Image Classification by Histogram Features Created with Learning Vector Quantization

Conference Paper · September 2008

DOI: 10.1007/978-3-540-87536-9_85 · Source: DBLP

2 authors:

Marcin Blachnik
Silesian University of Technology
60 PUBLICATIONS **448** CITATIONS
SEE PROFILE

Jorma Tapio Laaksonen
Aalto University
251 PUBLICATIONS **6,448** CITATIONS
SEE PROFILE

Proceedings of International Conference on Artificial Neural Networks - ICANN 2008
Volume 5163 of the series Lecture Notes in Computer Science pp 827-836
http://link.springer.com/chapter/10.1007/978-3-540-87536-9_85

Image classification by histogram features created with Learning Vector Quantization

Marcin Blachnik[1,2] and Jorma Laaksonen[2]

[1] Silesian University of Technology, Electrotechnology Department, Katowice Krasiskiego st. 8, Poland
[2] Helsinki University of Technology, Adaptive Informatics Research Centre, P.O. Box 5400, FI-02015 TKK, Finland.
marcin.blachnik@polsl.pl jorma.laaksonen@tkk.fi

Abstract. Histogram-based data analysis is one of the most popular solutions for many problems related to image processing such as object recognition and classification. In general a histogram preserves more information from the first-order statistics of the original data than simple averaging of the raw data values. In the simplest case, a histogram model can be specified for a specific image feature type independently of any real image content. In the opposite extreme, the histograms can be made dependent not only the actual image contents, but also on the known semantic classes of the images. In this paper, we propose to use the Learning Vector Quantization (LVQ) algorithm in fine-tuning the codebook vectors for more efficient histogram creation. The performed experiments show that the accuracy of the Interest Point Local Descriptors (IPLD) feature for image classification can be improved by the proposed technique.

1 Introduction

In image analysis many different problems can be identified, but most of them are related to one of two categories: classification where the goal is prediction of presence or absence of a certain class (object) in the test image; and detection where the goal is prediction and selection of the bounding box of a certain object. In this paper the first problem will be considered based on histogram features created from low-level visual descriptor vectors extracted from interest points detected in the images.

Histogram-based image descriptors have proofed to be one of the most useful methods for image representation in classification tasks. They allow high compression of an image represented as single vector with relatively small set of attributes. Histogram representation can be used for many different types of features extracted from images. Some of the most popular types are color histograms [1], and SIFT [2] or IPLD [3] feature histograms. Histogram and interest point based features allowed winning the PASCAL NoE[3] Visual Object Classes Challenge 2006[4].

[3] http://www.pascal-network.org
[4] http://www.pascal-network.org/challenges/VOC/voc2006/

There exist many ways of creating various types of histograms from simple 1-D to spatial. However, the basic approach is common and based on clustering features extracted from images and selecting special *reference vectors* also called *prototypes* or *codebook vectors*. These prototypes will then act as the cluster centers of the histogram bins in the high-dimensional feature space. The image-independent, fixed prototypes will then be further utilized for creating a content-describing histogram for each image to be classified. The number of low-level vector samples assigned to each prototypes in the nearest neighbor mapping is used as a new vector describing single image. Each histogram can then be normalized by the number of samples originating from the image, leading to an estimate of a discrete probability distribution.

In the simplest case, a histogram model can be specified for a specific image feature type independently of any real image content. A more evolved technique is to calculate the histogram prototypes from an existing image collection. In that case, one might not have any information available concerning the categories of objects in the images. If such category information is available, the creation of histograms can be based on clustering raw image features extracted from each object category independently. In such a case, however, no knowledge is gained from the class distribution, which may lead to low quality of final classification task of images set.

One of the methods that can utilize statistical inter-class knowledge is the Learning Vector Quantization (LVQ) algorithm [4]. The LVQ algorithm optimizes the positions of the prototypes (codebook vectors) according to the distances between samples from different classes, leading to improved discriminative abilities. We assume that this better discrimination will benefit also the process of histogram creation. The goal of this paper is to present improvements in creating histograms that can be achieved by applying a supervised codebook optimization algorithm – such as the LVQ algorithm – on features extracted from interest points of the images.

The rest of the paper is organized as follows: Section 2 first discusses the general question of discriminative prototype selection. Our approach for it is then described in detail, briefly introducing the Fuzzy C-Means clustering and the LVQ algorithm. The next Section 3 describes our data and experiments, evaluating SVM classifier on IPLD feature histograms, and presents a comparison of results obtained with and without LVQ tuning of the histogram codebooks. The last Section 4 concludes the paper and discusses the obtained results, pointing out possible future research directions.

