Using Dominant Sets for k-NN Prototype Selection

Sebastiano Vascon[1], Marco Cristani[1], Marcello Pelillo[2], and Vittorio Murino[1]

[1] Pattern Analysis and Computer Vision (PAVIS), Istituto Italiano di Tecnologia
Via Morego 30, 16163 Genova, Italy
{sebastiano.vascon,marco.cristani,vittorio.murino}@iit.it
[2] DAIS, University Ca'Foscari of Venice, Via Torino 155, 30172 Venezia Mestre, Italy
pelillo@dsi.unive.it

Abstract. k-Nearest Neighbors is surely one of the most important and widely adopted non-parametric classification methods in pattern recognition. It has evolved in several aspects in the last 50 years, and one of the most known variants consists in the usage of prototypes: a prototype distills a group of similar training points, diminishing drastically the number of comparisons needed for the classification; actually, prototypes are employed in the case the cardinality of the training data is high. In this paper, by using the dominant set clustering framework, we propose four novel strategies for the prototype generation, allowing to produce representative prototypes that mirror the underlying class structure in an expressive and effective way. Our strategy boosts the k-NN classification performance; considering heterogeneous metrics and analyzing 15 diverse datasets, we are among the best 6 prototype-based k-NN approaches, with a computational cost which is strongly inferior to all the competitors. In addition, we show that our proposal beats linear SVM in the case of a pedestrian detection scenario.

Keywords: K-nearest neighbors, Prototype selection, Classification, Dominant set, Data reduction.

1 Introduction

The k-Nearest Neighbors (kNN) method [3] is one of the fundamental strategies of non-parametric supervised classification, with a large spectrum of variations proposed in the last 50 years. It lies on a simple principle: a test sample is classified as one of the N available classes by evaluating the majority of the labels of the k nearest training neighbors in the feature space. k-NN, in its original form, suffers two major drawbacks. The first is the scarce *efficiency*, especially when the training dataset becomes large [10]: this is due to the fact that all the distances between the test element and the training set should be computed, to find the k closest elements. The second problem is the *sensitivity* to the outliers [6]: during the classification, all the training data are treated in the same way, ignoring the possibility of taking into account class outliers or

A. Petrosino (Ed.): ICIAP 2013, Part II, LNCS 8157, pp. 131–140, 2013.

noise. Methods that mitigate both problems take advantage of data reduction techniques [14]. The idea builds upon the design of prototypes, *i.e.*, surrogates of similar training points that are evaluated during the classification, substituting the whole training population.

The design of prototypes is crucial since, while good prototypes bring to classification performance higher than those obtained with the full training set [6], bad prototypes lead to a severe decrease of the classification score.

A common strategy for the choice of the prototypes is to adopt clustering strategies [8,9]. In the same line, this paper promotes the use of the dominant set (DS) framework [12] as a prototype extraction technique. DS is a pairwise-clustering technique based on similarities, that generalizes the notion of maximal clique to edge weighted graph. In practice, DS is employed to partition a graph into coherent and compact subsets, *i.e.* cliques, which correspond to clusters under a similarity-based perspective. DSs are extracted optimizing a quadratic function, exploiting whatever similarity measure among points. DS are particularly interesting in our setting due to their insensitivity to outliers [11].

Our method starts by clustering the training data into dominant sets *without using the label information*, in order to extract the natural underlying data structure. Then, we design four different strategies for prototype selection considering the class label information. In essence, DS are employed to discover the nature of the data and validate the information of the original sample labeling so as to improve the classification task. A similar idea but in a completely different scenario has been successfully employed in [7].

To test our approach, we compare the performance of 21 diverse prototype-based k-NN techniques over 15 heterogeneous datasets of binary classification tasks. Our approach is globally in the top 6 positions for what concerns accuracy, accuracy without random hits (the so-coled *Cohen's kappa factor* [2]) and reduction rate, *i.e.*, number of prototypes. To ground our technique with other standard classification methods, we report also a comparison with a standard k-NN and a linear SVM in a pedestrian detection scenario.

The rest of the paper is organized as follows. In Sec. 2, our approach is detailed regarding the DS clustering technique, the four prototype extraction methods and the classification algorithm. The experimental results are provided in Sec. 3, and, finally, Sec. 4 summarizes the work and envisages future perspectives.

