Weighted k-Nearest Leader Classifier for Large Data Sets

V. Suresh Babu and P. Viswanath

Department of Computer Science and Engineering,
Indian Institute of Technology–Guwahati, Guwahati-781039, India
{viswanath,vsbabu}@iitg.ernet.in

Abstract. Leaders clustering method is a fast one and can be used to derive prototypes called leaders from a large training set which can be used in designing a classifier. Recently nearest leader based classifier is shown to be a faster version of the nearest neighbor classifier, but its performance can be a degraded one since the density information present in the training set is lost while deriving the prototypes. In this paper we present a generalized weighted k-nearest leader based classifier which is a faster one and also an on-par classifier with the k-nearest neighbor classifier. The method is to find the relative importance of each prototype which is called its weight and to use them in the classification. The design phase is extended to eliminate some of the noisy prototypes to enhance the performance of the classifier. The method is empirically verified using some standard data sets and a comparison is drawn with some of the earlier related methods.

Keywords: weighted leaders method, k-NNC, noise elimination, prototypes.

1 Introduction

Nearest neighbor classifier (NNC) and its variants like k-nearest neighbor classifier (k-NNC) are popular because of their simplicity and good performance [1]. It is shown that asymptotically (with infinite number of training patterns) k-NNC is equivalent to the Bayes classifier and the error of NNC is less than twice the Bayes error[2,1]. It has certain limitations and shortcomings as listed below. (1) It requires to store the entire training set. So the space complexity is $O(n)$ where n is the training set size. (2) It has to search the entire training set in order to classify a given pattern. So the classification time complexity is also $O(n)$. (3) Due to the curse of dimensionality effect its performance can be degraded with a limited training set for a high dimensional data. (4) Presence of noisy patterns in the training set can degrade the performance of the classifier.

Some of the various remedies for the above mentioned problems are as follows. (1) Reduce the training set size by some editing techniques where we eliminate some of the training patterns which are redundant in some sense [3]. For example, Condensed NNC [4] is of this type. (2) Use only a few selected prototypes from the training set [5]. Prototypes can be selected by partitioning the training set by using some clustering techniques and then taking a few representatives for each block of the partition as the prototypes. Clustering methods like *Leaders* [6], *k-means* [7], *etc.,* can be used to derive the prototypes. (3) Reduce the effect of noisy patterns. Preprocessing the training set and removing the noisy patterns is a known remedy.

With data mining applications where typically the training set sizes are large, the space and time requirement problems are severe than the curse of dimensionality problem. Using only a few selected prototypes can reduce the computational burden of the classifier, but this can result in a poor performance of the classifier. *Leaders-Subleaders* method [5] applies the leaders clustering method to derive the prototypes. The classifier is to find the nearest prototype and assign its class label to the test pattern. While the *Leaders-Subleaders* method reduces the classification time when compared with NNC or k-NNC which uses the entire training set, it also reduces the classification accuracy.

This paper attempts at presenting a generalization over the *Leaders-Subleaders* method where along with the prototypes we derive its importance also which is used in the classification. Further, we extend the method to k nearest prototypes based classifier instead of 1-nearest prototype based one as done in the *Leaders-Subleaders* method. To improve the performance, noisy prototypes, which is appropriately defined as done in some of the density based clustering methods, are eliminated.

The rest of the paper is organized as follows. Section 2, first describes the leaders clustering method briefly and then its extension where the *weight* of each leader is also derived. It also describes regarding the noise elimination preprocessing. Section 3 describes the proposed method, *i.e.,* the k nearest leaders classifier. Section 4 gives the empirical results and finally Section 5 gives some of the conclusions.

2 Leaders and Weighted Leaders

2.1 Notation and Definitions

1. *Classes:* There are c classes *viz.,* $\omega_1, \ldots, \omega_c$.
2. *Training set:* The given training set is $\mathcal{D}$. The training set for class ω_i is $\mathcal{D}_i$, for $i = 1$ to c. The number of training patterns in class ω_i is n_i, for $i = 1$ to c. The total number of training patterns is n.
3. *Apriori probabilities:* Apriori probability for class ω_i is $P(\omega_i)$, for $i = 1$ to c. If this is not explicitly given, then $P(\omega_i)$ is taken to be n_i/n, for $i = 1$ to c.

