

# ROC analysis and cost-sensitive optimization for hierarchical classifiers

Pavel Paclík\*, Carmen Lai\*, Thomas C.W. Landgrebe†, Robert P.W. Duin‡

\*PR Sys Design, Delft, The Netherlands, Email: pavel@prsysdesign.net

†University of Sydney, Sydney, Australia, Email: thomas.landgrebe@sydney.edu.au

‡PRLab, TU Delft, Delft, The Netherlands, Email: r.duin@ieee.org

**Abstract**—Instead of solving complex pattern recognition problems using a single complicated classifier, it is often beneficial to leverage our prior knowledge and decompose the problem into parts. These may be tackled using specific feature subsets and simpler classifiers resulting in a hierarchical system. In this paper, we propose an efficient and scalable approach for cost-sensitive optimization of a general hierarchical classifier using ROC analysis. This allows the designer to view the hierarchy of trained classifiers as a system, and tune it according to the application needs.

**Keywords**—Hierarchical classifiers, ROC analysis, cost-sensitive optimization

## I. INTRODUCTION

Pattern recognition problems often exhibit high complexity. A powerful approach to tackle complexity employs problem decomposition. Instead of building one complicated classifier in a large feature space, we may design a hierarchical classifier focusing on separate sub-problems. Each sub-problem classifier may leverage specific features and hence a simpler model. The benefits of hierarchical classifiers have been demonstrated for problems with large number of classes in remote sensing [1], target detection [2] or pose estimation [3].

Although hierarchical classifiers provide much needed simplification, they are difficult to optimize. Receiver Operating Characteristic (ROC) analysis became the standard tool for tuning of trained two- and multi-class classifiers according to performance requirements [4]. The benefits of joint ROC optimization of a two-stage detector/classifier system have been also demonstrated [5]. However, in case of general classifier hierarchies, the designer currently faces a dilemma: Either to decompose the system in parts and optimize each step independently or to build a monolithic classifier.

In this paper, we propose an algorithm for cost-sensitive optimization of an apriori-defined hierarchical classifier. It allows the designer to view the entire hierarchy of individually-tuned classifiers as a system, and optimize it using the system-wide tools.

## II. HIERARCHICAL CLASSIFIERS

For the sake of this research we consider the hierarchical classifier as a tree with nodes representing the classifiers, and edges the flow of data samples during execution. We focus

on a sub-class of *decoupled* hierarchical classifiers that are constructed based on problem prior knowledge. We assume that the training and test data sets for each node classifier are known apriori and do not depend on setting of a decision operating point anywhere in the system.

Decoupled systems naturally arise, for example, in two-stage recognition applications where the detector is designed using low-level image processing techniques while the second-stage multi-class classifier is trained with statistical pattern recognition approach. Other candidates for apriori decomposition are the medical diagnostic or industrial sorting problems with known sub-classes such as tissues, materials or varieties. While the separation of sub-class groups may often be performed with apriori-known simple features, the finer discrimination of similar materials may require more sophisticated data representations.

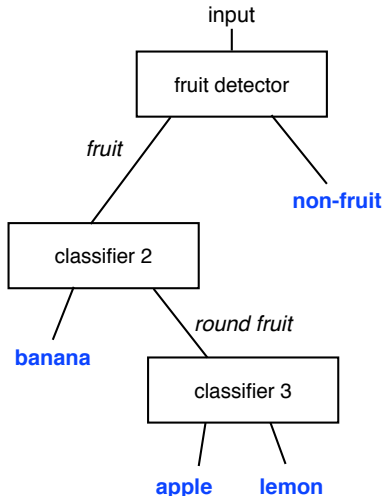


Figure 1. Hierarchical fruit classifier. Terminal decisions are in boldface, meta-class decisions in italics.

On the contrary, an example of a *coupled* hierarchical system is a cascade of AdaBoost detectors widely used in computer vision applications [6]. Each stage in the AdaBoost cascade is trained on the training data containing the false positives of the preceding stage which was tuned using ROC analysis not to loose target examples.

An example of a decoupled hierarchical classifier is presented in Figure 1. This fruit sorting system consists of

a fruit detector rejecting outlier objects on the conveyor belt such as stones. Only examples labeled as *fruit* by this detector are passed on to the classifier discriminating between the round fruit and bananas. Defining the *round fruit* meta-class permits the use of simple shape-based features in this step. The objects labeled as *round-fruit* are passed on to the last stage, the discriminant separating apples and lemons which may leverage informative color features.

