Genetically Found, Neurally Computed Artificial Features

Hiram Firpi and Javier Echauz

Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-9042
hiramf@ece.uprm.edu

Abstract

The specification of distinguishable features is the most important key to intelligent sensing in pattern recognition, but few research results have been reported in the field. We investigated genetically learned artificial features that are customized for a given classification task from a list of raw features. The pioneering work in [6] was significantly extended by representing and computing features by networks. A parallel/non-parallel random vector decision problem and k-nearest neighbor classifiers were used to assess the feasibility of genetically found, neurally computed (GFNC) features. We show how a novel artificial feature quickly evolves to achieve 100% accuracy in this problem.

1. INTRODUCTION

Features are quantitative or qualitative measures that distill relevant information from raw data for tasks such as classification and regression. The specification of distinguishable (sensitive, specific, separable, parsimonious, accurate, etc.) features is the most important key to intelligent sensing. This has been recognized many times: "Feature extraction is the least understood and most difficult part of the pattern recognition problem" [1], "Learning will always succeed, given the right preprocessor" [2], "The key to achieving robust recognition performance is the integration of 'good' features into the classifier architecture" [3]. Given a set of features, it is known how to prescribe optimal classifiers (in Bayes sense for example), and how to create near-optimal ones empirically using neural networks. However, the power set (all possible subsets) of these features may not convey maximum information available in the raw data. The act of prescribing the features themselves is routinely dismissed as an "art"—an inductive problem guided only by trial-and-error or intuition of the physics. It may be fruitful to challenge this long-standing view and explore ways of systematizing to some degree the determination of features in a given problem, beyond traditional subset selection.

The following terminology: feature extraction, feature creation, feature optimization, feature learning, feature discovery, feature mapping, feature augmentation, feature transformation [4], and signal or data projection [3], has appeared in the literature in the following contexts:

(1) selecting a feature subset from a predefined list with methods such as forward and backward sequential selection, or combined add-on/knock-out,
(2) creating features as linear combinations of input features (the classical definition of feature extraction) such as principal components, or creating feature vectors as linear combinations of raw inputs with methods such as adaptive noise filtering and time-frequency transforms,
(3) creating features as nonlinear combinations of the input features considering a hidden layer in a neural network as a nonlinear feature extractor [5], or joining inputs by algebraic operators [6].

Recognition rate improvements obtained from these methods stem from the fact that patterns are made more obvious before presentation to the decision structure.

The idea investigated here is to genetically learn **artificial features** that are customized for a given task. We significantly extend the pioneering work in [6] by representing and computing features by networks. This is motivated by the following observation: Since a feature (quantitative or qualitative-turned-quantitative) is obtained from a formula or algorithm that maps a raw input set into a scalar (in other contexts, a whole feature vector is considered to be a single feature), then an suitable neural network is theoretically capable of learning and implementing the map. This extension will also allow the future development of artificial features from raw data and not just from a finite list. The networks are represented genotypically as binary strings (chromosomes) and are considered as

individuals in a genetic algorithm or other evolutionary algorithm. Therefore, **genetically found, neurally computed (GFNC) features** are the range of feedforward networks, or the stable equilibria of recurrent networks, and can mimic conventional features or be completely novel artificial features. They are presented to a classifier as if they were conventional features computed procedurally. Table 1 highlights the contrasting characteristics between conventional and artificial features.

Table 1 Conventional vs. artificial features.

Conventional Features	Artificial Features
serial	parallel
Von Neumann computer	neural computer
programmed	learned
combinatorial	inductive
ad-hoc	optimized
based on intuition	based on given data

