Proceedings of the AI '93 Australian Joint Conference on Artificial Intelligence, pp. 450, 1993.

Learning and Generalization with Bounded Order Critical Feature Sets†

Timothy L. Andersen and Tony R. Martinez

Department of Computer Science, Brigham Young University, Provo, UT

e-mail: tim@axon.cs.byu.edu, martinez@cs.byu.edu

Abstract

It is the case that many real world learning problems exhibit a great deal of regularity. It is likely that the solutions to learning problems which exhibit such regularity can be approximated utilizing only simple (low-order) features gathered from analysis of pre-classified examples. However, little specific work has been done to demonstrate the utility of low-order features. This paper presents methods for gathering low order features from an existing data set of preclassified examples, and using these features for classification of novel patterns from the problem domain. It then conducts experiments using the methods presented on several real world classification problems and reports the results. The results show that pattern classification methods involving low-order feature sets have promise and warrant further research.

I. Introduction

Many of the problems that fall into the area of computerized pattern recognition, such as speech and handwriting recognition, have proven inherently difficult to solve with traditional computing methods. Because of this, several neural, statistical, and machine learning methods have been proposed as solutions to the problems of machine learning and pattern recognition.

The statistical methods used for pattern recognition and classification include nearest neighbor techniques, bayesian estimation techniques, and parametric and nonparametric clustering

†This research was partially funded by grants from WordPerfect Corporation and Novell Inc.

techniques (Afifi and Clark 1990; Dasarathy 1991; Fukun 1990; McLachlan 1992; Schalkoff 1992). Machine learning methods include incremental version space merging, decision trees, and other methods that incorporate meta-knowledge in the learning algorithm (Brazdil 1990; Quinlan 1986; Hirsh 1990). Neural methods are designed to use simple algorithms which are applied over and over again in parallel until the network has successfully learned the problem. In general, neural networks do not require a priori knowledge about problems which they are intended to solve, but rather learn about the problem through the presentation of pre-classified examples from the problem domain. Processing in the net has inherent parallelism, and most nets are capable of varying degrees of self organization (Beale 1990; McCLelland and Rumelhart 1986; Schalkoff 1992).

This paper utilizes a learning method which extracts critical features from an existing data set through direct analysis of the examples contained in that data set. The features are then used to classify novel examples from the problem domain. Since the number of potential features for a given data set can be quite large (it grows exponentially with the number of inputs), it is not feasible to search the entire feature space for critical features. The method proposed in this paper to prune the search space is to limit the order of considered features. In other words, the search through the feature space encompasses only lower-order features. There are a number of advantages to low-order features. Low-order features have a more compact representation than high-order features. Also, low-order features cover more of the input space than high-order features. Because of this, low-order features generally occur more often in the training set than high-order features. It is therefore easier to assess their strength or reliability, and they are less susceptible to noise. On the other hand, high-order features occur less often in the training set (and will therefore be used less often during generalization), are more susceptible to noise, and the amount of search required to discover them can be very large.

The major goal of this paper is to show whether or not satisfactory learning can be accomplished on real world problem domains if the order of stored critical features is bounded. If this can be shown, then it would allow for a large reduction in the size of the search space required to find critical features. In order to do this, this paper first proposes methods which utilize bounded order critical features to perform learning and generalization, then applies these methods to several data sets and analyzes the results. This paper attempts to demonstrate how limiting the order of stored critical features affects the ability of the proposed methods to learn and generalize correctly. In so doing, an assessment of the viability of learning using only low-order features is made.

II. Background

A common approach used in the field of pattern recognition is to try learn about a particular problem by analyzing an existing data set that contains pre-classified examples from the problem domain (called a training set). Each example in this data set has associated with it characteristics or attributes (inputs), and the class to which the example belongs (outputs). The goal then is to find a function which reasonably approximates the correct mapping of inputs to outputs for all possible examples in the problem domain through direct analysis of the examples in the data set.

One approach to this problem has been to search through the possible input pattern space for combinations of input attribute values (or features) which, from examination of the examples in the training set, appear to be strongly correlated with a particular output attribute value (or pattern class), then store these features for future reference. Novel examples are classified by finding matching features and utilizing them to predict an appropriate classification. An example is matched by a feature if the example contains the same combination of input attribute values as the stored feature. Obviously, conflicts between matched critical features can arise. A conflict arises whenever the set of features which an example matches are not all correlated with the same pattern class. Therefor, some method for resolving these conflicts must be implemented. The critical features such a system discovers, the metric it assigns to these features, and the way it resolves conflicts between matched critical features are all integral to how accurately the system will be able to generalize on novel inputs.

