A LINEAR PROGRAMMING MASKING ALGORITHM FOR PATTERN CLASSIFICATION

Somnath Mukhopadhyay, Department of Information & Decision Sciences, The University of Texas at El Paso, 500 W. University Ave., El Paso, TX 79968-0544, 915-747-7720, smukhopadhyay@utep.edu
Adriano O. Solis, Management Science Area, School of Administrative Studies, York University, 4700 Keele St., Toronto, ON Canada M3J 1P3, 416-736-2100 ext 22239, asolis@yorku.ca

ABSTRACT

The current study seeks to build linear programming (LP) models for dealing with pattern classification problems. The method uses LP to construct and train a higher order LP network. We propose a sequential masking algorithm using different masking functions to classify patterns in data. The study also compares the performance of LP models with neural network and decision tree classifiers.

INTRODUCTION

Pattern Classification

Pattern classification has become useful for a company that competes for new customers and attempts to retain existing ones. It allows companies to make decisions based on facts, rather than purely on management intuition. It holds the key to squeezing the best value out of the huge repositories of information stored in a company data warehouse. The process of separating patterns in data is called *discriminant analysis*. The knowledge gained from discriminant analysis is used to classify patterns for new data. Patterns in data can exist in many different ways. Likewise, there are many different approaches and technologies in use in pattern classification. The current study focuses on the use of linear programming (LP) as a classification tool.

Decision Boundary and Masking in Pattern Classification

The input to a pattern classifier is a set of N measurements, and the output is the classification. The input may be represented by a vector **x**, $\mathbf{x} = <x_1, x_2, \ldots, x_N>$, called the pattern vector. We call $W_{\mathbf{x}}$, the set of all possible values that the vector **x** may assume, the *pattern space*. Let us suppose that there are K possible classes in the problem domain. A classifier will divide $W_{\mathbf{x}}$ into K disjoint regions. The class regions are separated from each other by a decision boundary which could be a linear or nonlinear function. The pattern classification task is to create this decision boundary in the $\mathbf{R}^N$ input space so that a new pattern can be classified in one of those classes.

The parameters of the classifiers are estimated from a set of pattern samples with known classifications, $\mathbf{x}_1$, $\mathbf{x}_2$, ….. , $\mathbf{x}_T$. This set is called the training set. The classifier training approaches optimum performance as T becomes large. Conceptually, any classifier draws boundaries for each class region based on the information in the training set. If a new sample pattern is within the boundary of a specific class, the pattern is assigned to the class. The idea of drawing these decision boundaries around a class during the training phase is called *covering* or *masking*. Any complex, non-convex region can be covered by a set of elementary convex forms of varying size, such as circles or ellipses in the two-dimensional case and spheroids and ellipsoids in the N-dimensional case.

Mangasarian [3] discussed linear and nonlinear separation of data in detail. Roy and Mukhopadhyay [5] used simple fixed quadratic functions for all classes.

THE CURRENT STUDY

The current study will use different combinations of quadratic and radial basis functions. The study will also attempt to use different masking functions for different classes, as well as even within a class. The proposed study will also use pruning techniques to get smaller nets.

LP Masking Algorithm for Pattern Classification

Let S be the maximum number of mask refinement passes, and s be the pass counter. K is the total number of classes and k denotes a class. δ is the lower bound for the normalization constraint variable γ when it is positive and $-\delta$ is the upper bound for γ when it is negative. r denotes the number of in-class patterns (i.e., those belonging to the class being masked) and t the number of out-of-class patterns (i.e., those belonging to classes other than the class being masked) in a mask. A mask is considered to be "small" and pruned if the total number of patterns $(r + t)$ is $\leq \theta$, where θ is a predefined tolerance. In-class patterns are said to be in a majority in a mask if $r/(r + t) \geq \phi$, where ϕ is a predefined tolerance (e.g., $\phi = 2/3$ for a two-thirds majority requirement). M_s is the cumulative number of patterns masked (all classes combined) up to pass s. The relative change in total number of patterns masked from pass s to pass $(s + 1)$ is denoted by IC_{S+1}, where $IC_{S+1} = 100\ (M_{S+1} - M_S)/M_S$. If this relative change is small ($IC_{S+1} \leq \Psi$, where Ψ is a predefined tolerance), the masking process is stopped. NM_S is the total number of masks over all classes in pass s. λ is the error rate of the problem.

The current study proposes to use different masking functions for different classes and/or in different passes. The study will also use a set of different masking functions (Φ functions) for the same class within a pass and pick the one that generalizes the best. The ordering of classes is determined by the analyst. A pattern is never reclassified more than once in this algorithm.

