Linear model combining by optimizing the Area under the ROC curve

David M.J. Tax and Robert P.W. Duin
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: D.M.J.Tax@ewi.tudelft.nl

Yulia Arzhaeva
Image Sciences Institute, University Medical Center Utrecht,
Heidelberglaan 100, 3584 CX Utrecht, The Netherlands

Abstract

In some classification problems, like the detection of illnesses in patients, classes are very unbalanced and the misclassification costs for different classes vary significantly. Then it is better not to minimize the classification error, but to optimize the ordering of the data, or to optimize the Area under the ROC curve (AUC). In this paper we propose to optimize a linear combination of features (or base model outputs) by optimizing AUC. The advantages are that a relatively small training set is required for the optimization and that the training set can have a large class imbalance. Furthermore, the classifier does not make distributional assumptions, making it very suitable to combine the outputs of base classifiers. In the application of the detection of interstitial lung diseases it is shown to be very advantageous and to outperform standard classification rules.

Keywords: chest radiography, pattern recognition, combining classifiers, class imbalance, area under the ROC curve

1 Introduction

In medical detection problems or other screening applications one often has to deal with imbalanced class priors or misclassification costs. For these problems, the Area under the ROC curve (AUC) is often a more suitable error measure than the classification error [3]. Not only is it insensitive to class priors and costs, it also appears to be more stable for small sample sizes. There are some proposals for optimizing the AUC, for instance the support vector classifier[8] or decision trees[5]. In [9] a linear classifier is proposed that is a simplified version of [8], that is sparse in the features and is simpler to optimize.

In this paper we propose to use this linear weighting of models for the detection of interstitial lung diseases (ILD) in radiographs. These illnesses often reveal themselves by a changed textural appearance of the lungs with respect to healthy lungs. Although healthy lungs are relatively simple and more reliably to be found, examples of ill lungs are often unreliable. This problem has therefore skewed class priors and clearly also very different misclassification costs per class. Furthermore, the interpretation is very challenging because the image contains several superimposed anatomical structures. Not only do the different structures occlude each other, the tissue textures interfere and create artifacts. Therefore, classifying these diseases is one of the most difficult tasks for a chest radiologist. Constructing a classifier that aids the radiologists and that can be trained using a small set of ill patients is therefore a big challenge.

The proposed approach to classify patients, is to make density models for both healthy and ill patients, and combine these densities using a linear classifier that optimizes the AUC. In 2 we explain the AUC optimization. Then in section 3 we discuss the ILD data, and the different approaches to solve the classification problem. Finally, we show some experiments and results in section 4.

2 AUC optimization

Assume we have data $\{\mathbf{x}_i, y_i\}, i = 1..N$, where $y_i \in \{-1, +1\}$, and we would like to fit a model $h(\mathbf{x})$ to this data that optimizes the the Area under the ROC curve (AUC) [7, 10]. This measure counts how often an object of class $+1$ ($\mathbf{x}_+$) is ranked higher than an object of class -1 ($\mathbf{x}_-$):

$$\text{AUC} = Pr(h(\mathbf{x}_+) > h(\mathbf{x}_-)). \tag{1}$$

Clearly, a perfect separation of the two classes is obtained when $AUC = 1$. The AUC performance can be simply estimated on a finite dataset $\mathcal{X}^{tr}$. Assume that counter k^+

runs over all objects with $y_i = +1$, and k^- runs over all objects with $y_i = -1$. Then, the AUC can be computed as follows:

$$1-\hat{e}_{auc}(h, \mathcal{X}^{tr}) = \frac{1}{n^+ n^-} \sum_{k^+=1}^{n^+} \sum_{k^-=1}^{n^-} \mathbf{1}\left(h(\mathbf{x}_{k^+}) > h(\mathbf{x}_{k^-})\right). \tag{2}$$

Note that for the optimization of this performance, no reference is being made to a classification threshold θ. For the practical application of the classifier, an operating point has still to be defined by choosing a specific threshold θ.

