ELSEVIER

Pattern Recognition Letters 20 (1999) 1347–1352

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Effective supra-classifiers for knowledge base construction

Kurt D. Bollacker, Joydeep Ghosh [*]

Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712, USA

Abstract

We explore the use of the *supra-classifier framework* in the construction of a classifier knowledge base. Previously, we introduced this framework within which labels produced by old classifiers are used to improve the generalization performance of a new classifier for a different but related classification task (Bollacker and Ghosh, 1998). We showed empirically that a simple *Hamming nearest neighbor* is superior to other techniques (e.g., multilayer perception (MLP), decision trees, Naive Bayes, Combiners) as a supra-classifier. Here, we describe theoretically how the probability that the Hamming nearest neighbor supra-classifier will predict the true target class approaches certainty at an exponential rate as more classifiers are reused. The scalability of the Hamming nearest neighbor with large numbers of previously created classifiers makes it a good choice as a supra-classifier in the application of building a repository of domain knowledge organized as a *classifier knowledge base*. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Knowledge reuse; Evolving knowledge base; Hamming nearest neighbor

1. Introduction

1.1. Knowledge reuse via supra-classifiers

Knowledge reuse is the process of applying knowledge embodied in previously learned tasks to help solve a new, related task.

We recently introduced the *supra-classifier* knowledge reuse framework (Bollacker and Ghosh, 1997), which can apply the results of existing classifiers to help solve a new classification task. A schematic of this framework can be seen in Fig. 1. The supra-classifier knowledge reuse process is to present the training samples for a new, current target classification task to all available previously trained classifiers and then use the resulting output vector of classification labels as the input for a second stage supra-classifier. Only label information is used from support classifiers, thus allowing the reuse of any type of classifier architecture. The supra-classifier then makes the final classification decision for the current target classification task τ. Previously trained classifiers, termed *support classifiers*, are generally (but not always) designed for tasks other than the current target classification task of interest.

1.2. Supra-classifier design

Supra-classifiers make classification decisions on a vector of categorical (support class label) input values. Just like any ordinary classifier, they use the training samples for the target classification task (and perhaps a priori information) to make a decision. For an ordinary classifier, the

[*] Corresponding author.
E-mail addresses: kdb@lans.ece.utexas.edu (K.D. Bollacker), ghosh@lans.ece.utexas.edu (J. Ghosh)

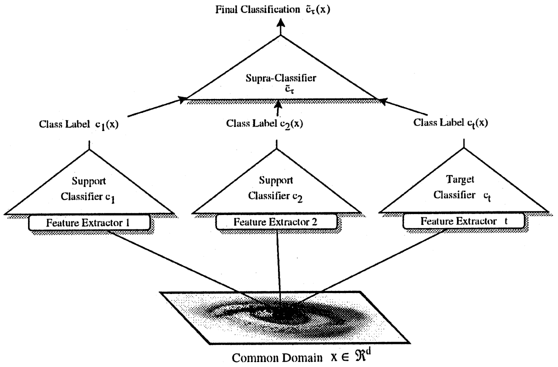

Fig. 1. A *Supra-classifier* based knowledge reuse framework.

feature set is static, and so to improve classification performance, more and/or better training samples are needed. In contrast to this, the premise of the supra-classifier framework is that knowledge can *also* be added by increasing the number of relevant support classifiers (input features). Designing a good supra-classifier is simply designing a classifier for a large number of discrete valued features that works well even with very few training samples. Although several architectures are possible, the Hamming nearest neighbor (HNN) classifier has been shown empirically to outperform decision trees, multilayer perceptron neural networks, Naive Bayes and other types of classifiers in the supra-classifier role. Furthermore, we have theoretically demonstrated HNN supra-classifier scalability to large numbers of support classifiers (Bollacker, 1998). We briefly reiterate here.