2 Selecting prototype positions

Statistical pattern recognition algorithms are typically based on finding similarities and dissimilarities between training samples and using this knowledge for further reasoning. In image analysis this is equivalent to comparing representations of images with different depicted objects in order to evaluate similarities between them. This process, however, can be very complicated, especially when one considers the huge number of pixels of a single image to be analyzed.

A commonly-used solution for the image representation problem is to extract low-level visual features from the image – or some special interest points in it – and to either average or create a histogram of those features. For preparing the histograms, some computationally simple methods – such as k-means clustering – are used to first create the codebook of prototypes and then to extract the actual histogram feature vectors. Each single image is then analyzed according to the low-dimensional histogram vectors with more complex classification methods like the Support Vector Machine (SVM).

In classification tasks, histograms are usually created in one of two principal ways. First, they can be obtained by clustering the whole dataset; second, they can be obtained clustering each semantic object class independently. Similar concept is well known also in similarity-based methods of learning classification problems, where these two approaches are also called *post-supervised* and *pre-supervised*, respectively [5].

Simple clustering algorithms are often used for selecting prototypes (codebook vectors) which are further used to create histograms of the images. However, clustering does not produce any knowledge of the inter-class distribution of the classes. This may lead to poor quality of obtained histograms, and suboptimal classification results. The same problem appears in classical classification problems, as reported eg. by Blachnik et al. [6] and Kuncheva [7]. Moreover, while creating histograms the goal is to find the least possible number of prototypes because too high a number can decrease the quality and generalization ability of the histograms. This appears when just a few samples from each image are used per one prototype.

The aim in selecting good prototypes is to look for prototypes that assure maximum discrimination abilities. In traditional nearest-neighbor classification the same goal has been sought for by using the Learning Vector Quantization (LVQ) algorithm [4]. The LVQ optimization tends to place prototypes of one class (say positive) inside the centers of homogeneous clusters of that class, far from representatives of the second (negative) class, and vice versa. After the LVQ training, the obtained prototypes represent the nearest-neighbor decision borders between the classes, providing good discrimination abilities.

In the LVQ training the main problem is the proper initialization of the codebook vectors. However, this can be solved sufficiently by using prototypes obtained after clustering each class separately. In the presented approach for LVQ-based histograms, Fuzzy C-Means algorithm was used to find initial position of the codebook vectors, and these prototypes were then optimized by the LVQ training.

2.1 Fuzzy C-Means clustering

One examples of modern clustering methods is the Fuzzy C-Means algorithm (FCM) [8]. It is an extension of the classical k-means clustering algorithm. Instead of binary $\{0, 1\}$ membership of a vector $\mathbf{x}$ to the cluster represented by reference vector $\mathbf{p}_i$, such membership is in FCM a continuous function in bounds $[0, 1]$.

FCM belongs to batch clustering methods, where prototypes are iteratively updated after processing of the whole dataset. Batch clustering methods require a special partition matrix $\mathbf{U}$ of size $N \times k$, where N is the number of vectors and k is the number of clusters. This matrix of elements u_{ji} represent membership value of assigning vector $\mathbf{x}_j$ to cluster i defined by a prototype $\mathbf{p}_i$. FCM is based on the minimization of a cost function defined as:

$$J_m(\mathbf{U}, \mathbf{P}) = \sum_{i=1}^{k} \sum_{j=1}^{N} (u_{ji})^m \left\| \mathbf{x}_j - \mathbf{p}_i \right\|^2 \tag{1}$$

where $m > 1$ is a fuzziness parameter, typically $m = 2$, and $\mathbf{P}$ is a matrix whose columns are the prototype vectors $\mathbf{p}_i$.

For obtaining good clustering results, the partition matrix $\mathbf{U}$ has to fulfill three conditions:
1^o each vector $\mathbf{x}_j$ belongs to the i-th cluster to a degree between $[0, 1]$:

$$\underset{1 \le i \le k}{\forall} \; \underset{1 \le j \le N}{\forall} u_{ji} \in [0, 1] \tag{2}$$

2^o the sum of membership values of j-th vector $\mathbf{x}_j$ in all clusters is equal to 1

$$\underset{1 \le j \le N}{\forall} \sum_{i=1}^{k} u_{ji} = 1 \tag{3}$$

3^o no clusters are empty.

