Generalizing the Claim Lattice with Applications for Learning by Local Clustering

Robert McCartney* **Dale E. Fish**
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269-3155
robert@cse.uconn.edu fish@cse.uconn.edu

Abstract

A central problem in case-based reasoning systems is that of salience: which features are important in determining similarity, and how might the closeness of situations be evaluated on the basis of their features. In this paper we use a graphical representation of a feature space that represents points in the space as nodes and links adjacent points by edges. This structure supports certain local reasoning within the feature space, allows the salience of features in particular situations to be learned over time, and provides a basis for predicting from situations of limited data.

Introduction

For case-based reasoning to be feasible, it must be possible to find the appropriate cases given a situation. A particularly efficient way to do so is to treat a case as a vector of features, with the assumption that cases whose feature vectors match those derived from a given situation will be useful in that situation. This scheme has a number of drawbacks in practice: if salient features are not represented in the vector, cases that are indexed the same may not in fact be sufficiently similar to be useful; if non-salient features are in the vector, similar cases may be indexed differently; and the feature vector for a given situation may not index any cases.

In this paper, we propose a structure, the feature space graph (FSG), as a solution to this problem. The FSG is a graphical representation of a feature space that represents points in the space as nodes and links adjacent points by edges. The FSG is appropriate for representing a finite number of features that take on a finite or countable number of discrete values. It supports certain local reasoning within the feature space, allows the salience of features in particular situations to be learned over time, and provides a basis for predicting from situations of limited data. We first describe the FSG, and discuss its implementation in a program called Old Hand which plays the two-player card game Set Back.

*This work has been supported in part by the National Science Foundation, grant IRI-9110961

The Graph

The FSG is an extension of the Claim Lattice, a data structure used in HYPO (Ashley 1988) which assigns nodes to a current fact situation (CFS) and retrieves cases on the basis of each node's intersection with the facts in the CFS, considering the partial order induced by the subset relation. There is potentially a node for each element of the power set of the CFS; these are arranged so the top element is the CFS, and each node has its maximal subset nodes as its immediate successors.

In this structure, the successors of the CFS are those cases that differ the least from the CFS in that there are no cases whose intersection with the CFS is a superset of any of these. HYPO takes advantage of this structure to choose the closest cases without having to determine the importance of any feature.

We can build a claim lattice from the power set of the CFS; this "complete" claim lattice will include all of the nodes of the original lattice, and will contain edges from each set to all of its subsets of one less element. In this structure, each edge corresponds to a difference of a single feature; for any node, the set of nodes that are one edge away in any direction consists of all the nodes that differ by one feature, and the minimum distance between any two nodes is equal to the number of feature value differences. This seems like a potentially useful feature in finding related cases–the cases that differ the least should be proximate within the graph–and is the basic assumption behind this work.

The FSG structure we present here is a generalization of the complete claim lattice, differing in the following ways: we remove the order inherent in presence and absence of features, we remove the restrictions that features be binary, and we allow traversal of the structure from arbitrary nodes rather than just the top (since the order is removed, there is no longer a top and the structure is no longer a lattice).
These differences notwithstanding, the FSG is closely analogous to the claim lattice; both are structures for which a fact situation corresponds to a node, and for which the

most closely related fact situations are represented as proximate neighbors or successors. They both have complete analogs corresponding to a complete feature space. All of the claim lattice operations can be done in the corresponding FSG (see (Fish 1994) for details).

The Feature Space Graph

More formally, we can define the FSG in terms of the domain space of feature vectors. For convenience, we start with the complete FSG, then show how the general FSG follows by the process of node merging.

The FSG is based on a fixed-length vector of features, each of which takes on a discrete set of values. There is a node in the complete FSG corresponding to each possible vector. For each feature's set of values, some value pairs are neighbors, so the values within each feature are arranged in a graph on the basis of these relations. These within-feature graphs are expressed in the topology of the FSG as follows: there is an edge between any two nodes if their feature vectors differ by exactly one feature and the differing feature values are connected by an edge in the within-feature graph. It can easily be seen that the undirected analog of the complete claim lattice is an FSG where each feature can take on two values (*present* or *absent*) and that *present* and *absent* are neighbors within each feature.

