The Generalized Condensed Nearest Neighbor Rule as A Data Reduction Method

Chien-Hsing Chou, Bo-Han Kuo, and Fu Chang
Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
Email: {ister, phkuo, fchang}@iis.sinica.edu.tw

Abstract

In this paper, we propose a new data reduction algorithm that iteratively selects some samples and ignores others that can be absorbed, or represented, by those selected. This algorithm differs from the condensed nearest neighbor (CNN) rule in its employment of a strong absorption criterion, in contrast to the weak criterion employed by CNN; hence, it is called the generalized CNN (GCNN) algorithm. The new criterion allows GCNN to incorporate CNN as a special case, and can achieve consistency, or asymptotic Bayes-risk efficiency, under certain conditions. GCNN, moreover, can yield significantly better accuracy than other instance-based data reduction methods. We demonstrate the last claim through experiments on five datasets, some of which contain a very large number of samples.

1. Introduction

Most data reduction schemes are based on certain prototype learning methods (Devroye et al. [1], chapter 19), which can be divided into two types. The first employs samples as prototypes and are thus called instance-based learning (IBL) algorithms. The CNN rule (Hart [2]) is the original, and perhaps simplest, of many such methods, all of which attempt to extract a subset from an entire set of samples. The common idea of these algorithms is to execute a process iteratively to check the satisfaction of certain criteria for the current set of prototypes, and add or drop prototypes until a stop condition is met. Wilson and Martinez [3] described and compared many algorithms of this type. Methods of the second type can be called clustering-based learning (CBL) algorithms. In these algorithms, prototypes are not samples per se, but are weighted averages of samples. The various proposals for CBL include k-means clustering algorithm, fuzzy c-means algorithm, etc.

In this paper, we focus on IBL algorithms. They have the advantage of extracting prototypes rapidly, since they adopt samples as prototypes and thereby avoid the rather costly computation of clustering. The proposed IBL algorithm is called the generalized condensed nearest neighbor rule, which is similar to CNN but adopts a stronger criterion for the absorption or representation of samples by prototypes. GCNN has the following advantages. First, it incorporates CNN as a special case and can thus outperform the latter. Second, under certain conditions, GCNN is consistent. Third, one of the above conditions requires that any two sets of data with different labels have a positive separation. This is more flexible than the corresponding condition for SVM (Vapnik [4]), which requires a margin between two such datasets (Figure 1). Fourth, GCNN creates prototypes for all labels simultaneously, in contrast to SVM, which creates support vectors for one pair of labels at a time.

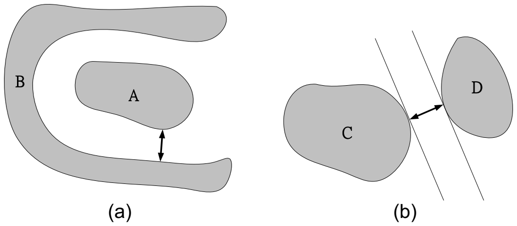

Figure 1. (a) A positive distance exists between two sets A and B. (b) A margin exists between two sets C and D.

The remainder of this paper is organized as follows. In Section 2, we describe GCNN and its relation to CNN. In Section 3, the convergence of GCNN and its consistency under certain conditions is presented. Section 4 contains experimental studies of GCNN and comparisons with two other instance-based algorithms. Finally, in Section 5, we present our conclusion.

2. The GCNN Algorithm

We assume that a set of observed data, or samples, $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)$ is given, where $\mathbf{x}_i$ is a sample and y_i denotes the label of $\mathbf{x}_i$ for $i = 1, 2, \ldots, n$. Let $X_n = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Our goal is to extract a subset U_n from X_n such that if $\mathbf{u}$ is the nearest member of U_n to $\mathbf{x}_i$, then $l(\mathbf{u}) = y_i$, where $l(\mathbf{u})$ is the label of $\mathbf{u}$. Members of U_n are called prototypes and those of $\{(\mathbf{u}, l(\mathbf{u})): \mathbf{u} \in U_n\}$ are called prototype data pairs. Moreover, samples that match in label with their nearest prototypes are said to be *absorbed*.

The CNN rule provides a simple solution to the above problem. Starting with $U_n = \{\mathbf{x}_0\}$, where $\mathbf{x}_0$ is randomly chosen from X_n, CNN scans all members of X_n, and adds to U_n a member $\mathbf{x}$ of X_n whose nearest prototype does not match in label with $\mathbf{x}$. The algorithm scans X_n as many times as necessary, until all members of X_n have been absorbed or, equivalently, no more prototypes can be added to U_n. For convenience, we say that two labeled entities are homogeneous if they have the same label, and heterogeneous otherwise.