2 The Proposed Approach

Given a binary classification problem supported by labeled training set, the main pipeline of our approach is illustrated in Fig. 1. It is composed by four steps:

1. **Dominant Set clustering:** the original training dataset is clustered by finding its dominant sets, without the label information (sec.2.1).
2. **Cluster labeling:** each cluster is labeled by considering the label distribution of its members (sec. 2.2)

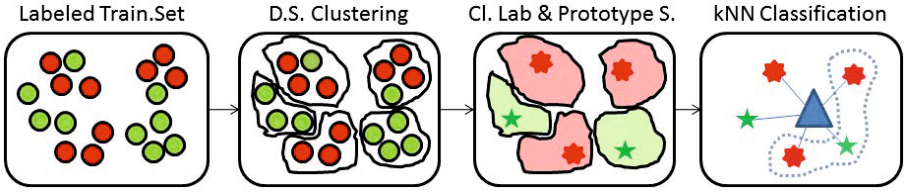

Fig. 1. The main pipeline of the algorithm. Starting from a labeled training set on left, data reduction and prototypes extraction are obtained through dominant set clustering, followed by k-NN prototype-based classification.

3. **Prototype selection:** from each cluster, prototypes are extracted following 4 alternative methods (sec 2.3)
4. **Classification:** k-NN classification over prototypes is applied (sec. 2.4)

2.1 Dominant Set Clustering

Dominant set clustering is a graph-based method that generalizes the problem of finding a maximal clique to edge-weighted graphs. A natural application of this method is for partitioning (clustering) a graph. In our case, we have a standard dataset that is modeled as an undirected edge-weighted graph $G = (V, E, w)$ with no self loops, in which the nodes V are items of the dataset (represented by feature vectors), the edges $E \subseteq V \times V$ represent similarity between pair of nodes and the weight function $w : E \to \mathbb{R}_+$ calculates pairwise similarities. The $n \times n$ symmetric adjacency matrix $A = (a_{ij})$ is employed to summarize G:

$$a_{ij} = \begin{cases} w(i,j) & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

In whatever clustering approach, the clusters should satisfy two properties: high *intra-cluster homogeneity* and high *inter-cluster inhomogeneity*. In the DS framework, these two properties translate in having the weights of the edges within a cluster higher than the ones connecting external nodes. Given a subset $S \subseteq V$, $S \neq \emptyset$ and a vertex $i \in V$, the *average weighted degree* of i with respect to S is $\delta_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij}$. If $j \notin V$ we can define $\phi_S(i,j) = a_{ij} - \delta_S(i)$, which compares the similarity between i and j with the average similarity of i with respect to S. Let $S \subseteq V, S \neq \emptyset$ and $i \in S$, the *weight* of i with respect to S, $w_S(i)$, is recursively computed as:

$$w_S(i) = \begin{cases} 1 & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j,i) w_{S \setminus \{i\}}(j) & \text{otherwise} \end{cases}$$

This definition indicates a measure of the overall similarity between i and the vertices of $S \setminus \{i\}$, with respect to the overall similarity among the vertices in $S \setminus \{i\}$. The total weight of a cluster S is the sum of all node weights

$$W(S) = \sum_{i \in S} w_S(i)$$

Given a non-empty subset of vertices $S \subseteq V$ such that $W(T) > 0$ for any non-empty $T \subseteq S$, S is said to be dominant if:

1. $w_S(i) > 0 \quad \forall i \in S$ internal homogeneity
2. $w_{S\cup\{i\}}(i) < 0 \quad \forall i \notin S$ external inhomogeneity

Pavan and Pelillo in [12] provided a connection between clusters, dominant sets and local solutions of the following quadratic problem:

$$\begin{aligned} &\text{maximize} \quad \mathbf{x}^T A\mathbf{x} \\ &\text{subject to} \quad \mathbf{x} \in \triangle^n \end{aligned} \tag{1}$$

where A is the adjacency matrix defined above and $\mathbf{x}$ is the *weighted characteristic vector*[12], living in the standard n-dimensional simplex $\triangle^n$, that is, $(\sum_i \mathbf{x}_i = 1, \forall i\ x_i \geq 0)$. If S is a dominant subset of vertices, then its weighted characteristic vector $\mathbf{x}^S = (x_i^S)$, defined as

$$x_i^S = \begin{cases} \frac{w_S(i)}{W(S)} & \text{if } i \in S \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