In general, an FSG is a graph that can be obtained from a complete FSG by a sequence of pair-wise merges. These reduce the number of nodes by combining two neighboring nodes into one node, and reduce the number of edges by eliminating edges between merged nodes and redundant edges (edges between a node and more than one node that has been merged together).

The intuitions behind this structure are as follows. We are interested in some prediction to be made from a point in the feature space (some function of the feature vector), and associate an FSG with that function. If two adjacent nodes give the same prediction, we merge them into a single node; in that case, the difference in the one feature value between the nodes is insignificant in terms of prediction. What we envision is a structure to support local learning: start with a complete FSG, observe the predictive function over time, and merge neighboring nodes for whom the value of the predictive function is equal. This local process of merging adjacent nodes that predict the same outcome should lead to a predictor whose domain is just the set of distinguishable feature vectors; over time the non-salient features will be ignored (that is, vectors that differ only on non-salient features will be associated with the same node in the graph).

In addition, this structure should prove useful in predicting with insufficient data. In a standard CBR approach, we would use the cases indexed by the appropriate FSG node to predict behavior. If the cases among the "exact" matches provide insufficient or contradictory information, we may wish to broaden our set of cases to closely related situations. In the FSG, these situations are those associated with neighboring FSG nodes, so should be easily retrievable via a local search process.

Feature Types and the Resulting FSG

By the way we have defined the complete FSG, its topology is exactly determined by the within-feature topology: the number of nodes and the within-feature graph relating values. We can identify four natural cases of features that seem useful, and from these describe the resulting complete FSG.

The feature space (the nodes in the FSG) is simply the cross product of the feature value sets. The number of elements is the product of the cardinalities of the feature value sets. In practice this may overstate the actual attainable feature space if features are not independent.

For every feature within the vector, we define a within-feature graph based on the neighbor relations among the possible feature values. The connectivity of feature values depends on the nature of the feature. We define four natural feature types where the connectivity of the feature values is implied by the type: binary features (two values, each of which is adjacent to the other), ordered features (values in a total order, neighbors of any value are the adjacent values within the ordering), unordered features (every value is a neighbor of every other value), and cyclic features (similar to ordered features except that the first and last values are also adjacent). More general feature value relationships are also possible, described by any irreflexive symmetric relation on the values, but these provide a natural starting place. The average number of edges going out of each node in the FSG is the sum of the average number of edges going out for each feature; multiplying this average by the number of nodes gives the number of edges, so the number of edges in the complete FSG can be given in terms of the number of values and the number of within-feature graph edges for each feature. The number of nodes and edges in a complete FSG can be calculated from the features as

$$|N| = \prod_{1 \le i \le m} |V_i|$$

$$|E| = \left(\prod_{1 \le i \le m} |V_i| \right) \bullet \sum_{1 \le i \le m} \frac{|E_i|}{|V_i|}$$

where m is the number of features, V_i is the range of values for feature i, and E_i is the set of edges in the within-feature graph for feature i. The number of edges increases faster than nodes; the additive increment $|E_i|/|V_i|$ is 1/2 for binary

features, $(n - 1) / n$ for ordered features, $(n - 1) / 2$ for unordered features, and 1 for cyclic features, where n is the number of values associated with that feature.

The following figures illustrate two FSGs: A simple FSG that consists of two binary features and an ordered three-value feature (Figure 1) and two binary features and an unordered four-value feature (Figure 2).

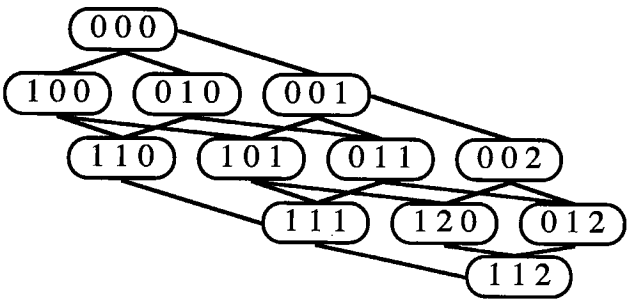

Figure 1: Complete FSG for two binary features and one ordered feature with three values.

