Patient-Specific Modeling of Medical Data

Guilherme Alberto Sousa Ribeiro[1(✉)], Alexandre Cesar Muniz de Oliveira[1], Antonio Luiz S. Ferreira[1], Shyam Visweswaran[2], and Gregory F. Cooper[2]

[1] Departament of Informatics, Federal University of Maranhao, Av. of Portugueses, São Luís, MA 1966, Brazil
{guilherme.ufma,aluizsf22}@gmail.com, acmo@deinf.ufma.br
[2] Department of Biomedical Informatics, 5607 BAUM, University of Pittsburgh, Boulevard BAUM 423, Pittsburgh, PA 15206-3701, USA
{shv3,V.gfc}@pitt.edu

Abstract. Patient-specific models are instance-based learn algorithms that take advantage of the particular features of the patient case at hand to predict an outcome. We introduce two patient-specific algorithms based on decision tree paradigm that use AUC as a metric to select an attribute. We apply the patient specific algorithms to predict outcomes in several datasets, including medical datasets. Compared to the standard entropy-based method, the AUC-based patient-specific decision path models performed equivalently on area under the ROC curve (AUC). Our results provide support for patient-specific methods being a promising approach for making clinical predictions.

Keywords: Classification problems · Approach instance-based · Area under the roc curve

1 Introduction

Clinical decision-making may be improved by using predictive models [1]. Predicting patient outcomes under uncertainty constitute an important healthcare problem. Enhanced decision models can lead to better patient outcomes as well as efficient use of healthcare resources.

The typical paradigm in predictive modeling is to learn a single model from a database of patient cases, which is then used to predict outcomes for future patient cases [2]. This approach is known as *population-wide model* because it is intended to be applied to an entire population of future cases. Examples of popular population-wide methods are decision trees, logistic regression, neural networks and Bayesian networks.

In contrast to that general approach, a *patient-specific model* consists of learning models that are tailored to the particular features of a given patient. Thus, a patient-specific model is specialized to the patient case at hand, and it is optimized to predict especially well for that case. Moreover, patient-specific models can also be seen as examples of instance-based learning schemes, of which k-nearest neighbor, local regression and lazy decision trees are examples.

P. Perner (Ed.): MLDM 2015, LNAI 9166, pp. 415–424, 2015.
DOI: 10.1007/978-3-319-21024-7_29

Instance-based algorithms learn a distinct model for a test instance and take advantage of the features in the test instance to learn the model [3]. Typically, the instance-based algorithms are lazy, since no model is constructed a priori before a test instance becomes available, and a model is learned only when a prediction is needed for a new instance [4]. In contrast, algorithms that learn population-wide models are eager since such explicitly build a model form the training data before a test instance becomes available.

There are several advantages of patient-specific models over population-wide methods. For instance, patient-specific models may have better predictive performance for taking advantage of any particular characteristic of the case at hand, whereas population-wide methods converge to an average method, derived for an entire population. Second, a patient-specific model structure is usually simpler than that of its population-wide counterpart. Thus, a patient-specific model can provide a more succinct explanation of its decision. Third, the construction patient-specific models may be computationally faster, though this advantage is offset by the observation that a patient-specific method has to construct a distinct model for each patient case of interest while a population-wide method has to construct just a single model. Finally, the task of handling of missing features is simplified on patient-specific approach.

In this paper, we investigate the performance of two patient-specific methods, based on the lazy decision tree approach. We compare the performance of the AUC-based patient-specific methods with the entropy-based model. We focus on the discriminative performance of the three methods and evaluate them using the area under the ROC curve (AUC) [5].

The remainder of this paper is organized as follows. Section 2 presents background and related work on instance-based methods. Section 3 provides details of the patient-specific decision path algorithms that we have developed. Section 4 describes the datasets, experimental methods and presents and discusses the results of the patient-specific decision path algorithm on several datasets. Section 5 presents our conclusions.

2 Background

The canonical instance-based method is a kind of nearest-neighbor technique, that is, the most similar training instance to a given the test instance is located its target value is returned as the prediction [6].For a test instance, the k-Nearest Neighbor (KNN) method, for example, selects the k most similar training instances and either averages or takes a majority vote of their target values. Modified version of kNN have been applied successfully to medical databases for diagnosis and knowledge extraction [7]. Other instance-based methods are not as reliant on a similarity measure as is the case for the nearest-neighbor methods, taking advantage of the values of the predictors in the test instance to learn a model.