Consider a space $\Omega = \Re^d \times \mathscr{S}$ where $\mathscr{S}$ is a set of all possible discrete labels. Let $\Omega_\tau = \Re^d \times \mathscr{S}_\tau$ be a space defined for a classification task τ where $\mathscr{S}_\tau \subset \mathscr{S}$ and $|\mathscr{S}_\tau|$ is finite. $\Re^d$ is the input space where values to be classified exist, and $\mathscr{S}_\tau$ is the set of possible labels for that data for task τ. Let $X_\tau : X_\tau \in \Re^d$ and $Y_\tau : Y_\tau \in \mathscr{S}_\tau$ be the random variables for the prior distribution on Ω_τ associated with task τ. It is assumed that $\forall(x,y) \in (\Re^d, \mathscr{S}_\tau)$, $P(y|x)$ is defined. The most likely value of a class label y, given an input value x is defined as the maximum posterior probability function $t_\tau(x) = \arg\max_y P(Y_\tau = y \mid X_\tau = x)$. Thus, $t_\tau(\cdot) : \Re^d \rightarrow \mathscr{S}_\tau$ is the target function to be approximated by a target classifier. Let $\mathscr{B}$ be a set of support classification tasks which have the same input domain space $\Re^d$ as task τ. For each classification task $b \in \mathscr{B}$, there is an associated set $\mathscr{S}_b$ and classifier $c_b(\cdot) : \Re^d \rightarrow \mathscr{S}_b$. [1] Let the function $\vec{c}(\cdot) = (c_b(\cdot))_{b \in \mathscr{B}}$. For all b, let $C_b : C_b = c_b(X_\tau)$ be the random variables resulting from the application of X_τ to support classifier b and let $\vec{C} = \vec{c}(X_\tau)$. Let $T_\tau : T_\tau = t_\tau(X_\tau)$ be defined as the random variable associated with the target function on X_τ. Assumed in this context is a support classifier set $\mathscr{B}$, which has n members; i.e. $n = |\mathscr{B}|$ and for convenience it is assumed that $\mathscr{B} = \{1, \ldots, n\}$ to allow b to index entries of $\mathscr{B}$.

Using these definitions, the HNN can be defined as a simple classifier for discrete input features (e.g., support classifier labels) similar to traditional nearest neighbor classifiers, which operate in a Euclidean space. If $1(\cdot)$ is the indicator function, then the Hamming distance between two sample input values x_r and x_s can be calculated as

$$D_n(x_r, x_s) = \sum_{b=1}^{n} 1(c_b(x_r) \neq c_b(x_s)),$$

where $c_b(\cdot)$ is the labelling bth support. For each test sample s, the HNN supra-classifier will choose the class label of the training sample with the smallest Hamming distance from it. If the previous classifiers are conditionally independent given the current classification task but are not independent of that current task, then the HNN approaches a perfect classification test rate with an exponential lower bound for an increasing number of previous classifiers. This is more precisely stated in Theorem 1.

Theorem 1. *If the random variables $C_b : b \in \mathscr{B}$ are independent of each other conditionally on the target class Y_τ, $\mu_\infty \geqslant 0$, the μ_n are equal for all values of n, making μ_n a constant, the training samples α, β and test sample γ are value pairs that are randomly and independently drawn trials from the random variable pairs (X_α, Y_α), (X_β, Y_β) and (X_γ, Y_γ), respectively, X_α,*

[1] Although some of the support classifiers may have been trained for task τ directly, in general $b \neq \tau$ and $\mathscr{S}_\tau \neq \mathscr{S}_b$, as the tasks are different.

X_β, X_γ and X_τ are IID, Y_α, Y_β, Y_γ and Y_τ are IID, samples α and β and test sample γ, and $\forall j,k \in \mathscr{S}_\tau$, $p_j = p_k$, then

$$P(D_n(X_\beta, X_\gamma) > D_n(X_\alpha, X_\gamma) \mid Y_\alpha = Y_\gamma, Y_\beta \neq Y_\gamma) \geqslant 1 - (1 - \mu_n^2)^{n/2}. \qquad (1)$$

Note that the only information this bound uses is the mean of the $E[Z_b]$ values and the boundedness of Z_b. As discussed in (Hofri, 1995), a better bound may be achieved by considering higher moments, but in general it is often difficult to do so. Since the Z_b random variables can only take the values $\{-1, 0, 1\}$, they are completely described by their first and second moments, potentially making this more tractable.

