Object Detection by Simple Fuzzy Classifiers Generated by Boosting

Marcin Gabryel, Marcin Korytkowski, Rafał Scherer, and Leszek Rutkowski

Institute of Computational Intelligence, Częstochowa University of Technology
al. Armii Krajowej 36, 42-200 Częstochowa, Poland
{marcin.gabryel,marcin.korytkowski,rafal.scherer,
leszek.rutkowski}@iisi.pcz.pl
http://iisi.pcz.pl

Abstract. Finding key points based on SURF and SIFT and size of their vector reduction is a classical approach for object recognition systems. In this paper we present a new framework for object recognition based on generating simple fuzzy classifiers using key points and boosting meta learning to distinguish between one known class and other classes. We tested proposed approach on a known image dataset.

1 Introduction

The problem of image recognition on the basis of its content is a one of the most important computer science challenges [12][13][14]. Finding keypoints based on SURF [2] and SIFT [9] and keypoint vectors dimensionality reduction is the classical approach for object recognition systems. The main contribution of this paper is to find the set of fuzzy rules which are representative for some class of objects. Fuzzy systems are very efficient method for describing partial membership to a set [3][4][5][11][16]. This approach could be very useful for the search based on the image content in a set of complex graphical objects in a database. In addition, creating an optimal set of indexes could accelerate this process. The general scheme of our approach is as follows:

- Determining key points for a set of images, the content of which belong to the same class (e.g. airplanes) using SURF or SIFT algorithms (positive examples),
- Determining key points for different classes using SURF or SIFT algorithms (negative examples),
- Dimensionality reduction using principal component analysis (PCA) algorithm by using both the positive and negative samples (vector size of 128 numbers is reduced to 36),
- Design of a fuzzy classifier based on the AdaBoost algorithm and assigning weights to each of its rules.

The main idea of this paper is suggested in papers [17] and [20] where the authors changed slightly the basis of Adaboost algorithm. They use the whole set of training examples to build many classifiers, then they choose the best model according to the error value. We are going back to the original AdaBoost. In each step we randomly choose one positive example according to their weights. This example is a base to build final classifiers. We also introduced other changes to the original concept where the authors searched for the most important samples. We are going to find the most important classes of descriptors representing objects of certain type. In our case, all classifiers are neuro-fuzzy models.

The paper is organized as follows. In the first section we present basis of algorithms which are used to build the proposed solution. It consists of algorithms to generate keypoints (SURF, SIFT etc.), algorithms for size of keypoints reduction (PCA), algorithms for modular systems building (AdaBoost) and fuzzy logic. Next section describes the proposed method and Section 3 provides simulation results.

2 Methods Used in Proposed Approach

2.1 Boosting

This section describes the AdaBoost algorithm which is the most popular boosting method [15]. The algorithm described here is designed for binary classification. Let us denote the l-th learning vector by $\mathbf{z}^l = [x_1^l, ..., x_n^l, y^l]$, $l = 1...m$ is the number of a vector in the learning sequence, n is the dimension of input vector $\mathbf{x}^l$, and y^l is the learning class label. Weights D^l assigned to learning vectors, have to fulfill the following conditions

$$\begin{aligned} &\text{(i) } 0 < D^l < 1 \ , \\ &\text{(ii) } \sum_{l=1}^{m} D^l = 1 \ . \end{aligned} \tag{1}$$

The weight D^l is the information how well classifiers were learned in consecutive steps of an algorithm for a given input vector x^l. Vector $\mathbf{D}$ for all input vectors is initialized according to the following equation

$$D_t^l = \frac{1}{m}, \quad \text{for } t = 0, ..., T \ , \tag{2}$$

where t is the number of a boosting iteration (and a number of a classifier in the ensemble). Let $\{h_t(\mathbf{x}) : t = 1, ..., T\}$ denotes a set of hypotheses obtained in consecutive steps t of the algorithm being described. For simplicity we limit our problem to a binary classification (dichotomy) i.e. $y \in \{-1, 1\}$ or $h_t(\mathbf{x}) = \pm 1$. Similarly to learning vectors weights, we assign a weight c_t for every hypothesis, such that

$$\begin{aligned} &\text{(i) } \sum_{t=1}^{T} c_t = 1 \ , \\ &\text{(ii) } c_t > 0 \ . \end{aligned} \tag{3}$$

Now in the AdaBoost algorithm we repeat steps 1-4 for $t = 1, \ldots, T$:

