International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:7, 2007

Genetic Programming Based Data Projections for Classification Tasks

César Estébanez, Ricardo Aler, and José M. Valls

***Abstract*—**In this paper we present a GP-based method for automatically evolve projections, so that data can be more easily classified in the projected spaces. At the same time, our approach can reduce dimensionality by constructing more relevant attributes. Fitness of each projection measures how easy is to classify the dataset after applying the projection. This is quickly computed by a Simple Linear Perceptron. We have tested our approach in three domains. The experiments show that it obtains good results, compared to other Machine Learning approaches, while reducing dimensionality in many cases.

***Keywords*—**Classification, Genetic Programming, projections.

I. Introduction

THE idea of projecting data spaces into other, more relevant, spaces in order to improve classification tasks has been widely used under many names. For instance, Support Vector Machines implicitly project data into a high number of dimensions (even infinite) by means of kernel functions, so that they are more easily separable [1]. In other cases, projections are used to reduce the number of dimensions, and in many cases, to improve classification accuracy (Fisher Linear Discriminant, Principal Component Analysis,...). Similarly, projections can construct relevant attributes from low-level attributes or to reformulate the pattern recognition problem by constructing more relevant features (feature induction or constructive induction [2, 3, 4]). These new features can be either added to the original attribute set, or replace them.

However, most projections are closed-forms (linear, polynomic, . . .). It would be interesting to obtain the most appropriate projection for the case at hand, given a set of primitives. In this paper, we have used Genetic Programming (GP) to do so [5]. GP is a stochastic population-based search method devised in 1992 by John R. Koza. It is inspired in Genetic Algorithms, being the main difference with them the fact that in the later, chromosomes are used for encoding possible solutions to a problem and making them evolve until converging to a valid solution. GP, nevertheless, proposes the idea of evolving whole computer programs. Within the scope of Evolutionary Algorithms, it exists a main reason for using GP in this problem: A projection is, in essence, a mathematical formula and so, its size and structure are not defined in advance. Thus, finding a codification that can fit a GA is a difficult problem. GP, nevertheless, does not impose restrictions to the size of evolved structures. There is another reason for using GP: its results are sometimes surprising, and may find some projection a human programmer might not think about. Finally, an advantage of GP is that some domain knowledge can be injected by selecting relevant primitives, whereas other Machine Learning methods use a predefined, unmodifiable set (neurons in NN, attribute comparisons in ID3,...).

In this paper, we present a GP-based method for finding projections that increase, leave equal, or decrease the data space dimensions, so that classification in the projected space is improved. Fitness is determined by computing the degree of linear separation of data in the projected space. This has been implemented as a Linear Perceptron. We believe that, although more powerful classification methods (like C4.5, SVM, or NN) could be used, choosing a predictor with few degrees of freedom is an important decision: if we let GP to evolve any projection, and in addition, we use powerful classification scheme that can separate the projected data using complex surfaces, there is a large risk of overfitting. Of course, there are other ways of preventing overfitting, both in GP and in the classification method, but we prefer to try the simplest approach first. In addition, using simple methods means that fitness computation will be fast which is important in evolutionary computation. Also, other simple classification methods (like nearest neighbor) could be used, and will be tested in the future.

The structure of this paper is as follows. Section II describes the approach. Then, Section III applies the method to several domains. Next, Section IV reports on the related work. And finally, Section V draws some conclusions and describes possible future research directions.

II. Description of the Method

We will learn from a set E of n examples expressed in a space U of N dimensions. Our objective is to be able to represent the examples in the space V, of P (projected) dimensions, and in which the examples will be linearly separable. P can be larger, equal, or smaller than N.

Manuscript received July 15, 2005. This article has been financed by the Spanish founded research MCyT project TRACER, Ref: TIC2002-04498-C05-04M.

C. Estebanez, R. Aler and J. M. Valls are with the Department of Computer Science, Universidad Carlos III de Madrid, Av. de la Universidad 30, 28911, Leganés (Madrid), (e-mail: cesteban@inf.uc3m.es, aler@inf.uc3m.es, jvalls@inf.uc3m.es).