2.2 Leaders Method

Leaders method [6] finds a partition of the given data set like most of the clustering methods. Its primary advantage is its running time which is linear in the size of the input data set. For a given threshold distance τ, leaders method works as follows. It maintains a set of leaders L, which is initially empty and is incrementally built. For each pattern x in the data set, if there is a leader $l \in L$ such that distance between x and l is less than τ, then x is assigned to the cluster represented by l. In this case, we call x as a *follower* of the leader l. Note that even if there are many such leaders, only one (the first encountered one) is chosen. If there is no such leader then x itself becomes a new leader and is added to L. The algorithm outputs the set of leaders L. Each leader can be seen as a representative for the cluster of patterns which are grouped with it. The leaders method is modified as enumerated below.

1. For each class of training patterns, the leaders method is applied separately, but the same distance threshold parameter, *i.e.,* τ is used. The set of leaders derived for class ω_i is denoted by L_i, for $i = 1$ to c.
2. Each leader l has an associated weight denoted by *weight*(l) such that $0 \leq$*weight*(l) ≤ 1.
3. For a training pattern x which belongs to the class ω_i, if there are p leaders that are already derived such that their distance from x is less than τ, then weight of all these p leaders is updated. Let l be a leader among these p leaders, then its new weight is found by $weight(l) = weight(l) + 1/(p \cdot n_i)$. If $p = 0$, then x itself becomes a new leader whose weight is $1/n_i$.

The modified leaders method called *Weighted-Leaders* is given in Algorithm 1.

Algorithm 1. Weighted-Leaders($\mathcal{D}_i$, τ)

```
$L_i = \emptyset$;
for each $x \in \mathcal{D}_i$ do
  Find the set $P = \{l \mid l \in L_i, ||l - x|| < \tau\}$;
  if $P \neq \emptyset$ then
    for each $l$ such that $l \in P$ do
      $weight(l) = weight(l) + 1/(\mid P \mid \cdot n_i)$.
    end for
  else
    $L_i = L_i \cup \{x\}$;
    $weight(x) = 1/n_i$;
  end if
end for
Output $L_i$ which is a set of tuples such that each tuple is in the form $< l, weight(l) >$ where
$l$ is a leader and $weight(l)$ is its weight.
```

The leaders and their respective weights derived depends on the order in which the data set is scanned. For example, for a pattern x, there might be a leader l such that $||l - x|| < \tau$ which is created in a later stage (after considering x) and hence the weight of l is not updated. By doing one more data set scan these kind of mistakes can be avoided. But since the method is devised to work with large data sets, and as the training set size increases the effect these mistakes diminishes, they are ignored.

We assume that the patterns are from a Euclidean space and Euclidean distance is used.

2.3 Eliminating Noisy Prototypes

Since noise (noisy training patterns) can degrade the performance of nearest neighbor based classifiers, we propose to eliminate noisy prototypes in this section.

A leader l which belongs to the class ω_i is defined as a noisy prototype if (1) the class conditional density at l is less than a threshold density and (2) there are no neighbors for l which are dense (*i.e.,* the class conditional density at each of these neighbors is less than the threshold density). This definition is similar to that used in density based clustering methods like DBSCAN [8] and its variants [9].

This process is implemented as follows. For a given ϵ distance, for each leader l (say this belongs to the class ω_i), we find all leaders from class ω_i which are at a distance less than ϵ. Let the set of these leaders be S. Let the cumulative weight of these leaders be W. Then we say that l is a non-dense prototype if $W < \delta$, for a predefined threshold weight δ. If all leaders in the set S are non-dense, then we say l is a noisy prototype and is removed from the respective set of leaders.

Since any two leaders that belongs to a class are separated by distance of at least τ (the threshold used in deriving the leaders), ϵ is normally taken to be larger than τ. Section 4 describes about how these parameters are chosen.