1. Create hypothesis h_t and train it with a data set with respect to a distribution d_t for input vectors.
2. Compute the classification error ε_t of a trained classifier h_t according to the formula

$$\varepsilon_t = \sum_{l=1}^{m} D_t^l(z^l) I(h_t(\mathbf{x}^l) \neq y^l) , \tag{4}$$

where I is the indicator function

$$I(a \neq b) = \begin{cases} 1 \text{ if } a \neq b \\ 0 \text{ if } a = b \end{cases} . \tag{5}$$

If $\varepsilon_t = 0$ or $\varepsilon_t \geq 0.5$, stop the algorithm.
3. Compute the value

$$\alpha_t = 0.5 \ln \frac{1 - \varepsilon_t}{\varepsilon_t} . \tag{6}$$

4. Modify weights for learning vectors according to the formula

$$D_{t+1}(\mathbf{z}^l) = \frac{D_t(\mathbf{z}^l) \exp\{-\alpha_t \mathbf{I}(h_t(\mathbf{x}_l) = y^l)\}}{N_t} , \tag{7}$$

where N_t is a constant such that $\sum_{l=1}^{m} D_{t+1}(\mathbf{z}^l) = 1$. To compute the overall output of the ensemble of classifiers trained by AdaBoost algorithm, the following formula is used

$$f(\mathbf{x}) = \sum_{t=1}^{T} c_t h_t(\mathbf{x}) , \tag{8}$$

where

$$c_t = \frac{\alpha_t}{\sum_{t=1}^{T} \alpha_t} \tag{9}$$

is classifier importance for a given training set, $h_t(\mathbf{x})$ is the response of the hypothesis t on the basis of feature vector $\mathbf{x} = [x_1, ..., x_n]$. The coefficient c_t value is computed on the basis of the classifier error and can be interpreted as the measure of classification accuracy of the given classifier. Moreover, the assumption (1) should be met. As we see, the AdaBoost algorithm is a meta-learning algorithm and does not determine the way of learning for classifiers in the ensemble.

2.2 Image Descriptors for Interest Regions

Local image descriptors for interest regions are utilized in applications such as image matching, image or texture recognition. There are many local descriptors developed to date [18]. The most common are SIFT (Scale Invariant Feature Transformation) [8] and SURF (Speeded Up Robust Features) [2]. Generally, SURF is faster than SIFT thanks to lower dimensionality (64 vs. 128 dimensions. SIFT

[9], consists of four major stages: (1) scale-space peak selection; (2) keypoint localization; (3) orientation assignment; (4) keypoint descriptor. A SIFT descriptor is constituted of a 128-dimensional vector (8 orientation bins for each 4x4 location bins). This representation allows significant levels of local distortions and changes in illumination. Vector of 128 key points is reduced in our approach to a 36-dimensional feature vector [6].

3 Proposed Approach

The main idea of this paper is suggested in [17], however we introduced many changes to the original concept. The authors of [17] used boosting algorithm to find a set of representative keypoints for selected class of images. They also changed the basis of the Adaboost algorithm: in each step of their modified algorithm they use the whole set of training examples to build many classifiers, then they choose the best model according to error value. We revert to the original AdaBoost. In each step we randomly choose one positive example according to their weight. Such an example $\mathbf{c} = [c_1, \ldots, c_n] = [x, \ldots, x_n]$ is a base to build the final classifier. This classifier is a very simple nonstandard neuro-fuzzy model, because it consists of one fuzzy rule. Thus we construct for every element of a vector $\mathbf{c}$ a Gaussian function $G_i(c_i, \delta_i)$. Each Gaussian function has the center in the value of x_i, $i = 1, .., n$. Then we train our classifier in the following way:

1. We calculate the distance from the point designated by the selected vector $\mathbf{c}$ to each of the remaining positive examples $\mathbf{x}$.
2. During this process we have to check whether this distance is less than the threshold chosen by user (in our simulations we chose 0.5).
3. In this case (distance is less then threshold) we record for each of coordinates of vector $\mathbf{c}$ how far a coordinate is distant from it to the right or left, that is we calculate
   If $(LeftMargin_i > c_i)$ then $LeftMargini = c_i$
   If $(RightMargini < c_i)$ then $RightMargin_i = c_i$
   And at the start of the algorithm $RightMargin_i = LeftMargin_i = c_i$
4. After training process, we modify parameters of each of the Gaussian functions which constitute the classifier in this step. The new centers are designated using formula $c_i = |RightMargin_i - LeftMargin_i|/2$. The second parameter of the Gaussian functions is their width. Due to the fact that our Gaussian functions are equated with membership functions of fuzzy sets we assume that throughout the interval $|LeftMargin, RightMargin|$ it must take values greater than 0.5; classifier assigned samples from this area to the class of positive samples. Hence, we set the width of the Gaussian functions by
$$\delta_i = \frac{|RightMargin_i - LeftMargin_i|}{\sqrt{-\ln 0.5}} \tag{10}$$
5. After determining the parameters of the new Gaussian functions we also obtain a set of fuzzy rules antecedents. For the entire set of samples (positive