2. METHODS

We tested the feasibility of GFNC artificial features on the problem of deciding whether two random vectors are parallel in the plane. Given the starting points and increments of the two vectors, $(x_1,y_1,\Delta x_1,\Delta y_1)$ and $(x_2,y_2,\Delta x_2,\Delta y_2)$, a decision structure must output 1 for parallel, and 0 for non-parallel. It will be instructive to note that the starting points and the relative size of the increments are irrelevant, and from knowledge of analytic geometry, that the absolute value of the cosine between the vectors:

$$|\cos q| = \frac{\left|\left(\Delta x_{1,}\Delta y_1\right)\cdot\left(\Delta x_{2,}\Delta y_2\right)\right|}{\left\|\left(\Delta x_{1,}\Delta y_1\right)\right\|\left\|\left(\Delta x_{2,}\Delta y_2\right)\right\|} = \frac{\left|\Delta x_1\Delta x_2+\Delta y_1\Delta y_2\right|}{\sqrt{\left(\Delta x_1^{\,2}+\Delta y_1^{\,2}\right)\left(\Delta x_2^{\,2}+\Delta y_2^{\,2}\right)}}, \qquad (1)$$

is an optimal feature for this task, with 0 meaning orthogonal, and 1 meaning completely parallel.

To test classification accuracy using various features, a *k*-nearest neighbor (*k*-NN) classifier was chosen. The *k*-NN is a nonlinear, nonparametric classifier widely accepted for benchmarking. Though computationally expensive, it is easily obtained without training by direct memorization of the whole training set of N_{obs} observations. Given an input pattern vector, the *k*-NN searches for the *k* most similar (closest in Euclidean distance) vectors in the training database, and declares the corresponding target class by majority vote. Figure 1 shows an input vector being tested against the training patterns in order to reach a target conclusion in a table look-up fashion.

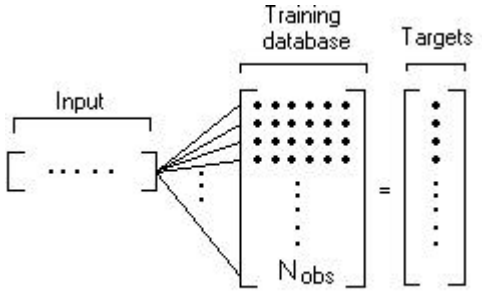

Fig. 1 *k*-NN classifier database.

In a first experiment, we trained a 3-NN classifier using 256 random examples of raw data containing the two input vectors $[x_1,y_1,\Delta x_1,\Delta y_1,x_2,y_2,\Delta x_2,\Delta y_2]$. The first 128 vectors were category 1 (parallel) while the next 128 were category 0 (non-parallel). The starting points were uniformly distributed between 0 and 10, the increments $\Delta x_1,\Delta y_1$ were uniformly distributed between –5 and 5, while $\Delta x_2,\Delta y_2$ were proportional to $\Delta x_1,\Delta y_1$ by a random factor uniformly distributed between –1 and 1 for the parallel cases, which included parallel vectors pointing at opposite directions. Similarly, a testing database of 20 new random examples was created for performance assessment. This was repeated using only the four relevant raw features $[\Delta x_1,\Delta y_1,\Delta x_2,\Delta y_2]$ as inputs. In a second experiment using the same training and testing data sets, a single-input 3-NN classifier was trained after processing the raw data vector into the absolute cosine feature in Eq. (1).

In the last experiment, we implemented the GFNC procedure for creating artificial features equal to Eq. (1) or better that could grasp the concept of parallelism. Figure 2 shows the general GFNC block diagram.

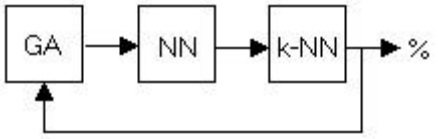

Fig. 2 Block diagram for creating GFNC features.

The neural network (NN) in this system encodes a formula that computes an artificial feature. In order

to initially limit the scope of a vast time-consuming investigation, the four relevant variables $[\Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2]$ were defined as inputs to the network. The problem is still far from trivial. The general network used in the experiment is shown in Fig. 3. Each single-input node represents one of the unary mathematical operators $\{I(\bullet), (\bullet)^2, \sqrt{\bullet}, |\bullet|\}$, where $I(\bullet)$ is the identity, while the two-input nodes represent binary operators $\{+, -, *, /\}$. The latter type of nodes can be easily generalized to *n*-ary operators. All connection weights shown are equal to 1. Conventional neural networks can also be employed.

