zhang Z, Hu Y, Chan S, Chia LT. Motion context: A new representation for human action recognition. In: ECCV. 2008.
26. Bregonzio M, Gong SG, Xiang T. Recognising action as clouds of space-time interest points. In: CVPR. 2009. p. 1948–55.
27. Liu J, Ali S, Shah M. Recognizing human actions using multiple features. In: CVPR. 2008. p. 1–8.
28. Sun X, Chen M, Hauptmann A. Action recognition via local descriptors and holistic features. In: CVPR workshop on human communicative behavior analysis. 2009. p. 58–65.
29. Seo HJ, Milanfar P. Action recognition from one example. IEEE Trans Pattern Anal Mach Intell (PAMI). 2011;33(5):867–82.
30. Shechtman E, Irani M. Space-time behavior based correlation. In: CVPR. 2005. p. 405–12.
31. Ta A-P, Wolf C, Lavoué G, Baskurt A. Recognizing and localizing individual activities through graph matching. In: International conference on advanced video and signal-based surveillance. 2010.
32. Abdelkader MF, Almageed WA, Srivastava A, Chellappa R. Silhouette-based gesture and action recognition via modeling trajectories on riemannian shape manifolds. Comput Vis Image Underst (CVIU). 2011;115(3):439–55.
33. Boiman O, Irani M. Detecting irregularities in images and in video. Int J Comput Vis (IJCV). 2007;74(1):17–31.
34. Cuntoor NP, Yegnanarayana B, Chellappa R. Activity modeling using event probability sequences. IEEE Trans Image Process. 2008;17(4):594–07.
35. Shi Q, Cheng L, Wang L, Smola A. Human action segmentation and recognition using discriminative semi-Markov models. Int J Comput Vis (IJCV). 2010;93(1):22–32.
36. Xiang T, Gong S. Activity based surveillance video content modelling. Pattern Recogn. 2008;41(7):2309–26.
37. Xiang T, Gong S. Incremental and adaptive abnormal behaviour detection. Comput Vis Image Underst (CVIU). 2008;11(1):59–73.
38. Zhang D, Perez DG, Bengio S, McCowan I. Semi-supervised adapted hmms for unusual event detection. In: CVPR. 2005. p. 611–8.
39. Zhou H, Kimber D. Unusual event detection via multi-camera video mining. In: ICPR. 2006. p. 1161–6.
40. Jhuang H, Serre T, Wolf L, Poggio T. A biologically inspired system for action recognition. In: ICCV. 2007. p. 1–8.
41. Taylor GW, Fergus R, Lecun Y, Bregler C. Convolutional learning of spatio-temporal features. In: ECCV. 2010.
42. Baccouche M, Mamalet F, Wolf C, Garcia C, Baskurt A. Sequential deep learning for human action recognition. In: International workshop on human behavior understanding: inducing behavioral change, 2011.
43. Fathi A, Mori G. Action recognition by learning mid-level motion features. In: CVPR. 2008. p. 1–8.
44. Dyana A, Das S. Trajectory representation using gabor features for motion-based video retrieval. Pattern Recogn Lett. 2009;30(10):877–92.
45. Stauffer C, Grimson WEL. Learning patterns of activity using real-time tracking. IEEE Trans Pattern Anal Mach Intell (PAMI). 2000;22(8):747–57.
46. Ryoo MS, Aggarwal JK. Stochastic representation and recognition of high-Level group activities. Int J Comput Vis (IJCV). 2010;93(2):183–200.
47. Wang L, Wang Y, Gao W. Mining layered grammar rules for action recognition. Int J Comput Vis (IJCV). 2011;93(2):162–82.
48. Niebles JC, Fei-Fei L. A hierarchical model of shape and appearance for human action classification. In: CVPR. 2007. p. 1–8.
49. Bishop CM. Neural networks for pattern recognition. Oxford: Oxford university press; 1994. p. 140–45
50. Laptev I, Marszalek M, Schmid C, Rozenfeld B. Learning realistic human actions from movies. In: CVPR. 2008. p. 1–8.
51. Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. 1998;10(7):1895–924.
52. Chang C-C, Lin C-J. LIBSVM a library for support vector machines. ACM Trans Intell Syst Technol. 2011;2(3):27:1–27:27.