$$\underset{1 \le i \le k}{\forall} 0 < \sum_{j=1}^{N} u_{ji} < N \tag{4}$$

Cost function (1) is minimized under these conditions by [8]:

$$\underset{1 \le i \le k}{\forall} \mathbf{p}_i = \sum_{j=1}^{N} (u_{ji})^m \mathbf{x}_j \Bigg/ \sum_{j=1}^{N} (u_{ji})^m \tag{5}$$

$$\underset{\substack{1 \le i \le k \\ 1 \le j \le N}}{\forall} u_{ji} = \left(\sum_{l=1}^{k} \left(\frac{\left\| \mathbf{x}_j - \mathbf{p}_i \right\|}{\left\| \mathbf{x}_j - \mathbf{p}_l \right\|} \right)^{2/(m-1)} \right)^{-1} \tag{6}$$

The above-described clustering method applied to classification problems does not optimize position of the codebook vectors according to inter-class distributions. This drawback can be improved by prototype optimization algorithms used in classification problems.

2.2 LVQ training

The Learning Vector Quantization (LVQ) is a family of neurally-motivated algorithms which during the training phase optimize the positions of codebook vectors to the maximize classification accuracy. This type of training can be seen as a special form of the k-NN or rather nearest prototype classifier (NPC) [9]. The family of LVQ networks consist of many algorithms, where the most basic one is called LVQ1 and based on iterative optimization of the position of the prototype vectors.

Generally there exist two approaches to the training phase: batch and online learning. In the first approach the positions of the codebook vectors are updated after presentation of the whole training set, while in the online mode the codebook vectors $\mathbf{p}_i$ are updated after presentation of each training instance using the formula:

$$\begin{array}{l} \mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \alpha(t)\left(\mathbf{x}(t) - \mathbf{p}_i(t)\right),\ \text{if class}(\mathbf{p}_i) = \text{class}(\mathbf{x}(t)) \\ \mathbf{p}_i(t+1) = \mathbf{p}_i(t) - \alpha(t)\left(\mathbf{x}(t) - \mathbf{p}_i(t)\right),\ \text{if class}(\mathbf{p}_i) \neq \text{class}(\mathbf{x}(t)) \end{array} \tag{7}$$

$$i = \underset{a=1,\ldots,l}{\operatorname{argmin}} \|\mathbf{x}(t) - \mathbf{p}_a\| \tag{8}$$

where t is the current iteration step, i denotes the index of the prototype which is the closest one to the current training vector $\mathbf{x}(t)$ according to formula (8), and $\alpha(t)$ is the learning rate, a monotonically decreasing function which takes values between $[0, 1]$.

Self-performed experiments and other researchers experience proofs that the second approach – online learning – is more stable and less sensitive to outliers. It also imposes smaller memory requirements.

3 Experiments and results

Datasets used in this experiments were taken from the Visual Object Classes Challenge 2007 (VOC2007) dataset [10]. It includes 2493 training images and 2500 images used for validation. In each image, may appear one or more of $n = 20$ object classes, such as *person, train, cat, dog, cow, bicycle*, etc.

Instead of average precision (AP), the measure suggested by the authors of the challenge, we decided to use receiver operator characteristic (ROC) curves and especially the area under curve (AUC) measure to determine the accuracy of classification. The switch between these accuracy measures was motivated by the problem that the AP measure depends on the number of intervals used to determine the value, and changing the number of intervals may affect the order of otherwise similar results.

The Support Vector Machine (SVM) was used as the final classifier with SVM-light implementation of T. Joachims [11]. The SVM used in the experiments had Gaussian kernels, and the SVM parameters such as C and γ (the kernel width) were optimized by using the greedy search procedure.

3.1 IPLD features

The Interest Point Local Descriptors (IPLD) [3] feature extraction technique first detects a set of interest points in an image and then calculates features describing the local neighborhoods of those points. The interest points have in our experiments been the union of points with strongest responses to the Harris–Laplace corner detector and the difference of Gaussians (DoG) keypoint filtering. The typical number of interest points found in each image has been around one thousand.

The features extracted around the interest points are based on the SIFT [2] descriptors. They are comprised of edge orientation histograms with eight directions. There are a total of 16 histograms extracted around the interest point in a 4×4 grid. The total dimensionality of the feature vectors is thus 128.

3.2 Training strategy

The process we used to create a classifier can be divided into several steps:

1. IPLD interest point detection and feature extraction from images
2. randomly sampling 300 interest points from each image
3. grouping feature vectors from images of the *studied class*, and those from the remaining classes as the *background class*
4. dataset normalization, so each variable has 0 mean and variance equal to 1.
5. clustering the studied class and the background class independently into 150 clusters per class with the Fuzzy C-Means algorithm
6. prototype optimization using labeled data and the LVQ algorithm
7. creating the histogram feature vector for each image
8. training and optimizing SVM classifier based on the obtained histogram features

First of all each image of the VOC2007 database was processed and the IPLD interest points and features were calculated from it. The mean number of interest points per image was around 1000, and from that number 300 interest points were randomly selected for further analysis.