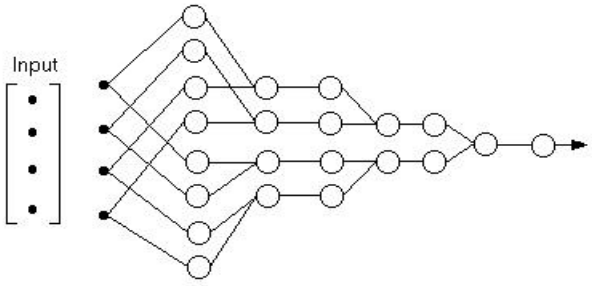

Fig. 3 General feature network investigated here.

The output of the network is a rather general nonlinear feature. Raw training and testing patterns go through the network in order to create a single-input 3-NN classifier that is trained on the artificial feature, and to compute a percent correct metric on the test set:

$$P_c = \frac{|\{3\text{-NN output} = \text{Target}\}|}{N_{obs}} \times 100\%, \qquad (2)$$

where $|\{\bullet\}|$ is the cardinality of the set of 3-NN test outputs that are equal to the target outputs (number of correct classifications) and N_{obs} is the number of test observations.

The genetic algorithm (GA) uses the performance metric as a fitness function. Our GA is of ranking type with elitism, and maintains a constant-size population of candidate solutions (artificial features or artificial formulas). Since the network has 22 nodes with 4 possible operators per node, this problem is a combinatorial search in a 22-dimensional space containing $4^{22} \approx 1.8(10)^{13}$ candidate solutions. At current processing speeds, every fitness evaluation takes several seconds to compute, therefore it would take on the order of several years to solve the problem by brute force (enumeratively) or by purely random search. In the GA's chromosome for each individual, we represented the node operations using 2 binary genes (because there are 2^2 possible operators per node), for a total of 44 genes per individual. This chromosome is decoded into the NN that represents an artificial feature. The evolution of artificial features in the population progresses by genetic crossover and mutation.

3. RESULTS

Figure 4 shows the 10 parallel and 10 non-parallel pairs of test vectors used in the experiments.

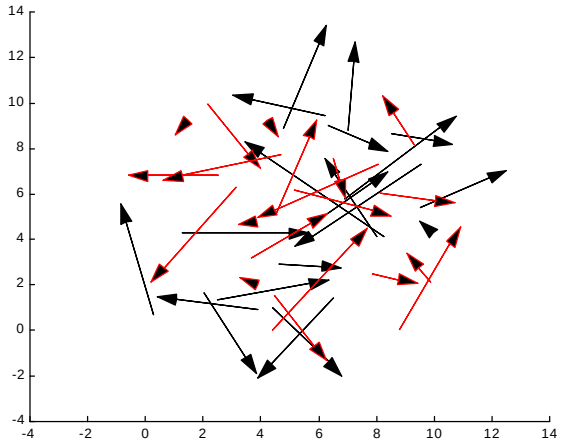

Fig. 4 Parallel and non-parallel pairs of test vectors.

The classification accuracies on this set of vectors is summarized in Table 2.

Table 2 Summary of classification results.

Raw Inputs	Processed Inputs to *k*-NN	P_e
8 raw features	8 raw features	55%
4 delta features	4 delta features	75%
	1 absolute cosine feature	100%
	1 GFNC feature	100%

Figure 5 shows the minimization of the fitness function expressed as percent incorrect (100% minus percent correct). The top graph is the average fitness of whole population in each generation, while the lower graph shows the best individual in each generation. It took only 10 generations to find the

network shown in Fig. 6. From this network, we can obtain the analytical form of the GFNC artificial feature as

$$\sqrt{\left(\sqrt{\frac{\Delta x_1}{\Delta x_2}}-\left(\sqrt{\Delta y_1}-\Delta y_2\right)^2\right)^2}-\sqrt{\left(\Delta x_1-\Delta x_2{}^2\right)^2\cdot\left(\Delta y_1{}^2-\Delta y_2{}^2\right)} \quad (3)$$