To define a practical classifier, we assume a linear classifier, $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. For the linear classifier we can cast the optimization problem in a linear programming form similar to the l_1-SVM, [2][1]:

$$\min \|\mathbf{w}\|_1 + C \sum_{k^+} \sum_{k^-} \xi_{k^+k^-} \tag{3}$$

$$\text{s.t.} \quad \forall k^+, k^- : \mathbf{w}^T(\mathbf{x}_{k^+} - \mathbf{x}_{k^-}) \geq 1 - \xi_{k^+k^-},\ \xi_{k^+k^-} \geq 0.$$

This can easily be rewritten in a linear programming formulation:

$$\min \sum_i (u_i + v_i) + C \sum_{k^+} \sum_{k^-} \xi_{k^+k^-} \tag{4}$$

$$\text{s.t.} \quad \forall k^+, k^- :\ (\mathbf{u}^T - \mathbf{v}^T)(\mathbf{x}_{k^+} - \mathbf{x}_{k^-}) \geq 1 - \xi_{k^+k^-}, \tag{5}$$

$$\xi_{k^+k^-} \geq 0, \quad \forall i :\ u_i \geq 0,\ v_i \geq 0.$$

A threshold θ is not defined, but can be derived when misclassification costs and class priors have been supplied. We will refer to this as the optimized AUC Linear Programming classifier, or AUC-LPC.

Another similar classifier that directly optimizes the AUC is the support vector machine as defined in [8]. It minimizes the l_2 norm of $\mathbf{w}$ with constraints on the ordering of the objects. The optimization problem is defined as follows:

$$\min \|\mathbf{w}\|^2 + C \sum_{k^+} \sum_{k^-} \xi_{k^+k^-} \tag{6}$$

$$\text{s.t.} \quad \forall k^+k^- : h(\mathbf{x}_{k^+}) - h(\mathbf{x}_{k^-}) \geq 1 - \xi_{k^+k^-}, \xi_{k^+k^-} \geq 0. \tag{7}$$

By introducing a kernel, thanks to the self-duality of the l_2-norm, the above formulation remains valid for nonlinear SVM as well [8].

Notice that the constraints (5) and (7) express the pairwise differences between objects from different classes $\mathbf{x}_{k^+} - \mathbf{x}_{k^-}$. The slack variables $\xi_{k^+k^-}$ in these constraints approximate the indicator function $\mathbf{1}$ that is part of (2). This approximation has the serious drawback that the number of constraints is quadratic in the number of objects, so it becomes very large. To cope with this, different strategies can be constructed. The first strategy is to start by *randomly drawing objects* from both classes, and iteratively update this set by considering the objects that are violating many constraints[1]. In this case only the very hard objects are considered. To avoid that the classifier flips its labels (because only the bad constraints are considered), this set is extended to include also well-ordered object pairs.

In [8] a second sub-sampling strategy is suggested. Here only the *objects and their nearest neighbors from the other class* are considered. The user has to define a number m indicating how many neighbors are assumed to be informative. Only the constraints between each object $\mathbf{x}$ and m of its nearest neighbors are imposed. This sub-sampling heuristic can fail, in particular when there is a large overlap and the number of neighbors is not sufficiently large. Furthermore, extra problems may occur when the features are poorly scaled and the distance between the objects does not reflect the classification problem well. In that case the nearest neighbors are not distributed in directions that are required for a high AUC performance.

In this paper we use a third constraint sub-sampling approach. By utilizing the linear programming formulation, it is very simple to *randomly subsample the constraints* in (5) instead of sampling the objects. This random sampling with M constraints avoids the focus on the local structure in the data (as given by the m nearest neighbors), but characterizes the structure on a larger scale. Therefore, it will give a less biased estimation of the optimal solution based on all constraints.

3 Interstitial lung disease detection

The task is to classify patients as being healthy or being ill, based on the classification of the individual pixels in the radiographs. First a pixel classifier has to be trained. Due to the high number of pixels, it is hard to construct a single classifier. Therefore the pixel classifier will actually be a combination of models, where each of the models is trained on one patient. To obtain one outcome per patient, the individual pixel classifications have to be combined. In table 1 the notation of the mathematical symbols is given.

Table 1. Notation used in combining models.

$\mathbf{x}$	158-dimensional pixel feature vector
$\mathcal{X}_i = \{\mathbf{x}_{(i)}\}$	the set of pixel vectors from patient i
$f(\mathbf{x})$	pixel classifier
$f_i(\mathbf{x})$	pixel classifier trained on $\mathcal{X}_i$
$h(\mathbf{x})$	pixel classifier that combines several $f_i(\mathbf{x})$
$g(\mathcal{X}_i)$	classifier that combines $h(\mathbf{x})$ for all pixels in $\mathcal{X}_i$

[1] We use the same notation for the counters k^+ and k^- as in the previous section.

The experiments were performed on a database obtained at the University of Chicago hospitals. It contains 100 normal PA chest radiographs and 100 abnormal radiographs with interstitial disease. The normal cases were selected based on consensus of independent review on each radiograph by four experienced radiologist. The abnormal cases were selected based on radiologic findings, CT, clinical data and/or follow-up radiographs, by consensus of the same radiologists. Some images contain artifacts due to clothing or catheters. The radiographs were digitized to 2000 by 2000 pixels with 0.175 mm pixel size and 10 bits intensity. A chest radiologist classified each upper, middle and lower lung field of the abnormal cases as normal, possibly abnormal and definitely abnormal. When an image contains possibly abnormal or definately abnormal areas it will be labeled diseased.