The process of eliminating noise can take time atmost $O(\mid L\mid^2)$ where L is the set of all leaders. Since $\mid L \mid << n$, where n is the data set size, the noise elimination process will not take much time.

3 Weighted k-Nearest Leaders Classifier

This section describes the proposed classifier. Let L_i be the set of leaders obtained after eliminating noisy leaders for class ω_i, for $i = 1$ to c. Let L be the set of all leaders. That is, $L = L_1 \cup \ldots \cup L_c$. For a given query pattern q, the k nearest leaders from L is obtained. For each class of leaders among these k leaders, their respective cumulative weight is found. Let this for class ω_i be W_i, for $i = 1$ to c.

Let the k nearest leaders are from the region R at q. Approximately class conditional density at q for class ω_i is:

$$\hat{p}(q \mid \omega_i) = \frac{m_i}{n_i \cdot V}$$

where m_i is the number of patterns that are present in the region R that belongs to the class ω_i and n_i is the total number of training patterns for the class ω_i, and V is the volume of the region R. Asymptotically as $n_i \to \infty, m_i \to \infty, m_i/n_i \to 0$ and $V \to 0$, it can be shown that $\hat{p}(q \mid \omega_i) \to p(q \mid \omega_i)$ [1].

It is easy to see that

$$W_i \approx \frac{m_i}{n_i}$$

and hence is proportionate to the $\hat{p}(q \mid \omega_i)$. So, $W_i \cdot P(\omega_i)$ is proportionate to the posterior probability $\hat{P}(\omega_i \mid q)$ where $P(\omega_i)$ is the apriori probability for class ω_i.

The classifier chooses the class according to $argmax_{\omega_i}\{W_1P(\omega_1), \ldots, W_cP(\omega_c)\}$. If $P(\omega_i)$ is not explicitly given then it is taken to be n_i/n where n_i is the number of training patterns from class ω_i and n is the total number of training patterns.

From the above argument, it is clear that the proposed method is approximately doing the Bayes classification as done by the k-nearest neighbor classifier.

The proposed k nearest leader classifier is given in Algorithm 2.

4 Experimental Results

Experimental studies are done with one synthetic data set and two standard data sets, *viz., Covtype.binary* available at the URL: *htpp://www.csie.ntu.edu.tw/ cjlin/libsvmtools*

Algorithm 2. k-Nearest-Leader(L, q)

```
{$L$ is the set of all leaders derived from all classes. $q$ is the query pattern to be classified}
Find $k$ nearest leaders of $q$ from $L$.
Among the $k$ nearest leaders find the cumulative weight of leaders that belongs to each class.
Let this be $W_i$ for class $\omega_i$, for $i = 1$ to $c$.
Class label assigned for $q$ = $argmax_{\omega_i}\{W_1 P(\omega_1), \ldots, W_c P(\omega_c)\}$.
```

/datasets/binary.html and *Pendigits* available at *http://www.ics.uci.edu/mlearn/MLRepository.html*.

A two dimensional *synthetic* data for a two class problem is generated as follows. First class having 60000 patterns were *i.i.d.* drawn from a normal distribution having mean as $(0, 0)^T$ and covariance matrix as $I_{2\times2}$(*i.e.,*identity matrix). Second class also is of 60000 patterns which is also *i.i.d.* drawn from a normal distribution with mean $(2.56, 0)^T$ and covariance matrix $I_{2\times2}$. The Bayes error rate for this synthetic data set is 10%. The data set is divided randomly into two parts consisting of 80000 and 40000 patterns which are used as training and testing sets respectively.

Covtype.binary is a large data set consisting of 581012 patterns of 54 dimensions which belongs to two classes. The data set is divided randomly into two parts consisting of 400000 and 181012 patterns which are used as training and test sets respectively.