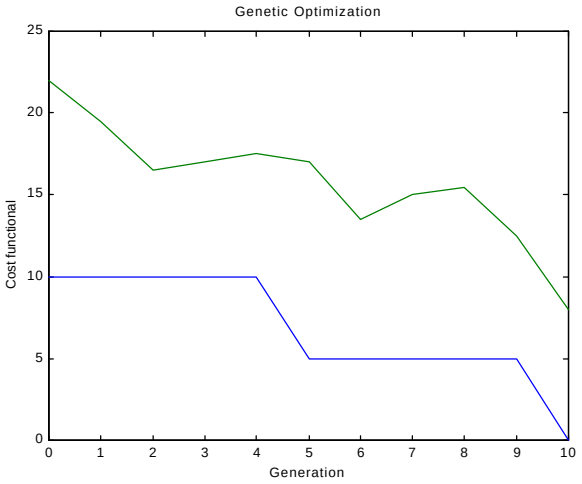

Fig. 5 Fitness function vs. number of generations.

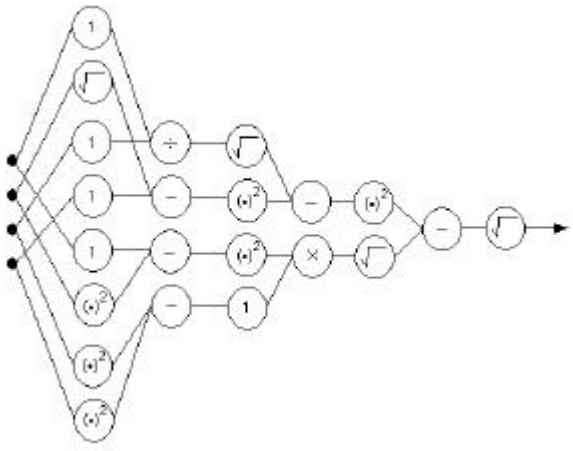

Fig. 6 NN representation of the new artificial feature found by the GFNC process.

4. CONCLUSIONS

From Table 2 it is clear that, keeping the number of training examples to a constant reasonable value, less features and better performance is possible simultaneously. We note that the 8-input *k*-NN immediately increases its accuracy (from 55% to 75%) when the 4 relevant features are given as inputs but is still far from 100%. This demonstrates the crucial role that a single feature such as the optimal absolute cosine feature can play in the discrimination task. Moreover, artificially created GFNC features by definition match or outperform conventionally chosen features, especially in situations where domain knowledge is insufficient to achieve satisfactory levels. In our experiments, a feature never thought of by a domain expert was quickly found to achieve the same zero-error performance as the common-sense analytic feature. It is reasonable to conjecture that there are hundreds of additional artificial features with maximum accuracy or other attractive properties. GCNC is a very promising approach to optimizing features for signal analysis, pattern recognition, machine learning, etc.

REFERENCES

[1] R. M. Glorioso, *Engineering Cybernetics*. NJ: Prentice-Hall, 1975.

[2] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation: A Lecture Notes Volume in the Santa Fe Institute Studies in the Sciences of Complexity*. Redwood City, CA: Addison-Wesley, 1991.

[3] D. H. Kil and F. B. Shin, *Pattern Recognition and Prediction with Applications to Signal Characterization*. Woodbury, NY: AIP Press, 1996.

[4] B. Zupan, M. Bohanec, J. Demsar, and I. Bratko, "Feature Transformation by Function Decomposition," *IEEE Intelligent Systems*, pp. 38-43, March/April 1998.

[5] V. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory, and Methods*. NY: Wiley, 1998.

[6] E. Chang, R. Lippmann, and D. Tong, "Using Genetic Algorithms to Select and Create Features for Pattern Classification," in *Proc. IEEE International Joint Conference on Neural Networks*, 1990, vol. III, pp. 747-752.