Adaptive Potential Active Hypercontours

Arkadiusz Tomczyk[1] and Piotr S. Szczepaniak[1,2]

[1] Institute of Computer Science, Technical University of Lodz
Wolczanska 215, 93-005, Lodz, Poland
tomczyk@ics.p.lodz.pl
[2] Systems Research Institute, Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland

Abstract. In this paper, the idea of *adaptive potential active hypercontours* (APAH) as a new method of construction of an optimal *classifier* is presented. The idea of *active hypercontours* generalizes the traditional *active contour* methods, which are extensively developed in image analysis, and allows the application of their concepts in other *classification* tasks. In the presented implementation of APAH the evolution of the *potential hypercontour* is controlled by *simulated annealing* algorithm (SA). The method has been evaluated on the IRIS and MNIST databases and compared with traditional *classification* techniques.

1 Introduction

The concept of *active hypercontours* (AH) was first introduced in [17] as a generalization of the *active contour* (AC) techniques ([1,2,3,4,5]) which are used in the image analysis. As shown in [16], *active contour* techniques can be in fact considered as search methods of an optimal *classifier* of pixels (it is usually a *contextual classifier*). The main advantage of *active contours* methods, in comparison with traditional *segmentation* techniques, is the possibility of an arbitrary choice of *energy function* which makes those techniques much more intuitive and allows an easier use of experts' knowledge than other *classification* methods (e.g. k-NN, *neural networks* NN ([6,7,10,13]), etc.). The concept of AH was proposed to enable an exchange of experience between those so far separately developed methodologies. In this article, a practical realization of AH is presented.

The paper is organized as follows: in section 2 the basic concepts of *active hypercontours* are described and their relationship with traditional *classifiers* is revealed, section 3 focuses on the description of the proposed APAH algorithm, section 4 presents obtained results comparing them to results achieved by means of traditional *classification* methods and finally the last section is devoted to conclusions and main ideas for future research directions.

2 Active Hypercontours and Classifiers

2.1 Hypercontours and Classifiers

In AC methods the *contour* of the object is sought in an optimization process of *energy* function $E : \mathcal{C} \rightarrow \mathbf{R}$, where $\mathcal{C}$ is the space of acceptable contours. The

L. Rutkowski et al. (Eds.): ICAISC 2006, LNAI 4029, pp. 692–701, 2006.

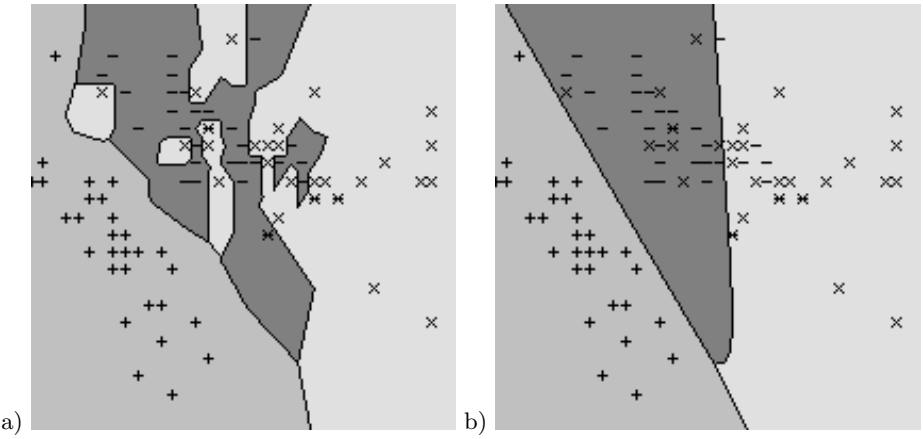

Fig. 1. Sample *hypercontours* generated by traditional *classifiers* (IRIS database, *features* $1-2$): (a) - k-NN method with Euclidean metric and $k=1$ (70), (b) NN (MLP) with 10 neurons in one hidden layer (74)