and negative ones) we test the performance of the classifier by treating it as a single neuro-fuzzy rule. The aggregation of rules antecedents is made with the minimum operator according to $\varepsilon = \sum_j w_j \left| r\left(\mathbf{x}_j\right) - y_j \right|$ which corresponds to (4) and r is the output of a given classifier (rule). In this step, we determine also the importance of the rule, according to the formula (6). Now we modify weights of all samples depending on the classification quality

$$w_{t+1,j} = w_{t,j} \beta_t^{1-e_j} \tag{11}$$

which corresponds to the formula 7 and $e_j = 0$ when the sample is classified correctly and $e_j = 1$ otherwise and $\beta_t = \ln \frac{1-\varepsilon_t}{\varepsilon_t}$ where t is a boosting step.

In [17] the membership to the a given classifier h is determined by computing distance d and then checking if d is smaller than threshold θ:

$$h\left(f, \theta\right) = \begin{cases} 1, d < \theta \\ 0, d \geq \theta \end{cases}, \tag{12}$$

where $h\left(f, \theta\right)$ is the response of a classifier, f – vector of features, which is identified with a classifier. The operation of this classifier can be presented in two-dimensional feature space as a circle with a radius of θ within which all samples are similar (in the sense of close proximity) to the feature vector f (Fig. 1c). In the figure a circle and a cross denote the position of the feature vectors of the two classes of objects in two-dimensional space. Despite its simplicity, the classifier has a major disadvantage, as it requires the selection of an appropriate threshold value θ. In the case of high-value threshold θ, the classifier can be extended to reach samples from a different class (Fig. 1d). On the other hand, lower threshold θ may not include all of the samples.

Our approach, consisting in the formation of the Gaussian functions, is more immune to the relative position of the samples in the feature space. Each Gaussian function is determined during learning, constantly changing its width as learning new samples. Since every dimension of the feature vector is represented by one Gaussian function, the space is divided into rectangles of varying width, the shape of which is better suited to the distribution of samples in space. Figure 1 a and b show how the shape of the Gaussian functions fits to the next sample during the learning process, not including samples from other classes in its range.

4 Results

Two hundred images was scaled to the size of 120x80. The files were taken from the Corel Database [19]. Images are divided into two classes – the first class consists of the pictures of dinosaurs. The second class consists of randomly selected images from the other ones. Figure 2 shows nine examples of positive samples used for learning. After training process described in Section 3 we obtained a set of ten classifiers. The examples of Gaussian functions for one feature after pruning process is shown in Figure 3. All positive images were classified correctly.

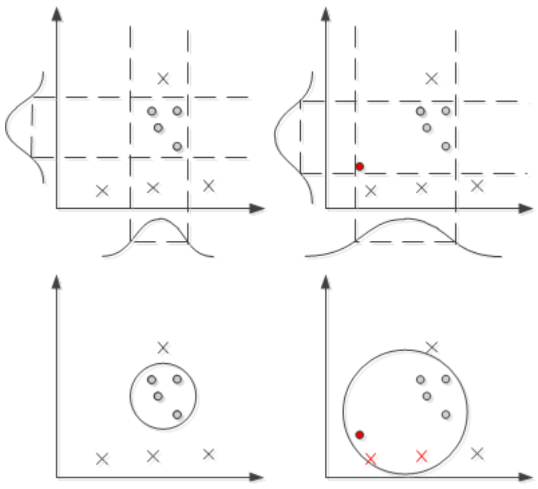

Fig. 1. Effect of change of the way of key point comparison, a) based on the distance b) using fuzzy logic

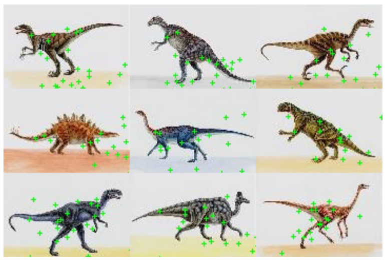

Fig. 2. Example of positive samples used for learning

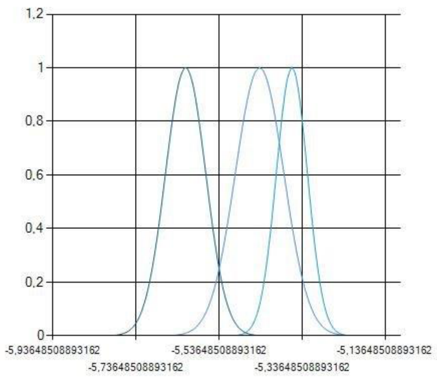

Fig. 3. Example of Gaussian functions of one feature for three simple classifiers after pruning process