Friedman et al. have described the LazyDT method [4] that searches for the best CART-like decision tree for a test instance. When compared to standard

population-wide methods, for inducing decision trees, as ID3 and C4.5, LazyDT does not perform pruning, handles only discrete variables, and has higher accuracies on average.

Zheng et al. have developed an instance-based method called the Lazy Bayesian Rules (LBR) learner that builds a rule tailored to the values of the predictors of the test instance, used to classify it [8]. A LBR rule consists of: (1) a conjunction of the features (predictor-value pairs) present in the test instance as the antecedent, and (2) a consequent nave Bayes classifier that consists of the target variable as the parent of all other predictors that do not appear in the antecedent. The classifier parameters are estimated from those training instances that satisfy the antecedent. A greedy step-forward search selects the optimal LBR rule for a test instance to be classified. LBR uses values of predictors in the test instance to drive the search for a suitable model in the model space and, when compared to a variety of population-wide methods, LBR has reached higher accuracies on average.

Visweswaran et al. have developed and applied an instance-based algorithm that performs Bayesian model averaging over LBR models [2,9], using the features of the test case to drive the search. The prediction for the test case target variable is obtained by combining the predictions of the selected models weighted by their posterior probabilities. This method has obtained higher accuracies than LBR on a range of non-medical datasets and also performed better than several population-wide methods on a pneumonia dataset, when evaluated within a clinically relevant range of the ROC curve. Furthermore, instance-based algorithms that use the test instance to drive the search over a space of Bayesian network models have been developed and applied to patient-specific modeling with good results [10,11].

Ferreira et al. developed patient-specific decision path (PSDP) algorithms that can build a decision path to predict patient outcome [12]. Given a patient for whom the values of the features are known, these algorithms construct a decision path using a subset of those features. Two selection criteria were investigated for selecting features: balanced accuracy and information gain. Results obtained with those algorithms using clinical datasets were compared with CART using the AUC metric.

3 Patient-Specific Decision Path Algorithms Based on AUC

The proposed patient-specific decision path algorithm uses AUC as a metric to select patient‘s features that will compose the path [13]. The Area under the ROC Curve (AUC) is a widely used measured of performance of supervised classification rules. It has the attractive property of circumvent the need to specify misclassification costs.

The use of AUC as a metric to select attributes in a decision tree was introduced by Ferri and colleagues [14,15]. Based on the optimal choice of possible labellings of the tree, the AUC-split criterion leads to good AUC values, without

```
PSDP-AUC-Split (labels, dataset, testset)
  INPUT: labels contain a set of possible labels to predict,
          dataset is database,
          testset is a test case to predict
  OUTPUT: an array with the label, estimates probability of the label and the path.

  LOOP: until dataset to be empty, all cases in dataset have the same target or the set of atributes to be empty

      FOR each testeset(i)
          subset = getSubSet(dataset, testset(i))
          diff_subset = dataset - subset

          partition(1) = counting of positive and negative values of subset
          partition(2) = counting of positive and negative values of diff_subset

          coordinates = sort(partition, 'descend')

          fpr = false positive rate based on negative coordinates matrix
          tpr = true positive rate based on positive coordinates matrix

          AUC = trapz(fpr, tpr) %this function calculate the area of trapezoid formed by coordinates

          v_AUC(i) = AUC

          IF AUC < BestAUC THEN
                BestAUC = AUC
          END IF
      END FOR

      path = attribute selected according BestAUC
      dataset = dataset according attribute selected
      dataset = remove atribute selected
      eProb_path = calculate probability of predicted label to belong to positive or negative class
      predicted_label = max(eProb_path)
  END LOOP
RETURN eProb_path, predicted_label and path
```

Fig. 1. Pseudocode for PSDP-AUC-Split algorithm.

compromising the accuracy if a single labelling is chosen. Thus, for a two class classification problem and a tree with n leaves, there are 2^n possible labellings, of which $n+1$ are optimal. That optimal labelling gives the convex hull for the considered leaves. Figure 1 shows the pseudocode of the patient-specific decision path that uses the AUC-split proposed by Ferri.