From the radiographs, 30 features per pixel are computed, including intensity and Gaussian derivative features on different scales. Furthermore, for each pixel also some local statistics (the mean, standard deviation, skew and kurtosis) in a circular region around the pixel was computed. Together with the own pixel features, two positional features and a binary feature indicating if a pixel is inside a rib or not (found by applying a rib segmentation [6]), this results in 158 features.

First we are going to model the lungs of the training patients. This will result in N_t trained models. These models are used to characterize new pixels by N_t new features (model outputs). To make a model of each of the patients, the distribution of the pixels from the healthy or ill patient are estimated using a mixture of Gaussians model, containing K Gaussian distributions:

$$f_i(\mathbf{x}) = \sum_{k=1}^{K} p_{(i),k} \mathcal{N}(\mathbf{x}; \mu_{(i),k}, \Sigma_{(i),k}) \tag{8}$$

where $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ is the Gaussian distribution with mean μ and covariance Σ, and index i runs over all patients: $i = 1, ..., N_t$. The model parameters $p_{(i),k}, \mu_{(i),k}, \Sigma_{(i),k}$ are optimized on the data from patient i, $\mathcal{X}_i$, using Expectation-Minimization [4]. Because we are operating in a relatively high dimensional feature space ($p = 158$), the covariance matrices have to be regularized, in order to be able to invert these covariance matrices: $\tilde{\Sigma}_i = \Sigma_i + 0.01 \cdot \Sigma_{\text{total}}$.

When N_t models f are available, a pixel is characterized by the set of model outputs. A combined pixel classifier has to be defined to provide one output per pixel:

$$h(\mathbf{x}) = h(f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{N_t}(\mathbf{x})) \tag{9}$$

We can use fixed combining rules, like the mean or product combination rule, but the fact that we can make numerous models which may become redundant, and the fact that we have a large sample size, suggest that a trained combiner may be more suitable. Unfortunately, in this setup we assume that outlier examples are not very reliably labeled. Therefore we will use a more advanced AUC optimization.

In order to classify a patient, the classification outputs $h(\mathbf{x})$ for all the pixels have to be integrated to one output $g(\mathcal{X}_i)$. Due to the varying size of the lungs, a simple combination rule is used that is independent on the number of pixel classifications. This rule first sorts the outputs $h(\mathbf{x})$ for all the pixels $\mathcal{X}_i$ of patient i. Then the 95% or the 99% percentile output is used as the final output $g(\mathcal{X}_i)$ for patient i. This means that when at least 5% or 1% of the pixels is classified as being ill with high confidence, the output for this patient will be label 'ill'.

4 Results

To compare the classification performances, we first train simple standard classifiers and one-class classifiers on the same data, but without making separate models per lung. The supervised classifiers are the Linear discriminant analysis (LDA) and the quadratic discriminant (QD), the one-class classifiers are the single Gaussian and finally Mixture of Gaussian models using 5 clusters per class (MoG5+5). All the data (from all the patients, containing both healthy and ill examples) will be pooled into one dataset. On this data a classifier $f(\mathbf{x})$ is trained. To make the training feasible (in terms of memory usage), only 1000 pixels per patient are used. The pixels from one patient are again combined using a combination rule $h(\mathcal{X}_i)$.

Table 2. The AUC performances ($\times 100$) obtained from training a single classifier $f(\mathbf{x})$ on a dataset containing pixels from all training patients. Results are averaged over 10 runs, standard deviations are around 15.0.

	0.5	0.95	0.99
LDA	68.4	73.8	62.0
QC	64.4	70.3	57.8
Gaussian OC	66.1	74.6	75.5
MoG 5+5	56.7	48.8	48.9
AUC-LPC	81.6	73.1	67.8

Table 2 shows the AUC performances for the six classifiers and for three different percentile values (see section 3). 90 healthy and 90 ill patients are used for training the classifier. The different classifier have similar performance, but the absolute performance is not satisfactory with all AUC's below 0.80, and it appears not to improve significantly when more training patients are used. Performance improves with

increasing complexity of the model, at the expense of much larger variances in performance.

