KYBERNETIKA — VOLUME 34 (1998), NUMBER 4, PAGES 369 – 374

CASCADING CLASSIFIERS

Ethem Alpaydin [1]
Cenk Kaynak

We propose a multistage recognition method built as a cascade of a linear parametric model and a k-nearest neighbor (k-NN) nonparametric classifier. The linear model learns a "rule" and the k-NN learns the "exceptions" rejected by the "rule." Because the rule-learner handles a large percentage of the examples using a simple and general rule, only a small subset of the training set is stored as exceptions during training. Similarly during testing, most patterns are handled by the rule-learner and few are handled by the exception-learner thus causing only a small increase in memory and computation. A multistage method like cascading is a better approach than a multiexpert method like voting where all learners are used for all cases; the extra computation and memory for the second learner is unnecessary if we are sufficiently certain that the first one's response is correct. We discuss how such a system can be trained using cross validation. This method is tested on the real-world application of handwritten digit recognition.

1. INTRODUCTION

A great percentage of the training cases in many applications can be explained by a simple rule with a small number of exceptions. Our previous experience on handwritten digit recognition [2] shows a small difference in accuracy between linear models and nonlinear multilayer perceptron type neural network models with many hidden units, indicating that digits are almost linearly separable. Instead of finding a complex rule that explains all the cases, our idea is to have a simple, e.g., linear, model that explains a large percentage of the cases, keeping also a list of the cases not covered by the rule as exceptions. This is a multistage pattern recognition approach [5] where inputs rejected by the first stage are handled by a second stage using costlier features or decision making mechanism that is too expensive to use for all inputs.

In Section 2, we discuss how to learn the rule and exceptions. Section 3 gives the results on the application of handwritten digit recognition and Section 4 concludes. The appendix gives an upper bound on the complexity of the exception-learner for the overall system to decrease average risk.

[1] Supported by Tübitak Grant EEEAG-143. The form processing routines are by NIST.

2. LEARNING THE RULE AND EXCEPTIONS

Assume we are given a training set $\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{y}^t\}_{t=1}^N$ of input $\boldsymbol{x}^t \in \Re^d$ and associated class index $\boldsymbol{y} \in \{0,1\}^c$ where $y_i^t = 1$ implies that $\boldsymbol{x}^t \in \omega_i$. We define linear discriminants using the softmax nonlinearity and estimate class posteriors $P(\omega_i|\boldsymbol{x})$ [3]

$$\mu_i(\boldsymbol{x}|\boldsymbol{U}) = \frac{\exp U_i^T \boldsymbol{x}}{\sum_{k=1}^c \exp U_k^T \boldsymbol{x}} \tag{2.1}$$

and find parameters $\boldsymbol{U} = \{U_i\}_{i=1}^c$ that minimize the cross-entropy on the training set

$$E(\mathcal{X}; \boldsymbol{U}) = -\sum_t \sum_i y_i^t \log \mu_i(\boldsymbol{x}^t|\boldsymbol{U}) \tag{2.2}$$

This corresponds to maximizing the log likelihood of the sample under a multinomial logit model $P(\boldsymbol{y}^t|\boldsymbol{x}^t) = \prod_{i=1}^c \mu_i^{y_i}$. Assuming Gaussian density for $p(\boldsymbol{y}^t|\boldsymbol{x}^t)$, the model can also be applied to the case of approximation of continuous functions [1].

Given a validation set $\mathcal{V}$, separate from $\mathcal{X}$ with which we trained the discriminants μ_i, we check if the model is certain of its output. In pattern recognition, a classifier is certain if the highest posterior is higher than a threshold $0 < \theta < 1$. That is for $(\boldsymbol{x}', \boldsymbol{y}') \in \mathcal{V}$, we check if $P(\boldsymbol{y}'|\boldsymbol{x}') > \theta$

$$\mu_i(\boldsymbol{x}') = \max_k \mu_k(\boldsymbol{x}') \text{ AND } y_i' = 1 \text{ AND } \mu_i(\boldsymbol{x}') > \theta$$

If this is not the case, the learned model is not confident and rejects the sample and thus it should be taken as an exception. In this case, we add $(\boldsymbol{x}', \boldsymbol{y}')$ to our table of exceptions $\mathcal{Z}$. When we do this for all patterns in $\mathcal{V}$, learning is over.

