Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp331-336)

Perceptrons without output codes

A. Sierra[1], A. Echeverría[2]
[1] Escuela Politécnica Superior
Universidad Autónoma de Madrid
28049 Madrid Spain
[2] Escuela Politécnica Superior
Universidad Europea de Madrid
28670 Madrid Spain

Abstract: Neural networks for classification require our choosing output codes. Most often, the 1-of-c output code is used, with as many dimensions as classes. This coding scheme can turn into a burden for datasets with many classes such as the 19 class UCI soybean problem. In this paper, a procedure is introduced which allows to choose the number of output units of a neural network, independently of the number of classes. The weights of the network are learned by means of an evolution strategy whose fitness is the number of misclassifications incurred by assigning patterns to the class of the closest projected mean. In this way, we obtain two-dimensional views of multiclass problems such as the 19-class soybean database and the non-linear 6-class satellite problem.

Key-Words: Evolutionary computation, Neural networks, Dimensionality reduction, Evolution strategy, Pattern classification

1 Introduction

This paper is devoted to pattern classification and dimensionality reduction, paying special attention to multiclass problems. There are two main ways of addressing classification problems with more than two classes. We can either follow a single-classifier approach or construct models as combinations of binary classifiers. The "one-versus-all" strategy [1] and error correcting output codes [2] belong to the latter category of classifiers. In this paper, we try to solve multiclass problems by minimizing just one cost function, i.e., we adhere to the single-classifier approach.

More specifically, our way of approaching multiclass problems has been inspired by one key feature discriminant analysis. Fisher's discriminant analysis (FDA) [3], instead of enforcing fixed output codes, makes use of class means as targets. Rather than learning output codes by quadratic error minimization, a class separation measure is maximized. Broadly speaking, this measure is the distance between means divided by the dispersion around the means. Typically, once a projection is found, patterns are classified as belonging to the class of the closest projected mean. The number of output dimensions is necessarily equal to the number of classes minus one.

Apart from the restrictions on the number of output dimensions, one of the main limitations of this classical approach is its linear nature. Non-linear problems such as the XOR problem, need non-linear transformations in order to disentangle the overlapping between class means. One of the key reasons for the popularity of multilayer perceptrons (MLP) is precisely their ability to approach non-linear input-output relationships. The composition of one-dimensional sigmoidal functions, $\varphi(x) = 1/(1 + e^{-x})$, is behind these non-linear capabilities [4]. Our proposal consists in taking advantage of the MLP's functional approximation with class means as output codes.

As we prove in the next section, there is no simple way of training multilayer perceptrons by gradient descent with projected class means as output codes. Instead of trying to solve this problem by adding penalization terms to the error function, we propose to give up quadratic errors altogether and use instead the number of misclassified patterns [5]. Since this cost function is a discrete quantity, an evolution strategy [6] is used to learn the weights. The good news about our proposal is that there is no restriction on the number of output dimensions. We can always get two-dimensional views of datasets no matter how many classes we are confronted with. Besides, non-linear overlappings can be projected out thanks to the MLP's universal approximation capabilities.

The structure of the paper is as follows. Section 2 makes clear the limitations of quadratic error minimization. Section 3 explains how to use an evolution strategy to circumvent these problems. The datasets on which we have run our experiments are introduced in section 4, and the experiments conducted are reported in section 5. The paper is closed by a section devoted to conclusions and future work.

