An Automatic Classifier For Diaphragm Curves - Final Report

Emanuele Lovera, 901727, University of Milan

11-04-2017

1 Introduction

The purpose of this document is to present the work that I've done during the development of the project for the exam of Information Management. The problem to solve was a ternary classification – Yes, No, Maybe – of signals generated by the activity of the diaphragm during a registration of 5 minutes. This work is a part of a bigger research [1] lead by Dr. Giorgio Villani.

2 The Dataset

All the data are collected during a registration, usually 5 minute long, and written in a dataset. Each row is a breath curve and it is described by 21 attributes, each of them represent a certain characteristic of the signal. All the values are real or numeric. There aren't special values, such as `Nan`, `Inf` or `Nd`. Furthermore, all limit values piked by the sensor during non-common condition (for instance +/- 9999), there were removed by a simple average between some of the previous and the next values. Here there is a detailed description of all attributes: name, meaning and type of the data.

1. `classifications` (binary - ternary): the first attribute is a binary descriptor when the dataset is taken in input – as learning dataset –, and it is ternary when it's in the output phase of the classification.
   - 0: when a certain curve is an Anomaly Curve.
   - 1: when it is a correct curve.
   - 2: it means that the classifier is not able to compute in which class it belongs.
2. `peak` (real): is the highest point of a signal.
3. `integral` (real): is the integral of the curve. It is computed by the sum of all points belonging the signal.

4. `P/I Index` (real): is the ratio between the `peak` and the `integral`.
5. `TV` (real): also known as tidal volume. It is the total volume of air inspired during the inspiration phase.
6. `RR` (numeric): It is the respiration rate. It's the number of breath done during a minute.
7. `RR/TV` (real): Ratio between the respiration rate and the tidal volume.
8. `etco2` (real): total amount of CO_2 inspired during a breath.
9. `first value` (real): value of the first non-zero point.
10. `last value` (real): value of the last non-zero point.
11. `intercept` (real): mean of the first 15th points of the signal.
12. `integral after max` (real): value of the area of the curve after the maximum point.
13. `before concavity counter` (numeric): counter of the number of positive concavity before the maximum point.
14. `before area 1` (real): it is the biggest concavity area, before the maximum point.
15. `before area 2` (real): second biggest area.
16. `before area 3` (real): third biggest area.
17. `after concavity counter` (numeric): counter of the number of positive concavity after the maximum point.
18. `after area 1` (real): the biggest concavity area, after the maximum point.
19. `after area 2` (real): second biggest area.
20. `after area 3` (real): third biggest area.

For this project the Dr. Villani has provide us six dataset correctly classified.

3 Technologies used

During this project, I've used different technologies in order to solve the classification problem. Fist of all, the dataset has been preprocessed by `Python 3.5` and the `Numpy` library. For plotting all the graphs, I've used the `matplotlib` library. After this phase, I've done all the rest of experiments with the statistical software `R`. I have used two package of the R suite: `rpart` for the Decision Tree and the `e1071` for the Support Vector Machine. I have also used the `caret` library for the k-fold technique.

4 Classification Techniques

For the classification task I tested two different approaches: the support vector machine and the decision tree. Both have pros and cons. The support vector machine is an algorithm that transform the data space in a new higher dimensional space and try to separate it by an hiperplane. It has also the powerful capability of geometrically divide the dataset in rapid way, but has a learning phase a bit longer than the decision tree. This last one is an algorithm that has the ability of divide the dataset in subset of data, splitting it by a set of test done on the more important attributes of the dataset. The DT is a more simpler way of classifying data, and for this reason is less precise than the SVM when attributes are numerical.

5 Metrics for evaluating performance

To choose the correct classification technique, there are the necessity of some metrics. For this project were made:

Measure	Definition	Formula
Acc – Accuracy	Is the recognition rate of the system.	$\frac{TP+TN}{P+N}$
TPR – True Positive R.	Rate of positive (1) tuples correctly labeled.	$\frac{TP}{P}$
TNR – True Negative R.	Rate of negative (0) tuples correctly labeled.	$\frac{TN}{N}$
PR – Precision R.	Rate of OK tuples, that are marked correctly.	$\frac{TP}{TP+FP}$

As I told before, such that the dataset is unbalanced on the negative class, we are interested about having a high value of TNR. A graphical representation is made with the confusion matrix:

	0	1	tot
0	TN	FP	N
1	FN	TP	P

Where on the rows there are the correct classification and on the columns there are the predicted values.