Table 1. Synthetic Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time	k	classification time	CA(%)
1-NNC	Nil				1	4692	85.17
k-NNC	Nil				74	7884	89.61
1-NLC	0.06	7947		4.75	1	36.97	73.93
	0.05	10327		6.81	1	53.74	75.52
	0.04	13949		11.98	1	73.9	77.35
	0.03	20164		22.53	1	105.2	79.80
	0.02	31743		46.41	1	151.15	82.43
k-NLC	0.06	7947		4.75	25	152.57	87.42
	0.05	10327		6.81	25	184.71	87.65
	0.04	13949		11.98	25	250.26	88.48
	0.03	20164		22.53	25	357.73	88.92
	0.02	31743		46.4	25	561.84	89.24
wk-NLC	0.06	7947		4.95	25	182.23	89.55
	0.05	10327		7.35	25	240.16	89.56
	0.04	13949		12.50	25	322.75	89.57
	0.03	20164		23.07	25	444.96	89.60
	0.02	31743		47.33	25	714.25	89.50
wk-NLC with noise elimination	0.06	5147	2800	10.61	25	112.97	89.56
	0.05	7067	3260	19.61	25	159.35	89.59
	0.04	10123	3826	37.99	25	233.45	89.62
	0.03	15789	4375	74.95	25	364.46	89.63
	0.02	26784	4959	163.98	25	607.32	89.56

Table 2. Covtype Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time	k	classification time	CA(%)
1-NNC	Nil				1	1373066	94.89
1-NLC	0.3	4129		90.41	1	538.15	68.81
	0.25	6775		168.65	1	880.93	72.76
	0.2	12385		380.57	1	1577.86	77.36
	0.15	25746		1050.15	1	4137.72	82.78
	0.1	63433		3395.42	1	11927.2	88.72
k-NLC	0.3	4129		90.41	23	765.46	70.28
	0.25	6775		168.65	25	1311.98	73.28
	0.2	12385		380.57	10	2177.86	77.87
	0.15	25746		1050.15	7	4623.61	83.58
	0.1	63433		3395.42	3	12527.4	89.62
wk-NLC	0.3	4129		92.79	11	674.90	71.95
	0.25	6775		171.56	7	1011.11	74.75
	0.2	12385		385.67	5	1744.37	78.49
	0.15	25746		1059.80	3	4453.59	84.06
	0.1	63433		3415.32	2	12154.6	90.46
wk-NLC with noise elimination	0.3	3753	376	103.3	11	627.9	72.02
	0.25	6293	482	235.45	7	965.6	74.95
	0.2	11772	613	565.63	5	1684.28	79.14
	0.15	24844	902	1210.9	3	4086.28	84.98
	0.1	61877	1556	3812.85	2	11657.66	91.65

Pendigits is a medium sized data set consisting of 7494 training patterns and 3498 test patterns. The dimensionality is 16 and the number of classes is 10.

The classifiers chosen for the comparative study are: (1) the nearest neighbor classifier(NNC), (2) the k-nearest neighbor classifier(k-NNC), (3) the nearest leader classifier(NLC), (4) the k-nearest leader classifier(k-NLC) and (5) the weighted k-nearest leader classifier(wk-NLC) which is the proposed one in this paper. NLC and k-NLC are similar to NNC and k-NNC, except that, instead of nearest neighbor(s), nearest leader(s) are taken in to consideration. For wk-NLC experiments are done with noise elimination preprocessing and without it.

The experiments are conducted for various leader's threshold *i.e.,* τ values. For *Synthetic* data set the τ values chosen are $\{0.02, 0.03, 0.04, 0.05, 0.06\}$, for *Covtype.binary* data set the τ values chosen are $\{0.1, 0.15, 0.2, 0.25, 0.3\}$, and for *Pendigits* data set these are $\{20, 30, 40, 50, 60\}$. The ϵ value for noise elimination are 0.12, 0.6 and 120 for *Synthetic* data set, *Covtype.binary* data set and *Pendigits* data set respectively. The parameter δ used as weight threshold to eliminate noise is chosen as 5% of the average weight of the leaders in the respective data sets. Similarly, the k value for k-NNC, k-NLC and wk-NLC are chosen from a three fold cross validation from $\{1, 2, \ldots, 40\}$.