Anyway, the use of projections does not exclude the possibility of using this method with a set of examples expressed in a space U in which they already are linearly separable. In this case, our method can generate projections that take the examples to a space V of a smaller number of dimensions than U but maintaining linear separability. Thus, our method can have two different applications: on one hand, the improvement of classification tasks by means of a transformation of the dataset (towards higher, equal, or lower dimensionality); on the other hand, the reduction of dimensionality by constructing new attributes that are as good, at least, as the original ones. Of course, any combination of both applications fits our approach.

Our method uses standard GP to evolve individuals made of P subtrees (as many as dimensions of the projected space V). Fitness is computed by measuring the degree of linear separation after applying the individual to the original data (in fact, projecting from U to V). The system stops if a 100% linear separation has been achieved or if the maximum number of generations is reached. Otherwise, the system outputs the individual that separated better the training data.

For the implementation of our application, we have used Lilgp 1.1, the software package for Genetic Programming developed in Michigan State University by Douglas Zongker and Bill Punch, members of the group GARAGe (Genetic Algorithms Research and Applications Group) (http://garage.cse.msu.edu/).

1) Terminal and function set.

In our problem, terminal set will be formed by the attributes of the problem expressed in coordinates of U (u_0, u_1, ..., u_N), and by the so-called Ephemeral Random Constants, which are randomly generated numerical constants that the program can use.

The set of functions to use is difficult to determine: it must be sufficient for, along with the set of terminals, being able to express the solution to the problem, but, on the other hand, they must not be too many as for uselessly increase the search space. Of course, for different domains, different terminal and function sets will be more appropriate. We consider that the fact that they can be chosen is an advantage of GP over other methods. At this point, we have tested some generic sets, appropriate for numerical attributes:

- Basic arithmetical functions: +, -, *, and /
- Trigonometric Functions: sine, cosine, tangent, arcsine, arccosine and arctangent.
- Square and square root.

2) GP Individuals

Instead of having individuals work with vectorial data and return a vector of P dimensions, every individual will contain P subtrees, using the same set of functions and terminals, that will be ran independently. Thus, a projection is going to consist of a series of trees labeled (v_0, v_1 ..., v_M) that represent combinations of all the terminals (u_0, u_1 ..., u_N) and functions. Actually, we use the lilgp mechanism for ADF (Automatically Defined Functions). That is, an individual is made of P ADF's and no main program. It is the fitness function that calls each one of the independent (non-hierarchical) ADFs. It is important to remark this issue because crossover is homologous, in the sense that subtree vi from individual a will cross with subtree vi of individual b. This makes sense, because if different features in V are independent and even orthogonal, subtrees in v_i will not be useful for subtrees in v_j, and vice versa. If it is suspected that different features might share some code, the standard ADF approach (i.e. ADFs common to the P main subtrees) would be more efficient [6]. We will test the ADF approach in the future, but we believe that it is better to separate both approaches conceptually and experimentally.

3) The fitness function

We already have introduced the basic mechanism of the fitness function. It takes the examples expressed in space U, projects them using the GP individual, and obtains a point in space V with P coordinates. Next, a classification algorithm is applied to the projected data. In this case, we have chosen to apply a Simple Linear Perceptron. Adaline or a Fisher Linear Discriminant could have also be applied, but the SP is fast and stable enough. We have preferred to use simple classification schemes in order to avoid overfitting: if both GP projections and the classification scheme have a lot of degrees of freedom, overfitting should be expected. The Perceptron is run for 500 cycles (experimentally we have checked that this is more than enough). If the SP converges, the projection would be producing a linear separation of the data and it would be the solution to the problem. If the SP does not converge, the fitness assigned to the individual is the number of examples that the SP has been able to correctly classify in the best cycle: if projected data is not linearly separable, the SP will oscillate. Storing the best value guarantees stability of the fitness value. This way, fitness measure is gradual enough and has the resolution necessary to be able to exert a real selective pressure. Pseudocode of the implemented fitness function is shown next:

```
evaluate_fintness (individual) {
   for every instance i in the training data do
      read an instance expressed in U coordinates
      assign values to the terminals u0,u1,..,un
      express the instance in coordinates of V:
         examplesV[i].v0 = evaluate_tree
            (individual.tree[0]);
         examplesV[i].v1 = evaluate_tree
            (individual.tree[1]);
         ...
         examplesV[i].vM = evaluate_tree
            (individual.tree[M]);
   classification_hits = perceptron (examplesV);
   fitness = classification_hits;
   return fitness;
}
```

III. Experimental Validation

In this paper, we have applied the approach to three domains: the first one is a simple synthetic domain with a

known solution. We have included this domain in this article because it helps to check the system and to visualize the results. The second one is a synthetic domain, the Ripley data set, widely used as a benchmark. Finally, the third one is the well-known, real-world domain Pima-Indians diabetes data set from UCI database.