construction of a *classifier* is very similar. To find an optimal *classifier* every method bases on some *a priori* knowledge (e.g. on a training set of correctly labeled objects). That knowledge can be expressed in the form of *performance index* $Q : \mathcal{K} \to \mathbf{R}$ capable of the evaluation of the usefulness of each function $k \in \mathcal{K}$ (where $\mathcal{K}$ represents the space of all admissible *classifiers*). It is further assumed that $k : \mathcal{X} \to \mathcal{L}$ where $\mathcal{X} \subseteq \mathbf{R}^n$ denotes a *feature space* and $\mathcal{L} = \{1, \ldots, L\}$ denotes the set of labels. The *performance index* plays here a similar role to the *energy function*.

To exchange the experience between those groups of methods, the relationship between them was presented in [16] and further developed to the general definition of *hypercontour* in [17]. The *hypercontour* (*contour* is a special case of *hypercontour* for $n=2$ and $L=2$) can be defined in the following way:

Definition 1. *Let ρ denote any metric (e.g. Euclidean metric) in $\mathbf{R}^n$, $\mathcal{L} = \{1, \ldots, L\}$ denote the set of labels and let $K(\mathbf{x_0}, \varepsilon) = \{\mathbf{x} \in \mathbf{R}^n : \rho(\mathbf{x_0}, \mathbf{x}) < \varepsilon\}$ denote the sphere with center $\mathbf{x_0} \in \mathbf{R}^n$ and radius $\varepsilon > 0$. The set $h \subseteq \mathbf{R}^n$ together with information about labels of regions it surrounds is called a hypercontour if and only if there exists a function $f : \mathbf{R}^n \to \mathbf{R}$ and $p_0 = -\infty, p_1 \in \mathbf{R}, \ldots, p_{L-1} \in \mathbf{R}, p_L = \infty$ $(p_1 < p_2 < \ldots < p_{L-1})$ such that:*

$$h = \{\mathbf{x} \in \mathbf{R}^n : \exists_{l_1, l_2 \in \mathcal{L}, l_1 \neq l_2} \ \forall_{\varepsilon > 0} \ \exists_{\mathbf{x_1}, \mathbf{x_2} \in K(\mathbf{x}, \varepsilon)} \ p_{l_1 - 1} \leq f(\mathbf{x_1}) < p_{l_1} \wedge p_{l_2 - 1} \leq f(\mathbf{x_2}) < p_{l_2}\} \tag{1}$$

and the region $\{\mathbf{x} \in \mathbf{R}^n : p_{l-1} \leq f(\mathbf{x_1}) < p_l\}$ represents class $l \in \mathcal{L}$.

In [17] it has been proved that *hypercontours* are equivalent to *classifiers*. Each *classifier* generates a *hypercontour* in every *feature space* which has a sufficient discriminative power to distinguish *classified* objects:

$$p_0 = -\infty, \forall_{l\in\{1,\ldots,L-1\}}\ p_l = l + \frac{1}{2}, p_L = \infty$$
$$\forall_{\mathbf{x}\in\mathbf{R}^n}\ f(\mathbf{x}) = k(x) \tag{2}$$

Similarly, each *hypercontour* unambiguously generates the corresponding *classification* function:

$$\forall_{\mathbf{x}\in\mathbf{R}^n}\ k(\mathbf{x}) = l \text{ if } p_{l-1} \leq f(\mathbf{x}) < p_l \tag{3}$$

The name *hypercontour* is used to emphasize the relationship of the proposed technique with *active contour* methods.

Having the *hypercontour* defined as a generalization of *contour*, it is easy to generalize the AC technique for other *classification* problems than *image segmentation*. Only the *energy function* $E : \mathcal{H} \to \mathbf{R}$ must be properly defined to evaluate the usefulness of *hypercontours* $h \in \mathcal{H}$ (where $\mathcal{H}$ is the space of all the available *hypercontours*) and the optimization technique must be chosen to be able to find an optimal *classifier*. That leads to the formulation of the AH technique and thanks to that all the advantages of AC can be used not only in image analysis tasks but also in other *classification* problems.