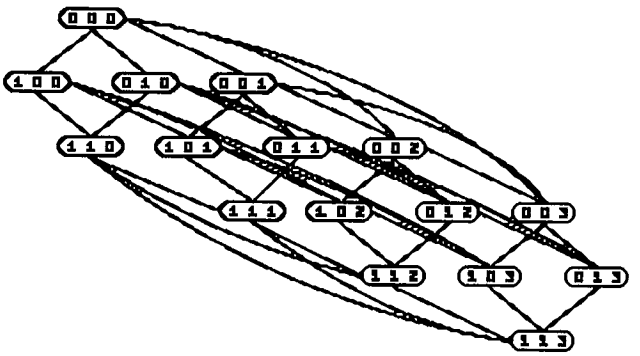

Figure 2 Complete FSG for two binary features and one unordered feature with four values.

Merging Nodes

The complete FSG provides us with a useful structure for case-based reasoning as each node in the FSG corresponds to a possible vector and can be used to index each case corresponding to that feature vector. The edges of the FSG allow accessing of similar cases by locally traversing from the node of interest to its neighbors. Many of the FSG nodes may not be interesting for a given problem however: cases that differ only in non-salient features might better be grouped and indexed by the salient features only. To attain this grouping with the FSG, we propose a merging mechanism whereby neighboring nodes that are not different in the context of the problem of interest are considered as one, and used to index the same cases from a number of different feature vectors.

We start with a complete FSG. The context of the FSG is a function from the feature space to a set of responses (which might be probabilistic) that we would like to predict. The purpose of merging is to combine neighboring nodes in the FSG for which the function is equal, retaining all of the predictive information with a smaller graph. This is equivalent to determining that differences in certain features (over a certain range in certain parts of the feature space) have no effect on the function of interest.

Merging can be implemented as an incremental process: take observations of the function of interest and associate this information with the appropriate FSG nodes. As information increases at a node, test the hypotheses that the function is the same for the node and each of its neighbors; if the information is sufficient to say that the function is equal for two neighboring nodes, those nodes are merged into a single node. This new node is indexed by all of the feature vector values that indexed either of the pair of nodes being merged, and has as immediate neighbors all of the immediate neighbors of the merged nodes. Over time, the FSG reflects the information gained about the relationships between feature vector values in the context of the function.

For example, consider Figure 3, which is an FSG obtained from the one in Figure 1. Suppose we determined that the nodes (0 0 1), (0 1 0), and (0 1 1) have the same response, *i.e.* the associated feature vectors do not distinguish the situations. We obtain Figure 3 by merging these three into a single node, reducing the number of nodes and edges in the FSG by two and three respectively. This merging has the effect that whenever we are interested in predicting from either (0 0 1), (0 1 0), or (0 1 1) we predict from a node having more information.

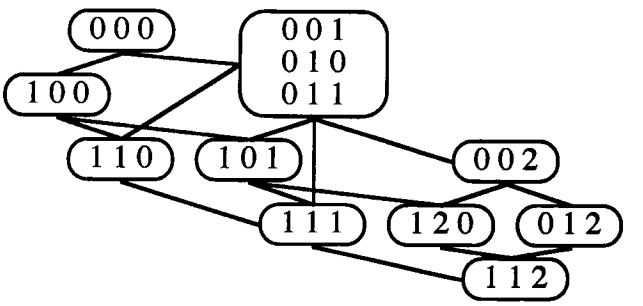

Figure 3. FSG after nodes (0 0 1), (0 1 0), and (0 1 1) are merged.

The existence of a merge operation does not preclude the existence of an unmerge operation: one that allows nodes that were aggregations of other nodes to be split into groups on the basis of differences in the predictions as information increases. We did not implement such an operation, but empirical testing (see Results) suggests it would be useful. Allowing such reformulation would require more information to be stored about individual vectors within a merged node.

In addition to merging nodes over time, it may be useful to temporarily cluster groups of nodes. This can be a useful strategy to deal with insufficient or contradictory evidence– if the cases that match the situation are not sufficient to lead to a decision, consider closely related cases as well. In the FSG, these related cases are stored at adjacent nodes in the graph so their information can be easily combined with that from the matched cases when necessary.

