A Cooperative Game Theoretic Approach to Prototype Selection

N. Rama Suri, V.S. Srinivas, and M. Narasimha Murty

Electronic Commerce Laboratory, Dept. of Computer Science and Automation,
Indian Institute of Science, Bangalore, India
{nrsuri,srini,mnm}@csa.iisc.ernet.in

Abstract. In this paper we consider the task of prototype selection whose primary goal is to reduce the storage and computational requirements of the Nearest Neighbor classifier while achieving better classification accuracies. We propose a solution to the prototype selection problem using techniques from cooperative game theory and show its efficacy experimentally.

1 Introduction

Though nearest neighbor (NN) [1] rule is a well-known supervised non-parametric classifier, it demands huge memory and computational requirements as the entire training data set needs to be stored in the main memory. There have been several efforts to alleviate these problems associated with the NN classifier. *Prototype selection is the problem* of finding representative patterns from a given data set to reduce the storage and computational requirements of a classifier while achieving better classification accuracies. In the literature, two different families of prototype selection methods exist [4]. The first method is called *condensing algorithm* that aims at selecting a minimal subset of prototypes that leads to the same performance as using the whole training set. The difficulty [4] with condensing algorithms is that noisy examples are preferred to be selected into prototype set, which harms the accuracy of the final classifier. The second type of algorithms for prototype selection are called *editing algorithms*. The main difficulty [4] with the editing algorithms is that they may not reduce the training set size.

Motivation: In this paper, we use cooperative game theoretic techniques to address the prototype selection problem. Cooperative game theory is a branch of game theory and we refer the reader to [6] for a detailed discussion on game theory and its variants. To analyze cooperative games, there are a set of solution concepts such as Shapley value, core, bargaining sets [6], etc. Among these, Shapley value is a fair solution concept in the sense that it divides the collective (i.e. total) value of the game among the players according to their marginal contribution in achieving that collective value. That is, the higher the Shapley value of a player, the more important that player among the other players. We can make use of this notion of importance among the available patterns to pick a small set of prototypes from the given data set, if we can formulate the prototype

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 556–564, 2007.

selection problem as a cooperative game. Hence the techniques from cooperative game theory can be useful in proposing a solution approach to the prototype selection problem.

Contributions and Outline of the Paper: To the best of our knowledge, *it is the first time that the techniques from (cooperative) game theory are used in proposing an algorithm to the prototype selection problem.* In Section 2, we present basic concepts underlying cooperative game theory that are useful in understanding the rest of the paper. We formulate a convex game [8], which we call *PS-GAME*, that helps in solving the prototype selection problem in Section 3.1. We propose an exact algorithm for prototype selection in Section 3.2 and also investigate the intractability of this algorithm. Then in Section 3.3, we propose an approximate prototype selection algorithm (APSA), which runs in polynomial time. We finally show the efficacy of the APSA using empirical studies on different data sets in Section 4. We conclude the paper in Section 5 along with a few pointers to the future work.

2 Preliminaries

A cooperative game [9] with transferable utility is defined as the pair (N, v) where $N = \{1, 2, \ldots, n\}$ is the set of players and $v : 2^N \rightarrow R$ is a characteristic function, with $v(\phi) = 0$.

We can analyze a cooperative game using solution concept, which is a method of dividing the total value of the game to the individual players. We now briefly discuss two such solution concepts, namely core and Shapley value.

The Core of a Cooperative Game: A payoff allocation $x = (x_1, x_2, \ldots, x_n)$ is any vector in R^n. In this payoff allocation, x_i is the utility of player i, $i \in N$. An allocation $x = (x_1, x_2, \ldots, x_n)$ is said to *feasible* if $\sum_{i \in C} x_i \leq v(C), \quad \forall\, C \subseteq N$. A payoff allocation is said to be *individual rational* if $x_i \geq v(\{i\}), \quad \forall\, i \in N$. It is said to be collectively rational if $\sum_{i \in N} x_i = v(N)$. A payoff allocation is said to *coalitionally rational* if $\sum_{i \in C} x_i \geq v(C), \quad \forall\, C \subseteq N$. A payoff allocation vector that is individually rational, collective rational and coalitionally rational is said to be an element in the *core* of the game (N, v). The allocation vectors in the core are stable in the sense that no subset of players can deviate from them.