2.2 Traditional Classifiers as Hypercontours

As stated in the previous section, each *classifier* generates a corresponding *hypercontour* in the proper *feature space*. Two typical *classifiers*: k-NN and NN (*multilayer perceptron* MLP) can be considered as examples of that fact. Each classifier assigns labels to vectors from the *feature space* and divides it into L regions of different topology. The boundaries of those regions are in fact a visual representation of *hypercontour*. In Fig. 1, sample *hypercontours* for the above mentioned two types of *classifiers* are presented.

In the case of NN, the similarity to AH is even more evident. The process of *neural network* learning is in fact a search for the optimal *classifier* basing on knowledge contained in the training set. Each iteration of the *back propagation* algorithm actually creates a new *classifier* (encoded in weights of *neural network*) and, in consequence, a new *hypercontour*. Thus, the adaptation of weights of neurons is in fact an evolution process controlled by the objective function (*performance index*, *energy*) which evaluates the progress of learning.

3 Potential Active Hypercontours

3.1 Potential Hypercontour

In this article *potential hypercontour* as a method of practical realization of *hypercontour* is introduced. For given values of n (the number of *features*) and L (the number of *classes*), the *potential hypercontour* is defined by means of a set of labeled control points: $D^c = \{\langle \mathbf{x}_1^c, l_1^c\rangle, \ldots, \langle \mathbf{x}_{N^c}^c, l_{N^c}^c\rangle\}$ where $\mathbf{x}_i^c \in \mathcal{X} \subseteq \mathbf{R}^n$ and $l_i^c \in \mathcal{L}$ for $i = 1, \ldots, N^c$. Each point is a source of potential the value of which decreases with the increase of distance from the source point (that concept

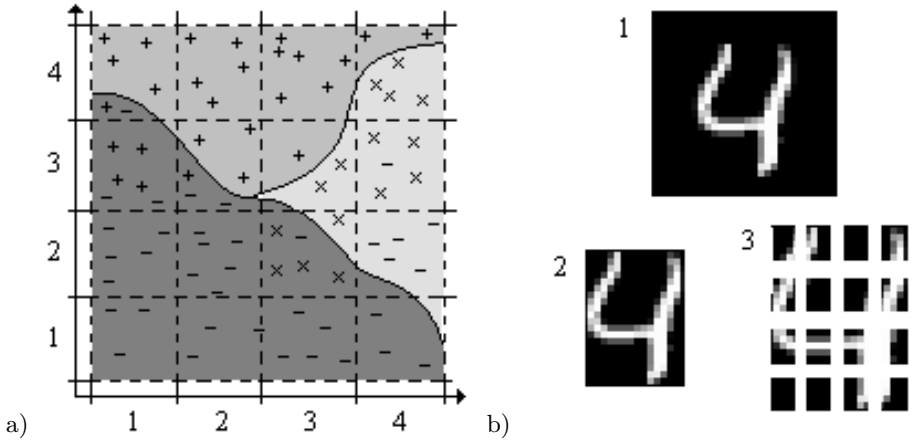

Fig. 2. (a) The method of *potential hypercontour* adaptation ($M = 4$). In the presented situation ($n = 2$, $L = 3$) new control points should be added in centers of regions $(1,3)$, $(3,2)$, $(4,2)$. (b) The method of the extraction of *features* for MNIST database: 1 - sample object from MNIST database, 2 - the smallest region containing the whole digit, 3 - subregions where the number of pixels with intensity above given threshold is calculated.

