Parametric Classification over Multiple Samples

Barbara Russo
Faculty of Computer Science
Free University of Bozen-Bolzano
Bolzano, Italy
Barbara.Russo@unibz.it

***Abstract*—This pattern was originally designed to classify sequences of events in log files by error-proneness. Sequences of events trace application use in real contexts. As such, identifying error-prone sequences helps understand and predict application use. The classification problem we describe is typical in supervised machine learning, but the composite pattern we propose investigates it with several techniques to control for data brittleness. Data pre-processing, feature selection, parametric classification, and cross-validation are the major instruments that enable a good degree of control over this classification problem. In particular, the pattern includes a solution for typical problems that occurs when data comes from several samples of different populations and with different degree of sparcity.**

I. Problem

The pattern we propose applies in the case of classification problems with two parametric output categories, replicated over several independent samples (not drawn from the same population), but with common research objectives. The pattern applies in case of curse of dimensionality of samples, that is when the feature spaces are highly dimensional. Large samples are needed to apply cross validation techniques over sample splittings. Techniques of balancing categories in samples might be necessary if any category is scarcely represented.

II. Solution

The classification problem is illustrated in Fig. 1. The solution we propose is structured in answers to a series of questions that drive the structure of our composite pattern.

Should I merge all my samples? This is a typical question when multiple samples are available. The answer depends on whether there is a common population for all / part or no samples. The pattern we propose here concerns the last case in which samples are not drawn from any common population, but still the research we want to perform on them has the same common objectives. In this case, we do not merge the samples and apply the classification on the merged set, but we rather replicate the analysis on each sample and then use technique of meta-analysis over samples to draw sample independent conclusion.

Which features should I consider? In the single-sample case, we can correlate the variables describing the input with the variable determining the output categories. Significant high correlation values can be used to indicate variables to consider as input features. The significance and the value of the correlations vary over samples, though. A sound technique to select the types of features relevant in a classification problem independently from the samples is the *weighted estimators of a common correlation*, a meta-analysis technique that determines whether the weighted mean of correlations is a good estimator of the correlation across samples, [2]. It is based on the homogeneity test of the estimations across the samples with the Null Hypothesis:

N_0 : All correlations of the samples are equal.

The Cochran Q statistics for homogeneity test of the correlation mean with significance level $0,05$ is used to test N_0 against the chi-square threshold value for a suitable degree of freedom [3].

Which features are redundant? This is a typical problem that relates to *curse of dimensionality*: data can be scarce in high dimensional spaces and classifiers' performance is low with scarce data. In machine learning theory, there are various approaches; the one we propose here is Information Gain (IG). IG ranks attributes from the most informative to least informative. Compared with other techniques as Principal Component Analysis, IG is rather intuitive and gives ready-available sets of variables that most influence the information to be used with the classifier. As illustrated in [6], IG measures the reduction in entropy of one class distribution C achieved by conditioning it with a new variable X:

$$\text{Information Gain}(X) = H(C) - H(C|X)$$

where H() is the expected mean of number of bits required to encode C:

$$H(C) = -\sum_{o \in C} p(o) log_2 p(o)$$

How do I classify my features? The answer depends on the context of research or the percentage of available faulty sequence types. We propose a parametric approach that defines the output categories as function of a parameter that can be valued ex-ante to satisfy specific research constraints or data scarcity (as in traditional literature) or ex-post by the best accuracy of the classifiers.

Given ρ the variable defining the output categories and c a parameter, we define the two parametric output groups as:

$$G_1(c) = \{feature : \rho(feature) \geq c\}$$
$$G_2(c) = \{feature : \rho(feature) < c\}$$

For each value of the parameter c, we get a variation of the classification problem.

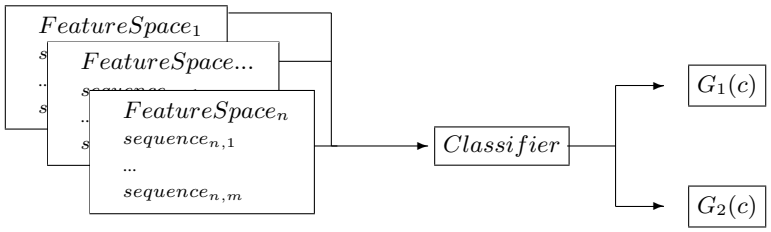

Fig. 1. Input and output of a classifier in case of multiple samples

Which model should I select? With classifiers the answer depends on the good use of the *cross validation technique*. We illustrate here the approach presented in [4] and a way to compare different models by their accuracy.