is a strict local solution of (1). In other words, if $\mathbf{x}$ is a strict local solution of (1) then its support[1] $\sigma(\mathbf{x})$ is a dominant set, provided that $w_{S\cup\{i\}}(i) \neq 0$, $\forall i \in S$. In order to extract a DS, it is sufficient to take the support of a local solution of (1), employing whatever optimization algorithm, like the *replicator dynamics* strategy (RD)[2]. RD is a dynamical system, taken from the evolutionary game-theory that is used for finding a local solution to the problem (1)

$$x_i(t+1) = x_i(t)\frac{(A\mathbf{x}(t))_i}{\mathbf{x}(t)^T A\mathbf{x}(t)} \tag{3}$$

The convergence of the process is guaranteed if the matrix A is non-negative and symmetric, and it brings to an equilibrium condition in which $\mathbf{x}(t) = \mathbf{x}(t+1)$. The optimization starts with a point $\mathbf{x}$, whose size is equal to the training set size, sited in the barycenter of the simplex $(x_i = 1/|\mathbf{x}|, \forall i)$ and (3) is iterated until convergence. The RD operates a selection process over the components of vector $\mathbf{x}$, leading certain elements to emerge $(x_i > 0)$ and others to extinct $(x_i = 0)$: in practice, these last components are almost close to zero, so that a pruning threshold is applied. For compactness we redefine the support as follow: $\sigma(\mathbf{x}) = \{i | x_i \geq max(\mathbf{x}) \cdot cvTh \quad , \quad cvTh > 0\}$. In practice, the pruning parameter $cvTh$ (which stands for *characteristic vector threshold*) has to roles: 1) discarding outliers (samples with low participation x_i); 2) defining a sort of minimal participation, normalized by the sample which has maximal participation. At each iteration, a dominant set is found, and it is subsequently removed from the dataset (*peeling-off strategy*), iterating again on the remaining nodes until every element belongs to a cluster.

[1] The support σ of $\mathbf{x}$ is the set of indices corresponding to the positive components of $\mathbf{x}$, $\sigma(\mathbf{x}) = \{i \in V | x_i > 0\}$

[2] In our implementation we used a more efficient algorithm [13] that grows linearly on each step rather then quadratically like the RD. Same approach has been used in [7]. Matlab implementation at `http://www.dsi.unive.it/~rodola/sw.html`

2.2 Clusters Labeling

After the DS clustering, we need to consider the label information. Since the clustering finds the natural structure of the data set, highlighting compact areas of samples, this has to be married with the label information to highlight the clusters which are coherent with the labeling and those which are not. First of all, given a cluster c, a generic label is assigned to it by using this rule:

$$c_- = \{c_i | class(c_i) = -\} \quad c_+ = \{c_i | class(c_i) = +\}$$

$$label(c) = \arg \max_{l \in \{+,-\}} (|c_l|)$$

with c_- the set of element of the negative class in cluster c (and viceversa with c_+), and $|c_-|$ is its sample cardinality. Therefore, since it is not guaranteed that a cluster is composed only by elements of the same class, we design a confidence measure with respect to the distribution of the original sample labels:

$$confidence(c) = \frac{abs(|c_-| - |c_+|)}{|c_-| + |c_+|} \tag{4}$$

In practice, this measure accounts for the "purity" of a cluster: it is equal to 0 in the case we have the same amount of elements of both classes, indicating an uninformative (for classification's sake) group of data, and it is equal to 1 in case that the cluster is totally homogeneous. In this approach, we use the confidence to prune out entire clusters (those with $confidence = 0$).

2.3 Prototypes Design

Each DS (cluster) is represented by a weighted characteristic vector $\mathbf{x}$ in which every i-th component corresponds to its degree of participation (see eq. 2). Four different prototype selection strategies are proposed:

Max. (*max*): Given a DS c and the corresponding characteristic vector $\mathbf{x}^c$, the prototype is taken as the most participating element and the corresponding label is selected as:

$$p\max(c) = \arg\max_i (x_i^c), \quad label(p\max(c)) = label(c)$$

The rationale is that if the i-th element has a high degree of participation in its cluster, it means that it shares a large amount of similarity with the other items in c, being thus a good representative.

Max Coherence. (*maxco*): Prototypes following this strategy are a subset of the **max** prototype class. Especially in presence of clusters with a low degree of confidence (formed mostly by outliers), the original sample label of the **max** prototype could not correspond to the one inferred by its cluster. In such a case, this strategy forces the *confidence* of the cluster to be 0, so that it has null weight in the classification phase.