1) Synthetic Domain

In order to verify the correct operation of our method and to make it more comprehensible, we have decided, as a previous step, to apply it to a toy domain composed of two datasets. In this domain the direct solution is known and the solution given by our method can be easily verified. Datasets Ellipse and EllipseRT were created with this purpose. Both are two-class classification problems with 1000 two-dimensional points. In dataset Ellipse, the examples belonging to class 0 are situated inside an ellipse that is centered in the origin, and whose focuses are placed at points (-10,0) and (10,0). Class 1 instances are situated outside the ellipse. Dataset ElipseRT is similar, but the ellipse has been rotated and translated, being its focuses located at points (10, -10) and (1,7).

We ran our application on the data set Ellipse with the following parameters: G = 500; M = 5000; function set = {+, -, *, /, SQR, SQRT}. The number of dimensions selected for the projected space V is 2 in this case, due to considering them sufficient for a so simple problem.

The graphical representation (not shown here) shows an almost perfect linear separability of projected data. A Simple Perceptron on the projected data obtains 100% accuracy.

The same process was followed with dataset ElipseRT. Parameters for the execution stay the same, but this time, due to the greater complexity of the problem, the dimension of the projected space is 3. A Simple Perceptron applied to it obtains 99,9% accuracy, which means that data has also been separated almost linearly. Fig. 1 displays the projected data. Points belonging to the inside of the ellipse appear blacker and placed in the bottom of the valley-like distribution, whereas points belonging to the outside appear grey, in the rest (upwards) of the figure.

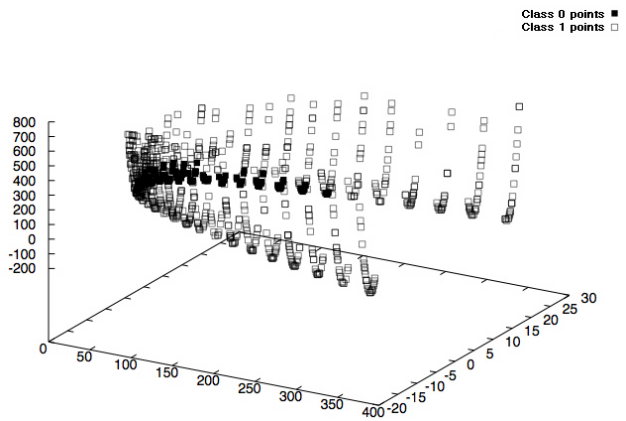

Fig. 1 Projected data for the rotated and translated ellipse. Two classes: black circles and grey squares

2) Ripley Data Set

This artificially generated dataset was used in [7]. Each pattern has two real-valued co-ordinates and a class that can be 0 or 1. Each class corresponds to a bimodal distribution that is a balanced composition of two normal distributions. Covariance matrices are identical for all the distributions and the centers are different. The training set has 1000 patterns and the test set has 250. This domain is interesting because there is a big overlap between both classes and the number of test examples is much bigger than the number of training patterns. On this domain, we have projected the data from its original two-dimensional space into a three-dimensional one were this data can be more easily classified. Five GP-runs were carried out. In all of them, GP has run for 350 generations.

TABLE I
CLASSIFICATION RESULTS ON THE RIPLEY DATA SET

Exp.	Pop. Size	Train C. rate (%)	Test C. rate (%)
1	1000	91.98	92.11
2	1000	**94.81**	**94.74**
3	250	95.75	86.84
4	500	94.34	84.21
5	250	91.04	78.95

Fig. 2 and Fig. 3 Ripley's data before and after applying the projection (the projection used is the best GP individual obtained in GP-run 2). The projected space has been projected itself to 2 dimensions (simply ignoring one coordinate), for visualization purposes. It can be observed that data can be, almost, linearly separated.