In the classical approach used for selecting codebook vectors, each class is after sampling clustered independently. This requires keeping $300 \times n_i$ vectors in memory, where n_i is the number of images in the i-th class. On the average each class of the VOC2007 dataset had around 140 images, except the *person* class which had 1025 images.

The number of samples is especially important for the LVQ training which requires labeled data and combination of samples from different classes into the training set. This may create a problem with the dataset size of $(\sum n_i) \cdot 300$ samples. The class-wise dataset combination can be done either as *one against all* strategy or *one against one*. In the first option, all vectors of all the other classes except the studied class, labeled as $C_1 = 1$, are grouped together and labeled as $C_2 = -1$. In the one against one strategy, $n-1$ datasets are required to create a codebook for single class because prototypes are generated to discriminate pairs of classes. From these two approaches, the one against all strategy was selected here because it has linear complexity to the number of classes. Moreover, to avoid imbalanced representation of vectors in the labeled LVQ training set, the data was resampled to obtain an equal number of samples in the studied and background classes. In the same time the memory complexity was also reduced.

The training vectors in both the studied and background classes were clustered independently by using the Fuzzy C-Means algorithm to obtain 150 prototypes per class. The prototypes were then further optimized by using the LVQ1 training on the training data. Histograms were then calculated for all the images in both the training and validation data sets. The histograms were normalized be the number of samples in each image to produce proper discrete probability estimates. The prepared histogram features from the training set images were used for training the SVM classifier and those from the validation set images were used for evaluating its accuracy.

3.3 Classification results

In order to evaluate the performance of the proposed approach, three classes, *dog*, *cat* and *person*, were selected from the dataset. For each of these three classes we ran a separate experiment and built a classifier by following the steps described in the previous section.

Accuracy measures such as the ROC curve and its AUC value were evaluated by using the validation image set of the challenge. The obtained ROC curves are presented in Figures 1, 2 and 3 for each class, *dog*, *cat* and *person*, respectively. One can clearly see that the ROC curves of the LVQ-optimized histogram features are always better than those obtained without LVQ optimization.

The AUC values and the SVM accuracies obtained during the training with five-fold cross-validation are presented in Table 1. The “FCM” results have been been obtained with the Fuzzy C-Means clustering alone, whereas the results in the “FCM+LVQ” row have included LVQ1 training after the class-wise clustering.

The presented results show clear improvement from the classical approach based on only independent class clustering. The positions of the prototypes optimized by the LVQ1 algorithm allow obtaining more discriminative histograms. Therefore, the difference in accuracy between just clustering and clustering followed by LVQ is significant. This phenomenon is visible both for the final validation results in the “AUC” columns of Table 1 and for the cross-validation results during the training.

	dog		*cat*		*person*	
	AUC	5xCV	AUC	5xCV	AUC	5xCV
FCM	0.6980	0.6837	0.7023	0.7150	0.6913	0.7080
FCM+LVQ	0.7499	0.8457	0.8583	0.9179	0.7654	0.8010

Table 1. AUC value on validation data and after 5xCV training of the SVM classifier

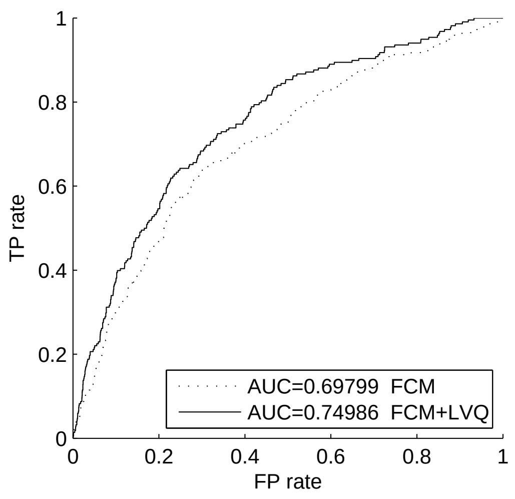

Fig. 1. ROC curve for the *dog* class obtained by using FCM clustering alone and using FCM+LVQ optimization