5 Conclusions

The presented method is a starting point for further work on the fuzzy representation of similar feature vectors (key points). Gaussian functions acquired during learning can be reduced by finding similar ones and their merging. It will allow to reduce the number of parameters describing all the key points of the class. These functions are assigned with labels describing images that contain specific data points. This in turn will allow easier storage in the database and index generation for faster search for similar images. Similar sets of key points described by sets of Gaussian functions allow to perform some fuzzy logic operations and will also allow for the calculation of the degree of similarity between images. The system presented in this paper can only distinguish one class of objects from the other ones. If we build several such classifier systems, we can combine them with appropriate assumptions about fuzzy rules normalization [10].

Acknowledgments. The project was funded by the National Center for Science under decision number DEC-2011/01/D/ST6/06957.

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: European Conference in Computer Vision, pp. 404–417 (2006)
2. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-Up Robust Features (SURF). International Journal of Computer Vision and Image Understanding (CVIU) 110(3), 346–359 (2008)
3. Cpałka, K.: A method for designing flexible neuro-fuzzy systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 212–219. Springer, Heidelberg (2006)
4. Cpałka, K., Rutkowski, L.: Flexible Takagi Sugeno fuzzy systems. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2005, Montreal, pp. 1764–1769 (2005)
5. Cpałka, K.: On evolutionary designing and learning of flexible neuro-fuzzy structures for nonlinear classification. Nonlinear Analysis Series A: Theory, Methods and Applications 71(12), e1659–e1672 (2009)
6. Ke, Y., Sukthankar, R.: PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In: Computer Vision and Pattern Recognition (2004)
7. Liu, Y., Zhang, D., Lu, G., Ma, W.-Y.: Asurvey of content-based image retrieval with high-level semantics. The Journal of The Pattern Recognition 40, 262–282 (2007)
8. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, vol. 2, pp. 1150–1157, doi:10.1109/ICCV.1999
9. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. Int'l J. Computer Vision 60(2), 91–110 (2004)
10. Korytkowski, M., Rutkowski, L., Scherer, R.: From Ensemble of Fuzzy Classifiers to Single Fuzzy Rule Base Classifier. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 265–272. Springer, Heidelberg (2008)

11. Nowicki, R., Rutkowski, L.: Soft Techniques for Bayesian Classification. In: Rutkowski, L., Kacprzyk, J. (eds.) Neural Networks and Soft Computing. Advances in Soft Computing, pp. s.537–s.544. Springer Physica-Verlag (2003)
12. Ogiela, L., Tadeusiewicz, R., Ogiela, M.R.: Cognitive Computing in Intelligent Medical Pattern Recognition Systems. In: Huang, D.-S., Li, K., Irwin, G.W. (eds.) ICIC. LNICST, vol. 344, pp. 851–856. Springer, Heidelberg (2006)
13. Ogiela, M.R., Tadeusiewicz, R.: Syntactic pattern recognition for X-ray diagnosis of pancreatic cancer. IEEE Engineering in Medicine and Biology Magazine 19(6), 94–105 (2000)
14. Ogiela, M.R., Tadeusiewicz, R., Ogiela, L.: Intelligent Semantic Information Retrieval in Medical Pattern Cognitive Analysis. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3483, pp. 852–857. Springer, Heidelberg (2005)
15. Schapire, R.E.: A Brief Introduction to Boosting. In: Conference on Artificial Intelligence, pp. 1401–1406 (1999)
16. Starczewski, J.T.: On defuzzification of interval type-2 fuzzy sets. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 333–340. Springer, Heidelberg (2008)
17. Tieu, K., Viola, P.: Boosting Image Retrieval. International Journal of Computer Vision 56(1/2), 17–36 (2004)
18. Tuytelaars, T., Mikolajczyk, K.: Local Invariant Feature Detectors: A Survey. In: Foundation and Trends in Computer Graphics and Vision, pp. 177–280 (2008)
19. Wang, J.Z., Li, J., Wiederhold, G.: SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture Libraries. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 23(9), 947–963 (2001)
20. Zhang, W., Yu, B., Zelinsky, G., Samaras, D.: Object class recognition using multiple layer boosting with heterogenous features. In: Proc. CVPR, pp. 323–330 (2005)