Shapley Value of a Cooperative Game: Let us consider a cooperative game, (N, v) as defined above. Shapley value suggests a fair allocation scheme to the players in the cooperative game in the sense that it takes the marginal contribution of each player into account. The Shapley value, $\Phi_i()$, for each player i is defined [6] as,

$$\Phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(S_i(\pi) \cup i) - v(S_i(\pi))] \tag{1}$$

where Π is the set of permutations over N, and $S_i(\pi)$ is the set of players appearing before the ith player in permutation π. The Shapley value of a player is a weighted mean of its marginal value, averaged over all possible permutations of players.

Note: In special class of cooperative games called convex games, the core is non-empty and the Shapley value belongs to the core.

Convex Games: Now we introduce a special class of cooperative games called *convex games*. The cooperative game (N, v) is called *superadditive* [8] if

$$v(S) + v(T) \leq v(S \bigcup T), \quad \forall S, T \subseteq N \quad with \quad S \bigcap T = \phi. \tag{2}$$

It is called *convex game* [8] if

$$v(S) + v(T) \leq v(S \bigcup T) + v(S \bigcap T), \quad \forall S, T \subseteq N. \tag{3}$$

It is called *strictly convex* if the inequality holds in (3) whenever neither $S \subseteq T$ nor $T \subseteq S$. Another condition [8] that is equivalent to (3) (provided N is finite) is,

$$v(S \bigcup \{i\}) - v(S) \leq v(T \bigcup \{i\}) - v(T) \tag{4}$$

for all $i \in N$ and all $S \subseteq T \subseteq N - \{i\}$. This expresses a sort of increasing marginal utility for coalition membership, and is analogous to the *increasing returns to scale* associated with convex production functions in economics [8].

Shapley Value of a Convex Game: We have explained briefly the Shapley value of a cooperative game in the previous section. If the cooperative game turns out to be a convex game, then it is possible to give a nice interpretation for the Shapley value using the *core* of the cooperative game [8].

Let $|N| = n$ in the cooperative game (N, v) under consideration. Any solution concept suggests a vector with n tuples in an $n-$dimensional linear space E^N with coordinates indexed by the elements of N. If $x \in E^N$ and $S \subseteq N$ we shall often write $x(S)$ to represent $\sum_{i \in S} x_i$. The hyperplane in E^N defined by the equation $x(S) = v(S)$, $0 \subseteq S \subseteq N$, will be denoted by H_S.

A payoff vector $x \in E^N$ is said to be feasible (for $v(.)$) if $x(N) \leq v(N)$. Recall that the core of (N, v) is defined as the set C of all feasible $x \in E^N$ such that $x(S) \geq v(S)$, $\forall S \in N$. The core is obviously a subset of the hyperplane H_N and since the inequalities $x_i \geq v(i)$ are included in the previous set of inequalities, i.e., $x(S) \geq v(S)$, $\forall S \in N$, the core is bounded. Thus, C is a compact convex polyhedron, possibly empty, of dimension at most $n - 1$.

Let ω represent a simple ordering of the players. Specifically, let ω be one of the $n!$ functions that map N onto $\{1, 2, \ldots, n\}$. Define

$$S_{\omega,k} = \{i \in N : \omega(i) \leq k\}, \quad k = 0, 1, \ldots, n. \tag{5}$$

these are called the *initial segments* of the ordering. Thus $S_{\omega,0} = 0$ and $S_{\omega,n} = N$. Consider the equations $x^\omega(S_{\omega,k}) = v(S_{\omega,k})$, $k = 0, 1, \ldots, n$ and it is easy to solve the set of equations to find the coordinates of the intersection of the hyper planes $H_{S_{\omega,k}}$, namely

$$x_i^\omega = v(S_{\omega,\omega(i)}) - v(S_{\omega,\omega(i)-1}), \quad \forall\, i = 1, 2, \ldots, n \tag{6}$$

Thus each ordering ω defines a payoff vector x^ω.