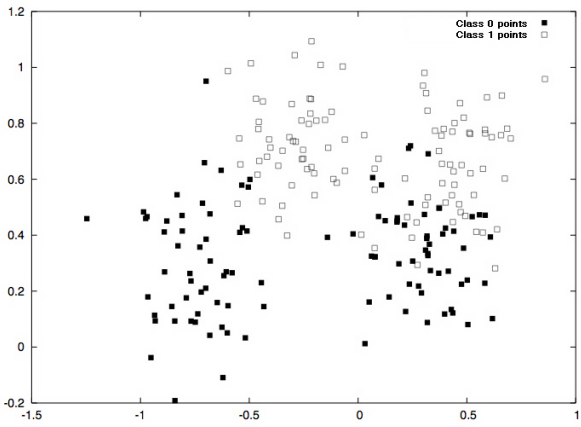

Fig. 2 Original Ripley's Data. Two classes: filled squares and empty squares

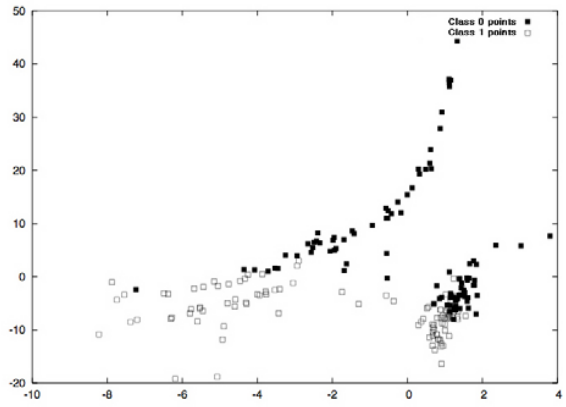

Fig. 3 Projected Ripley's Data in 2-D. Two classes: filled squares and empty squares

For comparison purposes, we have applied to this particular domain some well-tested tools from the Weka package. Results are displayed in Table 2. In any case, it is known that the Bayes error on this problem is 0.08% (that is, a 92% accuracy) and that complex Machine Learning techniques values around 92%. Thus, our projection method fares well in this problem compared to other Machine Learning methods (94.81% is definitely an optimistic estimation of the actual accuracy).

TABLE II
SUMMARY OF EXPERIMENTS CARRIED OUT.

Algorithm	Test C. rate (%)
SMO	77.21
Simple Logistics	77.86
MultiLayer Perceptron	76.69

3) Pima Indians Diabetes

The Pima Indians Diabetes data set studies the influence of diabetes on the American population of Pima Indians. A population of women of Pima Indians was tested for diabetes in accordance with World Health Organization criteria. These data belongs to the National Institute of Diabetes and Digestive and Kidney Diseases and is part of the UCI database. The original data set is composed of 768 instances, with 8 numeric attributes and a class variable labeled 1 or 0 showing whether diabetes was present. There are 268 examples belonging to class 1 and 500 belonging to class 0. The original data set has been split into a training set with 576 examples and a test set with 192 examples, maintaining the proportion between the examples of each class. Our method will project the data from its original eight-dimension space to a new three-dimensional one. Five GP-runs were carried out with different population sizes. In Table 3 we can see the classification accuracy obtained by the experiments.

TABLE III
CLASSIFICATION RESULTS ON THE PIMA INDIANS DIABETES DATA SET

Exp.	Pop. Size	Train C. rate (%)	Test C. rate (%)
1	500	**78.65**	**80.73**
2	500	78.65	75.52
3	250	77.08	78.12
4	500	78.47	79.17
5	1000	79.69	77.08

By taking the minimum value from training and test, it could be said that our method, with a few number of runs, achieves a 78.65% accuracy. In order to compare the method, we ran a support vector machine (SMO), the simple logistics algorithm, and the Multilayer Perceptron from the Weka tool. Results are displayed in Table IV.

TABLE IV
SUMMARY OF EXPERIMENTS CARRIED OUT

Algorithm	Test C. rate (%)
SMO	77.21
Simple Logistics	77.86
MultiLayer Perceptron	76.69

This domain has been very well researched. [8] applied 22 algorithms, with 12-fold cross validation. The best result is 77.7%, some other results being even below 70%. Therefore, the results obtained by our approach are comparable to other results shown in the literature. But it has to be remarked that dimensionality has been reduced from 8 to 3, maintaining similar results to other methods.