We want to apply the GCNN rule in a similar fashion to CNN, but we need to modify the absorption criterion. For CNN, a sample $\mathbf{x}$ is absorbed if

$$\|\mathbf{x}-\mathbf{q}\|-\|\mathbf{x}-\mathbf{p}\|>0, \qquad (1)$$

where $\mathbf{p}$ and $\mathbf{q}$ are prototypes: $\mathbf{p}$ is the nearest homogeneous prototype to $\mathbf{x}$, and $\mathbf{q}$ is the nearest heterogeneous prototype to $\mathbf{x}$. For GCNN, however, we adopt the following criterion:

$$\|\mathbf{x}-\mathbf{q}\|-\|\mathbf{x}-\mathbf{p}\|>\rho\delta_n, \text{ for } \rho\in[0,1), \qquad (2)$$

where $\delta_n = \min\{\|\mathbf{x}_i - \mathbf{x}_j\|: l(\mathbf{x}_i) \neq l(\mathbf{x}_j), \text{ and } \mathbf{x}_i, \mathbf{x}_j \in X_n\}$. We say that a sample is *weakly* absorbed if it satisfies (1), and *strongly* absorbed if it satisfies (2). Note that (1) corresponds to the case when $\rho=0$ in (2). Below, we detail the steps of GCNN.

S1 *Initiation*: For each label y, randomly select a y-sample as a new y-prototype.
S2 *Absorption Check*: Check whether all samples have been strongly absorbed. If so, terminate the process; otherwise, proceed to the next step.
S3 *Prototype Augmentation*: For each y, if there are any unabsorbed y-samples, randomly select one as a new y-prototype; otherwise, no new prototype is added to label y. Proceed to step S2.

3. Consistency of GCNN

Lemma 1. GCNN prototypes satisfy the following properties. (1) For each prototype $\mathbf{p}$, $\|\mathbf{x}-\mathbf{p}\|\geq \delta_n$ for all $\mathbf{x}$ with $l(\mathbf{x})\neq l(\mathbf{p})$. (2) For any two heterogeneous prototypes $\mathbf{p}$ and $\mathbf{q}$, $\|\mathbf{p}-\mathbf{q}\|\geq\delta_n$. (3) For any two homogeneous prototypes $\mathbf{m}$ and $\mathbf{n}$, $\|\mathbf{m}-\mathbf{n}\|>(1-\rho)\delta_n$.

Lemma 2 The number of GCNN prototypes cannot exceed $[(2R_n+\delta_n)/(1-\rho)\delta_n]^d$, where R_n is the radius of the smallest ball containing all samples in X_n. Moreover, the number of GCNN iterations cannot exceed this bound either.

Since $\{\delta_n\}$ is non-increasing, there exists a δ such that $\delta_n \downarrow \delta$ as $n\to\infty$, and δ is the minimal distance between heterogeneous samples. The consistency of GCNN can be proved under the following conditions: (1) all samples are included in a bounded set; and (2) the minimal distance δ between heterogeneous samples is positive.

Theorem 1. Let g_n be the classifier using the 1-NN rule based on GCNN prototype data pairs. The conditions of boundedness and minimal positive distance ensure that for any $\varepsilon>0$,

$$\lim_{n\to\infty}\Pr\{L(g_n)>\varepsilon\}=\lim_{n\to\infty}\Pr\left\{\sup_{g:R^d\to\Lambda}|L(g_n)-L(g)|>\varepsilon\right\}=0. \qquad (3)$$

Note that when a classification rule produces classifiers $\{g_n\}$ in such a way that $\lim_{n\to\infty}\Pr\left\{\sup_{g:R^d\to\Lambda}|L(g_n)-L(g)|>\varepsilon\right\}=0,$ it is said to be consistent.

4. Experimental Results

To test the effectiveness of GCNN, we set up experiments using five datasets as a test bed. In the first experiment, we compare GCNN with two IBL algorithms: CNN and DROP3 [3]. While CNN starts with an empty set and gradually builds prototypes, DROP3 starts with all samples and gradually removes them. In the latter, a sample is removed if at least k of its nearest neighbors can be classified correctly without it. DROP3 also uses a filter to remove noisy samples from the set of prototypes.

Among the five datasets listed in Table 1, “Segmentation” and “Forest” are from the UCI repository [5], “Letter” is from the Statlog collection [6], “UPS” is used in [4], and “Multilingual” is used in [7]. The last dataset contains samples of Chinese, Japanese, and English characters segmented from scanned images of newspapers and magazines. The first three datasets are considered small, as each has less than 20,000 samples; while the other two are considered large, as each contains more than 581,000 samples.

Table 1. Datasets used in the experiments.