Theorem 1. In convex games, the extreme points of the core are precisely the points x^{ω} [8].

Theorem 2. In convex games, the Shapley value is the center of gravity of the extreme points of the core [8]. That is, $\forall i \in N$,

$$\Phi_i = \frac{1}{n!} \sum_{\omega \in \Omega} x_i^{\omega}. \tag{7}$$

where Ω is the set of all permutations of N.

This gives a nice interpretation of the Shapley value for convex games.

3 Our Solution Approach to Prototype Selection

In this section, we propose our solution approach to the prototype selection problem. We first start with modeling this problem as a convex game.

3.1 Modeling Prototype Selection as a Convex Game (PS-GAME)

Here we consider *the data sets where the clusters corresponding to different classes of patterns are non-overlapping.* In one of the following sections, namely Section 3.4, we consider the *general scenario* where the clusters of patterns corresponding to different classes may be overlapping.

We now give interpretations for the two components of a cooperative game, (N, v), in the context of prototype selection. Let the set of players, N, be the patterns in the given training data set. Now we define the characteristic function, $v(.)$, in the following way. $\forall\ S \subseteq N$, $v(S)$ is *the error rate of the training data set* when only the patterns in S are used as the prototypes. Now it is easy to check that $v(.)$ satisfies the desired condition (4) for convexity for the data sets where the clusters corresponding to different classes are non-overlapping. Hence prototype selection problem can be modelled as convex game and we refer this *PS-GAME.*

3.2 Algorithm-1: An Exact Algorithm for Prototype Selection

Here we propose an exact algorithm for prototype selection that arises naturally as a consequence of the framework that we have developed so far. Due to *Theorem-1*, it is easy to see that *PS-GAME* has non-empty core (since PS-GAME is a convex game). Hence the Shapley value of *PS-GAME* is the center of gravity of the extreme points of the core by means of *Theorem-2.* There are $n!$ extreme points of the core. Now, we can compute the Shapley value (7) of each pattern (i.e. player) with the help of the extreme points of the core. By sorting the patterns in the non-decreasing order of their Shapley values, we can pick and add the patterns one by one to the desired set of prototypes until there is an increase in the error rate. Since this algorithm needs to work with all possible $n!$ permutations of the patterns, its time complexity is $O(\frac{n}{e})^n$ due to Stirling's

approximation. To get around this difficulty, we present an approximate algorithm that runs in polynomial time for prototype selection.

3.3 An Approximate Polynomial Time Algorithm for Prototype Selection

Recall that the Shapley value of *PS-GAME* is the center of gravity of the extreme points of the core. We can compute approximate Shapley values of the patterns by randomly sampling a set of extreme points. Here the cardinality of the sampled set is polynomial in the number of given patterns.

Let us represent the randomly sampled set of extreme points with Ψ. Also let t be the cardinality of Ψ, i.e., $t = |\Psi|$. So we approximate the Shapley value using Ψ. This approach is called *multi perturbation Shapley value analysis* [3]. Note that the size of each permutation in the set Ψ is n. In general the size of a permutation is n. For our approximation, it is sufficient even if we randomly pick d patterns out of available n patterns to form each element of the set Ψ, where d is a positive integer constant and $d << n$. We call each element of Ψ a *partial permutation*. Using such a sampled set Ψ consisting of t number of partial permutations where the size of each partial permutation is d, we present an approximate algorithm to solve the prototype selection problem. We call this algorithm *approximate prototype selection algorithm (APSA)*. Following is the description of *APSA*.

Algorithm 2: APSA

1. Construct a set, Ψ, of size t by sampling from available $n!$ permutations. The size of each partial permutation in Ψ is bounded by size d instead of n. Here $d << n$. The elements of Ψ are partial permutations.
2. For each partial permutation in Ψ of size d, compute the contribution of each of the d patterns using expression (6).
3. Compute the Shapley value of each pattern using expression (7).
4. Sort the patterns in the non-decreasing order of their Shapley values.
5. Pick and add patterns one by one, in the sorted order, to the desired set of prototypes until there is an increase in the error rate of the classification.