The first problem that needs to be addressed in any system which utilizes search to discover critical features is how to perform that search. One possible search method that could be used to gather critical features with such a system would be to perform an exhaustive analysis of all possible combinations of attributes which occur in the training set. First order features could be tested by considering the state of only one of the input variables and how strongly that variable correlates with the expected output, while disregarding all other inputs. Second order features would consider the states of only two of the input variables, and so on. Unfortunately, using an exhaustive search to find critical features is in general an unrealistic approach to feature extraction, since the number of possible features is bounded by S^k, where k is the number of input variables and S is the number of possible states for each input variable (this formula holds if each of the k input variables has the same number of possible states). If a learning problem requires higher order features to be solved, then exhaustively searching the entire input space of that problem is impractical, and the search space must somehow be pruned.

However, if the order of the features searched for is limited to a low number k (i.e. 2 or 3), the time required to complete an exhaustive search of the resultant search space is then polynomialy bounded by k. But if the search space is bounded by placing a restriction on the order of possible features, the main question that arises is whether or not the set of features considered by the learning algorithm will allow the system to learn and generalize upon the problem with a sufficient degree of accuracy? In the arbitrary case the answer to this question is certainly no. If we restrict the order of considered features to k, then we can always come up with an artificial problem that requires features of order $k+1$ (or higher) to solve it. However, due to the high degree of regularity found in many real world learning problems, it is the case that many of these problems do not require the use of higher order features to be solved. If it is true that a large number of real world learning problems can be solved with lower order features, then it is possible to vastly reduce the search space which would otherwise be required to find these critical features.

Besides reducing the search space, there are several other advantages to using lower order features. The first advantage is that they are by nature more general. That is, in general lower order features cover more of the input space (they match more of the possible examples in the problem domain) than higher order features. One of the main goals in any learning problem is to be able to accurately predict classification for examples which have not been seen before. Since lower order features have a higher probability of matching a novel example due to the fact that they cover more of the potential input space, it would appear that they will be matched more often, and will therefore be more useful in predicting classification for novel examples. The second advantage of lower order features is due to the tendency they have of occurring more often in the training set than higher order features. Since lower order features occur more often, it is easier to tell if the correlation's they exhibit with particular pattern classes are valid. If a higher order feature only occurs once or twice in a relatively large training set, it is impossible to tell if the occurrence of that feature is merely a statistical aberration, or a valid predictor of pattern classification.

This paper examines the learning and generalization capabilities of lower order features. In so doing, it describes an algorithm for gathering bounded order features and using them for pattern classification, and it explores three different methods for assigning metrics to extracted features.

III. Methods

Features are gathered through a bounded search of the possible features in each training set. Since the method used for feature extraction is currently designed to handle only multi-state variables, analog inputs contained in the data sets were discretized utilizing a simple histogram equalization technique. The search space is bounded by limiting the order of gathered features to a low number k. For this paper, k equal to 1, k equal to 2, and k equal to 3 were tried. Other than placing an upper bound on the order of stored critical features, no attempt is made to minimize the size of the critical features set.

Each feature is associated with a single output state or pattern classification for a particular output variable. The output state or pattern classification that the feature is associated with is the one with which it has the highest degree of correlation for that output variable. The degree of correlation is determined by analyzing the examples in the training set. This is done as follows: Each feature has a set of counters associated with it. The number of counters that the feature has associated with it is equal to the number of states that the output variable can take. These counters are initially set to zero. The examples in the training set are then examined, and for each example in the training set that matches the feature, the counter which corresponds to the matched example's output state is incremented by 1. Then the output state that has the highest count is said to be the one with which the feature has the highest degree of correlation.

Each feature is also assigned a metric that indicates the relative strength or predictive power of that feature in relation to other features. In other words, the metric indicates how well that feature predicts the output state with which it is associated. While there are a number of possible alternatives for resolving conflicts between matched critical features, this paper uses a simple voting method which works as follows. Each of the matched features is allowed to vote according to the strength of its metric. When a new example is presented to the system, the metrics of all of the features which match the example are summed for each of the possible output states, after which the output state with the highest sum is chosen as that examples most likely classification.