Theorem 1 states that as the number of conditionally independent and (at least barely useful) support classifiers increases, the probability that the (weighted) HNN classifier will predict the true target class approaches certainty bounded by an exponential rate. It should be noted that Theorem 1 holds even if there is only one training sample of each target class. This result leads to the observation that *under certain conditions, a wealth of features may substitute for a dearth of samples*. This is counter to the conventional wisdom that classifiers require more training samples when more features are present (Jimenez and Landgrebe, 1998). The trick is in avoiding a Euclidean feature space and the resulting typical curse of dimensionality problems.

2. A classifier knowledge base

A classifier knowledge base is a set of classifiers for a domain of interest. For a sample in this domain, each classifier can provide a label. The resulting set of labels is a description of the sample, and if (at least some of) the classifiers "say something useful" about the sample, the labels can be used to aid in new characterizations of that sample. The new labels that become available when a new classifier is built can be contributed back to the classifier knowledge base. In this manner, each new classifier both takes existing knowledge from the knowledge base and gives some new knowledge back. As the number of classifiers grows, the knowledge base learns its domain better, and can thus provide increasing knowledge to help build new classifiers.

2.1. Experiments in prediction of personal preference

In order to investigate the supra-classifier framework as applied to a classifier knowledge base, an experimental knowledge base on a small, fixed set of images was created. This experimental knowledge base was used to predict whether a user would "like" an image or not, based on previous classifications made by that user about those images. Although this target class is similar to those used in collaborative recommender systems (e.g., Balabanović, 1997; Menczer, 1997), the difference is that in many collaborative filtering systems, only like/dislike classifications are allowed, while in a classifier knowledge base, the previous classifications may be output for any set of labels. Like most collaborative filtering systems, our setup uses humans as support classifiers, making generalization to new images difficult. However, in this experiment we are primarily considering the situation where the data set is important and slow to change. We are trying to predict the user's behavior, not simply an image classification.

A dataset was created consisting of 30 photographic images, mostly acquired from the author's personal collection and a commercial CD-ROM. Some of these images can be seen in Fig. 2. Seventy-three classifications were designed for this set of images. Most of class labels sets were high level, such as "big versus small", "clean versus dirty", "busy versus calm" and "solid versus liquid versus gas". A Web site was created to present the 30 images to six human users (graduate students), who were asked to classify the images in each of 72 ways. These 72 manual (support) classifications can be thought of as a "personal profile" of knowledge about the images for each user. For the 73rd classification, users were asked to decide whether they "liked" each image more than average. This "like/dislike" label was used as the target classification for experiments in supra-classifier testing. One hundred experimental trials each of

Fig. 2. Nine of the 30 images in the image dataset.

several combinations of training set and support classifier sizes were performed for each user and the results were averaged over all the trials and users. For each trial, the 30 images were randomly split into training and test sets, where the training sets ranged in size from 3 to 27. The order of inclusion of the 72 support classifiers was also random for each trial.

2.1.1. Comparative results

Although it is useful for a supra-classifier to perform better when there are more training samples, the most important role of a classifier knowledge base is to be able to perform well when there are only few training samples available for a new task but a large number of previously created classifiers. We compared the HNN supra-classifier with other architectures, including a Naive Bayes (NBAY) classifier, a C4.5 decision tree, a multilayer perceptron neural network (MLP), and a combiner based classifier (VOTE) where each component was a Naive Bayes classifier that saw only one of each of the 72 previous classifications. A most common class (MCC) classifier always chose the most frequent target class in the training set, and was given as a baseline. The test rate of using each type of supra-classifier versus the number of support classifiers in the case of six training samples is given in Fig. 3. The combiner (VOTE) supra-classifier has the best performance when the number of support classifiers was very small, but as the number of support classifiers increased, its performance was reduced. Many of the supra-classifiers had increasing performance with more support classifiers up to about 32, but only the HNN continued to have increased performance up to the full set of 72 support classifiers.