Average Prototype. (*avg*): The idea of this strategy is to use an average measure of all the elements in a cluster for a democratic representative, inheriting the class label defined in Sec. 2.2. Given a cluster c, its average prototype is:

$$p_{\text{avg}}(c) = \frac{1}{|c|} \sum_i c_i \quad label(p_{\text{avg}}(c)) = label(c)$$

where $|c|$ is the number of elements in c and c_i is the i-th element in c.

Weighted Average Prototype. (*wavg*): Taking into account the meaning of the characteristic vector (degree of participation of an element in the cluster), the prototype is generated as a weighted sum of the cluster components. Given a cluster c and its characteristic vector $\mathbf{x}^c$:

$$p_{\text{wavg}}(c) = \sum_i x_i \cdot c_i \quad label(p_{\text{wavg}}(c)) = label(c)$$

where c_i is the i-th element of the cluster c.

2.4 Classification

The classification phase derives easily from the original kNN recipe: given an unlabeled item, its k nearest neighbors (NN) are searched only among the prototypes (we now use a single strategy for extracting a prototype from each DS cluster), evaluating a given metrics detailed in the following. Once the k NN are found the majority class is assigned to the item in a standard kNN setting.

3 Experiments

The main goals of the experiments are, first, to compare the approach against the state-of-the-art prototypes-based k-NN methods [6], and, second, to analyze our performances against two of the most common classifiers in pattern recognition, i.e., the simple k-NN and the standard linear Support Vector Machine (SVM).

To implement our strategy, we have to define two entities, *i.e.*, the similarity function to calculate the affinity matrix A and the threshold $cvTh$ (see end of Sec. 2.1) used to extract the DS's. As for the matrix A, we adopt the Euclidean distance (but this is not mandatory in general), opportunely cast as an exponential to make A symmetric, non negative and with zero diagonal, as requested by the DS framework. In short, we have $A = a_{ij}$ where:

$$a_{ij} = \begin{cases} e^{-||i-j||} & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

Regarding the $cvTh$ value, we explore the behavior of our approach with $cvTh = [.0001, .001, .01, .1, .3, .5, .7]$. In general, we did not find dramatic differences using different $cvTh$ values; anyway, interesting behaviors for particular settings will

Table 1. The several datasets used in the experiments. #ex is the number of examples in the dataset, #att is the size of each data sample.

Name	#ex	#att	Name	#ex	#att	Name	#ex	#att
appendicitis	106	7	haberman	306	3	sonar	208	60
australian	690	14	heart	270	13	spambase	4597	57
banana	5300	2	mammographic	961	5	spectfheart	267	44
bands	539	19	monk-2	432	6	titanic	2201	3
bupa	345	6	pima	768	8	wisconsin	699	9

be explained in the following. we experimentally found in this work that values ≤ 0.3 lead to good results in terms of classifier generalization.

In the first scenario, we compare our four prototype strategies vs. all the available prototype-based techniques reported in [6], *i.e.*, 21 methods [3]. We actually considered a subset of the available datasets (see Table 1), only focusing on binary problems with samples represented by numerical features and a defined range of dataset cardinality (# elements ≤ 6000)[4].

As figures of merit, we adopt the standard accuracy *Acc*, the *Cohen's Kappa* Kp and the *reduction rate* R, given a confusion matrix cm:

$$Acc(cm) = \frac{trace(cm)}{\sum_{i,j} cm} \tag{5}$$

$$Kp(cm) = \frac{N * trace(cm) - [a * b + c * d]}{N^2 - [a * b + c * d]} \tag{6}$$

$$a = (TP + FP),\ b = (TP + FN),\ c = (FN + TN),\ d = (FP + TN)$$

$$R = \frac{|\text{training set}| - |\text{prototypes}|}{|\text{training set}|} \tag{7}$$

The Cohen's Kappa Kp [2] measure is a compensation for random success in the accuracy, it varies from -1 to 1 and provides a measure of the agreement between classes in the classifier. Negative values mean disagreement, 0 means that the classifier is acting completely random, and positive values mean that the agreement is very high. The reduction rate R indicates how much the prototypes diminish the number of comparisons needed for the k-NN with respect to exploiting all the original training data.