is similar to the electric potential field). The *classifier* (and consequently the corresponding *hypercontour* (2)) is defined as follows:

$$\forall_{\mathbf{x}\in\mathcal{X}}\ k(\mathbf{x}) = \arg\max_{\mathbf{l}\in\mathcal{L}} \sum_{i=1}^{N^c}\ P_{Q_i\mu_i}(\mathbf{x}_i^c, \mathbf{x})\delta(l_i^c, l) \tag{4}$$

where $\delta : \mathcal{L}\times\mathcal{L} \to \{0,1\}$, $l_1 \neq l_2 \Rightarrow \delta(l_1,l_2) = 0$, $l_1 = l_2 \Rightarrow \delta(l_1,l_2) = 1$ and $P : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ is a *potential function* e.g. *exponential potential* function:

$$P_{Q\mu}(\mathbf{x}_0, \mathbf{x}) = Qe^{-\mu\rho^2(\mathbf{x}_0,\mathbf{x})} \tag{5}$$

or *inverse potential* function:

$$P_{Q\mu}(\mathbf{x}_0, \mathbf{x}) = \frac{Q}{1+\mu\rho^2(\mathbf{x}_0,\mathbf{x})} \tag{6}$$

In both cases Q and μ are parameters characterizing the potential field and ρ denotes a metric (e.g. Euclidean metric) ([6]). The *potential hypercontour* defined in this way is able to describe almost each division of *feature space* (its shape depends both on the distribution of control points and on parameters of *potential* functions) and at the same time is very simple and intuitive in use.

3.2 Evolution of Potential Hypercontour

The shape of the *potential hypercontour* and its *classification* ability depend on the position of control points in the *feature space* and on the parameters characterizing *potential* functions. The search for the optimal distribution of control

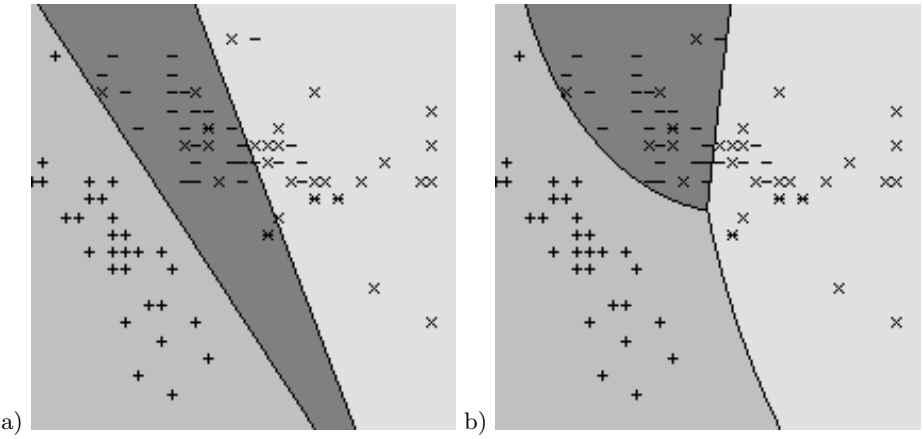

Fig. 3. The influence of the parameter Q of *potential* functions on the results of PAH algorithm (IRIS database, *inverse potential*, $\mu = 1$, *features* $1-2$): (a) - the parameter Q was not modified during optimization (76), (b) - the parameter Q was modified by *simulated annealing* algorithm (the search space contains more elements - the *hypercontours* are more flexible) (80)

points only appeared, after first experiments, to be sometimes unsatisfactory (the *hypecontours* are less flexible and its harder to achieve a desired shape especially if the initial configuration is far from the optimal one) (Fig. 3). Because of that reason also the optimal value of the parameter Q for each source of potential is sought (parameter μ is assumed to be constant during the optimization). In the proposed implementation of *potential active hypercontour* PAH the *simulated annealing* (SA), as an optimization algorithm, was used ([18]). That method on the one hand does not require any gradient information about *objective function* (only its values) and, on the other hand, it allows avoiding the local minima. Any other optimization techniques (e.g. *genetic algorithm* etc.) can also be used here.

3.3 Energy of Hypercontour

The main advantage of AH is its ability to define *energy* (*objective function*) in an almost arbitrary way. In this paper the *a priori* knowledge about the problem is hidden in a training set. In general, however, the *energy* can use any other information obtained from an expert as well as it can put any arbitrary chosen constraints on the shape of the desired *hypercontour*.