2 Projected means or fixed output codes?

Let us consider a set of patterns $(\mathbf{x}, \mathbf{t})$ in d dimensions with c classes. Each pattern $\mathbf{x}$ is labelled by the class $\mathbf{t}(\mathbf{x})$ it belongs to. In general, learning a classifier consists in finding a function $\mathbf{y}(\mathbf{x})$ from the input space into the class space which minimizes the classification error of unseen patterns. In particular, multilayer perceptrons learn these functions by the minimization of quadratic errors [4] such as

$$L = \sum_{\mathbf{x}} (\mathbf{y}(\mathbf{x}) - \mathbf{t}(\mathbf{x}))^2. \quad (1)$$

It is most common to use a 1-of-c coding for the output targets so not to enforce spurious order relationships:

$$\mathbf{t}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \mathbf{t}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \dots \mathbf{t}_c = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \quad (2)$$

Let us see why it is not a simple task to use class means instead of codes such as 2. In order to simplify the equations, let us consider a neural network without hidden units and without activation functions. In this simple case, the outputs are the following:

$$y_j = \sum_{i=0}^{d} x_i w_{ij}, \quad (3)$$

where $x_0 = 1$ for all patterns. The gradient of the quadratic error function for a fixed pattern $\mathbf{x}$

$$\frac{\partial(\mathbf{y}(\mathbf{x}) - \mathbf{t}(\mathbf{x}))^2}{\partial w_{ij}} = 2(y_j - t_j)x_i, \quad (4)$$

leads to the classical delta rule or back-propagation rule when hidden units are present.

Next, let us try to substitute output codes by class means. To start with, replacing $\mathbf{t}(\mathbf{x})$ by $\mathbf{m}(\mathbf{x})$ (the mean of the class of $\mathbf{x}$) in equation 1 is dangerous when there exists non-linear overlapping among classes. For instance, in the XOR problem, we would end up using identical codes for all of the classes. Even in linear problems, if we used means as targets, the output dimension would have to be equal to the input dimension, loosing the dimensionality reduction property we are seeking. This is why we have to project means along with their patterns.

If we choose to project patterns along with their class means, the error function becomes

$$L = \sum_{\mathbf{x}} (\mathbf{y}(\mathbf{x}) - \mathbf{y}(\mathbf{m}))^2. \quad (5)$$

The new gradient is proportional to the weights we are trying to learn and therefore it will drive them to zero:

$$\frac{\partial(\mathbf{y}(\mathbf{x}) - \mathbf{y}(\mathbf{m}))^2}{\partial w_{ij}} = 2w_{ij}(x_i - m_i)^2. \quad (6)$$

Therefore, in order to make sense of quadratic errors while projecting class means, the cost function in equation 6 has to be changed somehow. Perhaps, a distance between means' term might help. However, instead of trying to improve this cost function, we have addressed the problem in a new way. We have given up quadratic error functions altogether and used the number of misclassified patterns as the new cost function. There is only one drawback: gradient descent can not be used to minimize this error due to its discrete nature. We have to resort to an evolution strategy [6].

Notwithstanding, the process followed by the evolution strategy is very intuitive. In its simplest version, it can be described as follows. We start from a randomly chosen neural network without output codes. In order to calculate the fitness of this network, i.e., its number of misclassifications, all we have to do is to project patterns and class means, and then classify patterns as belonging to the class of the closest projected mean. Our evolution strategy, after mutating the weights of this network, will update it whenever the number of errors is decreased. This simple mechanism and the absence of quadratic errors allow us to choose the number of projections without restrictions. The output codes are now the projections of the class means and these projections are learned during the learning process. Next section is devoted to explain the detailed workings of this algorithm.

3 Evolutionary discriminant analysis

One of the main ingredients of our algorithm is the substitution of the traditional quadratic error by the number of misclassified patterns. This discrete quantity is to be minimized by means of an evolution strategy, which is detailed in this section. Let us start by saying a few words about the coding scheme.

3.1 Coding scheme

The real valued weights of our MLPs clearly invite us to try evolution strategies as the optimization algorithm, rather than genetic algorithms, for example. Besides, since most of the experiments conducted in this paper make use of networks with two layers of weights, we will illustrate our coding scheme with one of these networks.

