Using nondeterministic learners to alert on coffee rust disease

Oscar Luaces[a], Luiz Henrique A. Rodrigues[b], Carlos Alberto Alves Meira[c], Antonio Bahamonde[a,*],

[a] *Artificial Intelligence Center. Universidad de Oviedo at Gijón, Asturias, Spain*
[b] *Universidade Estadual de Campinas, Faculdade de Engenharia Agrícola, Cx. P. 6011, CEP 13083-875 Campinas, SP, Brazil*
[c] *Embrapa Informática Agropecuária, Cx. P. 6041, CEP 13083-970 Campinas, SP, Brazil*

Abstract

Motivated by an agriculture case study, we discuss how to learn functions able to predict whether the value of a continuous target variable will be greater than a given threshold. In the application studied, the aim was to alert on high incidences of coffee rust, the main coffee crop disease in the world. The objective is to use chemical prevention of the disease only when necessary in order to obtain healthier quality products and reductions in costs and environmental impact. In this context, the costs of misclassifications are not symmetrical: false negative predictions may lead to the loss of coffee crops. The baseline approach for this problem is to learn a regressor from the variables that records the factors affecting the appearance and growth of the disease. However, the number of errors is too high to obtain a reliable alarm system. The approaches explored here try to learn hypotheses whose predictions are allowed to return intervals rather than single points. Thus,

*Corresponding author
Email address: antonio@aic.uniovi.es (Antonio Bahamonde)

in addition to alarms and non-alarms, these predictors identify situations with uncertain classification, which we call warnings. We present 3 different implementations: one based on regression, and 2 more based on classifiers. These methods are compared using a framework where the costs of false negatives are higher than that of false positives, and both are higher than the cost of warning predictions.
Keywords: Machine Learning, Agriculture Application, Risk Assesment, Coffe Rust

1. Introduction

In this paper we discuss how to learn alarm functions which should predict the values of a continuous target variable. The research was motivated by an agriculture case study: the prevention of coffee rust, the most devastating disease for coffee plants, caused by the fungus *Hemileia vastatrix* Berk. & Br.

In Brazil, damage leads to yield reductions of up to 35% in regions where climate conditions are propitious to the disease. The impact of the disease is thus considerable due to the economic importance of the coffee crop.

The traditional way to prevent the disease is to apply agrochemical fungicides on fixed calendar dates. However, the fungicides pollute the environment and reduce the quality of the coffee. Moreover, as the intensity of the disease between seasons suffers major variations, the use of agrochemicals is not always justified.

The symptoms of infection include the appearance of spots on the upper leaf surface of plants. The incidence of the disease is defined as the percentage

of infected leafs. The purpose of alarm functions is to alert on high incidences of the disease with time enough to apply the fungicide in advance. The objective is to build economical tools that will allow applying agrochemicals only when necessary, leading to healthier products as well as reductions in costs and environmental impact.

Throughout the paper, we use a dataset [1, 2, 3] that comprises monthly accounts of the incidence of disease on an experimental farm in Brazil over 8 years. Additionally, the dataset registers the values of variables known to stimulate the growth of fungus: weather conditions, fruit load of the plantation, and spacing between plants.

A baseline approach to build an alarm function is to learn a regressor that predicts incidences. Thus, in addition to predicting whether or not the incidence will require the use of fungicides, we also have an assessment of how serious the situation may be. We trained a regression Support Vector Machine (SVM) with quite good results. The correlation between predicted and actual incidences is about 0.94 in a cross-validation experiment. However, if we use regression predictions to decide whether the incidence of the disease will be greater than a given threshold, the number of misclassifications is too high.

To try to overcome this weakness, we used an approach similar to that of classifiers with a reject option, [4, 5]. The situations that are likely to be misclassified are rejected; they are not classified and will be tagged, in this case, as *warnings*. In other words, we can convert misclassifications into a type of situation that may require deeper analysis. However, when the alarm system predicts an alarm, and especially a non-alarm, the confidence level of

these predictions increases.

The distinction between false positive (alarm) predictions, and false negative, is very important in this case. The cost, in the economic sense, is quite different in these kinds of errors. False positives may lead to the unnecessary application of fungicides, but false negative predictions can lead to the loss of coffee crop.