### III. ROC ANALYSIS FOR HIERARCHICAL CLASSIFIERS

We consider a statistical classifier as a composition of a model returning *soft output* such as estimated class posteriors, and a decision function converting this soft output into a decision. The classifier *operating point*  $\phi$  is defined by the parameters of its decision function such as the per-class weights for a discriminant or the threshold for a detector [7]. ROC analysis derives a set of relevant operating points from a labeled test set and estimates desired performance measures at these points [8].

In order to perform ROC analysis for a hierarchical classifier with  $N$  nodes, we need to define the operating point of such system. Because our hierarchical system is decoupled, we can estimate an ROC for each node classifier separately. We define the *system operating point* as a tuple of per-node operating point indices  $\phi_{\text{sys}} = \{\phi_n\}_{n=1}^N$ . Therefore, the hierarchical classifier ROC is composed of a set of system operating points representing the  $N$ -ary Cartesian product between per-node sets of operating points.

We propose the ROC analysis scheme for hierarchical classifiers using a two-step procedure. In the first step, individual node ROCs are estimated using the design data set. In the second step, the operating point of the hierarchical classifier is optimized according to application-specific costs.

- 1) *Per-node ROC estimation* (arbitrary node order)
  - Define the node data set using specific features and subsets of classes or meta-classes.
  - Define the node model training and validation sets.
  - Train the model on the node training set.
  - Estimate model soft outputs on the val.set.
  - Estimate ROC from the soft outputs.
- 2) *Optimization of the system operating point* using the system ROC optimizer

### IV. ROC OPTIMIZER FOR HIERARCHICAL SYSTEMS

In the proposed algorithm, we do not evaluate entire ROC of a hierarchical system explicitly. Instead, we use an ROC optimizer identifying the system operating point  $\phi_{\text{sys}}$  minimizing the loss  $L$  of the hierarchical classifier on a given

test set:

$$L(\phi_{\text{sys}}) = \sum_{i=1}^C P(\omega_i) \sum_{j=1, i \neq j}^C \text{CM}_{i,j} M_{i,j} - \sum_{k=1}^C P(\omega_k) \text{CM}_{k,k} M_{k,k}, \quad (1)$$

where  $\text{CM}$  represents normalized confusion matrix at the system operating point  $\phi_{\text{sys}}$ ,  $M$  the cost matrix of the same dimension, and  $P(\omega)$  the prior probability of the class  $\omega$ .

---

#### Algorithm 1 System ROC greedy optimizer

---

- 1: **Input:** Trained hierarchical classifier with per-node ROCs, labeled data set, cost matrix  $M$ .
  - 2: Find initial system operating point  $\phi_{\text{init}}$  by minimizing the loss for a set of  $K$  randomly selected system operating points.
  - 3: Cycle through nodes.
  - 4: Compute the set of system loss values  $L$  for all operating points at the node  $n$  while fixing the operating points at remaining nodes.
  - 5: Update  $\phi_{\text{sys}}$  minimizing  $L$ .
  - 6: Repeat until  $\phi_{\text{sys}}$  does not change.
  - 7: **Output:** System op.point  $\phi_{\text{sys}}$  and its loss  $L$ .
- 

The proposed optimizer first selects a suitable initial system operating point using random sampling. Then it performs a node-per-node greedy search evaluating the system loss on all operating points of a given node while fixing the operating points at remaining nodes.

## V. EXPERIMENTS

To demonstrate the proposed algorithm we use the artificial *Fruit* data set with four classes, namely *apple*, *banana*, *lemon* and *outlier*. The Fruit problem imitates a typical sorting application where objects on the conveyor belt need to be classified while rejecting possible outliers. The outlier class is not well represented during classifier design stage as arbitrary unseen outlier objects may appear in production. We simulate this by separating the training set and the test set used only for performance evaluation, see Figure 2. All classes in the training set are generated using bi-variate Gaussian distributions. The *outlier* class in the test set is composed of a Gaussian mode and additional uniformly-distributed examples.

#### A. Optimizing hierarchical classifier

The hierarchical classifier is constructed according to the scheme depicted in Figure 1. It comprises the Parzen-window based fruit detector, the quadratic discriminant assuming normal densities (QDC) for the *banana/round-fruit* classifier, and the linear discriminant assuming normal densities (LDC) for the last node.