Cross-validation provides techniques to measure the ability of a model to predict on new instances. The approach in [4] consists of three stages: training, validating, and testing a model. First a sample is randomly split into a fit set and a validation set and then the fit set is in turn randomly split into a training set and test set. The percentage of splitting can be fixed (typically 70-30) or variable (1/k - (k-1)/k) depending whether we want to control for the splitting size. The model learns its parameters on a training set (Training), calculates its error on a validation set (Generalization), and measure its accuracy on the test set (Quality of fit). The quality is evaluated averaging accuracy measures over the n-folds cross-validation (typically $n = 10$) on the test set.

We use the balance measure greater than a given threshold to determine the top ranked models for generalization and quality. Choosing the threshold is a key issue. Literature on classification problems in software can help to set it, (e.g., [6]). The balance measure is a distance from the ideal value $(1, 0)$ and the actual pair (true prediction rate, false prediction rate):

$$\text{bal}= 1 - dist((TP_r, FP_r), (1, 0)) = 1 - \frac{\sqrt{(TP_r-1)^2+(FP_r)^2}}{\sqrt{2}}.$$

High values of balance indicates proximity to the ideal value (1,0). Top ranked model are evaluated over all the sample splittings. Finally, we can compare different classification problems, by repeating this procedure at different values of c.

III. Consequence

Each of the technique that composes the pattern has some subtle aspects to consider and some limitations.

- The meta-analysis technique assumes that samples come from normally-distributed populations with the same correlation and sample correlations are computed according to Pearsons definition of correlation. When we cannot assume any normality in the distributions of our input variables, we need to normalize the sample correlations with the Fisher z transformation, compute of the 95% confidence interval confidence interval for the transformed correlations, and apply the inverse Fisher z transformation on the resulting range as in [8].
- With the homogeneity test, we aim at accepting the null hypothesis. The type of error we must control is the statistical power of the test and reducing the type II error for which we accept the null hypothesis when it is false. A typical means to increase statistical power of a test is to increase the individual sample size and the number of samples, [3], [2], [7], [8].
- The cross validation technique we propose requires a large number of instances in the samples and a sufficient representation of the categories in the samples. Large samples are needed to have sufficient number of instances in the three sets, training, test, and validation sets. In addition, each set must have a sufficient representation of the categories. When one category is not represented in one of the three sets derived from the sample, the sample must be discarded. Splitting the sample at different percentages can increase the chance (but does not ensure it) to get three sets with some representation of the categories. "Some" representation might also not be enough. Literature reports that classification machines need to run with at least 5-20% instances of each category [4]. In this case, we ca use techniques to artificially create splittings of samples with sufficient representations of categories, [5], [1], but again large samples are needed.
- High values of the parameter c in the definition of the output categories can decrease the balance of the categories in the samples and consequently the performance of the classifiers. For example, if we classify software classes by the number of errors they have, the number of classes that have more than three errors $G_1(3)$ is smaller or equal to the number of classes that have more than one error $G_1(1)$.

IV. Example

The example is adapted from a manuscript under review co-authored by the author.

Log files store sequence of events determined by a given application. A sequence starts with "log-in" event and ends with the last event at the end of the day, before a new "log-in" event, or with an event that is labeled as error. Each application is characterized by a set of event types and each sequence can be represented as a vector of multiplicities over this set of event types. For example, the event sequence {Login, Order, Order, Order, Run} of a given application is mapped into the

vector $[1, 0, 3, 1]$, if the event types for that application are {Login, Maintain, Order, Run}.

The feature spaces. The input vector is the vector of event multiplicities v augmented by the multiplicity of v, μ, and the number of users that generate the events v, ν. In this way, each input vector represents a unique feature $[v, \mu, \nu]$ that we call sequence type. A sequence type can have no, one or more errors, ρ, depending whether the sequences that it represents have errors. For each application, the final sample consists of sequence types equipped by the number of errors and the feature space consists of the sequence types.