Thus, we try to learn models to predict *approximations* to incidence values instead of exact values. To implement this idea, there are a number of possible alternatives. For instance, in [6, 7] the authors propose algorithms to learn confidence intervals. It is out of the scope of this paper to compare all available alternatives. We focus on a method to compare the performance of predictors devised to alert on coffee rust disease. Thus we employ one approach based on metric regression, another based on ordinal regression, and finally a classifier.

In [8], we explored regressors whose predictions were intervals of a fixed width, say 2ϵ, rather than points. Using a regression SVM, we can learn an interval regressor that optimizes the so-called ϵ-*insensitive* loss function for each width; i.e., regressors that try to include the targets in an interval of radius ϵ centered around the predicted value, or that at least try to place the interval as close as possible to the targets.

Additionally, we implemented approximate regressors by means of classifiers. In fact, the range of target values, in this case the interval $[0, 100]$ of percentages of incidence, can be discretized in, say, k bins with equal frequency. The prediction of one such bin is actually an approximation to the true target; however, we may go even further by allowing the classifiers to

predict one or more consecutive bins. Once again, uncertainty in the prediction is the rationale behind broader predictions. Following [9, 10], these predictors may be called *nondeterministic* classifiers.

The three learners considered here have in common that they use a parameter to state what can be seen as a *degree of nondeterminism*. Thus, we can achieve broader or narrower predictions simply by modifying the corresponding parameter both for classifiers and regressors. As expected, we found that the number of warnings is higher as we increase the radius of predicted intervals, causing a decrease in the number of misclassifications.

To compare the performance of each learner, we may plot the pairs of number of misclassifications and warnings obtained when varying the parameters that control the degree of nondeterminism. Thus, we obtain curves similar to Regression Error Characteristic (REC) curves [11] or reject tradeoff curves, see [5]. However, taking into account that we must consider different costs for false positives, false negatives, and warnings, the decisive comparison of learners has to be drawn from a 3D perspective. The economic and ecological costs of the use of fungicides and the risk of loss of coffee crops must be taken into account.

The conclusion is that the design of alarm functions for this agricultural problem is feasible. Under mild conditions for the relationships of costs of errors and warnings, we obtain that nondeterministic classifiers outperform the regressors compared in this study.

2. The dataset

The data used in this paper was obtained on a monthly basis from an experimental farm (Fundação Procafé, Varginha, MG, Brazil), from October 1998 to October 2006, with reports of coffee rust incidences. In September of each year (beginning of the agricultural season), eight coffee-producing plots were selected, four with thin spacing (approximately 4000 plants/ha) and four with dense spacing (approximately 8000 plants/ha). For each case, two plots were selected with a high fruit load (above 1800 kg/ha) and two with a low fruit load (below 600 kg/ha). No disease control was employed in these plots. Meteorological data was automatically registered every 30 minutes by a weather station close to where the incidence of coffee rust was being evaluated.

Bearing in mind that the alarm system can be used at any time, not only from the first day of one month to guess the incidence in the first day of the next month, we devised an input space, $\mathcal{X}$ (see Figure 1) as a set of vectors whose components are:

- Fruit load of the plantation: low (1) or high (2)
- Spacing between plants: dense (1) or thin (2)
- Percentage of leaf-fungal incidence on date d_0
- Days from d_0 till now (the day we make the prediction)
- Days from now till the target day: 1 month, 25, 20, 15 and 10 days
- Weather scores in the last 45 days

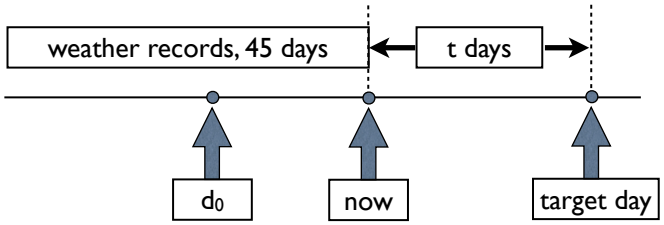

Figure 1: Structure of the entries of the coffee rust dataset. The distance from *now* till the target day, t, may be 1 month, or 25, 20, 15, or 10 days

The weather scores are 13 variables per day and include: temperature, solar radiation, number of hours of sunlight, wind speeds, rain, relative humidity, number of hours with a relative humidity above 95%, average temperature during these hours, and the same values, but during the night. For more details, see [2, 3].