6 DT vs SVM: Experimental Results

In order to choose the best classification technique, I have done different kind of test. All of them is done with k-fold technique, in order to correct the anomaly of having a small amount of dataset. All the values presented are to be intended as average of the k interation.

6.1 Resalistic test

The first test that Ive done is the realistic one possible. The k-fold is a rude version of it. Simply use a dataset as test and the other five as learning and try to classify the test and collect all the result. Iterate this procedure for six time choosing a different dataset as test. Here the results of the test.

k=6	DT	SVM
Avg Accuracy	0.9190726	0.9090671
Accuracy Variance	0.0037335	0.0043425
Avg TP Rate	0.9970307	0.9990020
TP Rate Variance	1.069666e-05	5.976072e-06
Avg TN Rate	0.4951935	0.4398819
TN Rate Variance	0.1023117	0.04678799
Avg PR Rate	0.9159684	0.9040619
PR Rate Variance	0.004730	0.0052743

As we can see, the Accuracy in both cases is high, but the TN Rate is pretty low. With this level of TNR, for each Anomaly Curve we met, one time is classified in correct way, and in the other, is incorrectly classified. The precision rate is about 0.90 in both cases, this means that the positive tuples are quite correctly classified.

6.2 Oversampling test

In order to improve the value of the TNR, I have tried with a simple technique of oversampling. First of all I have created a balanced dataset with the same number of rows of the previous test, so as to be able to compare the result with the others tests. As the previous one, I have done a k-fold validation with `k=6`.

k=6	DT	SVM
Avg Accuracy	0.8970373	0.8970312
Accuracy Variance	0.00035764	0.0005878
Avg TP Rate	0.8579582	0.874683
TP Rate Variance	0.00050561	0.0017492
Avg TN Rate	0.9355169	0.9203087
TN Rate Variance	0.0005575	0.00021229
Avg PR Rate	0.9311777	0.915702
PR Rate Variance	0.0004586	0.0003581

Again, in this case we have the Decision Tree is more accurate than the Support Vector Machines. We have, also, more or less the same accuracy but we have the doubled the precision of the TNR, reaching 0.93. We have also a reduction of the PR Rate about of 10% - 15%.

6.3 Split test with balanced dataset

This test I have done is the split test with balanced data set. With the previous we have seen that the oversampling technique help the improvement of the TNR leaving more or less the same accuracy. But, the main problem of the previous test is that the test set is also balanced on its classification class. So in this case I have done a balanced learning dataset with an unbalanced test set. Here the results, always with a k-fold validation.

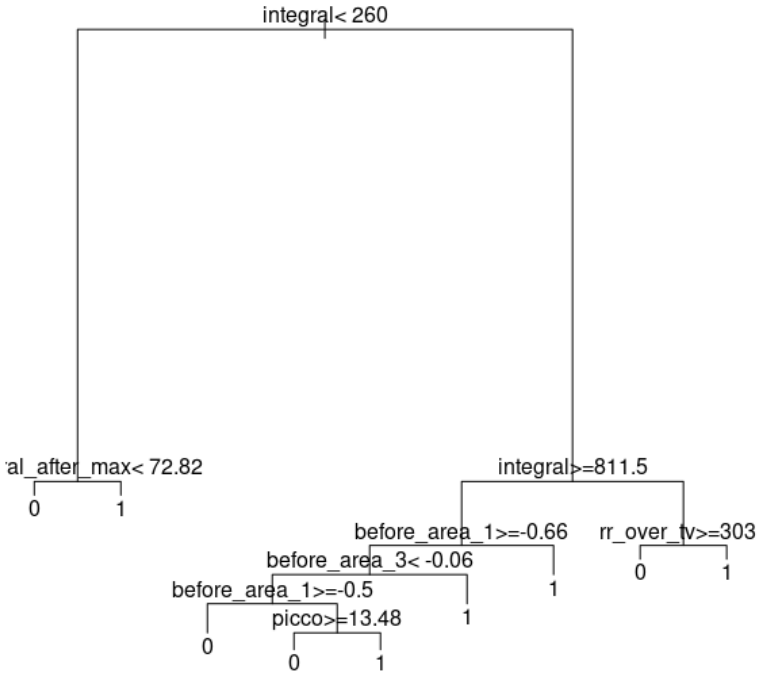
k=6	DT	SVM
Avg Accuracy	0.8530706	0.8578346
Accuracy Variance	0.0084403	0.025497
Avg TP Rate	0.8834463	0.8864275
TP Rate Variance	0.012885	0.046231
Avg TN Rate	0.7007271	0.5896648
TN Rate Variance	0.017058	0.014983
Avg PR Rate	0.9406947	0.9208221
PR Rate Variance	0.0033785	0.0026104

In the real application we cannot have a balanced test set, therefore this test is more realistic than the previous one. We have a bit less accuracy than

the previous one and also in this case, the Decision tree is more reliable than the SVM. Especially on the TNR and on the Precision Rate.