Time Complexity of APSA: We now discuss the running time of *APSA*. Here we have to compute the marginal contribution of each pattern corresponding to each partial permutation in Ψ. It takes $O(td)$ time. Note that $t << n$ and $d << n$. Sorting the patterns in non-decreaseing order of their Shapley values requires $O(n\ log(n))$ time. The overall running time of *APSA* is $O(td+n\ log(n))$. Since $t << n$ and $d << n$, note that it runs much much faster than a quadratic time algorithm.

3.4 The General Scenario

In Section 3.1 we assumed that the data sets where the clusters corresponding to different classes are non-overlapping. Here we relax this assumption. That is, the clusters of patterns corresponding to different classes may be over-lapping.

In such scenarios, if we want to formulate a convex game as in Section 3.1, the characteristic function $v(.)$ may not satisfy the convexity condition (4). To get around this problem, we have to relax the conditions (4) for convexity. *Average convex games [2]* and *partially average convex games [2]* essentially address this issue.

The class of average convex games strictly includes the class of convex games [2] and the Shapley value of an average convex is in the core [2]. In case of partially average convex games, they include the average convex games. Moreover partially average convex games need not be super-additive [2]. It is also true that Shapley value for these partially average convex games belongs to the core [2]. Hence in the general scenario of modelling prototype selection problem either as average convex games or as partially average convex games, we can use *APSA* to solve the prototype selection problem.

4 Experimental Results

In this section, we carry out two types of experiments to illustrate the performance of *APSA* on 6 bench mark data sets from UCI repository of machine learning databases [5]. Characteristics of the data sets are shown in Table 1.

Table 1. Description of Experimental Data Sets

Data Set	No. of Features	No. of Classes	Size of Data Set
Glass	9	6	214
Iris	4	3	150
Liver (BUPA)	6	2	345
Pima	8	2	768
Wine	13	3	178
Ionosphere	34	2	351

Experiment Type-1: Comparison with Editing Algorithms: Here we compare the performance of *APSA* with some popular editing algorithms [7] for prototype selection. These algorithms and *APSA* are respectively applied to the same data sets and 5-fold cross validation has been applied on the training data set to obtain the performance results. Each data set has randomly been divided into training and test samples as shown in Table 2.

Two main aspects of our interest to carry out these experiments are *classification accuracy*, and *number of prototypes selected* from the training set. The number of prototypes selected is a direct measure of the computational savings and storage space. Classification accuracy represents the ability of the algorithms to select the most representative prototypes. We present the comparison of *the classification accuracies* of APSA with the editing algorithms [7] in Table 3. Similarly, Table 4 presents the comparison of *the number of prototypes selected* by APSA and by the editing algorithms [7]. In both these tables, the numbers provided for editing algorithms are directly reproduced from [7] (since our experimental setup is the same as that reported in [7]).

Table 2. Data Sets with Corresponding Training and Test Set Sizes for Experimental Comparison of APSA with Editing Algorithms

Data Set	Training Set Size	Test Set Size
Glass	174	40
Iris	120	30
Liver (BUPA)	276	69
Pima	615	153
Wine	144	34

Table 3. Comparison of Classification Accuracies of *APSA* and the Editing Algorithms

	Glass	Iris	Liver	Pima	Wine
APSA	65.5 (3.67)	95.0 (3.08)	69.85 (1.58)	70.5 (3.7)	**76.2** (2.72)
Best k-NCN	**68.0** (5.79)	96.0 (1.34)	70.1 (6.71)	74.1 (2.64)	71.8 (3.99)
Wilson (k = 3)	63.0 (6.20)	**96.7** (2.11)	69.3 (6.24)	72.0 (2.59)	71.8 (8.02)
Repetition (k=3)	64.5 (7.58)	**96.7** (2.37)	**71.4** (8.50)	73.2 (3.67)	72.9 (7.61)
Edited k-NCN (k=3)	67.0 (5.34)	**96.7** (2.11)	68.1 (4.85)	72.2 (3.47)	70.0 (6.81)
Edited k-NCN (k=5)	65.0 (7.07)	**96.7** (2.11)	66.4 (6.94)	73.9 (2.45)	70.0 (9.0)
GG	67.0(5.79)	95.3 (1.64)	69.3 (4.71)	74.1 (3.27)	70.0 (5.39)
RNG	67.5 (6.52)	90.2 (1.34)	68.1 (4.85)	72.0 (2.35)	67.88 (6.86)
All k-NN (k=3)	64.2 (6.29)	**96.7** (2.36)	68.1 (7.39)	71.7 (3.84)	67.7 (5.5)
Depuration (k=3, k'=2)	67.0 (5.12)	96.7 (1.52)	70.3 (7.15)	**75.9** (2.58)	70.6 (11.76)