To design a single measure which summarizes the different performances, we use the product $A \times Kp \times R$; since A and R are bounded in $[0, 1]$, and K in $[-1, 1]$, the composite measure has the range $[-1, 1]$. The performance evaluation follows [6] in which, for each dataset and each technique, 10-fold cross-validation is employed, averaging the obtained results at each run. Finally, we consider k-NN with $k = 1, 3$ as in [6].

[3] Prototype's acronym used in Table 2, 3 and 4 are referred to [6].

[4] A more extensive analysis on a larger number of benchmarks would require more time, so that this will be postponed as future work.

The results are portrayed in the Tables 2 and 3, where we rank the approaches adopting our composite measure. Table 2 shows the best results obtained using

Table 2. 1-NN classifier ranking

Rank	1-NN Classifier	
#	A×K×R	Name
1	0,428	'CHC'
2	0,393	'SSMA'
3	0,393	'GGA'
4	0,369	'RMHC'
5	0,349	'RNN'
6	**0,340**	**AVG**
7	**0,332**	**WAVG**
8	**0,329**	**MAXCO**
9	**0,328**	**MAX**
10	0,296	'CCIS'
11	0,243	'MCNN'
15	0,238	'CPruner'

Table 3. 3-NN classifier ranking

Rank	3-NN Classifier	
#	A×K×R	Name
1	0,386	'GGA'
2	0,382	'RMHC'
3	0,344	'SSMA'
4	**0,334**	**AVG**
5	0,328	'CHC'
6	**0,322**	**MAX**
7	**0,306**	**MAXCO**
8	0,284	'RNN'
9	**0,268**	**WAVG**
10	0,244	'CCIS'
11	0,239	'DROP3'
12	0,224	'HMNEI'

Table 4. Time needed for the prototypes extraction

Rank	PS exec time (sec)	
#	Time	Name
14	10,073	'DROP3'
15	**52,625**	**MAX**
16	**52,625**	**MAXCO**
17	**52,625**	**AVG**
18	**52,625**	**WAVG**
19	80,508	'RNG'
20	127,942	'Recons.'
21	372,360	'CHC'
22	391,666	'RMHC'
23	513,965	'SSMA'
24	874,246	'RNN'
25	1525,734	'GGA'

1-NN classifier with $cvTh = 0.3$. The *avg* prototype locates in the 6-th position, the highest position reached by our prototypes. In particular, we reach 0.79 in accuracy, 0.47 in Kp and 0.91 in reduction rate. Generally, there is a small decay in the performance when $cvTh$ increases, in particular when $cvTh > 0.3$, but this does not affect the global final rank. The decay is motivated by a lack in the generalization: the $cvTh$ values, when > 0.3, constrains all the dominant sets to have high internal coherence, obtaining many prototypes very specialized, hence inducing a globally low generalization capability. In Table 3 the rank on the 3-NN classifier are shown. Even in this case the results are interesting since the best result (obtained with $cvTh = 0.5$) promotes our *avg* prototype-based method in 4th position. In Table 4 we show the timings needed to extract the prototypes in the *most time-demanding* case ($cvTh = 0.7$, since lots of small and coherent cluster should be generated). These results witness that all our proposals are almost one order faster that the best competitors, which definitely promote our idea as a good compromise between accuracy measures and computational effort.

In the second scenario, we compare our approach against a classic kNN approach and a Support Vector Machine[5] in a pedestrian detection task. The dataset involved is the Daimler Monocular Pedestrian [5], composed by a set of labeled gray scale images of size 48×96 pixels representing pedestrian and non pedestrian instances. In our setting, each image is represented by a concatenated set of HOG features [4] producing a vector of 1980 dimensions. Other

[5] The LibSVM [1] implementation has been used in this work.