Step 1. Define a set of masking functions to be used.
Step 2. Initialize pass counter: $s = 0$.
Step 3. Let $s = s + 1$. If $s >$ S, stop. Else, $NM_s, = 0$, $M_s = 0$.
Step 4. Initialize class counter: k = 0.
Step 5. Let k = k + 1. If k > K, go to Step 9. Else, initialize the sets G_C, G_{NC} and G_M for class k. G_C consists of unmasked class k patterns, G_{NC} consists of all other unmasked patterns, and G_M consists of all patterns masked in pass s so far.
Step 6. Set up and solve the LP (3)-(8) twice, once for $\gamma > \delta$ and again for $\gamma \leq- \delta$, to mask the remaining unmasked class k patterns.
Step 7. If the generated mask is too small, $r + t \leq \theta$, or if class k patterns in the mask are not in a majority, $r/(r + t) < \phi$, then ignore the mask and go to Step 5 to mask the next class. Otherwise, (a) select the mask with less "noise" for use, (b) reclassify non-class k patterns in the mask as class k patterns, but do not reclassify a pattern more than once, (c) remove the corresponding masked patterns from the sets G_C and G_{NC}, and (d) update M_s and NM_s: $NM_s = NM_s + 1$.
Step 8. If all class k patterns have been masked, go to Step 5 to mask next class. Otherwise, go to Step 6 for next class k mask.

Step 9. If the relative change in total number of patterns masked, IC_S, is less than Ψ and if the total number of masks is unchanged, $NM_S = NM_{S-1}$, then stop. Else, go to Step 3.

Neural Network and Decision Tree Classifiers

One of the popular methods for solving discriminant analysis problems from a prediction point of view is the multi-layered perceptron (MLP), one of the artificial neural network (NN) methods. The most popular brain-like supervised learning algorithm for MLP is the back propagation (BP) algorithm [7]. NN classifiers do not always generalize for many applications when used for classification unless the models are designed properly to fit the application phenomenon [6]. Another difficulty with the NN training algorithm is that in many cases the amount of training data required for convergence is large. For many practical applications, NN models must learn the mathematical function relating correct outputs to inputs with the desired degree of accuracy from a limited number of available data.

There are many decision tree algorithms in the pattern recognition literature. The most popular ones are classification and regression trees [1], chi-squared automatic interaction detection [2], and ID3 family of classification trees [4]. Limitations of decision tree algorithms have been listed [8]. One of the main problems of decision tree classifiers is over-fitting as the algorithm tries to draw decision boundaries surrounding each sample point with hyper-planes (decision boxes or tree leaves). As a result, additional tree pruning algorithms are required to improve generalization performance.

Model Validation and Results

All the classification methods (LP, NN, decision trees) will be rigorously tested using a *ten-fold cross-validation* procedure. This study will first collect data from the well known UCI benchmark repository http://www.kernel-machines.org/ for classification problems. The method will split data sets into ten-fold validation for rigorous testing. The performance of different types of classifiers, with variations on the test data sets, will be documented. Post validation work will include recommendations for various classification problems. Results of the study are to be reported at the time of the conference.

REFERENCES

[1] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. *Classification and regression trees*, Chapman and Hall, 1984.

[2] Kass, G.V. An exploratory technique for investigating large quantities of categorical data, *Applied Statistics*, 1980, *29*, 119-127.

[3] Mangasarian, O.L. Linear and nonlinear separation of patterns by linear programming, *Operations Research*, 1965, *13* (3), 444-452.

[4] Quinlan, J.R. *C4.5 programs for machine learning*, Morgan Kaufmann, 1993.

[5] Roy, A. and Mukhopadhyay, S. Pattern classification using linear programming, *INFORMS Journal on Computing*, 1991, *3* (1), 66-80.

[6] Roy, A. and Mukhopadhyay, S. Iterative generation of higher-order nets in polynomial time using linear programming, *IEEE Transactions on Neural Networks*, 1997, *8* (2), 402-412.

[7] Rumelhart D.E., Hinton G.E., and Williams R.J.. Learning internal representations by error propagation. In *Parallel distributed processing - Explorations in the microstructure of cognition*, D.E. Rumelhart, G.E. Hinton, and R.J. Williams (eds.), MIT Press, 1988.

[8] Safavian, S. and Landgrebe, D. A survey of decision tree classifier methodology, *IEEE Transactions on Systems, Man, and Cybernetics*, 1991, *21* (3), 660-674.