Applied Soft Computing 11 (2011) 1087–1097

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A generic optimising feature extraction method using multiobjective genetic programming

Yang Zhang, Peter I. Rockett*

Vision and Information Engineering Group, Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S1 3JD, UK

ARTICLE INFO

Article history:
Received 19 November 2007
Received in revised form 16 February 2010
Accepted 16 February 2010
Available online 26 February 2010

Keywords:
Feature extraction
Multiobjective optimisation
Genetic programming
Pattern recognition

ABSTRACT

In this paper, we present a generic, optimising feature extraction method using multiobjective genetic programming. We re-examine the feature extraction problem and show that effective feature extraction can significantly enhance the performance of pattern recognition systems with simple classifiers. A framework is presented to evolve optimised feature extractors that transform an input pattern space into a decision space in which maximal class separability is obtained. We have applied this method to real world datasets from the UCI Machine Learning and StatLog databases to verify our approach and compare our proposed method with other reported results. We conclude that our algorithm is able to produce classifiers of superior (or equivalent) performance to the conventional classifiers examined, suggesting removal of the need to exhaustively evaluate a large family of conventional classifiers on any new problem.

1. Introduction

Despite its prominence in the field of pattern recognition up to the 1970s, the area of feature extraction – also termed feature construction – together with the related area of feature selection, has been largely overtaken by work on classifier design, principally neural networks. Indeed many elegant theoretical results have been obtained in the classification domain in the intervening years. Nonetheless, feature extraction retains a key position in the field since the performance of a pattern classifier is well-known to be enhanced by proper preprocessing of the raw measurement data – this topic is the main focus of the present work.

Fig. 1 shows a prototypical pattern recognition system in which a vector of raw measurements is mapped into a decision space. Often the feature selection and/or extraction stages are either omitted or are implicit in the recognition paradigm – a multi-layer perceptron (MLP) is a good example of a classification paradigm where a distinct feature extraction stage is not readily identifiable. Addison et al. [1] and Park et al. [2] have reviewed existing feature extraction and selection techniques while Guyon and Elisseeff [3] have discussed feature extraction in terms of filter and wrapper methods. In this paper we focus on feature extraction.

The principal difficulty with designing the feature extraction stage for a classifier is that it usually requires deep domain-specific knowledge. (Indeed much of the work in image processing on detecting image cues such as edges and corners is actually feature extraction.) Even for feature extractors designed by domain experts, the issue of optimality is rarely addressed. Ideally, we would require some measure of class separability in the transformed decision space to be maximised but with handcrafted methods this is hard to guarantee.

In general terms, finding the optimal (possibly nonlinear) transformation, $\mathbf{x} \rightarrow \mathbf{y}$ from input vector $\mathbf{x}$ to the decision space vector $\mathbf{y}$ where $\mathbf{y} = f(\mathbf{x})$, is a challenging task. In the sense that the feature extraction preprocessing stage is a transformation or mapping from input space to decision space, for a given classification problem we seek the mapping which maximises the separability of the classes in decision space. Thus feature extraction can be regarded as the search for an optimal sequence of operations subject to some criterion.

Genetic programming (GP) is an evolutionary problem solving method which has been extensively used to evolve programs or sequences of operations [8]. Typically, a prospective solution in GP is represented as a parse tree which can be interpreted as a sequence of operations and thus evaluated. Fig. 2 shows example GP trees together with the crossover operation typically used in the search process; the the output of the tree on the left (Parent 1), for example, evaluates to the expression:

$$y = -\log(X_3 - X_4)$$

where $X_{3,4}$ are input features from the pattern being processed.

During evolutionary search two parents are selected biassed in their fitness and these may undergo *crossover* to produce two new

* Corresponding author.
E-mail addresses: hegallis@gmail.com (Y. Zhang), p.rockett@shef.ac.uk (P.I. Rockett).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.02.008

Fig. 1. Prototypical pattern recognition system.

offspring. A crossover point is selected in each parent and the the two subtrees – shown in the dashed boxes in Fig. 2 – are exchanged. The two offspring may each be modified by a *mutation* operator in which a subtree in an offspring tree is selected and replaced by a new, randomly-generated subtree. See Section 2.2 for details of the selection, crossover and mutation operations used in the present work. The cycle of selection/crossover/mutation is repeated either for a fixed number of iterations or until some some pre-specified error target is attained. (Genetic programming has been comprehensively reviewed in a recent book by Poli et al. [4]).

GP has been used before to optimise feature extraction and selection. Ebner [5,6] has evolved image processing operators using GP. Bot [7] has used GP to evolve decision space features, adding these one-at-a-time to a *k* NN classifier if the newly evolved feature improved the classification performance by more than a certain amount. Bot's approach is a greedy algorithm and therefore almost certainly sub-optimal. In addition, Koza [8] has produced character detectors using genetic programming while Tackett [9] evolved a symbolic expression for image classification based on image features.