The results are summarized in Tables 1, 2,and 3.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. The classification time and classification accuracy(CA) of the

Table 3. Pendigits Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time	k	classification time	CA(%)
1-NNC	Nil				1	302.6	97.74
k-NNC	Nil				3	320.8	97.83
1-NLC	60	385		0.10	1	0.34	91.76
	50	631		0.11	1	0.53	94.19
	40	1165		0.13	1	0.94	95.85
	30	2306		0.19	1	1.78	97.17
	20	4821		0.40	1	4.70	97.48
k-NLC	60	385		0.10	4	0.40	92.28
	50	631		0.11	4	0.63	94.72
	40	1165		0.13	5	1.19	95.11
	30	2306		0.19	3	2.16	96.04
	20	4821		0.40	1	4.70	97.48
wk-NLC	60	385		0.10	2	0.37	93.39
	50	631		0.11	2	0.58	95.48
	40	1165		0.13	3	1.13	95.19
	30	2306		0.19	2	2.20	96.68
	20	4821		0.40	2	5.48	97.57
wk-NLC with noise elimination	60	363	22	0.10	2	0.37	93.37
	50	606	25	0.13	2	0.57	95.43
	40	1133	32	0.21	3	1.09	95.23
	30	2260	46	0.47	2	2.11	96.71
	20	4753	68	1.48	2	5.43	97.57

leader based classifiers depends on the threshold τ. As the value τ reduces, the classification time increases, and also the CA increases. With $\tau = 0$ each distinct training pattern becomes a leader and hence NLC is same as NNC and k-NLC, wk-NLC both are same as k-NNC. With $\tau = 0.03$ for *synthetic* data set, with $\tau = 0.1$ for *Covtype.binary* data set, and with $\tau = 20$ for *Pendigits* data set, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 6%,1% and 2% for Synthetic, Covtype.binary and Pendigits data sets respectively. With noise elimination wk-NLC shows some improvement with respect to classification accuracy over wk-NLC without noise elimination. Also with noise elimination wk-NLC can be slightly faster than wk-NLC without noise elimination.

5 Conclusions

In this paper an improvement over using leaders as prototypes is given. For each of the leaders an associated weight is found which gives its relative importance. These weights are used in the k nearest leader based classifier called weighted k nearest leader classifier. A preprocessing step to eliminate noisy prototypes is also presented. The proposed method is experimentally compared with nearest neighbor classifier(NNC), k nearest neighbor classifier(k-NNC), nearest leader classifier and k nearest leader classifier. With suitable parameters, the proposed method can achieve classification accuracy similar to k-NNC and is superior than the other methods, but is a much fast classifier than k-NNC or NNC. Hence the proposed method is a suitable one to be used with large data sets as in data mining applications.

References

1. Duda, R.O., Hart, E., Stork, P.: Pattern Classification, 2nd edn. John Wiley & Sons, Chichester (2000)
2. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13, 21–27 (1967)
3. Dasarathy, B.V.: Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Los Alamitos, California (1991)
4. Hart, P.: The condensed nearest-neighbor rule. IEEE Transactions on Information Theory IT-4, 515–516 (1968)
5. Vijaya, P., Murty, M.N., Subramanian, D.K.: Leaders-subleaders: An efficient hierarchical clustering algorithm for large data sets. Pattern Recognition Letters 25, 505–513 (2004)
6. Spath, H.: Cluster Analysis Algorithms for Data Reduction and Classification. Ellis Horwood, Chichester, UK (1980)
7. Jain, A., Dubes, R., Chen, C.: Bootstrap technique for error estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 9, 628–633 (1987)
8. Ester, M., Kriegel, H.P., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd ACM SIGKDD, Portand, Oregon, pp. 226–231 (1996)
9. Viswanath, P., Pinkesh, R.: l-dbscan: A fast hybrid density based clustering method. In: Proceedings of the 18th Intl. Conf. on Pattern Recognition (ICPR 2006), Hong Kong, vol. 1, pp. 912–915. IEEE Computer Society, Los Alamitos (2006)