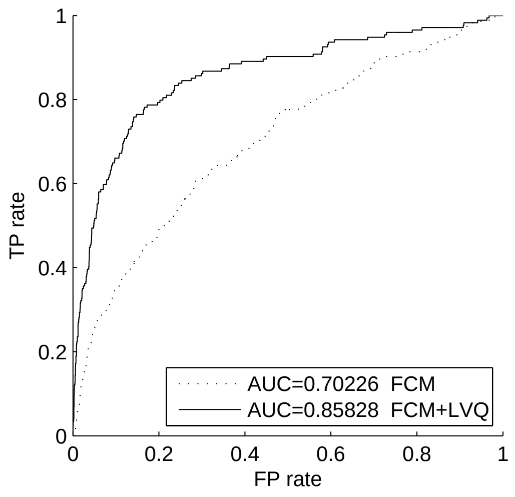

Fig. 2. ROC curve for the *cat* class obtained by using FCM clustering alone and using FCM+LVQ optimization

4 Conclusions and future directions

The proposed approach for selecting prototypes was in this paper applied only for IPLD histograms of images while it can be used as well for other types of visual

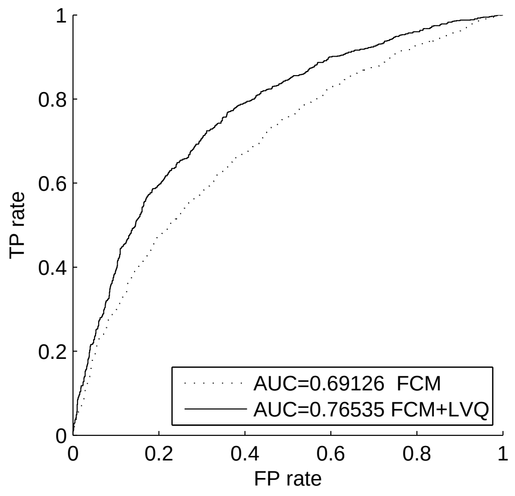

Fig. 3. ROC curve for the *person* class obtained by using FCM clustering alone and using FCM+LVQ optimization

features and other totally different data domains. However, the applicability of the technique will need further investigation.

An important issue that should be solved is the optimization of the number of prototypes selected for each class. Our current approach was based on a manually selected value. However, selecting an appropriate number of codebook vectors may further improve the quality of the histograms and enhance their discrimination abilities. The codebook size selection problem might perhaps be solved by using the Race algorithm proposed by Blachnik in [6] or by Dynamic LVQ solution developed by Bermejo in [12]. Instead of the initial optimization of the number of prototypes, feature selection algorithms could also be used during the SVM training for pruning any useless codebook vectors.

One more possible improvement that might lead to achieving good prototype sets is based on the utilization of the Relevance Vector Machine (RVM) [13]. The RVM belongs to the group of sparse Bayesian methods and tends to automatic selection of a small number of prototypes that lie far from the decision borders. This property might be desired for codebooks used for creating histograms.

All in all, we have shown in this paper that supervised tuning of histogram prototypes is a promising approach for creating efficient features for classification. The LVQ algorithm is a simple and well-known method for this purpose, but not the only one. Future research will include further investigations concerning both the unsupervised selection of the initial prototypes and their supervised tuning.

References

1. ISO/IEC: Information technology - Multimedia content description interface - Part 3: Visual (2002) 15938-3:2002(E).
2. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (2004) 91–110
3. Dorko, G., Schmid, C.: Selection of scale-invariant parts for object class recognition. In: Proceedings of Ninth IEEE International Conference on Computer Vision. Volume 01., Los Alamitos, CA, USA, IEEE Computer Society (2003) 634
4. Kohonen, T.: Self-Organizing Maps. Springer Verlag (2001)
5. Kuncheva, L., Bezdek, J.: Presupervised and postsupervised prototype classifier design. IEEE Transactions on Neural Networks **10**(5) (1999) 1142–1152
6. Blachnik, M., Duch, W., Wieczorek, T.: Selection of prototypes rules context searching via clustering. Lecture Notes in Artificial Intelligence **4029** (2006) 573–582
7. Kuncheva, L., Bezdek, J.: Nearest prototype classification: Clustering, genetic algorithms or random search? IEEE Transactions on Systems, Man, and Cybernetics **C28**(1) (1998) 160–164
8. Hoppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley (1999)
9. Kuncheva, L., Bezdek, J.: An integrated framework for generalized nearest prototype classifier design. International Journal of Uncertainty **6**(5) (1998) 437–457
10. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html (2007)
11. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Kluwer Academic Publisher (2002)
12. Bermejo, S.: Learning with nearest neighbor classifiers. PhD thesis, Technical University of Catalonia (2000)
13. Tipping, M.: The relevance vector machine. In: Advances in Neural Information Processing Systems, Morgan Kaufmann (2000)