Three different methods are used to calculate each feature's metric The first method is

$$[1] \qquad m = P[y/x]$$

where m is the metric value, x is the feature, y is the output state associated with that feature, and $P[y/x]$ is the probability of y occurring given that x has occurred. This metric does not consider the probability of x (the feature itself) and is therefore likely to find correlation's that exist in the training set that do not exist in the overall data set due to noise and singular events.

The second method for metric calculation is similar to the first, and uses the following formula

$$[2] \qquad m = P[y/x]\, P[x]$$

where m is the metric value, x is the feature, y is the output state associated with that features, $P[y/x]$ is the probability of y occurring given that x has occurred, and $P[x]$ is the probability of x occurring. This method takes into account the probability of the feature x occurring, which can also be viewed as a measure of that feature's generality, and thus favors features which occur more often in the training set over those that occur less frequently.

The first two metric assignment methods assume independence of matched critical features. In order to overcome the problems associated with this assumption, the third method utilizes a simple training algorithm to calculate a metric for each feature. The algorithm utilized to calculate the metrics is in essence the perceptron algorithm developed in the late 1950s (Rosenblat 1958). The algorithm works as follows:

[3] Until convergence (total error < ε)

> Punish matching features that are causing an instance to be incorrectly classified by decrementing their metric.
>
> Reward matching features that could help an instance to be correctly classified by incrementing their metric.

The main drawback of the perceptron algorithm is that it is unable to solve problems which are not linearly separable. However, by considering features of order greater than 1, the input pattern space of an initially non-linearly separable problem can be transformed into a higher-dimensional space in which the problem is linearly separable. Then, by incrementing several times through the training set, the algorithm can find a metric for each feature in the critical feature set that will satisfy a number of simultaneous constraints or inter-dependencies, thus overcoming the assumption of independence of matched critical features. As long as the critical feature set is sufficient and the training set has no inconsistencies, this algorithm will find a metric for each feature which allows the features to correctly predict the output of each training example (that is, the algorithm will converge to a solution). If this is not the case, the perceptron algorithm will not be able to converge to a perfect solution, but will in general still come up with a reasonable approximation to the solution.

IV. Data Sets And Methods

The goal of this paper being to determine the learning and generalization traits of bounded order feature sets, several data sets taken from real world problems, as well as three artificial data sets, are selected to conduct experiments upon. Each data set is randomly divided into two equal parts, a training set and a test set. The data in the training set is used during the training phase of the algorithm to obtain critical features and for metric calculation, while the data in the test set is used to test the generalization capabilities of the extracted features.

Five data sets which have previously been used to test other learning methods were chosen from real world problem domains. The real-valued attributes found in some of the data sets were discretized by performing a histogram equalization of the real-valued attributes and then partitioning the result into a discrete number of equal segments. The data sets chosen were as follows: The breast cancer data set contains 699 examples of cases of breast tumors discovered at University of Wisconsin Hospitals; the credit application (690 examples); the image segmentation data (420 examples); the iris data set (150 examples); and the tic-tac-toe data set (958 examples).

In addition to the five data sets listed above, the three Monk data sets supplied by Sebastion Thrun were also used. For each of these data sets, the training sets used in this paper are identical to the ones supplied by Professor Thrun.

During the training phase of the algorithm a search bounded by feature order for critical features contained in the training set is conducted, and each discovered feature is assigned a metric. After the training phase the test set is used to determine how well the algorithm is able to generalize (predicted classification) on novel patterns, utilizing only those features and feature metrics extracted during the training phase.

For each run the data for each data set is randomly divided into two equal parts. Since selection of the training set can affect the results, 5 runs using different training sets were conducted for each metric on each data set, and the average accuracy for the 5 runs was calculated.

V. Results

The results of the experiment are listed in tables 1 through 8. Each table is divided into 5 columns. The first column headed by "Metric" displays the metric calculation method used for the three rows associated it. The "Order" column displays the upper bound on feature order for that row. The third, fourth, and fifth columns display the percentage of examples correctly categorized by the algorithm for the training set, test set, and total data set respectively. Due to time constraints, the "Trained" or perceptron metric calculation method was limited to 500 iterations through the training set (some data sets that converged to a solution with second order features would not converge with third order features even after 2000 iterations, and going beyond 500 iterations did not appear to improve the results for these data sets).

