Computational intelligence and asthma

Pablo Vicente-Munuera
Bioinformatics MSc
Email: pablovm1990@gmail.com

***Abstract*—The aim of this work is apply some algorithms typical used in statistics and machine learning to a given dataset. With this, we'll try to find if some of these algorithms can classify our dataset and try to figure something out. The dataset selected for this purpose is an asthma dataset.**

I. Introduction

Asthma is a chronic disease that affects your airway

http://www.aaaai.org/about-the-aaaai/newsroom/asthma-statistics.aspx

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3893990/

A. Dataset

Bronchial Epithelial Cells were isolated processed as described (Chu et al., 2002 and Zhao et al., 2011). The objective of the study was to identify differentially expressed genes between normal control (NC), mild-moderate asthmatic (MMA) and severe asthmatic (SA) patients.

Overall design Isolated fresh bronchial epithelial cells were placed into trizol for further analysis.

Asthma Organism Homo sapiens Experiment type Expression profiling by array

II. Methods

The methods are statistics and bla bla

why use machine learning algorithms?

Brief Bioinform-2006-Larranaga-86-112.pdf

A. Supervised classification

1) Naive Bayes:

2) KNN, K-nearest neighbours (IBk): The nearest-neighbour rule [75] to classify x is to asign it to the label associated with the prototype nearest to the test point (Figure 6). An obvious extension of the nearest-neighbour rule is the k-nearest-neighbour rule. This rule classifies x by assigning it to the label most frequently represented among the k nearest samples. In other words, a decision is made by examining the labels on the k-nearest-neighbours and voting. A practical problem with this simple method is that it tends to be slow for large training sets because the entire set must be searched for each test instance. A strategy to avoid the computational complexity of the nearest neighbour algorithm is to classify each example with respect to the examples already seen and to save only those that are misclassified. This strategy is known as condensing.

3) Decision tree: C4.5 (j48):

4) Logistic regression: stic regression defined as pC 14 1jx 14 1=121 e0 i141 ixi; where x represents an instance to be classified, and 0, 1, . . . , n are the parameters of the model. These parameters should be estimated from the data in order to obtain a concrete model.

5) Bayesian Networks: TAN:

B. Filter selection subset

Bioinformatics-2007-Saeys.pdf

1) Univariate:

2) Multivariate:

3) Wrapper:

C. Complementary

book wittenyfrank.pdf

1) Random forest: RandomForest constructs random forests by bagging ensembles of random trees

2) Adaboost: There are many variants on the idea of boosting. We describe a widely used method called AdaBoost.M1 that is designed specifically for classification. Like bagging, it can be applied to any classification learning algorithm. To simplify matters we assume that the learning algorithm can handle weighted instances, where the weight of an instance is a positive number. (We revisit this assump- tion later.) The presence of instance weights changes the way in which a classi- fiers error is calculated: it is the sum of the weights of the misclassified instances divided by the total weight of all instances, instead of the fraction of instances that are misclassified.

3) Multilayer perceptron: Section 4.6 explained that a perceptron represents a hyperplane in instance space. We mentioned there that it is sometimes described as an artificial neuron. Of course, human and animal brains successfully undertake very complex classification tasksfor example, image recognition. The functional- ity of each individual neuron in a brain is certainly not sufficient to perform these feats. How can they be solved by brain-like structures? The answer lies in the fact that the neurons in the brain are massively interconnected, allowing a problem to be decomposed into subproblems that can be solved at the neuron level. This observation inspired the development of networks of artificial neuronsneural nets. Consider the simple datasets in Figure 6.10. Figure 6.10(a) shows a two- dimensional instance space with four instances that have classes 0 and 1, repre- sented by white and black dots, respectively. No matter how you draw a straight line through this space, you will not be able to find one that separates all the black points from all the white ones. In

other words, the problem is not linearly separable, and the simple perceptron algorithm will fail to generate a separating hyperplane (in this two-dimensional instance space a hyperplane is just a straight line). The situation is different in Figure 6.10(b) and Figure 6.10(c): both these problems are linearly separable. The same holds for Figure 6.10(d), which shows two points in a one-dimensional instance space (in the case of one dimension the separating hyperplane degenerates to a separating point). If you are familiar with propositional logic, you may have noticed that the four situations in Figure 6.10 correspond to four types of logical connectives. Figure 6.10(a) represents a logical XOR, where the class is 1 if and only if exactly one of the attributes has value 1. Figure 6.10(b) represents logical AND, where the class is 1 if and only if both attributes have value 1. Figure 6.10(c) repre- sents OR, where the class is 0 only if both attributes have value 0. Figure 6.10(d) represents NOT, where the class is 0 if and only if the attribute has value 1. Because the last three are linearly separable, a perceptron can represent AND, OR, and NOT. Indeed, perceptrons for the corresponding datasets are shown in Figure 6.10(f) through (h) respectively. However, a simple perceptron cannot represent XOR, because that is not linearly separable. To build a classifier for this type of problem a single perceptron is not sufficient: we need several of them.

4) RBF Network: Other kernel functions can be used instead to implement different nonlinear mappings. Two that are often suggested are the radial basis function (RBF) kernel and the sigmoid kernel. Which one produces the best results depends on the application, although the differences are rarely large in practice. It is interesting to note that a support vector machine with the RBF kernel is simply a type of neural network called an RBF network (which we describe later)

III. Results

In this section we'll show all the results we get from the expression set used. The software used to get all of those results is weka [2].

TABLE I
*: The X values means no results could be obtained due to computational problems. **The K used here is equal to one.

		FSS		
Method	No filter	Univariate	Multivariate	Wrapper
Naive Bayes	0.436	0.626	0.623	0.731
KNN**	0.594	0.644	0.67	0.781
Logistic	X*	0.731	0.7	0.793
Decision Tree	0.486	0.639	0.698	0.641
Bayesian Net	X*	0.657	0.663	X*

We had these problems mentioned before because the initial dataset had, approximately, 43300 variables. One of the first approachs to solve this, was a first univariate filter in which we selected the information gain ratio with respect to their own class. After this filter, all of these classify problems became more time-relaxed, due to the reductio of the number of variables (43300 to 32).

TABLE II
*: K equals to 4; ** K equals to 1.

	Multivariate		
Method	CFS Best first	CFS Genetic	Wrapper
Naive Bayes	0.623	0.599	0.74
KNN	0.773*	0.674*	0.687**
Logistic	0.699	0.675	0.824
Decision Tree	0.668	0.76	0.641
Bayesian Net	0.667	0.674	0.678

TABLE III
Results depending the K value on the KNN classifier

K	Univariate	Multivariate
1		
2		
3		
4		
8		

Another important thing which showed up with the output of the results was the classification between one class and the other ones.

TABLE IV
Some examples of the difference between classes. Measured by the ROC curve.

	Classes			
Method - Filter	Control	MMA	SA	Overall
KNN (k=1) - Gain ratio	0.642	0.589	0.717	0.644
Bayesian Net - Gain ratio & CFS	0.802	0.591	0.697	0.667
Logistic - Wrapper	0.82	0.755	0.829	0.793
Decision Tree - No filter	0.55	0.473	0.471	0.486
Naive Bayes - CFS Best first	0.766	0.534	0.664	0.623

IV. Discussion

References

[1] Voraphani N, Gladwin MT, Contreras AU, Kaminski N et al. An airway epithelial iNOS-DUOX2-thyroid peroxidase metabolome drives Th1/Th2 nitrative stress in human severe asthma. Mucosal Immunol 2014 Sep;7(5):1175-85.

[2] Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations. Department of Computer Science. University of Waikato. New Zealand.