Let $D^{tr} = \{\langle \mathbf{x}_1^{tr}, l_1^{tr} \rangle, \ldots, \langle \mathbf{x}_{N^{tr}}^{tr}, l_{N^{tr}}^{tr} \rangle\}$ where $\mathbf{x}_i^{tr} \in \mathcal{X} \subseteq \mathbf{R}^n$ and $l_i^{tr} \in \mathcal{L}$ for $i = 1, \ldots, N^{tr}$ denote a sample training set of correctly labeled vectors. The *energy* of *potential hypercontour* h described by means of control points D^c can be defined as:

$$\forall_{h \in \mathcal{H}}\ E(h) = \sum_{i=1}^{N^{tr}} (1 - \delta(l_i^{tr}, \arg\max_{\mathbf{l} \in \mathcal{L}} \sum_{i=1}^{N^c}\ P_{Q_i \mu_i}(\mathbf{x}_i^c, \mathbf{x}_i^{tr}) \delta(l_i^c, l))) \qquad (7)$$

3.4 Adaptive Potential Active Hypercontour

Experimental results revealed that in some situations the initial large number of random control points can cause some problems during the optimization process i.e. it can be hard to get out of the local minima. (especially when there is no *a priori* information about some fragments of *feature space*). Moreover, at the beginning of the *hypercontour* evolution it is usually not known how many control points are needed for a satisfactory description of a desired *classifier*. Too many control points can reduce its generalizing abilities. Due to those reasons and to improve the performance of the proposed algorithm, an adaptation mechanism can be added to PAH. This adaptation allows in APAH to start the optimization phase several times. After each phase additional control points can be added to D^c in those areas of *features space* where the number of incorrect *classifications* is the largest. Thus, the evolution can begin with a smaller number of control points and can be finished when the *classification* results are satisfactory.

In the presented implementation each adaptation step adds either one or L control points, one for each class. To choose the points that should be added, the smallest n-dimensional cube containing all the points from the *training set* is considered. After each optimization phase that cube is divided into M^n identical but smaller n-dimensional cubes (in every dimension the smallest interval containing all the possible values of the *feature* is divided into M equal subintervals) (Fig. 2). Next, in all of those cubes the number of incorrect *classifications* of objects from D^{tr} is calculated and a new control point, for a given *class* $l \in \mathcal{L}$, is placed in the center of that cube where the highest number of wrong *classifications* of objects from that *class* was observed.

4 Results

The method was tested on the IRIS and MNIST databases. In both cases the *training set* D^{tr} and *test set* $D^{te} = \{\langle \mathbf{x}_1^{te}, l_1^{te} \rangle, \ldots, \langle \mathbf{x}_{N^{te}}^{te}, l_{N^{te}}^{te} \rangle\} \subseteq \mathbf{R}^n \times \mathcal{L}$ were considered. The latter was used to evaluate the results of *classification* (the percent of correct *classifications* in that set was used as a measure of the quality of the *classifier*).

The first data set contains $L = 3$ classes referring to 3 types of iris plants (*iris setosa*, *iris versicolour* and *iris virginica*) ([19]). Each class is represented by 50 objects and each object is described using $n = 4$ *features* (*sepal length*, *sepal width*, *petal length* and *petal width*). For evaluation purposes the whole set was randomly divided into *training set* D^{tr} (100 instances) and *test set* D^{te} (50 instances). The achieved results of *classification* for traditional methods as well as for PAH and APAH are presented in Table 1 and in Fig. 3, Fig. 4.