IV. RELATED WORK

In [10] the authors use typed GP for building feature extractors. Terminals are arithmetic and relational operators. Terminals are the original (continuous) attributes of the original dataset. Every individual is an attribute and the fitness function uses the info gain ratio. Testing results, using C4.5, show some improvements in some UCI domains. Our approach differs in that our individuals contain as many subtrees as new attributes to be constructed and that the fitness function measures the degree of linear separation in the training data. [11] follows a similar approach to ours, where every individual contains several subtrees, one per feature. C4.5 is used to classify in feature-space instead of the simple scheme (linear separation) we prefer here. Although the author reports very good results in some domains, we believe that allowing GP to find a projection and then using powerful classification schemes can lead rapidly to overfitting. Of course, there are other ways to reduce overfitting, both in GP and in the machine learning scheme. Also, in our experiments we do not limit ourselves to constructing new features and reducing dimensionality. Rather, our intent is to improve classification accuracy, and this can be done by reducing but also by increasing the number of dimensions, in the spirit of Support Vector Machines. Finally, their work allows to cross over subtrees from different features, whereas we use homologous crossover so that only subtrees from the same features from two individuals can be crossed over. We believe

that it would be desirable for constructed features to be independent and, even, orthogonal. Therefore, they should evolve independently and not allow to share code between features via crossover, as we do. This assumption might not work in all domains, but in any case, differences in empirical results should be expected with our approach.

In [12], GP is used to evolve kernels for Support Vector Machines. Both scalar and vector operations are used in the function set. Fitness is computed from SVM performance using the GP-evolved kernel. The hyperplane margin is used as tiebreaker to avoid overfitting. No attempt is made so that kernel satisfies standard properties (like Mercer's) but results in testing datasets are very good, compared to standard kernels. Instead of evolving distance or kernel functions, we evolve projections to spaces with larger, equal, or smaller number of dimensions. We believe that evolving actual distance functions or kernels is difficult, because some properties (like transitivity or Mercer's) are not easy to impose in the fitness computation.

In [13], Genetic Programming was used to construct features to classify time series. Individuals were made of several subtrees returning scalars (one per feature). The function set contained typical signal processing primitives (like convolution), statistical, and arithmetic operations. SVM was then used for classification in feature-space. Cross validation on training data was used as fitness function. The system did not outperform the SVM, but managed to reduce dimensionality. This means that it constructed good features to classify time series. However, only some specific time-series domains have been tested. Similarly, [14, 15] assembles image-processing primitives (edge-detectors, ...) to extract multiple features from the same scene to classify terrains containing objects of interest (golf courses, forests, etc.). Linear fixed-length representations for the GP trees are used. A Fisher Linear Discriminant is used for fitness computation. Results are quite encouraging but they restrict themselves to image-processing domains.

Results from the bibliography show that, in general, the GP-projection approach has merit and obtains reasonable results, but more research in the subject is needed. New variations of the idea and more domains should be tested.

V. Conclusions

In this paper we have presented a GP-based method of automatically evolving projections, so that data can be more easily classified in the projected space. Every individual contains as many subtrees as dimensions in the projected space and are evolved independently. This is on purpose, as we want to evolve independent, and possibly orthogonal features, so we believe that they should not share code via crossover. Our approach can reduce dimensionality by constructing more relevant attributes, but also allows to increase dimensionality, in case classification is more feasible in higher dimensional spaces (in the spirit of Support Vector Machines). The fitness function projects the training data and computes the degree of linear separability by running a Simple Linear Perceptron. We have chosen a simple classification scheme because if GP is allow to evolve any projection, a complex classification scheme would add too many more degrees of freedom and lead easily to overfitting. Also, in evolutionary computation, it is desirable that the fitness function be fast to compute.

We have tested our approach in three domains: a toy ellipse classification domain, the overlapping Ripley's data and the UCI diabetes data. Some experiments reduced dimensionality, and some others increased it. Our method has obtained results comparable to other Machine Learning algorithms cited in the literature in most of the domains. In many cases, results are comparable, but dimensionality is greatly reduced. So, our method constructs good attributes from raw ones.