Testing the FSG on Old Hand

The FSG has been implemented and tested in a computer program called Old Hand which plays the two player card game Set Back, which is a bidding game where players bid on the strength of their hands for the right to choose the trump suit, then are required to score a certain number of points in the play of the hand. Old Hand has two players: a procedural player that uses hard coded bidding and playing rules, and a lattice player that uses FSGs to store and access case information used in bidding (the lattice player uses procedures for playing).

Rather than go through the details of how Old Hand operates, we will summarize. The system starts with a set of potential features, for which it generates an FSG with no associated cases. For each hand, the lattice player uses the information in the FSGs to bid. If the node corresponding to the current hand does not have enough case information to predict the outcome of a bid, the information from the adjacent nodes is added in, clustering nodes until there is sufficient experience (or no more nodes). After each hand, the node corresponding to the hand is updated as to success or failure. Periodically, the system scans the FSG and merges any adjacent nodes for which the data are sufficient to say that the success ratios are equal. The system continues playing hands and collecting statistics about the merges and success rates. For details, see (Fish 1994).

Results

Old Hand has been run numerous times with the lattice players using various feature sets. Lattice players typically bid poorly and erratically early and fairly well after enough hands have been played. They also make reasonable bids even when a hand is seen for the first time by using the neighbors of the initial node.

The effects of merging over time are often interesting. When an inconsequential feature is included, the FSG will frequently collapse entirely along that dimension. The FSGs eventually stabilize, but the rate at which this occurs is dependent on the size of the original graphs (as well as the number of observations needed at each node before merging is allowed and the number of non-salient features). Figure 5 shows the average number of nodes vs. number of hands for three different sets of features (FSGs with 56, 14, and 8 nodes to start). In each test, we ran 2000 hands and checked for possible merges every 100 hands. Merged nodes were often simpler than a disjunction of the original nodes (a superfluous feature will collapse along that dimension, for example). In some cases, a merged node can be distilled to a single feature value.

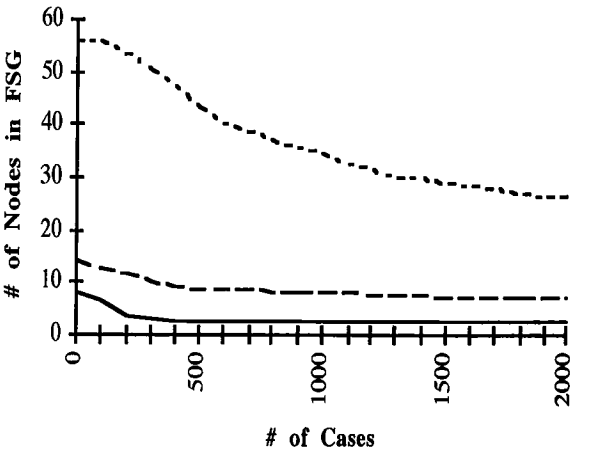

Figure 5. Number of nodes in FSG over time for initial FSG node numbers of 56, 14, and 8.

One observed result is that sometimes nodes that are distinguishable in the long run will merge, and then their combined observations are sufficiently different from their neighbors that the new node will not merge again even if a merge is appropriate. Suppose the expected response for nodes A and B is .2 and the expected response for nodes C and D is .4, but early on B and C are merged. In the long run, the combined B-C may never merge, which suggests either a more conservative merge strategy or the possibility of unmerging.

Related Clustering and Similar Structures

HYPO, a case-based reasoning system that works in the domain of trade secret law, uses a structure similar to the FSG, and was discussed earlier.

A case space representation scheme described in (Farley 1991) for the purpose of case-based classification represents cases as a vector of positive integers with each element of the vector representing some dimension of information. Our scheme generalizes this representation in that it can be implemented using our approach as an FSG of ordered features. Our incorporation of other feature types with their implied connectivity is arguably suitable for a wider range of domains.