Table 4. Comparison of the Number of Prototypes Selected by *APSA* and the Editing Algorithms

	Glass	Iris	Liver	Pima	Wine
Original Training Set Size	174	120	276	615	144
APSA	**12.0** (7.48)	**15.2**(8.68)	**26.20**(10.37)	**84**(25.53)	**9.0**(5.36)
Best k-NCN	115.8(5.49)	115.6(0.80)	190.8(4.45)	456.0(10.14)	105.6(2.06)
Wilson (k = 3)	114.8(6.43)	115.4(1.36)	176.0(7.16)	427.8(6.43)	105.6(2.06)
Repetition (k=3)	76.8(43.44)	33.0(47.54)	146.8(54.38)	338.2(47.74)	86.2(15.36)
Edited k-NCN (k=3)	114.6(5.75)	115.6(0.80)	176.8(8.06)	424.0(3.74)	103.8(2.93)
Edited k-NCN (k=5)	109.8(6.68)	116.2(0.75)	185.6(4.45)	438.6(3.07)	104.2(1.94)
GG	105.6(4.32)	115.2(1.17)	190.4(3.93)	472.8(8.42)	110.8(2.23)
RNG	132.2(6.18)	115.0(0.63)	195.2(6.73)	474.6(10.07)	118.0(2.61)
All k-NN (k=3)	111.0(6.24)	114.6(0.89)	145.4(12.95)	363.2(10.13)	92.8(4.15)
Depuration (k=3, k'=2)	142.4(26.34)	35.0(47.53)	232.0(48.39)	403.6(49.55)	109.8(16.66)

In Table 3, and Table 4, the values in brackets correspond to the respective standard deviation. We can observe from Table 4 that *APSA* is performing better than the editing algorithms [7] in terms of the number of prototypes selected. Results from Table 3 suggests that *APSA* can achieve competitive classification accuracies compared with the editing algorithms. Hence we can conclude that our proposed algorithm can select small size prototype set while maintaining comparable classification accuracies.

There are two parameters in the proposed algorithm. They are t and d. Recall that $t << n$ and $d << n$. The values of t and d in these experiments also convey the same message. For Glass data set $t = 10$, $d = 16$, and $n = 174$. Here we can observe that $t << n$ and $d << n$. This observation is true for the remaining data sets as well. For completeness, we provide the values of t and d for the remaining data sets also. For IRIS data set $t = 14$, $d = 21$, and $n = 120$. For WINE data

set $t = 19$, $d = 20$, and $n = 144$. For BUPA (Liver) data set $t = 21$, $d = 49$, and $n = 276$. For PIMA data set $t = 31$, $d = 48$, and $n = 615$.

Experiment Type-2: Comparison with Condensing Algorithms: Here we compare the performance of the *APSA* with some popular condensing algorithms [10] for prototype selection over the four bench-mark data sets reported in Table 1. 10-fold cross validation is used for each experiment. Each data set is divided into 10 sets and each algorithm is given a training set consisting of 9 of the partitions (i.e. 90% of the data), from which it picks the prototype set, S. The other 10% of the data is classified using the prototypes in S. Ten such trails are run for each data set with each algorithm, using a different one of the 10 partitions as the test set for each trail. The average classification accuracy and the average percentage of the number of prototypes selected in each trail are reported in Table 5. The numbers in Table 5 corresponding to the condensing algorithms are directly reproduced from [10] (since our experimental setup is same as that reported in [10]).