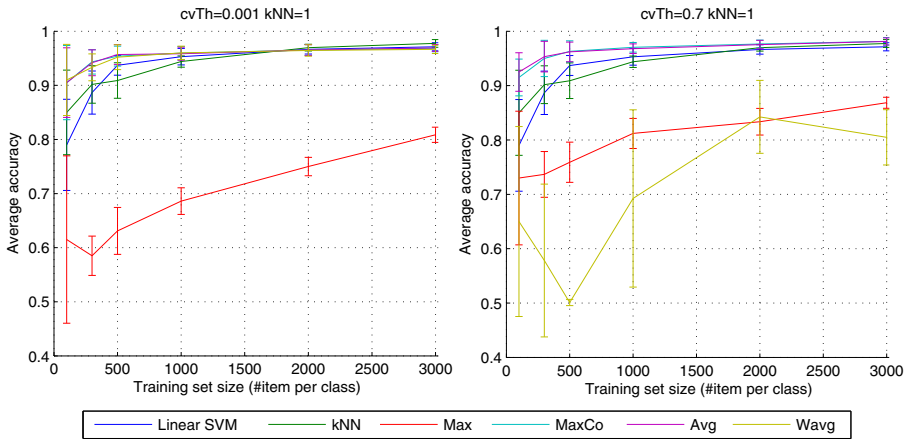

Fig. 2. Comparison of the accuracies achieved by the SVM, k-NN and the four prototypes when changing the $cvTh$. Best viewed in colors.

than evaluating the performance under different value of $cvTh$, we analyze the performances by changing the training set size with [100, 300, 500, 1000 , 2000 ,3000] samples per classes. The number of k-NN is [1 3 5 7]. Again, we perform 10 fold cross-validation, averaging the results. The default parameter for the linear SVM (C=1) has been used. The pairwise similarity matrix A, needed by the DS framework, is calculated using the Euclidean distance as $a_{ij} = e^{-\frac{||f_i - f_j||}{\sigma}}$ where f_i is the HOG vector of the i-th element in the training set. We experimentally found that $\sigma = 10$ gives the highest results. As we can see from Fig.2, all the proposed prototypes except the *max*, exploit better or comparable performances against the linear SVM and k-NN. This is true as long as $cvTh \leq 0.3$; for higher values the *wavg* prototype starts suffering of overfitting since all clusters are very compact and specialized, thus losing in terms of generalization. No plot regarding the changes in the k-NN are shown because the results are almost similar; The more stable results are provided by *max coherence*, *average* and *weighted average*. Typically, the average accuracy of SVM and k-NN increase with the number of examples considered, in our approach the same behavior occurs; in particular, in the cases of *avg, wavg* and *max coherence*, the average accuracy is larger than that achieved by the other methods, even when a small training set is considered. This confirms again the goodness of our approach.

4 Conclusions

In this work, we have proposed a novel approach for the prototype selection based on the Dominant Set clustering technique. In general, our method has the capability of keeping a good balance between accuracy, compression rate, and computational cost. From our experiments, we found that, on average, the

avg prototype produces better results in term of accuracy, this is mainly due to the fact the *avg* is a good statistical representative of the content of the entire cluster.

Our approach is theoretically motivated by the DS theory and provided good results as compared with the state-of-the-art prototype selection methods and classical classifiers like SVM or k-NN. Future work will be devoted to extend this technique to handle multi-class problems.

References

1. Chang, C.C., Lin, C.J.: {LIBSVM}: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2(3), 27:1–27:27 (2011)
2. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement 20(1), 37–46 (1960)
3. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
4. Dalal, N., Triggs, W.: Histograms of Oriented Gradients for Human Detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2005, vol. 1(3), pp. 886–893 (2004)
5. Enzweiler, M., Gavrila, D.M.: Monocular Pedestrian Detection: Survey and Experiments. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(12), 2179–2195 (2009)
6. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(3), 417–435 (2012)
7. Hou, J., Pelillo, M.: A simple feature combination method based on dominant sets. Pattern Recognition (2013)
8. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. 31(3), 264–323 (1999)
9. Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognition Letters 31(8), 651–666 (2010)
10. Kibriya, A.M., Frank, E.: An Empirical Comparison of Exact Nearest Neighbour Algorithms. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 140–151. Springer, Heidelberg (2007)
11. Pavan, M., Pelillo, M.: Dominant sets and hierarchical clustering. In: Proceedings. Ninth IEEE International Conference on Computer Vision, vol. 1, pp. 362–369 (2003)
12. Pavan, M., Pelillo, M.: Dominant sets and pairwise clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(1), 167–172 (2007)
13. Rota Bulò, S., Bomze, I.M.: Infection and immunization: A new class of evolutionary game dynamics. Games and Economic Behavior 71(1), 193–211 (2011)
14. Wilson, D.R., Martinez, T.R.: Reduction Techniques for Instance-Based Learning Algorithms. Machine Learning 38(3), 257–286 (2000)