During test, for a test pattern $\boldsymbol{x}$, we first check if

$$\mu_i(\boldsymbol{x}') = \max_k \mu_k(\boldsymbol{x}') \text{ AND } \mu_i(\boldsymbol{x}') > \theta$$

If this holds, we choose class ω_i as output otherwise we do k-nn on $\mathcal{Z}$ to find the output.

If a separate validation set is not available, we do k-fold cross-validation to have a division of $\mathcal{V}$ and $\mathcal{X}$. Note that these two sets should be distinct as otherwise with a complex rule we may have high confidence on all data which is misleading; we should get an idea about where the rule can be trusted and this can only be done with data different from the training data.

The linear model is fast and if it is certain for a large percentage of the cases, the overall speed is high. The k-nn is slow due to finding the k closest neighbors but it is only used for cases rejected by the linear model and even when it is used, the k nearest neighbors are searched for in a smaller set. In multistage classification methods [5], classifiers using simpler to extract features are used to recognize well-formed cases before those that use features that are more complex and costly to

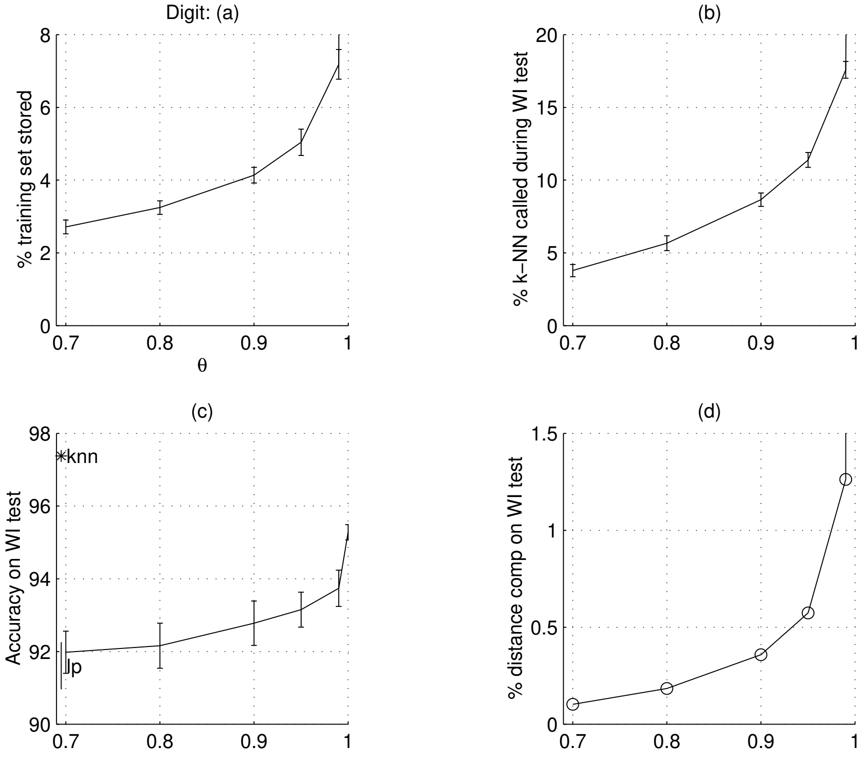

Figure 1: Results by cascading for $\theta \in \{0.70, 0.80, 0.90, 0.95, 0.99, 1.00\}$. (a) % training patterns stored, (b) % k-NN called during test, (c) % accuracy on the WI test set with one standard error bars (linear model 'lp' and 3-NN proper are given for comparison) and (d) % of distance computations made (d=a*c). When $\theta = 1$, (a), (b) and (d) are 100%; this is a simple vote over 'lp' and k-nn proper.

extract are used to recognize patterns of poorer quality. In our approach, it is not the features that get more complex but the classifier.

3. EMPIRICAL COMPARISON

The database we use to test performance contains handwritten digits created using the set of programs made available by NIST [4]. The 32 by 32 normalized bitmaps are low-pass filtered and undersampled to get 8 by 8 matrices where each element is an integer in the range 0 to 16. 44 people filled in forms which are randomly divided into two clusters of 30 and 14 forms. From the first 30, three sets are generated: A training set of 1,934 examples, a validation set of 946 examples and a writer-dependent set of 943 examples. The other 14 forms containing 1,797 examples from distinct writers make up the writer-independent (WI) test set. We use $k = 3$.