This algorithm receives as parameters the dataset, the label of each dataset instance and the test instance in which you want to classify. As shown in Fig. 1, the first step of the algorithm is to select an attribute of the test instance by times, calculate the partitions according to the value of this attribute in this instance using as reference the training dataset in order setting the partitions of each one of subsets (number of positive and negative cases). The matrix *coordinates* receives the matrix *partition* sorted in decreasing so that the AUC value obtained is the maximum. The calculation of the AUC is done by the

function *trapz* [16], written in Matlab [17], using as parameters the false positive rate and true positive rate which are calculated based on *coordinates* matrix. After this calculation, the value is stored in *v_AUC* vector. All these steps will be repeated for the other attributes. At the end of the calculation for all, the attribute with best AUC value will be selected and so that one of the stopping conditions is true, the path will be complete together the predicted label.

In contrast, another alternative way to use the AUC metric to select patient features requires a prediction for class probability or some other score as proposed by Fawcett. In this case, a leave-one-out cross-validation scheme was employed in order to generate that probability estimate and further calculating a Mann-Whitney-Wilcoxon test statistic, which direct relates to the AUC. To avoid overfitting, Laplace smoothing [18] was employed when estimating class probabilities. The patient-specific decision path that uses this standard approach just described is shown in the pseudo-code of Fig. 2.

The main difference between PSDP-STD-AUC and PSDP-AUC-Split algorithms are the steps prior to the calculation of the AUC. An operation probability estimates is performed for all cases of dataset. However, these estimates are calculated according to the subset of training instances that have the same value of the test instance and the subset of training instances that have different values of the test instance. After the calculations of these estimates according to Laplace smoothing, the AUC is calculated by the function *colAUC* [19] written in Matlab [17], using as parameters all probability the true class of the training set, accompanied by the labels of each instance. The following steps of this algorithm are identical to the steps described in the previous paragraph for the PSDP-AUC-Split algorithm.

4 Experimental Results

In this section we presented the datasets on which the algorithms were evaluated, the preprocessing of the datasets, the performance measures used in the evaluation, and the experimental settings used for the algorithms. Amongst those datasets, there are clinical datasets including two on heart disease, two on diabetes and one cancer patients. Brief descriptions of the datasets are given in Table 1.

4.1 Datasets

All datasets used to test the algorithms were taken from the *UCI Repository* and are used on classification tasks. The continuous variables were discretized using the entropy-based method developed by Fayyad and Irani [20]. Although big part of data presented in the datasets are numeric, they are not used to calculate the area under the curve. The AUC is calculated receiving as parameters the positive probabilities estimated for each instance of the dataset, with the respective labels of each of these instances. The discretization process is conducted in order to facilitate the handle of data by the algorithm, creating ranges of values and using

```
PSDP-STD-AUC (labels, dataset, testset)
  INPUT: labels contain a set of possible labels to predict,
          dataset is database,
          testset is a test case to predict
  OUTPUT: an array with the label, estimates probability of the label and the path.

  LOOP: until dataset to be empty, all cases in dataset have the same target or the set of atributes to be empty

      eProb = calculate the probability of happen two class for each instance of dataset

      FOR each testeset(i)
          subset = getSubSet(dataset, testset(i))
          diff_subset = dataset - subset

          eProb(subset) = calculate probability according number of testset
          eProb(diff_subset) = calculate probability different of testset

          AUC = colAUC(true_probability_of_dataset, labels)

          v_AUC(i) = AUC

          IF AUC < BestAUC THEN
                BestAUC = AUC
          END IF
      END FOR

      path = attribute selected according BestAUC
      dataset = dataset according attribute selected
      dataset = remove atribute selected
      eProb_path = calculate probability of predicted label to belong to positive or negative class
      predicted_label = max(eProb_path)
  END LOOP
RETURN eProb_path, predicted_label and path
```

Fig. 2. Pseudocode for PSDP-STD-AUC algorithm.

a single number or character that represents each interval. Missing values were imputed using an iterative non-parametric imputation algorithm described by Caruana [21] which has previously been applied to fill in missing predictor values for medical datasets with good results.

4.2 Test Settings

The proposed patient-specific decision path algorithms were implemented in MATLAB (R2013b version). We evaluated the algorithms using 20-fold cross-validation. This method randomly divided each dataset into 20 approximately equal sets such that each set had a similar proportion of individuals who developed the positive outcome. For each algorithm, we combined 19 sets and evaluated it on the remaining test set, and we repeated this process once for each possible test set. We thus obtained a prediction for the outcome variable for every instance in a dataset. The final result of the algorithms will be presented

Table 1. Brief description of used UCI datasets

Dataset	Instances	Attributes	Positive cases (%)
Australian	690	14	44%
Breast	699	9	34 %
Cleveland	296	13	45 %
Corral	128	6	44 %
Crx	653	15	45 %
Diabetes	768	8	35 %
Flare	1066	10	17 %
Glass 2	163	9	47 %
Heart	270	13	44%
Pima	768	8	35 %
Sonar	208	60	53 %
Tic-tac-toe	958	9	65 %
Vote	435	16	61 %

in terms of AUC and processing time. The algorithms that used the AUC measures to select an attribute were compared with the entropy-based algorithm proposed in [12].