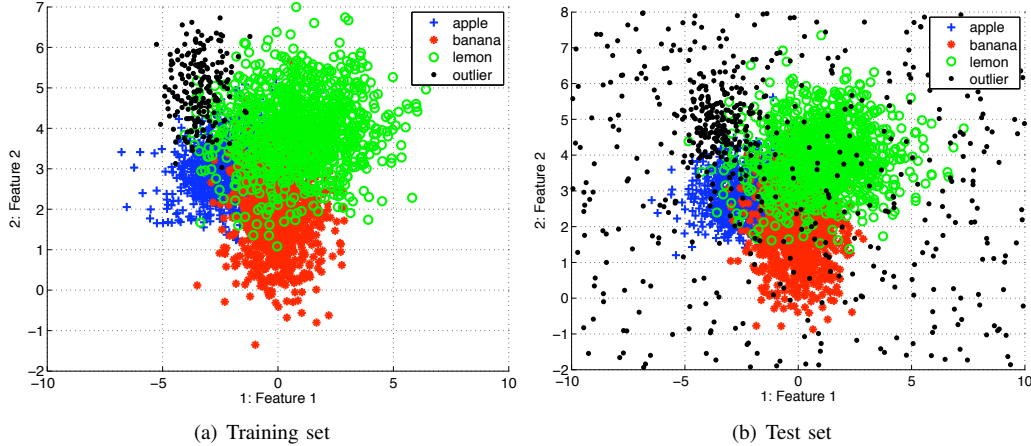


Figure 2. Training and test sets in the Fruit problem.

The hierarchy is trained using a random subset with 200 samples per class drawn from the training distribution. To avoid bias, the node models and ROCs are estimated on different 50% of the training data. The system operating point is then optimized using the proposed greedy scheme employing the complete training set. The node ROCs of the trained hierarchical classifier contain 400, 238, and 166 operating points, respectively. In all experiments, the search algorithm was initialized using  $K = 2000$  randomly selected points.

We first optimize the system using the cost matrix with equal costs. The decisions at the resulting system operating point are shown in Figure 3(a). The system confusion matrix on the test set is:

True Labels	Decisions				Totals
	apple	banana	lemon	outlier	
apple	0.793	0.046	0.118	0.043	1.00
banana	0.155	0.593	0.201	0.051	1.00
lemon	0.107	0.038	0.715	0.140	1.00
outlier	0.087	0.037	0.045	0.832	1.00

To illustrate the cost-sensitive system optimization, we define the second cost matrix increasing the costs for fruit rejection and for the *bananalemon* entry:

1	1	1	5
1	1	5	5
1	1	1	5
1	1	1	1

Invoking the cost optimization algorithm with the updated specification, we obtain the new system operating point resulting in the following confusion matrix:

True Labels	Decisions				Totals
	apple	banana	lemon	outlier	
apple	0.879	0.083	0.022	0.016	1.00
banana	0.197	0.737	0.052	0.014	1.00
lemon	0.260	0.073	0.606	0.061	1.00
outlier	0.187	0.050	0.040	0.723	1.00

Figure 3(b) shows the corresponding decisions. We may see that the detector decision boundary expanded and both discriminant boundaries shifted. Note that the *bananalemon* error is not intrinsically handled by any of the three node classifiers. This illustrates how the hierarchical classifier ROC allows us to work at the system level.

Finally, Figure 3(c) shows the decisions of the monolithic discriminant trained on all available classes including the known outliers. Naturally, it accepts unexpected outliers in the test set into one of the fruit classes.

### B. Learning curves

In the second experiment, we investigate classifier behaviour. We attempt to answer two questions. Firstly, is it beneficial to train the node models and estimate the node ROCs on different subsets of training data? In addition to the algorithm performing 50/50 split of node training data, we also include the algorithm re-using the entire node training set for both model training and ROC estimation. Secondly, we compare the hierarchical system to two types of monolithic classifiers. The first one trains QDC on four classes including outliers. The second one is training QDC only on the three *fruit* classes and adds a reject option with the threshold set to reject 1% of fruit samples. This effectively approximates optimal (Bayes) solution for our test set. The ROC analysis and the cost-sensitive optimization of all algorithms was performed in Matlab 7.5 using PRSD Studio package<sup>1</sup>.

Figure 4 shows the learning curves with mean loss and standard deviation of mean loss computed over 30 experiments. We observe that the four-class QDC provides the worst solution. This was expected as it cannot cope with the additional outliers in the test set. Although the hierarchical classifiers do not train a model on the outlier class, they provide better protection for the fruit distribution due to the