We will be using MLPs with one hidden layer of neurons, i.e., two layers of weights: $d+1$ input units, m hidden units, and n output units. The weights of the first layer are called w_{ij}, where $i = 0, \dots, d$ and $j = 1, \dots, m$. The second layer's weights or output weights are named v_{jk}, where $j = 0, \dots, m$ and $k = 1, \dots, n$. Sigmoidal functions $\varphi(z) = 1/(1 + \exp^{-z})$ are only used in the hidden units. The k-th component of the function represented by this MLP is the following:

$$y_k(\mathbf{x}) = \sum_{j=0}^{m} v_{jk} \varphi(\sum_{i=0}^{d} x_i w_{ij}). \quad (7)$$

The weights (w_{ij}, v_{jk}) are located into the strategy's chromosomes in the following order. First, the w_{ij} weights, from $i = 0$ till $i = d$, and then the v_{jk} weights, from $j = 0$ till $j = n$:

$$(\dots, w_{i1}, w_{i2}, \dots, w_{im}, \dots, v_{j1}, v_{j2}, \dots, v_{jn}, \dots). \quad (8)$$

This is a direct coding scheme and arguably the most straightforward way to encode the projecting functions.

3.2 Fitness function

The fitness value assigned to a chromosome (see equation 8) is the number of patterns misclassified by the network represented by this chromosome. This misclassification rate

is calculated by first projecting patterns and then assigning them to the class of the closest projected mean. Actually, the set of patterns is broken into three parts: training, validation and test sets. The means of each class are calculated out of the training patterns exclusively. Both, training and validation patterns are classified by distance to these projected means. The details of the algorithm for an n-dimensional projection are as follows:

- The training means (m_i^r), where $i = 1, \ldots, d$ and $r = 1, \ldots, c$, are projected according to the weights $(\mathbf{w}, \mathbf{v})$ as follows:

$$\bar{m}_k^r = \sum_{j=0}^{m} v_{jk} \varphi(\sum_{i=0}^{d} m_i^r w_{ij}) \qquad k = 1, \ldots, n. \tag{9}$$

- The patterns ($\mathbf{x}$) of both training and validation sets are projected accordingly:

$$\bar{x}_k = \sum_{j=0}^{m} v_{jk} \varphi(\sum_{i=0}^{d} x_i w_{ij}) \qquad k = 1, \ldots, n. \tag{10}$$

- The patterns are assigned to the class of the closest training mean in the projected space:

$$\text{class}(\mathbf{x}) = \arg \min_{r=1,\ldots,c} (\sum_{k=1}^{n} (\bar{x}_k - \bar{m}_k^r)^2) \tag{11}$$

- The fitness value assigned to $(\mathbf{w}, \mathbf{v})$ is the percentage of misclassified patterns for this projection:

$$J(\mathbf{w}) = \frac{1}{N} \sum_{\mathbf{x}} L(t(\mathbf{x}), \arg\min_r (\sum_{k=1}^{n} (\bar{x}_k - \bar{m}_k^r)^2)), \tag{12}$$

where N is the number of patterns, and the value returned by $L(t, t')$ is 0 when $t = t'$, and 1 otherwise. Actually, the error on the training ($J_{tr}(\mathbf{w})$) and validation sets ($J_{va}(\mathbf{w})$) are summed into a single figure $J(\mathbf{w}) = J_{tr}(\mathbf{w}) + J_{va}(\mathbf{w})$.

3.3 The complete algorithm

Once we know how to assign fitness to chromosomes, we are ready to study the algorithm called evolutionary discriminant analysis:

- The number of output dimensions is chosen.
- A neural architecture is chosen: number of hidden layers and number of units in each layer. Our experiments make use of one hidden layer at most.
- An evolution strategy $(\mu, \lambda|\rho)$ is chosen: μ is the size of the parent population, λ is the size of the offspring population, and ρ is the size of the family, or parents whose recombination leads to each offspring.
- The mutation step ($\sigma = 1$) is kept fixed during the evolutionary process.
- An initial population of μ neural networks is randomly generated with weights in $[-0.5, 0.5]$.
- The following steps are repeated until a prescribed number of generations (80 in our experiments) without improvements are done:
  - A group of λ neural networks is generated by the discrete recombination of ρ parent networks. Discrete recombination works as follows. For each offspring, ρ individuals are drawn at random from the parent population. Next, each allele of the offspring is equaled to the corresponding allele of one of the ρ parents, drawn at random from the family subset.
  - Each recombined network is mutated by adding independent normal deviations to each of the weights.
  - Comma replacement is used as generally recommended for continuous parameters [6]. Therefore, a new parent population is created out of the μ best among the λ offspring weights.

The following sections report on the results found by applying this algorithm to a couple of UCI databases.

4 Experimental setup

Two benchmarks have been used in this paper, the soybean set and the satellite image set [7]. In order to asses the generalization ability of our projections, the original sets have been broken into three subsets: training, validation and test sets. Only training and validation patterns take place in the evolutionary process. The training patterns are the only ones used for calculating means. Therefore, the error on the validation set helps to control overfitting, specially when non-linear projections are constructed. Let us start by describing the satimage dataset.

4.1 Satellite images

We have applied our algorithm to the discrimination of images taken by the Landsat satellite. This problem has been chosen because it is a multi-class dataset with non-linear overlapping among the classes. Although classical FDA gives rise to five dimensional linear projections, our algorithm easily yields two dimensional non-linear projections with state of the art recognition rates.

The dataset is publicly available under the name sat.trn, containing training patterns, and sat.tst, with testing patterns. Contrary to what one would probably think, we are not asked to classify whole images, rather we are confronted with the classification of individual pixels. In fact, the data contained in both satellite sets correspond to one image only or, better, a small area of one image.

In order to understand the information available for each pixel's classification, let us review how scenes are sensed by the satellite. Four images are taken of each scene in different spectral bands: two in the visible region and two in the infra-red region. Each of these four images contains 2340×3380 pixels and each pixel occupies 8 bits. The UCI satellite image database corresponds to a 82×100 pixel subarea of a certain scene, and each line of data corresponds to

a square 9 pixel neighborhood. More specifically, each line of data corresponds to one pixel and contains:

- The classification code of the pixel, i.e., its class membership. Although there are 7 classes, the pixels belonging to class 6 have been erased from the database.
- The 3×3 pixel neighborhood, i.e., the ASCII values of the 9 dimensional pixel lattice surrounding the classified pixel. Given that each frame is composed of four digital images, the 9 pixel neighborhood gives rise to 36 pixels, which have been converted to ASCII code (0 corresponding to black and 255 meaning white).

In short, we have 36 dimensional input patterns to be classified in one of six classes.

4.2 Soybean diseases

The soybean set is dedicated to soybean disorders. Although beans are characterized in the UCI repository by 35 continuous features, we will be using Prechelt's encoding of this problem into 82 real-valued features [8], including special codes to deal with the abundant missing values. More concretely, the so called soybean1 permutation will be used. The within-class covariance matrix of this set, S_w, is singular, even after removing those features (features 66 and 82) with null variance. This makes very difficult the application of traditional discriminant analysis.

5 Experimental results

Let us first report on the results found without the use of hidden units, i.e., the performance of linear projections which, for certain problems such as the soybean problem, yield state of the art recognition rates. In all of our experiments, a (15,100|15)-ES evolution strategy has been used. The mutation step is kept fixed ($\sigma = 1$) during the evolutionary process. The algorithm is stopped after 80 generations without improvements.