All experiments were performed on a PC with a processor Intel Core i5 two cores with frequency of 2.5GHz, 8GB of RAM and running the operating system Mac OS X 64-bit Yosemite.

4.3 Results

Table 2 shows the AUCs obtained by the three algorithms. For each dataset, we present the mean AUC based on the 20-fold cross-validation and the respective confidence intervals at the 0.05 level. Overall, the two AUC based split algorithms perform comparably to the entropy method. Except for the Crx and Tic-tac-toe datasets, there is no statistically significant difference between the three methods.

In the Crx dataset, the entropy based model was statistically significant better than the other two methods with an mean AUC of 0.92. As for the Tic-tac-toe dataset, the PSDP-STD-AUC performed better, with a mean AUC of 0.98. ANOVA analysis [22] was performed and we verify that there is not statistical significant different between the three methods ($p >> 0.05$).

Table 3 shows the execution time of the proposed algorithms (means pm standard deviation). Each dataset was run 30 times for each model. Because the entropy based algorithm requires less operations, it presented better run time methods. The PSDP-STD-AUC requires an estimation of the class probabilities, which demands several computational operations. Even though the PSDP-AUC-Split does not require class probability estimation. Sorting of the leaves is

Table 2. AUCs for the datasets in Table 1. For each algorithm the table gives the mean AUC obtained from 20-fold cross-validation along with 95 % confidence intervals. Statistically significant mean AUC are in bold.

Datasets	Entropy	PSDP-STD-AUC	PSDP-AUC-Split
Australian	0.919 [0.910,0.928]	0.896 [0.877,0.916]	0.889 [0.869,0.909]
Breast	0.984 [0.980,0.988]	0.985 [0.978,0.992]	0.983 [0.975,0.991]
Cleveland	0.862 [0.837,0.888]	0.845 [0.779,0.911]	0.834 [0.773,0.895]
Corral	1.000 [1.000,1.000]	1.000 [1.000,1.000]	1.000 [1.000,1.000]
Crx	**0.920** [0.911,0.929]	0.879 [0.855,0.903]	0.885 [0.861,0.909]
Diabetes	0.827 [0.812,0.842]	0.815 [0.781,0.850]	0.820 [0.785,0.855]
Flare	0.730 [0.717,0.745]	0.718 [0.680,0.757]	0.715 [0.681,0.749]
Glass 2	0.831 [0.795,0.867]	0.870 [0.794,0.946]	0.865 [0.785,0.945]
Heart	0.879 [0.862,0.898]	0.877 [0.832,0.921]	0.877 [0.831,0.923]
Pima	0.825 [0.810,0.840]	0.813 [0.769,0.858]	0.811 [0.781,0.841]
Sonar	0.889 [0.867,0.911]	0.862 [0.818,0.907]	0.887 [0.835,0.939]
Tic-tac-toe	0.960 [0.952,0.969]	**0.989** [0.978,1.000]	0.977 [0.968,0.986]
Vote	0.986 [0.982,0.990]	0.985 [0.970,1.000]	0.985 [0.975,0.995]

Table 3. Results of execution time for proposed algorithms

Datasets	Entropy	PSDP-STD-AUC	PSDP-AUC-Split
Australian	3.346 ±0.046	35.821 ±1.464	7.388 ±0.138
Breast	1.140 ±0.010	10.102 ±0.048	2.115 ±0.025
Cleveland	1.091 ±0.011	10.827 ±0.052	2.466 ±0.026
Corral	0.170 ±0.005	1.651 ±0.019	0.341 ±0.009
Crx	3.342 ±0.032	34.111 ±0.094	7.620 ±0.229
Diabetes	1.974 ±0.013	17.305 ±0.094	4.002 ±0.040
Flare	4.348 ±0.021	36.206 ±1.179	8.019 ±0.322
Glass 2	0.427 ±0.007	3.298 ±0.038	0.920 ±0.014
Heart	1.042 ±0.011	10.301 ±0.057	2.220 ±0.019
Pima	1.973 ±0.012	19.175 ±0.093	3.781 ±0.137
Sonar	3.730 ±0.029	34.608 ±0.134	8.040 ±0.038
Tic-tac-toe	2.527 ±0.013	26.731 ±0.124	4.647 ±0.029
Vote	1.366 ±0.017	14.300 ±0.089	2.686 ±0.018