Now we can compare these results with classifiers that combine the base pixel classifiers (given by (8)). The densities were estimated by using a Mixture of Gaussians using three clusters. In table 3 the AUC results for six classifier combiners are shown. The first two combiners are trained classifiers, the LDA and QD again, the second two are the mean and the product combination rules. Finally the L_1 support vector machine (L1-SVM) and the AUC linear classifier (AUC-LPC) are used to combine the base classifiers. The L1-SVM is closely related to the AUC-LPC, but it minimizes the error instead of maximizing the AUC. The pixel classifications are integrated into a final score by utilizing the 50%, 95% and 99% percentile pixel output value.

Table 3 shows the results using a varying number of training patients. In general, the supervised classifiers like LDA and QD show very poor performance. Only when a few ill patients are used for training, and the 50 percentile pixel-output is used, results become a bit better than random. Using more ill patients actually confuses the classifiers and deteriorates the results a bit more (although not very significant, the variance is very high). The same can be observed for the L1-SVM The fixed combining rules are not suitable for this problem, and give in some cases worse than random results. The AUC-LPC performs well in particular when more ill patients are used for training. Furthermore, the variance of the outcomes from the AUC-LPC is also lower than that of the other classifiers.

Table 3. AUC performances ($\times 100$) of 6 different classifiers, using a varying number of healthy and ill training patients. Results are averaged over 10 runs.

training patients	10% ill	50% ill	90% ill
pixel combining using 50 percentile (median)			
LDA	77.1 (16.5)	72.3 (19.1)	71.0 (19.1)
QC	77.1 (16.5)	68.8 (19.7)	67.8 (19.8)
average	76.5 (17.6)	70.1 (20.0)	68.9 (19.9)
product	71.8 (18.4)	67.8 (20.7)	64.7 (20.8)
L1-SVM	77.1 (16.5)	68.6 (19.6)	67.8 (19.8)
AUC-LPC	**86.4 (10.0)**	**95.6 (4.1)**	**95.5 (4.5)**
pixel combining using 95 percentile			
LDA	70.2 (17.6)	61.0 (18.2)	59.4 (15.4)
QC	68.6 (18.6)	61.4 (17.1)	59.0 (14.8)
average	52.6 (16.4)	52.4 (17.3)	52.3 (17.3)
product	53.8 (16.8)	52.2 (16.7)	51.8 (17.1)
L1-SVM	68.1 (18.8)	61.3 (17.7)	58.9 (15.6)
AUC-LPC	**90.4 (6.7)**	**93.9 (5.6)**	**94.1 (6.2)**
pixel combining using 99 percentile			
LDA	68.3 (20.1)	59.7 (21.9)	58.6 (21.5)
QC	68.4 (20.0)	59.7 (22.1)	58.7 (19.8)
average	59.1 (18.1)	53.6 (20.2)	48.8 (19.2)
product	58.8 (18.3)	51.5 (21.4)	48.8 (19.6)
L1-SVM	68.2 (20.1)	59.5 (22.0)	59.5 (20.5)
AUC-LPC	**87.7 (7.5)**	**93.8 (5.2)**	**92.0 (7.8)**

5 Conclusions

In this paper we discussed a linear classifier that optimizes the AUC. It can be trained on data with a large class imbalance and small sample sizes. Although the linear classifier in itself is not very complex, when the features are a complex function of the data, it can solve very complicated classification problems. In the application of the detection of interstitial lung diseases it is shown to be very advantageous and to outperform standard classification rules. An open issue in this research is the way in which the individual pixel classifications for one patient should be combined.

Acknowledgments This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

References

[1] K. Ataman and W. Street. Optimizing area under the ROC curve using ranking svms. In *KDD'05*, 2005. Under review.

[2] K. Bennett and O. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–24, 1992.

[3] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[4] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. B*, 39:1–38, 1977.

[5] C. Ferri, P. Flach, and J. Hernandez-Orallo. Learning decision trees using the area under the ROC curve. In *Proceedings of the ICML*, 2002.

[6] B. v. Ginniken and B. Romeny. Automatic segmentation of lung fields in chest radiographs. *Med.Phys.*, 27(10):2445–2455, 2000.

[7] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, VIII(4), October 1978.

[8] A. Rakotomamonjy. Optimizing AUC with support vector machine. In *European Conference on Artificial Intelligence Workshop on ROC Curve and AI*, 2004.

[9] D. Tax and C. Veenman. Tuning the hyperparameter of an auc-optimized classifier. In *17th Belgium-Netherlands conference on artificial intelligence*, pages 224–231, 2005.

[10] L. Yan, R. Dodier, M. C. Mozer, and R. Wolniewicz. Optimizing classifier performance via the Wilcoxon-Mann-Whitney statistic. In *The Proceedings of the ICML*, pages 848–855, 2003.