Metric	Order	Train	Test	Total
P[y/x]	1	96.17	95.24	95.71
	2	98.86	96.45	97.65
	3	99.83	96.22	98.03
P[y/x]P[x]	1	77.77	76.91	77.34
	2	85.83	82.69	84.26
	3	89.31	84.36	86.84
Trained	1	100.00	94.50	97.25
	2	100.00	95.87	97.94
	3	100.00	95.59	97.80

Table 1. Breast cancer results.

Metric	Order	Train	Test	Total
P[y/x]	1	74.20	73.80	74.00
	2	83.13	81.16	82.14
	3	89.86	85.74	87.80
P[y/x]P[x]	1	68.23	69.57	68.90
	2	79.42	79.65	79.54
	3	82.32	81.97	82.14
Trained	1	86.84	81.80	84.32
	2	99.25	83.07	91.16
	3	94.61	86.67	90.64

Table 2. Credit application results.

Metric	Order	Train	Test	Total
P[y/x]	1	64.00	64.95	64.48
	2	75.49	73.00	74.34
	3	94.33	84.13	89.33
P[y/x]P[x]	1	61.43	61.52	61.48
	2	67.00	64.50	65.85
	3	65.97	63.24	64.50
Trained	1	73.90	66.00	69.95
	2	99.90	88.65	94.28
	3	92.57	84.57	88.57

Table 3. Image segmentation results.

Metric	Order	Train	Test	Total
P[y/x]	1	94.13	92.40	93.27
	2	99.07	88.80	93.93
	3	100.00	92.53	96.27
P[y/x]P[x]	1	92.13	90.80	91.47
	2	95.73	87.20	91.47
	3	96.27	89.33	92.80
Trained	1	100.00	91.33	95.67
	2	100.00	87.33	93.47
	3	100.00	90.67	95.33

Table 4. Iris results.

Metric	Order	Train	Test	Total
P[y/x]	1	73.39	64.29	66.90
	2	82.26	69.16	72.92
	3	88.71	72.40	77.08
P[y/x]P[x]	1	64.52	52.60	56.02
	2	78.23	61.69	66.44
	3	84.68	63.96	69.91
Trained	1	70.97	65.91	67.36
	2	100.00	91.95	94.26
	3	99.84	91.17	93.66

Table 5. Monk 1 results.

Metric	Order	Train	Test	Total
P[y/x]	1	62.13	70.34	67.13
	2	62.13	70.34	67.13
	3	63.91	70.34	67.82
P[y/x]P[x]	1	62.13	70.34	67.13
	2	62.13	70.34	67.13
	3	62.13	70.34	67.13
Trained	1	62.13	70.34	67.13
	2	100.00	86.01	91.48
	3	100.00	59.85	75.56

Table 6. Monk 2 results.

Metric	Order	Train	Test	Total
P[y/x]	1	86.07	82.58	83.56
	2	91.80	87.10	88.43
	3	99.18	95.16	96.30
P[y/x]P[x]	1	78.69	67.10	70.37
	2	83.61	78.71	80.09
	3	87.70	80.65	82.64
Trained	1	100.00	98.52	98.94
	2	100.00	97.23	98.01
	3	100.00	96.84	97.73

Table 7. Monk 3 results.

Metric	Order	Train	Test	Total
P[y/x]	1	65.97	64.84	65.34
	2	64.94	65.71	65.37
	3	70.30	68.28	69.19
P[y/x]P[x]	1	65.97	64.84	65.34
	2	64.94	65.67	65.34
	3	65.83	66.47	66.18
Trained	1	89.65	90.28	90.00
	2	100.00	93.69	96.51
	3	100.00	94.37	96.89

Table 8. Tic-tac-toe results.

VI. Discussion

In almost every case the trained metric more accurately predicted classification than the other two metric assignment methods. It would appear from this that the assumption of independence of matched critical features is generally invalid. However, with the less complex data sets even the simple metric assignment method *P[y/x]* did relatively well. The other metric assignment method, *P[y/x]P[x]*, fared the poorest in almost every case. From analyzing the data, it appeared that the quantity *P[x]* was relatively large with some of the more general but less highly predictive features, creating a situation where feature generality overpowered feature/output correlation. This metric could probably be improved by decreasing the importance of $P[x]$ in some fashion (or discarding it altogether for some other measure of generality), and in any case would probably do better in comparison on data sets that contained more noise than the ones used for this paper.