The second database contains a set of images with handwritten digits ($L = 10$) ([20]). One image contains one digit only. This set was divided into *training set* D^{tr} and *test set* D^{te} with 6000 and 1000 instances respectively. In this case to conduct experiments, the *features* had to be first extracted. The method of extraction proposed here first finds the smallest region containing the whole digit and then divides it into a given number (here 16) of identical subregions

Table 1. Sample *classification* results (percent of correctly labeled objects from D^{te}) for each combination of *features* (IRIS database). Each column corresponds to one classifier: (a), (b) k-NN method with Euclidean metric and $k = 1$ and $k = 7$, respectively; (c), (d) - NN with one hidden layer containing 5 and 10 neurons, respectively; (e), (f), (g) - PAH method with *inverse potential* where $\mu = 1$ and the number of control points in each class is equal to 1, 2, 10 respectively; (h), (i) - APAH with *inverse potential*, $\mu = 1$, $M = 8$ and 1, 9 adaptation steps respectively, (j), (k), (l), (m), (n) - the same as (e), (f), (g), (h), (i) but with $\mu = 15$.

Features	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	(m)	(n)	**Best**
1	68	68	74	68	72	74	72	72	72	76	72	72	72	72	PAH
2	46	50	50	50	52	52	52	52	52	52	52	52	52	54	APAH
3	92	94	92	92	92	92	92	92	92	92	92	92	92	94	k-NN, APAH
4	92	92	94	92	94	92	92	92	92	92	96	96	92	96	APAH
$1-2$	70	76	74	82	76	76	74	78	80	78	76	78	78	84	APAH
$1-3$	94	92	92	94	94	94	94	94	94	94	94	94	94	96	APAH
$1-4$	94	96	94	94	94	94	94	94	94	94	94	96	94	94	k-NN, PAH
$2-3$	90	94	92	92	90	92	92	92	92	92	92	92	92	92	k-NN
$2-4$	92	92	94	92	92	94	92	94	94	92	92	92	90	94	NN, APAH
$3-4$	96	96	96	96	96	96	96	96	96	96	96	96	96	96	ALL
$2-3-4$	98	94	94	94	94	94	96	94	96	94	94	94	94	98	k-NN, APAH
$1-3-4$	96	98	96	94	96	96	96	98	96	94	94	98	96	98	k-NN, APAH
$1-2-4$	94	98	92	94	92	90	94	94	96	94	94	94	98	98	k-NN, APAH
$1-2-3$	92	90	94	96	94	96	94	94	94	94	94	96	94	96	NN, APAH
$1-2-3-4$	96	96	94	94	94	96	94	96	98	96	96	96	94	96	APAH

Table 2. The influence of the adaptation process on the *classification*. Comparison of the achieved results (percent of correctly labeled objects from D^{te}) with results obtained by means of traditional techniques (MNIST database). Each column corresponds to one classifier: (a), (b), (c) - k-NN method with Euclidean metric and $k = 1$ and $k = 7$, respectively; (d), (e), (f), (g) - NN with one hidden layer containing 10, 20, 50 and 100 neurons, respectively; (h), (i), (j), (k), (l) - APAH method with *inverse potential*, $\mu = 1$ and $M = 2$ after 0, 4, 8, 12, 16 adaptation steps, respectively.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
78.1	80.5	78.9	65.0	72.8	76.4	79.2	19.2	48.4	62.2	66.8	75.2

(Fig. 2). The *feature vector* is composed of the ratios of pixels with intensity above the given threshold (e.g. 128) to the whole number of pixels in every subregion (consequently $n = 16$). In spite of the fact that in the literature the better *features* can be found, those proposed here are sufficient for comparison of *classifiers*. The obtained results are gathered in Table 2.