necessary to obtain the convex hull, for each split, also demanding extra CPU time. As per an ANOVA analysis, we verified that there is significant difference on execution time, since the entropy-based model runs faster ($p << 0$).

5 Conclusions

We have introduced PSDP, a new patient-specific approach for predicting outcomes, based on the AUC metric to select patient attributes. We evaluated this method on several datasets, including medical data. The results show that the PSDP-AUC based methods performs equivalently to the standard information based method. These results are encouraging and provide support for further investigation into the PSDP methods and more extensive evaluation on a wide range of datasets.

In future work, we plan to examine the complexity of the models generated by the PSDP methods and also explore other criteria for selecting predictors. Moreover, the presentation of patient-specific decision paths as IF-THEN rules to a domain expert may provide insight into patient populations.

The current PSDP methods have several limitations. One limitation is that they handle only discrete variables and continuous variables have to be discretized. A second limitation is the execution time on large data samples.

References

1. Abu-Hanna, A., Lucas, P.J.: Prognostic models in medicine. AI and statistical approaches. Methods Inf. Med. **40**(1), 1–5 (2001)
2. Visweswaran, S., Cooper, G.F.: Patient-specific models for predicting the outcomes of patients with community acquired pneumonia. In: AMIA Annu Symposium Proceedings (2005)
3. Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill Inc, New York (1997)
4. Friedman, J.H.: Lazy decision trees. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, (AAAI 1996), v.1, pp. 717–724. AAAI (1996)
5. Foster J.P., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. In: Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), pp. 445–453, San Francisco, CA, USA (1998)
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
7. Gagliardi, F.: Instance-based classifiers applied to medical databases: diagnosis and knowledge extraction. Artif. Intell. Med. **52**(3), 123–139 (2011)
8. Zheng, Z.J., Webb, G.I.: Lazy learning of Bayesian rules. Mach. Learn. **41**(1), 53–84 (2000)
9. Visweswaran, S., Cooper,: G.F.: Instance-specific Bayesian model averaging for classification. In: Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems. Vancouver, Canada (2004)
10. Visweswaran, S., et al.: Learning patient-specific predictive models from clinical data. J. Biomed. Inform. **43**(5), 669–685 (2010)
11. Visweswaran, S., Cooper, G.F.: Learning instance-specific predictive models. J. Mach. Learn. Res. **11**, 3333–3369 (2010)
12. Ferreira, A., Cooper, G.F., Visweswaran, S.: Decision path models for patient-specific modeling of patient outcomes. In: Proceedings of the Annual Symposium of the American Medical Informatics Association, pp. 413–21 (2013). PMID: 24551347, PMCID: PMC3900188

13. Hand, D.J., Till, R.J.: A simple generalisation of the area under the roc curve for multiple class classification problems. Mach. Learn. **45**(2), 171–186 (2001)
14. Ferri, C., Flach, P., Hernndez-Orallo, J.: Learning decision trees using the area under the ROC curve. In: Proceedings of the 19th International Conference on Machine Learning, pp. 139–146, Sydney, NSW, Australia, 8–12 July 2002
15. Ferri, C., Flach, P., Hernndez-Orallo, J.: Rocking the ROC Analysis within Decision Trees. Technical report, 20 December 2001
16. MATLAB: Trapz: Trapezoidal numerical integration. The Mathworks Inc. (2006). Accessed 17 February 2015
17. MATLAB: Version 8.4.0 (R2014b) The Mathworks Inc. (2014). Accessed 17 February 2015
18. Zadrozn, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), pp. 609–616, San Francisco, CA, USA, (2001)
19. colAUC: Calculates Area under ROC curve (AUC). The Mathworks Inc. (2011). Accessed 17 February 2015
20. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI, pp. 1022–1029 (1993)
21. Caruana, R.: A non-parametric EM-style algorithm for imputing missing values. In: Proceedings of Artificial Intelligence and Statistics (2001)
22. Janez, Demar: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)