With cascading, the number of exceptions during training increase when θ is increased (Fig 1-a). But even for θ as large as 0.99, the exceptions are only 7% showing that the linear model does find a good underlying rule explaining the majority of the cases with sufficient confidence. We also see that this small extra set of stored patterns significantly increases the accuracy on the test set (Fig 1-c). During testing, the slow and cumbersome exception-learner is rarely consulted (Fig 1-b) and even when it is, response is faster because the table is much smaller (Fig 1-d). For example on this dataset, normal k-nn requires 1,934 distance computations for each test character and we have 1,797 WI test characters. With cascading when $\theta = 0.99$, the exception table stores on the average 7% of the cases and only 18% of the test set uses the exception-learner k-NN thus we need $0.07 * 0.18 = 1.3\%$ computations of k-nn proper (Fig. 1-d).

4. CONCLUSIONS

The method we propose, namely the cascading algorithm, is a multistage method which handles a large majority of the cases with the rule found by a simple method, resorting to the more complicated only for the cases that cannot be dealt with by the rule with enough certainty. We think that this is a better approach than multiexpert methods like voting [6] where multiple learners are used for all cases; the extra computation and memory required for the second learner is unnecessary if we are sufficiently sure that the first one's response is correct. The designer can choose between speed and accuracy by selecting the certainty threshold θ fitting best to the constraints set by the application. If high accuracy is required, we suggest to use a high θ value though this uses more memory and is slower. For a fast recognizer, we propose to use a lower θ to handle the large majority by the rule-learner. The optimal θ that balances these partially contradicting aims depends on the application and the losses of actions as given in the Appendix.

In this short paper, we explain the algorithm briefly and cite results on only one application; a more detailed discussion of the method, its variants and its comparison with similar models and applications to other domains is given in [1].

APPENDIX

By $\alpha_i, i = 1, \ldots, c$, we denote assigning input to class ω_i and by α_{c+1} we denote the action of rejecting. Let

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \; i, j = 1 \ldots c \\ \lambda & i = c + 1 \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

where λ is the loss incurred for rejecting ($0 < \lambda < 1$). Then the risk of using the exception-learner is:

$$R(\alpha_{c+1}|x) = \sum_{j=1}^{c} \lambda P(\omega_j|x) = \lambda \tag{2}$$

and the risk of choosing class i with the rule-learner is:

$$R(\alpha_i|x) = \sum_{j \neq i} P(\omega_j|x) = [1 - P(\omega_i|x)] \tag{3}$$

We decide ω_i if $P(\omega_i|x) > P(\omega_j|x)$ for all $j \neq i$, $j, i = 1 \dots c$ and if $P(\omega_i|x) > 1 - \lambda$. Otherwise we reject. θ of cascading is equivalent to $1 - \lambda$ defining the threshold of decision.

We aim finding a bound on the complexity of the second classifier that guarantees decreasing average risk. We follow work done by Pudil et al. [5] here. We are interested in using a second classifier to classify those rejected by the first. This second classifier may use costlier features or a more expensive classification scheme and thus is to be used as rarely as possible. This depends on: (i) How much additional cost the second classifier requires, let's call this c_2, and (ii) How good the second classifier is compared to the first.

If the first classifier does not reject, defining R_i as the region where x is assigned to class ω_i, the average risk is given as (λ_{ij} is short for $\lambda(\alpha_i|\omega_j)$)

$$\mathcal{R} = \sum_{i=1}^{c} \sum_{j=1}^{c} \int_{R_i} \lambda_{ij} P(\omega_j|x) p(x) dx = \sum_{i=1}^{c} \sum_{j=1}^{c} \lambda_{ij} P(\omega_j) \int_{R_i} p(x|\omega_j) dx \tag{4}$$

If we do reject, there is also the additional action of reject ($i = 0$). Defining R'_i as the region where x is assigned to class ω_i after reject

$$\mathcal{R}' = \sum_{i=0}^{c} \sum_{j=1}^{c} \lambda_{ij} P(\omega_j) \int_{R'_i} p(x|\omega_j) dx \tag{5}$$

which can be broken into a sum of making an assignment to one of the classes, $i = 1, \dots, c$ and that of rejecting, $i = 0$

$$\mathcal{R}' = \sum_{i=1}^{c} \sum_{j=1}^{c} \lambda_{ij} P(\omega_j) \int_{R'_i} p(x|\omega_j) dx + \sum_{j=1}^{c} \lambda_{0j} P(\omega_j) \int_{R'_0} p(x|\omega_j) dx \tag{6}$$