The first two metric assignment methods were often benefited somewhat by the addition of third order features. It is even possible that the continued addition of higher-order features would allow for greater classification accuracy with these two metric assignment methods. However, empirical evidence gathered from simulations with a few of the data sets used in this paper suggests that going to higher order features will at best bring diminishing returns or no return at all. Since the effort required to gather higher order features grows exponentially, the small increase in accuracy which might be gained by going to higher order features is usually not justified.

It is interesting to note that the trained metric, which was in general the most accurate at predicting classification, did not benefit appreciably from the addition of third order features. In fact, with five out of the six data sets the accuracy of the trained metric dropped markedly with the addition of third order features. One possible reason for this is that the algorithm for finding features does not yet make any attempt to minimize the feature list to only those features that are needed to solve the problem. Rather, all possible features bounded by the upper bound on feature order are saved and used by the algorithm. Some of these features are likely to lead the training algorithm astray, causing it to "memorize" the training set rather than learn the desired general concept description.

Although the first two metric assignment methods often did better with the addition of third order features, the best results obtained with these two methods did not significantly surpass the results obtained with the trained metric method on any of the data sets used in this paper. While the trained metric exhibited the best performance, its general lack of benefit from the addition of third order features leads to the conclusion that higher-order features are not necessarily required in order to find a satisfactory solution to many learning problems. And, depending upon the problem to be solved and the learning and generalization methods used, the use of higher order features can actually be detrimental to classification accuracy.

Using the trained metric, and limited to first and second order features, satisfactory solutions were obtained for all of the learning problems studied in this paper. It is likely that even better results could be obtained with feature sets limited to first and second order features by making improvements to the methods for metric assignment and conflict resolution, and by discarding features which are not needed.

VII. Conclusion

The results of the experiments were encouraging. It would appear that the use of lower order feature sets has potential for application in many real world problem domains and warrants further research. Although relatively simple methods were used for feature extraction and metric assignment, surprisingly good results were obtained even on the comparatively more complex problems such as the credit, image segmentation, and monk 2 data sets. The results also demonstrate the large amount of regularity that can be found in real world problems. While it is unlikely that all learning problems found in the real world will exhibit this kind of regularity, it is expected that a great many of them will. If this is indeed the case, then an approach to pattern recognition that limits itself (or highly favors) lower order features is not unreasonable.

Current research is focusing on the following:

- Different methods for feature set minimization.
- More efficient methods for feature extraction.
- Metric assignment methods.
- Other methods for conflict resolution.

VIII. Acknowledgments

The authors would like to thank the following people for supplying the data sets used in this paper: William H. Wolberg for the breast cancer data collected from the University of Wisconsin Hospitals; J. R. Quinlan for the credit application data; The Vision Group, University of Massachusetts for the image segmentation data; R. A. Fisher and Michael Marshall for the iris data set; Sebastion Thrun for the Monk data set; and David Aha for the tic-tac-toe data set.

IX. Bibliography

Afifi, A. A. and Virginia Clark. *Computer Aided Multivariate Analysis.* New York: Van Nostrand Reinhold Company, 1990.

Beale, Russell and Tom Jackson. *Neural Computing*. Bristol: IOP Publishing Ltd, 1990.

Brazdil, Pavel B. and Kurt Konolige. eds. *Machine Learning, Meta-Reasoning and Logics*. : Kluwer Academic Publishers, 1990.

Dasarathy, Belur V. ed. *Nearest Neighbor Norms: NN Pattern Classification Techniques*. Los Alamitos: IEEE Computer Society Press, 1991.

Fukun ga, Keinosuke. *Introduction to Statistical Pattern Recognition*. San Diego: Academic Press, Inc., 1990.

Hirsh, Haym. *Incremental Version-Space Merging: A General Framework for Concept Learning*. Boston: Kluwer Academic Publishers, 1990.

McClelland, James L. and David E. Rumelhart. *Parallel Distributed Processing*. Cambridge: The MIT Press, 1986.

McLachlan, Geoffrey J. *Discriminant Analysis and Statistical Pattern Recognition*. New York: John Wiley & Sons, Inc., 1992.

Quinlan, J. R. Induction of Decision Trees. *Machine Learning*, 1:81-106, 1986.

Rosenblatt, F. The Perceptron: A probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, Vol. 65, No. 6, 1958.

Schalkoff, Robert. *Pattern Recognition: Statistical, Structural, and Neural Approaches.* New York: John Wiley & Sons, Inc., 1992.