Harvey et al. [10] evolved pipelined image processing operations to transform multi-spectral input synthetic aperture radar (SAR) image planes into a new set of image planes and a conventional supervised classifier was used to label the transformed features. Training data were used to derive a Fisher linear discriminant and GP was applied to find a threshold to reduce the output from the discriminant finding phase to a binary image. However, the discriminability is constrained in the discriminant finding phase and the GP only used as a one-dimensional search tool to find a threshold.

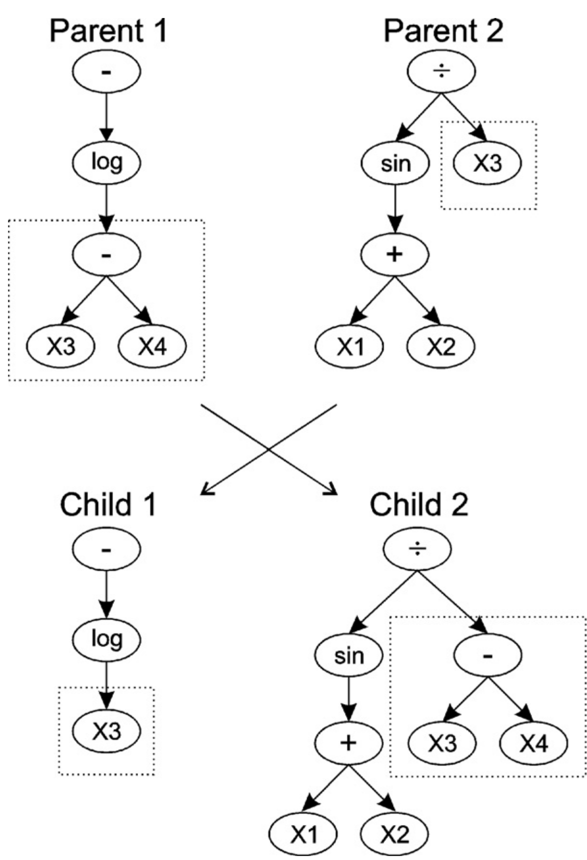

Fig. 2. Illustration of the crossover operation in genetic programming.

Sherrah et al. [11] proposed the Evolutionary PreProcessor (EPrep) system which used GP to evolve a good feature mapping by minimising misclassification error. Three typical classifiers: generalised linear machine (GLIM), *k*-nearest neighbour (*k* NN) and maximum likelihood classifiers were selected randomly and trained in conjunction with the search for the optimal feature extractors. The misclassification errors on the validation set from those classifiers were used as a fitness value for the individuals in the evolutionary population. The same procedure was used in the co-evolution of feature extraction/classifiers in [12]. This approach, however, makes the feature extraction procedure dependent on the classifier in an opaque way such that there is a potential risk that the evolved preprocessing can be excellent but the classifier can be poor giving a poor overall performance, or vice versa.

Kotani et al. [13] used GP to determine the polynomial combination of raw features to be fed into a *k* NN classifier and reported an improvement in classification performance. Krawiec [14] constructed a fixed-length decision vector using GP proposing an extended method to protect 'useful' blocks during the evolution. This protection method, however, contributes to the overfitting which is evident from his experiments. Indeed, Krawiec's results show that for some datasets, the application of his feature extraction method actually produces *worse* classification performance than using the raw input data alone. Estébanez et al. [15] have followed a similar approach to Krawiec in projecting to a vector decision space of pre-determined dimensionality. Recently, Guo et al. [16] have evolved features in a condition monitoring task although it is not clear whether the elements in the vector of decision variables were evolved at the same time or hand-selected after evolution. Smith and Bull [17] have used GP together with a GA to perform feature construction and feature selection.

Broadly, the previous work on GP feature extraction can be categorised as evolving either: A discrete feature extraction stage which then feeds into a traditional classifier, or evolving a combined feature extraction/classification method which directly outputs a class label. Of the two possible routes, we argue that there is little merit in investing computational effort in evolving classifiers since this area is well understood and has solid theoretical underpinnings. We argue that the available computational effort should be expended on producing good feature extraction; in addition, we question the speed of convergence when exploring a search space which contains not only the set of feature extractors but also the set of all classifiers. Consequently, we adopt the approach here of evolving optimal feature extraction algorithms and performing the classification task using a standard, simple and fast-to-train classifier since the classifier has to be included inside the evolutionary loop to evaluate an individual's fitness in terms of a separability measure in the decision space. We draw a distinction in the present