We note that we can write R_i as the sum of two regions: Of x that would not be rejected and of x that would be if we used reject. Thus we can write $\mathcal{R}$ as

$$\mathcal{R} = \sum_{i=1}^{c} \sum_{j=1}^{c} \lambda_{ij} P(\omega_j) \left[\int_{R'_i} p(x|\omega_j) dx + \int_{R'_{0i}} p(x|\omega_j) dx \right] \tag{7}$$

R'_i is the region of x that would not be rejected and R'_{0i} is the region of x that would be rejected if we used reject but assigned to class i if we did not reject. $\cup_i R'_{0i}$ contain x that are rejected. For rejection to be useful, $\mathcal{R}' < \mathcal{R}$ or $\mathcal{R} - \mathcal{R}' > 0$ and

$$\mathcal{R} - \mathcal{R}' = \sum_{i=1}^{c} \sum_{j=1}^{c} \lambda_{ij} P(\omega_j) \int_{R'_{0i}} p(x|\omega_j) dx - \sum_{j=1}^{c} \lambda_{0j} P(\omega_j) \int_{R'_0} p(x|\omega_j) dx \tag{8}$$

If we use a second classifier to handle the rejections, we replace λ_{0j} with the risk of that second classifier

$$\lambda_{0j} \equiv \sum_{i=1}^{c} (c_2 + \lambda_{ij}) \int_{R_{2i}} p_2(x|\omega_j)dx = c_2 + \sum_{i=1}^{c} \lambda_{ij} \int_{R_{2i}} p_2(x|\omega_j)dx \qquad (9)$$

c_2 is the constant overhead due to using a second classifier and the second term is the average loss conditioned on x being from ω_j. $p_2(x|\omega_j)$ is the class-conditional probability of the second classifier and R_{2i} are its decision regions.

By replacing Eq. (9) in Eq. (8) and requiring that $\mathcal{R} - \mathcal{R}' > 0$, a bound for c_2 can be found that guarantees decreasing average risk

$$c_2 < \frac{\sum_i \sum_j \lambda_{ij} P(\omega_j) \int_{R'_{0i}} p(x|\omega_j)dx - \sum_i \sum_j \lambda_{ij} P(\omega_j) \int_{R_{2i}} p_2(x|\omega_j)dx \int_{R'_0} p(x|\omega_j)dx}{\sum_j P(\omega_j) \int_{R'_0} p(x|\omega_j)dx} \qquad (10)$$

The first term in the numerator is the risk of not rejecting and the second term is using the second classifier after the first rejects. The second term should be less than the first to satisfy $c_2 > 0$. The denominator is the normalizing term that is the overall probability of reject.

(Received December 18, 1997.)

REFERENCES

[1] Alpaydın, E. 1997. REx: Learning A Rule and Exceptions. International Computer Science Institute TR-97-040 Berkeley, CA, USA.

[2] Alpaydın, E., Gürgen, F. 1995. Comparison of Kernel Estimators, Perceptrons and Radial-Basis Functions for OCR and Speech Classification. *Neural Computing and Applications*, vol. 3, 38–49.

[3] Bishop, C. M. 1995. *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

[4] Garris, M. D., Blue, J. L., Candela, G. T., Dimmick, D. L., Geist, J., Grother, P. J., Janet, S. A., Wilson, C. L. 1994. NIST Form-Based Handprint Recognition System, NISTIR 5469, 1994.

[5] Pudil, P., Novovicova, J., Blaha, S., Kittler, J. 1992. Multistage Pattern Recognition with Reject Option. *11th IAPR International Conference on Pattern Recognition B*, vol. II, 92–95.

[6] Xu, L., Krzyżak, A., Suen, C. Y. 1992. Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, 418–435.

E. Alpaydın, PhD, Associate Professor. Department of Computer Engineering, Boğaziçi University, Istanbul TR-80815 Turkey.

C. Kaynak, MSc. Department of Computer Engineering, Boğaziçi University, Istanbul TR-80815 Turkey.