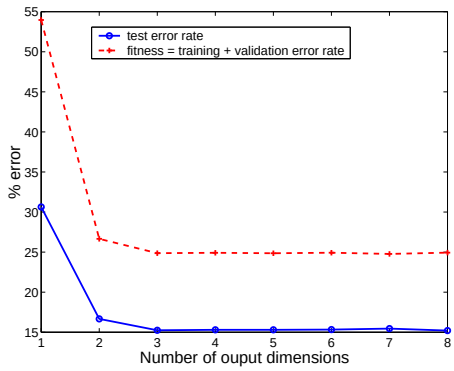

Figure 1: Average fitness and test error rate of 20 executions of EDA on the satellite set for an increasing number of output units. Three output units yield 15% test error rate.

Model	Train	Valid.	Test	Eval. (x10^3)	Fitness
1	15.08	11.43	15.90	59.64	26.51
2	15.01	11.75	17.15	32.18	26.76
3	15.43	11.85	16.42	18.02	27.28
4	15.21	10.71	16.53	64.28	25.92
5	15.37	11.43	17.05	25.74	26.80
6	15.41	11.23	17.05	52.51	26.64
7	15.06	11.12	16.11	52.65	26.18
8	15.10	11.43	16.94	47.58	26.53
9	15.06	11.43	16.94	40.57	26.49
10	15.34	11.12	16.94	41.69	26.47
11	15.59	11.12	16.74	35.28	26.71
12	15.28	11.43	16.22	39.39	26.71
13	15.34	11.85	16.01	21.97	27.19
14	15.30	11.12	17.67	59.43	26.42
15	15.43	11.54	17.46	22.37	26.97
16	15.37	11.64	16.53	41.93	27.01
17	15.37	11.54	16.11	47.68	26.90
18	15.28	11.43	16.63	42.05	26.71
19	15.48	10.71	16.53	47.02	26.18
20	15.14	11.54	16.42	29.00	26.68
Mean	15.28	11.37	16.67	41.05	26.65
Dev	0.16	0.31	0.47	12.94	0.33

Table 1: 20 runs of EDA without hidden units and two output dimensions for the satellite database. Training, validation and test error rates of the best individual of each run are shown. The number of evaluations till this best individual and its fitness are shown in columns 4 and 5, respectively.

5.1 Linear results

Classical FDA finds a 5 dimensional projection with a 19.02% test error rate. As can be seen in table 1, the linear projections evolved by means of EDA outperform FDA. This table shows fitness and error rates of the best individuals found in 20 executions of EDA with two output dimensions and no hidden units. The average test error rate of these 20 executions is 16.67%, more than a 10% improvement on FDA, even when both algorithms are linear projections and classify by distance to means. This difference is due to the fact that we are optimizing the classification rule directly. By contrast, FDA first maximizes the separation between classes and only then, classifies patterns by distance to means. The projection, as happens in this case, may end up being suboptimal with respect to classification by distance to means.

In order to check the dependence of the performance of our algorithm on the number of output units we have executed EDA without hidden units and with an increasing number of output dimensions. The results are gathered in figure 1. Each point of this figure is the result of averaging 20 executions of EDA. It shows average fitness and test error rate versus number of output dimensions. The number of dimensions varies from 1 till 8. Due to the linear nature of the projections, there is a strong correlation between fitness and generalization performance, i.e., test error rate. Three

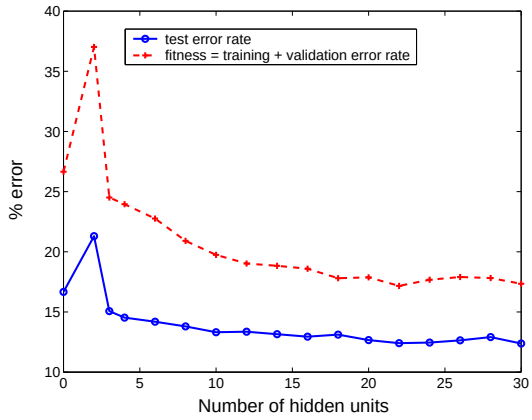

Figure 2: Average fitness and test error rate of 20 executions of EDA on the satellite image set for two output units and an increasing number of hidden units.

output dimensions already yield a 15% test error rate, which amounts to a 20% improvement with respect to FDA.