7 Final choice: motivation

The final choice is fallen on the Decision Tree, for three main reason. As told before, this algorithm create a model easily understandable from all the people involved by the main research. Here we can see a plot of the model.



As we can see it is a very simple way to understand how the single tuple are classified.

The second reason, but not least important, is the precision. In all the test that I have done, the decision tree is quite more accurate to understand the correct classification.

The third point is the stability. The k-fold is an approach used to mitigate the unbalance property of the single dataset, and all the results that I have presented before, are the mathematical mean of each result of the

k iteration. So, over the iterations the Decision Tree has a better stability compared to the Support Vector Machine.

8 Implementation

This project is the second part of a big one, and it was designed to be a inner component of a program. The first part, written in Python, has the main purpose of taking in input the data, visualize all the information about the curve, let a manual classification and displaying the final results. This project has the main task of design and building a classifier, which it is insert in the Python program, allow the final user to automatic classify the curves. The implementation of the Decision Tree is made by the R language with the use of the `rpart` library.

This script is made by a first phase of creation of the learning dataset, based on a knowledge base already present. After that there is the learning and prediction phase and finally the return of an array of probabilities, which values are real numbers between 0 and 1.

This list expresses the probability that each curve belongs to the class `1` (Ok). In opposit, when a element of index `i` has a low value, the `i-th` curve has `1 - probability` of belonging to the class `0` (anomaly).

The class assignment is made by the Python program by a double threshold (see Parametric Analysis section). This choice has done to make possible at the final user to set the desired threshold's level.

9 Learning Dataset Updating

More tuples the system use as learning, the more accurate the classifier will be. As told before I have decided to use the balanced version of the dataset. The application will use the six database as knowledge base and it will ask every time, after the classification phase, if the user want to add the latter at the learning base.