In the future, we would like to automate most parameter adjustment tasks so that the user need only introduce the examples and he receives them expressed in a space V where classification accuracy is as good as possible. In particular, the system itself should decide the dimension (higher or lower) of the projected space.

Clearly, overfitting is still a problem and we should modify the fitness function so that more realistic estimates are computed. Also, new ways of computing the fitness could be tested, by using other simple classification approaches like ADALINE, or nearest neighbor. In addition, we believe that more complex Machine Learning approaches could be used on the projected data, after evolution took place. That is, if near linear separability is achieved, it is likely that if a Neural Network is applied on the projected data, even better accuracies could be achieved. This was not the case on the domains tested here, probably because we were already on the limit of what could be obtained. But in other domains, this approach could work. We believe that this is better, with respect to overfitting, than using more complex classification schemes directly in the fitness function.

References

[1] N. Cristianini and J. Shawe-Taylor. An introduction to Support Vector Machines (and other kernel-based learning methods). Cambridge University Press, 2000.

[2] T. Fawcett and P. Utgoff. A hybrid method for feature generation. In Proceedings of the Eighth International Workshop on Machine Learning, pages 137– 141, Evanston, IL.

[3] S. Kramer. Cn2-mci: A two-step method for constructive induction. In Proceedings of ML-COLT'94.

[4] B. Pfahringer. Cipf 2.0: A robust constructive induction system. In Proceedings of ML-COLT'94, 1994. W. D. Doyle, "Magnetization reversal in films with biaxial anisotropy," in *1987 Proc. INTERMAG Conf.*, pp. 2.2-1–2.2-6. G. W. Juette and L. E. Zeffanella, "Radio noise currents n short sections on bundle conductors (Presented Conference Paper style)," presented at the IEEE Summer power Meeting, Dallas, TX, June 22–27, 1990, Paper 90 SM 690-0 PWRS.

[5] John R. Koza. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA, 1992.

[6] John R. Koza. Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge Massachusetts, May 1994.

[7] B.D. Ripley. Pattern Recognition and Neural Networks. Cambridge: Cambridge University Press, 1996.
[8] D. Michie, D. J. Spiegelhalter, and C.C. Taylor. Machine learning, neural and statistical classification. Ellis Horwood, 1994.
[9] Benjamin Blankertz, Gabriel Curio, and Klaus-Robert M¨uller. Classifying single trial eeg: Towards brain computer interfacing. In Advances in Neural Inf. Proc. Systems 14 (NIPS 01), 2002.
[10] Fernando E. B. Otero, Monique M. S. Silva, Alex A. Freitas, and Julio C. Nievola. Genetic programming for attribute construction in data mining. In Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa, editors, Genetic Programming, Proceedings of EuroGP'2003, volume 2610 of LNCS, pages 389–398, Essex, 14-16 April 2003. Springer-Verlag.
[11] Krzysztof Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genetic Programming and Evolvable Machines, 3(4):329–343, December 2002.
[12] Tom Howley and Michael G. Madden. The genetic kernel support vector machine: Description and evaluation. Artificial Intelligence Review, To appear, 2005.
[13] S. Davis S. Perkins J. Ma R. Porter D. Eads, D. Hill and J. Theiler. Genetic algorithms and support vector machines for time series classification. In Proceedings SPIE 4787 Conference on Visualization and Data Analysis, pages 74–85, 2002.
[14] John J. Szymanski, Steven P. Brumby, Paul Pope, Damian Eads, Diana Esch-Mosher, Mark Galassi, Neal R. Harvey, Hersew D. W. McCulloch, Simon J. Perkins, Reid Porter, James Theiler, A. Cody Young, Jeffrey J. Bloch, and Nancy David. Feature extraction from multiple data sources using genetic programming. In Sylvia S. Shen and Paul E. Lewis, editors, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII, volume 4725 of SPIE, pages 338–345, August 2002.
[15] Neal R. Harvey, James Theiler, Steven P. Brumby, Simon Perkins, John J. Szymanski, Jeffrey J. Bloch, Reid B. Porter, Mark Galassi, and A. Cody Young. Comparison of GENIE and conventional supervised classifiers for multispectral image feature extraction. IEEE Transactions on Geoscience and Remote Sensing, 40(2):393–404, February 2002.