Dataset	No. of Samples	No. of Features	No. of Classes
Segmentation	2,310	19	7
Letter	20,000	16	26
UPS	9,298	256	10
Forest	581,012	54	7
Multilingual	880,490	256	3

To evaluate the three algorithms we perform a five-fold cross-validation task on each dataset to obtain the average accuracy rate of the validation data. The computational experiments were conducted on an Intel Pentium 4 CPU 2.2GHz with a 2G MB RAM. For

DROP3, we employ the toolkit developed by Wilson and Martinez [3]. Whenever parameters need to be specified by users, we adopt the default values suggested by the software. For comparison, we also apply the *k*-nearest neighbor (*k*-NN) algorithm to the same datasets and the same five-fold cross-validation task to obtain the average accuracy rate. In the testing process, the three IBL algorithms behave like *k*-NN, in that they determine the label of each test sample based on its *k* nearest prototypes and a voting mechanism. The optimal value of *k* is determined by the same cross-validation.

We first apply GCNN with five values of ρ (ρ=0, 0.25, 0.5, 0.75, and 0.99) to the Letter dataset. This includes CNN as a special case, since GCNN is the same as CNN when ρ=0. The experiment results, presented in Table 2, show: (i) the average accuracy rate, expressed as a percentage; (ii) the training time, expressed in seconds; (iii) the data reduction rate, defined as the proportion of samples that are prototypes; and (iv) the optimal value of *k*. The results show that CNN yields the best reduction rate, but the lowest accuracy rate – 3.8% lower than that of *k*-NN. However, if we apply GCNN and increase ρ from 0 to 0.99, we obtain more prototypes and better accuracy rates. The results clearly show that the additional prototypes improve overall accuracy rates. In fact, when ρ=0.99, we obtain 95.2% accuracy, which is only 0.7 lower than that of *k*-NN, and a reduction rate of 42.6%. We find the same pattern in other datasets, i.e., the higher the value of ρ, the larger the number of prototypes, and the higher the accuracy rates. Thus, hereafter, we only report the GCNN results for ρ=0.99.

Table 2. Results of applying *k*-NN, CNN, and GCNN to the Letter dataset.

Method	Accuracy	Training Time	Reduction Rate	*k*
k-NN	**95.9**	0	100	4
CNN	92.1	**69**	**15.4**	1
GCNN ρ=0.25	93.7	89	22.2	1
GCNN ρ=0.50	94.4	138	28.5	1
GCNN ρ=0.75	94.9	170	35.8	1
GCNN ρ=0.99	95.2	187	42.6	1

Table 3 lists the results of applying the three IBL algorithms to the five datasets and the *k*-NN results for comparison. For each dataset the differences in accuracy between the IBL algorithms and *k*-NN are also shown. Some entries are missing from DROP3 because it needs a huge amount of computation resources and training time for the two large datasets, Forest and Multilingual, and we cannot produce results for them. The results show that CNN requires the least amount of training time and obtains the best reduction rate, and GCNN achieves better accuracy rates than the other two algorithms for all the datasets. We observe that all the differences in accuracy between GCNN and *k*-NN are less than 0.8%. In the Multilingual case, GCNN even outperforms *k*-NN by a slight margin.

We can apply SVM to post-process the set of prototypes produced by CNN or GCNN. To do this, we employ the soft-margin version of SVM and the RBF kernel function (Vapnik [4]). The RBF function involves a parameter γ, whose value range is $\{10^{-a}: a = 0, 1,\ldots, 8\}$. An additional parameter C is used as a penalty factor for the soft-margin version of SVM. Its value range is $\{10^{b}: b = -1, 0, ,1,\ldots, 5\}$. Since SVM only deals with one binary classification at a time, we need a decomposition scheme to apply it to multi-class datasets. We use the one-against-others scheme (Bottou et al. [8]) in our experiment. That is, if there are *n* labels in a dataset, we train *n* SVM classifiers, each of which classifies a sample as *A* or not *A*, where *A* is one of the *n* labels. The toolkit we use in the SVM training and testing steps is LIBSVM (Hsu and Lin [9]).

Table 4 details the results of three hybrid classifiers, Full SVM, CNN+SVM, and GCNN+SVM, in which SVM is applied, as a post-process, to the full datasets and the prototypes constructed by GCNN and CNN, respectively. As expected, CNN+SVM requires the least computing time for training, but its accuracy is substantially lower than that of Full SVM. Meanwhile, Full SVM achieves the best accuracy, but requires an intolerable amount of training time for the two large datasets. GCNN+SVM stands in the middle in terms of both accuracy and training time. Note that for Segmentation, the accuracy of GCNN +SVM is 1.4% lower than that of Full SVM. However, using GCNN alone yields better accuracy than GCNN+SVM (96.4% versus 95.7%), as Tables 3 and 4 show. This suggests that we should not use SVM as a post-process for Segmentation. For Letter and UPS, GCNN+SVM achieves comparable accuracy rates to those of Full SVM, while requiring only 21% and 2% of Full SVM's training time, respectively. For Forest and Multilingual, the accuracy of Full SVM is not available because the training process is too long.