It is worth mentioning that the choice of a *potential function* and its parameters affects the character of the *potential hypercontour* and consequently the *classification* results. For the *potential functions* considered here, the parameter

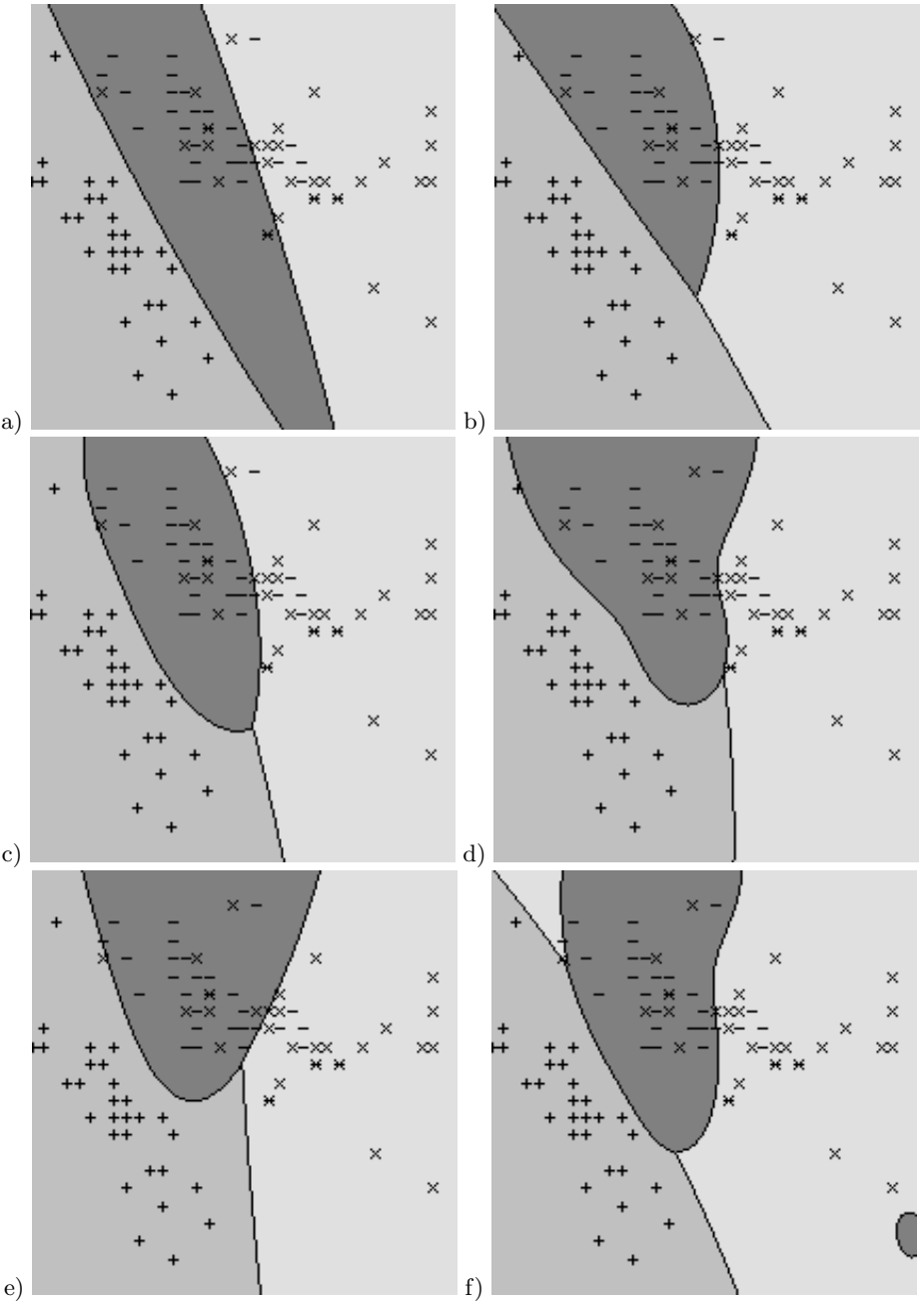

Fig. 4. The influence of parameters of APAH (IRIS database, *features* $1-2$, $M=8$): (a), (c), (e) - $\mu=1$, results after 0 (76), 4 (76), 7 (78) adaptations steps, respectively; (b), (d), (f) - $\mu=15$, results after 0 (78), 4 (80), 7 (82) adaptations steps, respectively

μ affects the flexibility of the APAH (Fig. 4). Thus μ can be used to control the ability of a *classifier* to generalize the available knowledge.