<sup>1</sup><http://prsdstudio.com>

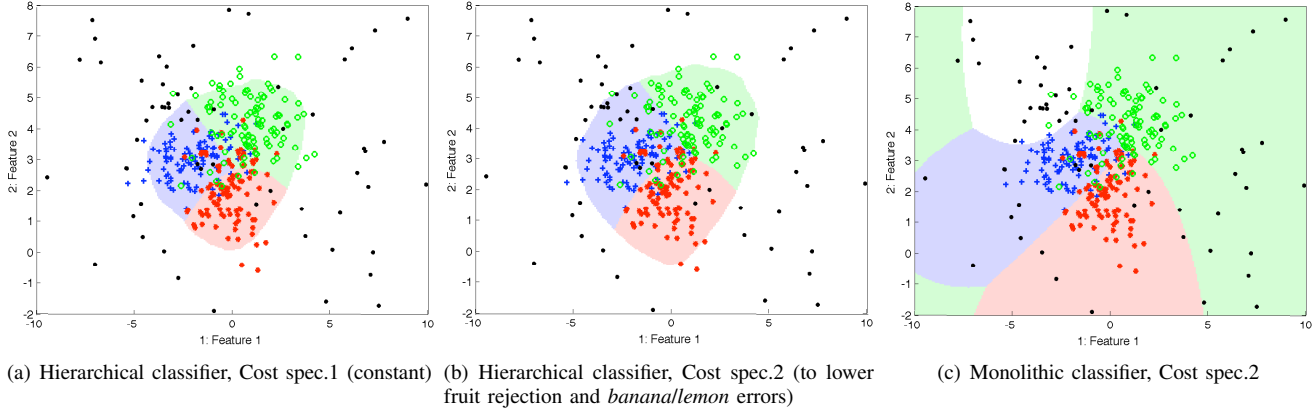


Figure 3. Decisions of a hierarchical system and of a monolithic discriminant

dedicated detector stage. It appears that estimating unbiased node ROCs using 50/50 node data split is a slightly better strategy.

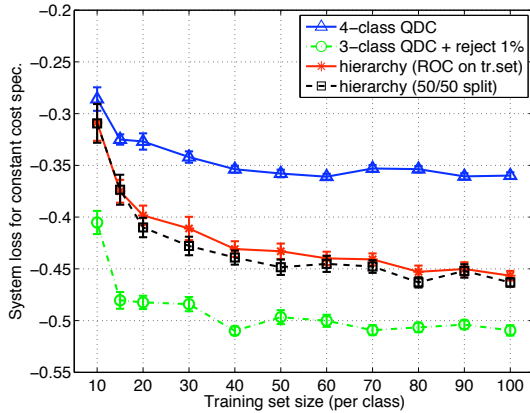


Figure 4. Learning curves for constant cost matrix (smaller loss is better).

We have also compared the proposed greedy optimizer with the full search varying the training set sizes for the constant cost specification. In total, more than 15 million operating points were evaluated in the full search. In all cases, the greedy algorithm found the optimal solution identified by the full search. The greedy search naturally requires only a tiny fraction of operating point evaluations (e.g. for data set with 70 samples per class, it evaluated only 3158 from the total of 6.2 million system op.points).

The optimization of the hierarchical fruit classifier on the data set with 3000 samples takes under 5 minutes on 2.8 GHz laptop.

## VI. CONCLUSIONS

In this paper, we propose a scalable algorithm for optimization of decoupled hierarchical classifiers, thus providing a novel approach for tackling certain types of complex classification problems. It allows the designer to simplify

the complex problem by decomposition, leveraging the most appropriate features and models for each of the sub-problems while still tuning the entire system using a cost-sensitive optimization scheme.

It is to be expected that for more complex problems and complicated cost specifications, the suboptimal nature of the greedy search may become apparent, but in this case the exhaustive approach becomes computationally intractable.

## REFERENCES

- [1] S. Kumar, J. Gosh, and M. M. Crawford, "Hierarchical fusion of multiple classifiers for hyperspectral data analysis," *Pattern Analysis & Applications*, vol. 5, pp. 210–220, 2002.
- [2] Y. C. Wang and D. Casasent, "A hierarchical classifier using new support vector machine," in *Proc. of Int.Conf.on Doc.Analysis and Recognition (ICDAR'05)*, vol. 2, 2005, pp. 851–855.
- [3] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla, "Hand pose estimation using hierarchical detection," in *Computer Vision in Human-Computer Interaction*, vol. LNCS 3058, 2004, pp. 105–116.
- [4] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.
- [5] T. C. W. Landgrebe, D. M. J. Tax, P. Paclík, and R. P. W. Duin, "The interaction between classification and reject performance for distance-based reject-option classifiers," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 908–917, 2006.
- [6] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [7] P. Paclík, C. Lai, J. Novovicova, and R. P. W. Duin, "Variance estimation for two-class and multi-class roc analysis using operating point averaging," in *19th Int.conf.on Pat.Rec. (ICPR 2008, Tampa, Florida, USA)*. IEEE Press, 2008.
- [8] T. C. W. Landgrebe and P. Paclík, "The ROC skeleton for multiclass ROC estimation," *Pattern Recognition Letters*, vol. 31, no. 9, pp. 949–958, July 2010.