We made the following interesting observation while conducting this experiment. In SVM+GCNN, finding the optimal values of the SVM parameters requires training SVM on GCNN prototypes with various combinations of parameter values. This lengthy process can be reduced if we train SVM on CNN prototypes. The optimal parameter values thus obtained are as accurate as those derived by training SVM on the GCNN prototypes; however, the test accuracy can deviate significantly. To remedy the latter discrepancy, we can further train SVM on the GCNN prototypes, using the optimal parameter values obtained previously. Thus, the training time for SVM+GCNN, as listed in Table 4, is comprised of the CNN training time, the time spent searching for the optimal values of the SVM parameters on the CNN prototypes, and the SVM training time spent on the GCNN prototypes using the optimal values. Full SVM, on the other hand, is applied to the complete datasets in the search for optimal parameter values and the computation of accuracy rates.

5. Conclusion

From theorectical and experimenal works, we find that GCNN serves as a good data-reduction algorithm, since it achieves reasonable reduction rates, as well as accuracy rates close to those for the full datasets. Also, applying SVM as a post-process to the GCNN prototypes can yield better accuracy rates. Note that what is described in this paper is not the most powerful version of GCNN, due to its use of random fashion in picking up seeds for clustering and its insistency of zero training error that can compromise the generalization power of the resultant classifiers. The GCNN in its full version as well as a systematic comparison of such a version with other instance-based learning methods will be reported in forthcoming papers.

References

[1] L. Devroye, L Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer, New York, 1996.

[2] P. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Information Theory*, vol. 14, pp. 515-516, 1968.

[3] D. R. Wilson and T. R. Martinez, "Reduction techniques with instanced-based learning algorithms," *Machine Learning*, 38, pp. 257–286, 2000.

[4] V. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer Verlag, 1995.

[5] C. L. Blake and C. J. Merz, *UCI repository of machine learning databases*, Dept. Inform. Comput. Sci., Univ, California, Irvine, CA, 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[6] D. Michie, D.J. Spiegelhalter and C.C. Taylor, *Machine Learning, Neural and Statistical Classification*. 1994. [Online]. Available: ftp.ncc.up.pt/pub/statlog/

[7] Y.-H. Liu, C.-C. Lin, and F. Chang, "Language Identification of Character Images Using Machine Learning Techniques," *Inter. Conf. Document Analysis and Recognition*, Seoul, pp. 630-634, 2005.

[8] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: A case study in handwriting digit recognition," *Proc. Int. Conf. Pattern Recognition*, pp. 77–87, 1994.

[9] C.-W. Hsu, C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415-425, 2002.

Table 3. Results of applying the three IBL algorithms to the five datasets.

Dataset		Segmentation	Letter	UPS	Forest	Multilingual	Average
Accuracy	*k*-NN	**97.0**	**95.9**	**96.9**	**96.9**	99.95	**97.33**
	GCNN	96.4	95.2	96.7	96.1	**99.96**	96.87
	CNN	95.2	92.1	95.3	95.4	99.83	95.57
	DROP3	95.1	92.4	87.3	------	------	91.6
Training Time	GCNN	3.1	259	350	123,031	301,155	------
	CNN	**0.8**	**74**	**78**	**48,620**	**28,290**	------
	DROP3	395	15,220	6,003	------	------	------
Data Reduction Rate	GCNN	16.2	42.6	44.0	19.2	3.2	25.04
	CNN	**11.2**	**15.4**	12.8	**11.8**	**0.6**	**10.36**
	DROP3	11.5	16.9	**9.9**	------	------	12.77
Differences in Accuracy	GCNN	**-0.6**	**-0.7**	**-0.2**	**-0.8**	**+0.01**	**-0.46**
	CNN	-1.8	-3.8	-1.6	-1.5	-0.12	-1.76
	DROP3	-1.9	-3.5	-9.6	------	------	-5.0

Table 4. Results of applying SVM to the full datasets, GCNN prototypes, and CNN prototypes.

		Segmentation	Letter	UPS	Forest	Multilingual
Accuracy	Full SVM	**97.1**	**97.6**	**97.8**	------	------
	GCNN+SVM	95.7	97.0	97.6	**94.5**	**99.91**
	CNN+SVM	93.7	95.0	97.3	92.7	99.52
Training Time	Full SVM	1,676	30,115	243,834	------	------
	GCNN+SVM	110	6,373	5,858	62 days	375,936
	CNN+SVM	**105**	**6,104**	**5,430**	**60 days**	**68,085**
Differences in Accuracy	GCNN+SVM	**-1.4**	**-0.6**	**-0.2**	------	------
	CNN+SVM	-3.4	-2.6	-0.5	------	------