Thus, we might have {Login, Order, Order, Order, Run} and {Login, Order, Order, Run, Order}. They are both represented by $[1, 0, 3, 1]$, and the users that generated the events in each sequence are {Smith, Taylor, Taylor, Taylor, Smith} and {Smith, Taylor, Murphy, Murphy, Wang} respectively. Thus the sequence type is $[1, 0, 3, 1, 2, 4]$ as there are two sequences that correspond to the vector $[1, 0, 3, 1]$ and four distinct users that have generated the two sequences. Assuming that only the sequence {Login, Order, Order, Order, Run} ends because the event "Run" is "error," the sequence type $[1, 0, 3, 1, 2, 4]$ has one error, $\rho = 1$.

Should I merge all my samples? In our example, the answer is no. Sequences generated by different applications are in general composed by different event types (the typical common even type is "Log-in"). As such, the sample of sequence types we get form each application is likely to come from an independent population. At the end, we may get as many populations as the number of applications running in a system.

Which features should I consider? The weighted estimators of a common correlation is applied to μ, ν, vs. ρ to understand whether the first two variables have any impact on the last across all our samples. We keep both variables as we cannot reject the null hypothesis when it is true and the sample size and the number of samples are large enough to get high statistical power, [3].

Which features are redundant? Altogether the information carried in sequence types can be redundant, for example not all entries (event types) in v contribute to the classification problem. In our case, we have found, for example, that after feature reduction, the number of event types reduces in the majority of the samples. This indicates that not all the types of events in a sequence contribute to the error-proneness of a sequence.

How do I classify my features? The output groups consist of sequence types with a number of errors greater or equal to c, $(G_1(c))$, and sequence types with number of errors less than c, $(G_2(c))$. In the previous example, for $c = 1$, the sequence type $[1, 0, 3, 1, 2, 4]$ is classified in $G_1(1)$ and for $c = 2$ it is classified in $G_2(2)$. Varying the value of c allows investigating sequence types at different degrees of error proneness. If we want to classify the most risky sequence types we increase the value of c.

Which model should I select? We have cross validated three types of classifiers: Linear Classifier (L), Radial Basis Function neural network (RBF) and Multilayer Perceptron neural network (MP) for five values of c and across four different splittings for cross validation ($k = 2, ...5$). We found that:

- When the performance is set at high values for quality of fit and generalization, MP generally outperforms, but there are samples in which the type of the classifier is uncertain as performance depends on the data distribution of the classification groups and the nature of classifier itself. Specifically, L appears to outperform with low percentages of defective sequence types and no feature selection. RBF is more flexible and can outperform with different percentages of defective sequence types and test sets.
- Feature selection helps determine with more precision the classifier for a specific sample reducing uncertainty and increasing the robustness of the classifier over different splittings. In particular, the linear classifier is less represented among the top ranked models.

V. R SNIPPETS

Key R Snippets can be downloaded from https://pro.unibz.it/staff/brusso/WorkInProgress.html

REFERENCES

[1] N. Chawla, and K. Bowyer, and L. Hall, and W. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research* vol. 16, pp. 321-357,2002.

[2] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, L. Erlbaum Associates, 1988.

[3] V.L. Hedges and I. Olkin, *Statistical Methods for Meta-Analysis*, 1st ed. Academic Press, 1985.

[4] T. M. Khoshgoftaar and D. L. Lanning, "A neural network approach for early detection of program modules having high risk in the maintenance phase," in *Selected papers of the sixth annual Oregon workshop on Software metrics*. New York, NY, USA: Elsevier Science Inc., 1995, pp. 85–91.

[5] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, vol. 1, oct. 2010, pp. 137 –144.

[6] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.

[7] J. Sanchez-Meca, and F. Martin-Martinez,"Homogeneity tests in meta-analysis: a Monte Carlo comparison of statistical power and Type I error",*Quality and Quantity*,vol. 31, no. 4, pp. 385–399,

[8] G. Succi, W. Pedrycz, S. Djokic, P. Zuliani, and B. Russo, "An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite," *Empirical Softw. Eng.*, vol. 10, no. 1, pp. 81–104, Jan. 2005.