5.2 Non-linear results

We have also run experiments with hidden units. More specifically, we have evolved neural networks with one layer of hidden units and two output units. Figure 2 shows how fitness and test error rate vary with the number of hidden units. The architecture 36:2:2, as can be seen in the peak of the figure, seems to get trapped into a local minimum. In general, the performance improves with the number of hidden units as expected but, around 22 units, both fitness and test error rate begin to worsen. The test error rate corresponding to the best fitness is under 13%.

Before closing this section, let us take a loot at Table 2, which compares EDA with some other algorithms on the satellite problem. As can be seen, one-versus-all (OVA) classification yields the best results. OVA makes use of $c = 6$ SVM binary classifiers, each one trained to distinguish the patterns of one class from the rest of the examples. New patterns are classified as belonging to the class of the classifier with a higher single-class output value. The performance on the satellite problem (7.8%) is excellent mainly due the combination of classifiers [1]. Among the single-classifier models, the $k = 1$ nearest neighbor rule yields the best result, with a 9.4% test error rate [9]. The results under EDA and non-linear EDA are the ones corresponding to the projections with best fitness. The non-linear result is close to the 1NN result even when only two output projections are used.

6 The soybean database

The soybean dataset is interesting in the following sense. The within covariance matrix which enters into the calculation of the eigenvectors of FDA happens to be singular. As a result, the classical algorithm does not give any sensible answer. By contrast, our algorithm yields quite good results even with two output units, despite of the fact this set has

Algorithm	Train %	Test %	Ref.
OVA	-	7.8	[1]
KNN	8.9	9.4	[9]
Non-linear EDA	11.04	12.41	this work
EDA	14.07	15.45	this work
FDA	17.49	19.02	this work

Table 2: Performance comparison between EDA and other algorithms on the satellite problem. KNN is $k = 1$ nearest neighbors. OVA is one-versus-all classification with SVM networks as binary classifiers.

19 classes. In figure 3, one of these two-dimensional projections is shown. This view allows to check the proximity between deceases, which might be enlightening for the expert.

7 Discussion and conclusions

The main goal of this paper has been to develop an algorithm capable of rendering two-dimensional views of multiclass problems. This work fills a gap because in most dimensionality reduction algorithms, the number of projections can not be chosen. Quite on the contrary, this number is fixed by the number of classes. For instance, classical discriminant analysis returns as many dimensions as classes minus one. Although it is true that this number can be reduced by discarding some of the linear combinations, the rejected features will give rise to loss of information.

We have succeeded in training multilayer perceptrons without enforcing output codes by giving up quadratic errors. Our new cost function, the number of misclassified patterns, is optimized by an evolution strategy. Direct error minimization has been tackled before by means of projection pursuit algorithms [10]. Projection pursuit is a procedure for searching interesting low-dimensional projections that maximize the so called projection index. The total probability of misclassification has been used as such an index in [11]. Recently, the classification error has been optimized directly by means of a simulated annealing technique [12]. The main difference between this technique and ours is the use of a population of solutions instead of a unique candidate.

Despite the success of our approach, some questions have been left unanswered in this paper. The search for adequate architectures is one of them. In section 5, we have checked how the number of hidden units affects performance. We have used only two output units because we were interested in obtaining two-dimensional views of our problems. However, in order to optimize performance, we have to search for optimal architectures. We are currently working on different ways of addressing this problem and hope to report our results soon. Besides, we are considering the possibility of using different distance metrics. The use of covariance matrices might help the task carried out by the evolution strategy. In our current algorithm, it is the pattern standardization what is playing this role.