Table 5. Comparison of Classification Accuracies of *APSA* and the Condensing Algorithms. The Column (%) indicates percentage of number of prototypes selected.

	Ionosphere	%	Liver (Bupa)	%	Pima	%	IRIS	%
APSA	85.75	5.73	68.58	7.66	73.49	7.02	90.66	11.81
k-NN	84.62	100	65.57	100	73.56	100	94.0	100
CNN	82.93	21.62	56.8	40.87	65.76	36.89	90.00	12.74
SNN	81.74	19.21	57.70	52.59	67.97	42.95	83.34	14.07
IB2	82.93	21.62	56.80	40.87	65.76	36.89	90.00	12.74
IB3	85.75	14.59	58.24	10.66	69.78	10.97	94.67	19.78
DEL	86.32	12.88	61.38	38.36	71.61	12.64	93.33	9.56
DROP1	79.77	3.23	58.24	10.92	65.23	6. 50	84.67	8.59
DROP2	86.6	7.79	67.77	24.77	70.44	17.59	94.67	14.22
DROP3	87.75	7.06	60.84	24.99	75.01	16.90	95.33	14.81
DROP4	86.90	10.60	62.60	32.56	72.53	21.76	95.33	14.89
DROP5	86.90	9.78	65.50	31.08	73.05	21.95	94.0	12.15
ENN	84.04	84.24	61.12	68.15	75.39	76.37	95.33	94.74
RENN	84.04	82.27	58.77	63.13	75.91	74.52	95.33	94.67
All k-NN	84.05	82.18	60.24	52.34	74.88	64.61	95.33	93.78
ELGROW	73.77	0.63	56.74	0.55	67.84	0.29	88.67	2.30
EXPLORE	80.89	0.63	57.65	0.64	75.27	0.29	92.67	2.30

We can see from Table 5 that the percentage of the number of prototypes selected by *APSA* is better than most of the condensing algorithms except ELGROW and EXPLORE. The classification accuracies obtained by *APSA* are better than the condensing algorithms for some data sets and comparable with the remaining algorithms for some other data sets.

5 Conclusions and Future Work

We considered the prototype selection problem in this paper and proposed a cooperative game theoretic based solution. This is a new direction to address the prototype selection problem which is not yet considered in the literature. Experimental results showed the efficacy of the presented approach.

It would be interesting to probe the use of the game theoretic ideas to address the prototype selection problem in case of incremental data sets.

Acknowledgements

We are very thankful to Prof. Y. Narahari for getting valuable advices and cooperation in carrying out this work.

References

1. Cover, T.M.: Nearest neighbor pattern classification. IEEE Transactions of Information Theory 13, 21–27 (1967)
2. Inarra, E., Usategui, J.M.: The shapley value and average convex games. International Journal of Game Theory 22, 13–29 (1993)
3. Keinan, A., Sandbank, B., Hilgetag, C., Meilijson, I., Ruppin, E.: Fair attribution of functional contribution in artificial and biological networks. Journal of Neural Computation 16(9), 1887–1915 (2004)
4. Li, Y., Hu, Z., Cai, Y., Zhang, W.: Support Vector Based Prototype Selection Method for Nearest Neighbor Rules. In: Advances in Natural Computation. LNCS, Springer, Berlin (2005)
5. Murphy, P.M., Aha, D.W.: UCI repository of machine learning databases (1994)
6. Myerson, R.B.: Game Theory: Analysis of Conflict. Harvard University Press (1997)
7. Sanchez, J.S., Barandela, R., Alejo, R., Marques, A.I.: Performance evaluation of prototype selection algorithms for nearest neighbor classification. In: Proceedings of 14th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'01) (2001)
8. Shapley, L.S.: Cores of convex games. International Journal of Game Theory 1(1), 11–26 (1971)
9. Straffin, P.D.: Game Theory and Strategy. The Mathematical Association of America (1993)
10. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learing algorithms. Machine Learning 38(3), 257–286 (2000)