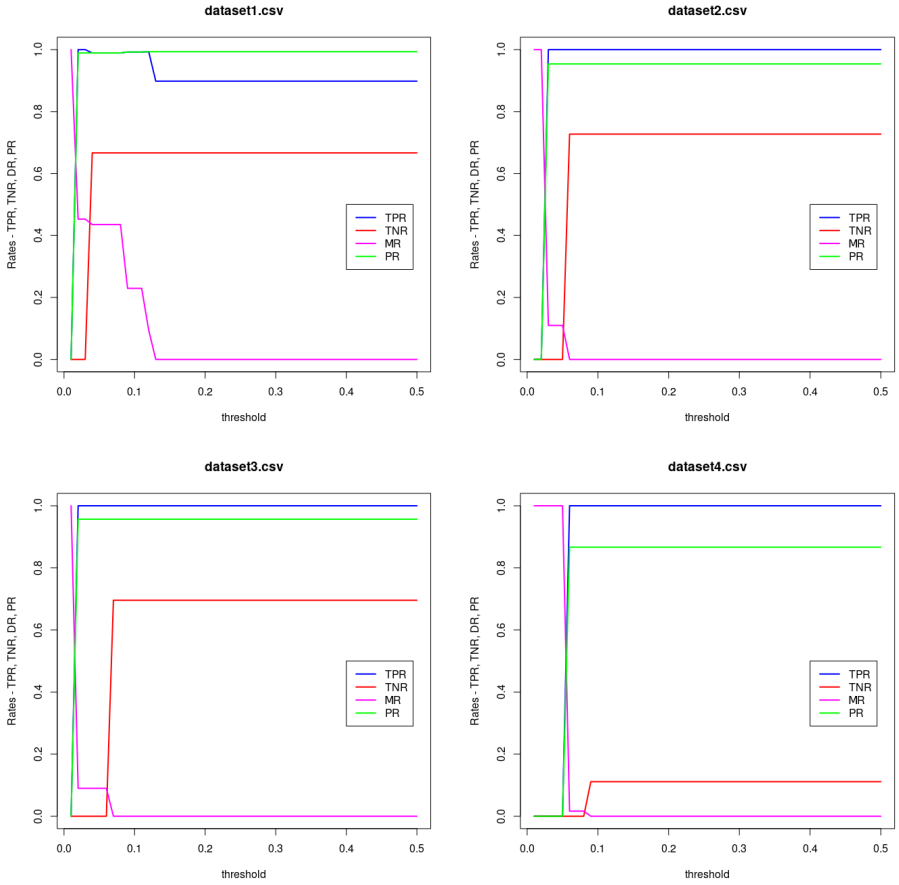
10 Parametric Analisy

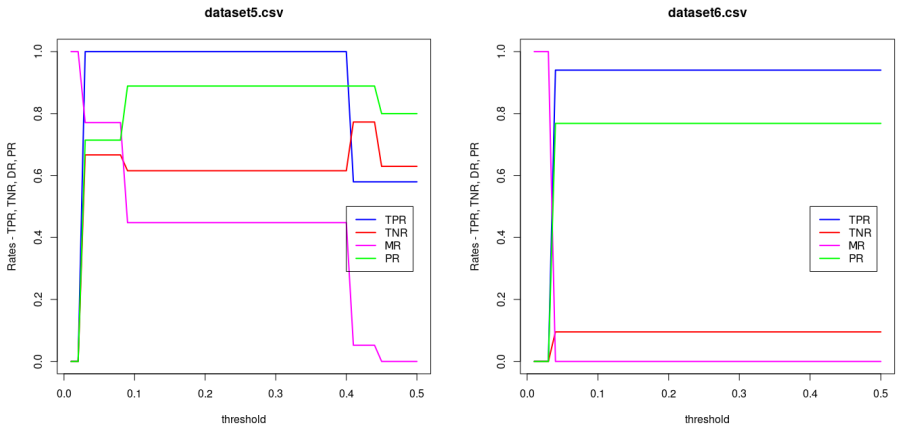
As I told in the final part of the Implementation Section, the program need an appropriate threshold level. For notation, we call α the lower one and the greater one is simply $\beta = 1$ - α. Moving the theshold change the final number of curves belonging to the three classes. If we choose a too low value of α, we will have a high number of `Maybe` curves. In opposite setting a threshold at a value too much high, means having an high number of misclassified curves.

I have done an analysis on the relation between α and the number of the element generated from the system. For each dataset, I've used the α as independent variable, and computed:

- **TPR** (blue curves): True Positive Rate. Computed on the sum of all poitive and negative curves (maybe counter excluded).
- **TNR** (red curves): True Negative Rate. Computed on the sum of all poitive and negative curves (maybe counter excluded).
- **MR** (Magenta curves): Maybe Rate. Computed on the total number of curves.
- **PR** (Green curves): Precision rate. Computed on the sum of all poitive and negative curves (maybe counter excluded).

All of the values are constrainted in the [0,1] range. After that, I have plotted the obtained values. Here the results.





As we can see the threshold has a big impact on the classification. The idea is to maintain the highest as possible all variables' values. All the four indexes are computed on the counter of correctly classificated curves.

The error in the classification aren't visible in the plots. This is made for a reason. An incorrect classification is a **weakeness** for the system. We like, instead, a **large number** of Maybe curves, that can be easily classified by the final User.

In all the graphs, we can see that when there is a decreasing curve, we are loosing correctly classified diaphragm curves. Now the question is: where they are gone? If we are lucky, the loosed breath curves are correctly classified in an other class. Instead, if we are unlucky or if we use a large value of the α theshold, the loosed curves are incorrectly classified: we don't want this beahavior.

So, in order to balance all of the previous consideration, the default threshold is choosen at the value of **0.05**. As I told in prevous sections, α can be set each time before the classification task, by the user, during the execution of the program.

11 Conclusions

All these test has been done in order to classify all the curves in a ternary classification. Obviously the human supervision is always need, in fact all the maybe are need to correctly classified by the final User, otherwise the program will not create the final dataset. Furthermore, all the other results of the automatic classification can be rechecked by a the user at the end of the phase by the User Interface of the Python program.

All the result presented in this project are constrained by the limited number of learning dataset. More dataset the user will add to the knowledge base, the more accurate will be the classification.

References

[1] Muttini Stefano, Villani Pier Giorgio, Trimarco Roberta, Bel- lani Giacomo, Grasselli Giacomo, Patroniti Nicol. *Relation between peak and integral of the diaphragm electromyographic activity at different levels of support during weaning from mechanical ventilation: a physiologic study.* Journal of Critical Care (2014), doi:10.1016/j.jcrc.2014.08.013

[2] Jiawei Han, Micheline Kamber, Jain Pei. *Data Mining – Concept and techniques.* Third Edition. Elsevier 2012.