Clustering is the partitioning of data into subset whose in-class members are similar and whose cross-class mem-

bers are dissimilar. In this work, we use clustering in two ways: whenever we merge nodes, and when we include data from neighboring nodes when faced with insufficient information; in any case, we cluster adjacent nodes in the graph. In (Jarvis & Patrick 1973) a clustering technique is described which uses the metric of shared nearest neighbors. Two data points are judged to be similar if their k-nearest neighbor lists match. Although in the FSG adjacent nodes will tend to share nearest neighbors (at least each other), it is possible that nodes with identical nearest-neighbor lists may not be adjacent, as in the case of an FSG created by a feature set of two binary features. An agglomerative clustering technique is described in (Gowda & Krishna 1978) based on mutual neighborhood value which indicates the mutual nearness of two points (as opposed to the number of shared neighbors in (Jarvis & Patrick 1973)). In this approach, two points are mutual nearest neighbors if they both are each others nearest neighbor, as opposed to only one of the points being nearer to the other than any other point. This is similar to the merging that takes place between indistinguishable neighbors in the FSG (those that differ the least are merged), but the merge is based on both the proximity (all neighboring nodes are mutual nearest neighbors) and the associated observations.

The COBWEB system (Fisher 1990) is similar to Old hand in that it is a general purpose incremental clustering method motivated by the maximizatioin of inference ability. However, conceptual clustering judges class quality by the quality of the concepts that describe them (e.g., simplicity of concepts) and the resultant class structure is hierarchical. Old Hand clusters cases solely on the basis of their behavior or performance and the FSG is flat. Although this often yields simple class descriptions, reclustering would be motivated by diverging performance and not the quality of the concepts.

Conclusions and Possible Extensions

We have presented a generalization of the claim lattice that is promising for problems where the cases can be indexed by a reasonably small set of feature vectors, and where the salience of the features is not completely understood. The FSG will support local merging, and can be used to learn where feature differences are not significant locally; ultimately this may lead to a global understanding of which feature sets can be used to efficiently characterize cases.

Preliminary empirical work with Old Hand suggests that the FSG (with non-salient features) will become smaller at a fairly rapid rate, but that a liberal merge policy may lead to an incorrect structure relative to distinguishing feature vectors. More systematic study is required to determine the effects of the number of significant features versus insignificant features, aggregation strategies to deal with insufficient data, and so forth.

In the future, we would like to develop and evaluate "unmerge" mechanisms to distinguish subsets of feature vectors that are combined within an aggregate node, with the eye toward a sequence of merge-do nothing-unmerge decisions modifying the FSG. This could be used to deal with the kinds of early over commitment problems that we have seen empirically. Alternatively, rather than successively testing for the equality of nodes, we could use sequential tests that test the three-possibility hypothesis same-different-no decision; once a decision is reached that the nodes are the same, we merge as before, but if we get enough information to determine that the nodes are different we could forego further comparisons. Another direction to explore is methods for assigning variable weights to the edges in the graph so that there is a better distance metric for distance of neighbors. An important first step could be to compare the behavior of the FSG using edge counts as a distance measure between nodes and the use of Euclidean distance between the associated vectors when clustering.

References

Ashley, K., and Rissland, E.L. 1987. Compare and Contrast, A Test of Expertise. *Proceedings of AAAI-87,* 273-278. Seattle, WA.

Ashley, K. 1988. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* Technical Report 88-01, Dept. of Computer and Information Science, Univ. of MA.

Farley, A.M. 1991. *Case-Based Classification: A New Approach.* Technical Report #CIS-TR-91-16, Department of Computer and Information Science, University of Oregon, Eugene, Oregon.

Fish, D.E. 1994. *A Dynamic Memory Organization for Case- Based Reasoning Supporting Case Similarity determination and Learning Via Local Clustering.* M.S. Thesis, Dept. of Computer Science and Engineering, Univ. of Connecticut.

Fisher, D.H. 1990. Knowledge Acquisition Via Incremental Conceptual Clustering. *Readings In Machine Learning*, 267-283. Morgan Kaufman, San Mateo, CA.

Kolodner, J.L. 1993. *Case-based Reasoning.* Morgan Kaufmann, San Mateo, CA.

Jarvis, R.A., and Patrick, E.A.. 1990. Clustering Using a Similarity Measure Based on Shared Near Neighbors. In Dasarathy, B.V., Ed. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques,* 388-397. IEEE Computer Society Press, Los Alamitos, CA.

Gowda, K. C, and Krishna, G. Agglomerative Clustering Using The Concept of Mutual Nearest Neighborhood. In Dasarathy, 398-405. *Op. cit.*