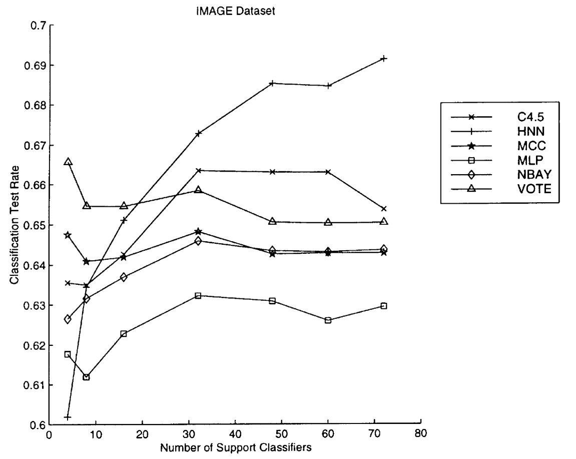

Fig. 3. Average classification test rate versus number of support classifiers for several supra-classifiers for the image dataset.

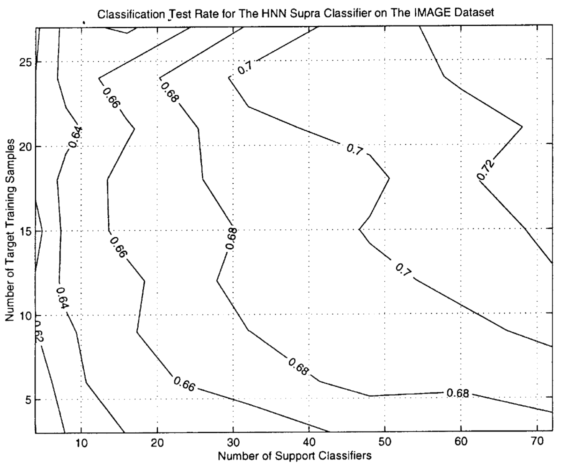

Fig. 4. Classification test rate contours in the knowledge space for the HNN supra-classifier on a classifier knowledge base for the image dataset.

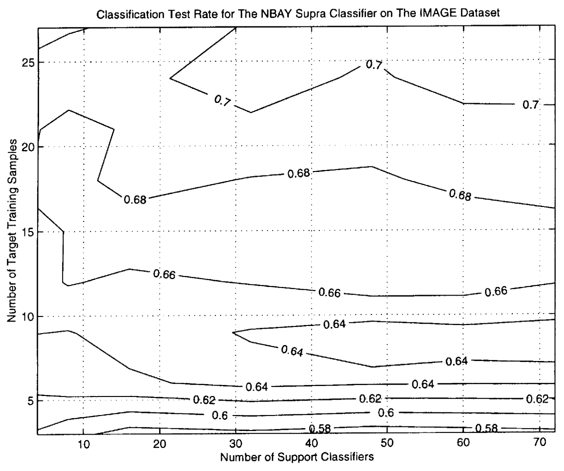

Fig. 5. Classification test rate contours in the knowledge space for the Naive Bayes supra-classifier on experimental knowledge base.

2.1.2. *Results in a knowledge space*

In order to understand the effect of knowledge reuse, consider a 2D *knowledge space* in which classifier performance is measured for a specific number of training samples and support classifiers. The average target classification test rate for the HNN supra-classifier given a range of number of training samples and support classifiers can be seen in the knowledge space represented in Fig. 4. To achieve a given classification test rate, the HNN supra-classifier needed fewer training samples as the number of support classifiers grew. This is evidence that in a knowledge base, the HNN would perform better as the knowledge base grows. Contrast this to the poor performance of a Naive Bayes classifier as a supra-classifier in Fig. 5. Here, the Naive Bayes classifier does not take advantage of support classifiers as they are added, probably because of increasing disparity with the required independence assumption.

3. Conclusions and future work

The supra-classifier framework (and the HNN in particular) is discussed as an approach to the creation of a classifier knowledge base and empirical evidence indicates its effectiveness. A prototype classifier knowledge base has been created and is available on the World Wide Web for testing and public use at http://www.lans.ece.utexas.edu/cgi-bin/cgiwrap/kdb/knowledgebase.