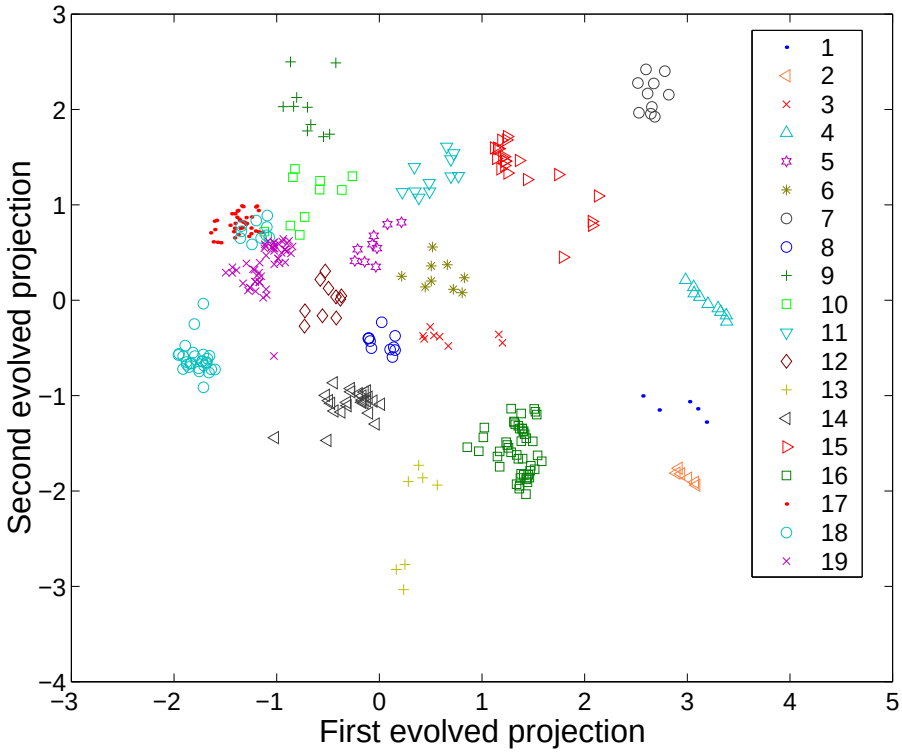

Figure 3: Two-dimensional projection of the soybean1 problem (training set only) found by EDA and a $(15, 100|15)$-ES. One of the key points of our approach is that it allows to visualize multi-class problems like this (19 classes) in two dimensions.

Acknowledgments

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TIN2004-07676-C02-02.

Bibliography

[1] R. Rifkin and A. Klautau, "In Defense of One-Vs-All Classification," *Journal of Machine Learning Research*, vol. 5, 2004, pp. 101–141.

[2] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems with error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, 1995, pp. 263–286.

[3] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, 1936, pp. 179–188.

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

[5] A. Sierra and A. Echeverría, "Skipping Fisher's Criterion," *Lecture Notes in Computer Science*, vol. 2652, 2003, pp. 962–969.

[6] H. G. Beyer and H. P. Schwefel, "Evolution Strategies. A Comprehensive Introduction," *Natural Computing*, vol. 1, 2002, pp. 3–52.

[7] C. L. Blake and C. J. Merz, UCI repository of machine learning databases, 1998. [www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. of Information and Computer Science.

[8] L. Prechelt, "Some Notes on Neural Learning Algorithm Benchmarking," *Neurocomputing*, vol 9, n. 3, 1995, pp. 343–347.

[9] D. Michie, D. J. Spiegelhalter and C. C. Taylor (eds.) *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Hemel Hempstead, 1994.

[10] J. H. Friedman and J. W. Tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on Computers*, vol. 23, n. 9, 1974, pp. 881–884.

[11] C. Posse, "Projection Pursuit Discriminant Analysis for Two Groups," *Commun. Statist.*, vol. 21, n. 1, 1992, pp. 1–19.

[12] M. Rohl, C. Weihs, and W. Theis, "Direct Minimization of Error Rates in Multivariate Classification," *Computational Statistics*, vol. 17, 2002, pp. 29–46.