5 Conclusions

The presented APAH method is a new approach to an optimal *classifier* construction. It allows (even in its basic form presented here) to obtain *classifiers* which give similar and in some situations even better results than traditional methods. The main advantage of APAH, however, is its ability to straightforwardly use any experts' knowledge. Here, the knowledge was gathered in the form of a *training set*, but sometimes experts, basing on their experience, can add a heuristic information which can improve the *classification* significantly (e.g. *fuzzy* information). This method allows also to take into account any additional constraints that can be put on the shape of desired *hypercontour* (as for example in *support vector machines* SVM) which is not always possible in traditional techniques. In the APAH approach all of that can be simply encoded in the *energy* function. Moreover, it is also possible to consider the *potential active contour* (PAC) method as a new method for image segmentation. It is worth mentioning that there are some analogies between this approach and other methods known from *pattern recognition* to call RBF *neural networks* as an example. All these aspects are presently under further, practical investigation.

References

1. Kass M., Witkin W., Terzopoulos D., (1988) *Snakes: Active Contour Models*, International Journal of Computer Vision, 321–331
2. Caselles V., Kimmel R., Sapiro G., (1997) *Geodesic Active Contours*, International Journal of Computer Vision 22(1) 61–79
3. Xu Ch., Yezzi A., Prince J., (2000) *On the Relationship between Parametric and Geometric Active Contours*, in Proc. of 34th Asilomar Conference on Signals, Systems and Computers 483–489
4. Cootes T., Taylor C., Cooper D., Graham J., (1994) *Active Shape Model - Their Training and Application*, CVGIP Image Understanding, 61(1) 38–59 Janvier
5. Grzeszczuk R., Levin D., (1997) *Brownian Strings: Segmenting Images with Stochastically Deformable Models*, IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 19 no. 10 1100-1013
6. Tadeusiewicz R., Flasinski M., (1991) *Pattern Recognition*, PWN, Warsaw (in Polish)
7. Kwiatkowski W., (2001) *Methods of Automatic Pattern Recognition*, WAT, Warsaw (in Polish)
8. Sobczak W., Malina W., (1985) *Methods of Information Selection and Reduction*, WNT, Warsaw (in Polish)
9. Nikolaidis N., Pitas I., (2001) *3-D Image Processing Algorithms*, John Wiley and Sons Inc., New York
10. Bishop Ch., (1993) *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford

11. Pal S., Mitra S., (1999) *Neuro-fuzzy Pattern Pecognition, Methods in Soft Computing*, John Wiley and Sons Inc., New York
12. Bennamoun M., Mamic G., (2002) *Object Recognition, Fundamental and Case Studies*, Springer-Verlag, London
13. Looney C., (1997) *Pattern Recognition Using Neural Networks*, Theory and Algorithms for Engineers and Scientists, Oxford University Press, New York
14. Sonka M., Hlavec V., Boyle R., (1994) *Image Processing, Analysis and Machine Vision*, Chapman and Hall, Cambridge
15. Gonzalez R., Woods R., (2002) *Digital Image Processing*, Prentice-Hall Inc., New Jersey
16. Tomczyk A., Szczepaniak P. S., (2005) *On the Relationship between Active Contours and Contextual Classification*, 4th International Conference on Computer Recognition Systems (CORES), Rydzyna (near to Leszno), Poland, Springer, 303-311.
17. Tomczyk A., (2005) *Active Hypercontours and Contextual Classification*, 5th International Conference on Inteligent Systems Design and Applications (ISDA), Wroclaw, Polska, IEEE Computer Society Press, 256-261.
18. Kirkpatrick S., Gerlatt C. D. Jr., Vecchi M.P., (1983) *Optimization by Simulated Annealing*, Science 220, 671-680.
19. IRIS, `http://www.ics.uci.edu/~mlearn`.
20. MNIST, `http://yann.lecun.com/exdb/mnist/`.