The most important direction for future development in the application of a practical classifier knowledge base is to create a knowledge base that provides some utility to a large number of users. If created with an easily accessible interface like the prototype shown here, a large classifier knowledge base could be built, and further investigations could be performed in the context of a truly practical system. Possible examples of such a system would be an example-queried Web page recommender or a predictor of legislative votes that uses post voting records as a knowledge base. [2]

Discussion

Kanal: I remember from when I was doing these kind of things, that there was an area of decision theory called compound decision theory. The idea of that is to make use of past decisions on similar or dissimilar problems, to come up with an improved decision. You may want to look into that: Compound Decision Theory. In a book that I

[2] Databases of such records are readily available on the Web at such sites as http://thomas.loc.gov

edited in 1966, which was published in 1968, there is a chapter by a student of mine, which introduces this subject briefly. (*Note of the editors: Abend, K., 1968. Compound decision procedures for unknown distributions and for dependent states of nature. In: L.N. Kanal (Ed.), 1968. Pattern Recognition. Thompson Book Company, Washington DC, pp. 207–249*). There is a whole literature and statistical area on compound decision theory. It makes various assumptions on the distributions, but it is still the only theory I know, about how to use past decisions on any problem, improving your decision making on the next problem.

Ghosh: Thanks for the pointer. We actually searched the decision fusion literature, as well as the Web, using certain keywords, to look for a similar concept, but apparently we did not use the right keywords. But I think, what is interesting is that now, because of the Web and faster computers, there are a lot of problems and data sets available where such a theory can be applied.

Raghavan: I just wanted to make a brief suggestion just like Kanal did. You talked about these students who were trying to do all these classifications. We have also, in some work that we did, tried to describe images by interviewing people, using an approach based on a clinical psychologist's work. His name is George Kelly and he introduced a method called "personal construct theory". If you are interested, I can give you the reference. (*Note of the editors: Kelly, G.A., 1995. The Psychology of Personal Constructs. Norton, New York.*) It is an idea that could be useful to extract the kind of class information that you are trying to get.

Ghosh: That would be very helpful. We have actually performed another experiment, where we looked at classifications of words, based on user feedback to help people who are taking GRE. What we found out is that if you just place this feedback facility on the Web and announce it, you do not get a random sample of people who try these methods; you get only those who are very interested and I am not sure how unbiased that distribution is. But I would certainly like to know about Kelly's method.

Duin: The idea of weak classifiers, is that similar to this, are you familiar with that?

Ghosh: Yes, the idea of weak classifiers is similar, yet different. In weak classifiers, you just need to do better than 50% for a given problem and, hopefully, if you have several of such classifiers, then you can combine, using boosting for good performance. But there again, the classifiers are trying to solve the same problem, or some aspect of the same problem, though they might be looking at different training samples. In the case I presented here, the support classifiers are generally solving different problems. So, there is a fundamental difference. It is only similar in the sense that we do not need each classifier to perform very well.

Acknowledgements

The authors would like to thank Ross Baldick, Gustavo De Veciana and Ray Mooney for useful discussions and the Army Research Office contracts DAAH-94-G-0417 and DAAH 049510494, Texas ATP grant #442, as well as the National Science Foundation for sustaining support.

References

Balabanović, M., 1997. An adaptive Web page recommendation service. In: Proc. of the First Internat. Conf. Autonomous Agents, 1997.

Bollacker, K.D., 1998. The supra-classifier framework for knowledge reuse. Ph.D. thesis, The University of Texas at Austin, Austin, TX.

Bollacker, K.D., Ghosh, J., 1997. Knowledge reuse in multiple classifier systems. Pattern Recognition Letters 18 (11–13), 1385–1390.

Bollacker, K.D., Ghosh, J., 1998. A supra-classifier architecture for scalable knowledge reuse. In: The Internat. Conf. on Machine Learning, pp. 64–72.

Hofri, M., 1995. Analysis of Algorithms. Oxford University Press, New York.

Jimenez, L.O., Landgrebe, D.A., 1998. Supervised classification in high-dimensional space: Geometric, statistical and asymptotical properties of multivariate data. IEEE Trans. Systems Man and Cybernetics, Part C: Applications and Reviews 28, 39–54.

Menczer, F., 1997. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In: Proc. 14th Internat. Conf. Machine Learning, pp. 227–235.