The dimensionally of the vectors of the input space, $\mathcal{X}$, is 590. On the other hand, the output space in this case is just the interval of real numbers, $\mathcal{Y} = [0, 100]$, to capture the percentage of coffee leaves infected by the fungus.

3. Nondeterministic alarm functions and metric regression

As pointed out in the introduction, unfortunately the accuracy of regressors is not always completely satisfactory when we are interested in deciding whether the target value is going to be greater than a given threshold. We therefore need to go beyond conventional regression to obtain useful alarm functions. The approach proposed here attempts to smooth the alert predictions somehow. The idea is to split inputs into 3 categories: *alarms*, *non-alarms*, and doubtful situations. We shall label the third category as

a *warning*: something between an alarm and a non-alarm.The core point of this paper is to show the usefulness of such warning labels.

To implement this idea, we can use interval regressors: these are allowed to return intervals rather than single point predictions. The idea is that the true class of an entry $\boldsymbol{x}$ may be *somewhere* within the predicted interval for $\boldsymbol{x}$. In other words, we are dealing with *nondeterministic (or set-valued) hypotheses* [9, 10]: h_{ND} functions from the input space to the set of non-empty intervals of $\mathcal{Y}$

$$h_{ND} : \mathcal{X} \longrightarrow Intervals(\mathcal{Y}). \tag{1}$$

In this context, if τ is a threshold in $\mathcal{Y}$, we shall interpret the outputs of h_{ND} as follows

$$Alarm(h_{ND}(\boldsymbol{x})) = \begin{cases} \text{non-alarm} & h_{ND}(\boldsymbol{x}) \subset (-\infty, \tau] \\ \text{alarm} & h_{ND}(\boldsymbol{x}) \subset (\tau, +\infty) \\ \text{warning} & otherwise. \end{cases} \tag{2}$$

To implement interval regressors in a simple way, we may content with intervals of fixed-width, say 2ϵ with $\epsilon > 0$. Notice that the aim is to include the target values in the predicted intervals, or at least as close as possible to them; in other words, regressors that minimize the following loss function

$$loss_{\epsilon}(h(\boldsymbol{x}), y) = \max\{0, |h(\boldsymbol{x}) - y| - \epsilon\}. \tag{3}$$

But this is the purpose of Regression Support Vector Machines that use the so-called ϵ-*insensitive zone*. Formally, given a learning task $S = \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, n\}$ and a positive number ϵ, we may use, for instance, LibSVM [12]

to solve the following convex optimization problem

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-), \\ \text{s.t.} \quad & (\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle + b) - y_i \leq \epsilon + \xi_i^+, \\ & y_i - (\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle + b) \leq \epsilon + \xi_i^-, \\ & \xi_i^+, \xi_i^- \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{4}$$

In this way, we obtain the regressor

$$h_\epsilon(\boldsymbol{x}) = \sum_{i=1}^{n}(\alpha_i^- - \alpha_i^+)K(\boldsymbol{x}_i, \boldsymbol{x}) + b^*, \tag{5}$$

where K is, for instance, the *rbf* kernel,

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}};$$

b^*, α^-, and α^+ are respectively the solution and the Lagrange multipliers of the convex optimization problem (Eq. 4).

Notice that for each ϵ we obtain an optimal hypothesis, h_ϵ, according to (Eq. 4). The nondeterministic version of this hypothesis is then given by

$$h_{ND(\epsilon)}(\boldsymbol{x}) = [h_\epsilon(\boldsymbol{x}) - \epsilon, h_\epsilon(\boldsymbol{x}) + \epsilon]. \tag{6}$$

4. Alarm functions and nondeterministic classifiers

Equations (Eq. 1) and (Eq. 2) can be used employing classifiers instead of regressors. In this case, intervals may be interpreted as classes. For instance, a partition of the output space, $\mathcal{Y}$, in intervals may be the set of classes.

To accomplish this transformation, we may discretize $\mathcal{Y}$ in a number, say k, of bins of equal frequency. Therefore, the new output space for classification purposes will be

$$Dis(\mathcal{Y}) = \{1, \ldots, k\}.$$

We can view this discretization as a translation into a ranking scale of k qualitative ranks. Moreover, we can use the fact that the learning task with this discretized output space is an ordinal regression task, seeing as the classes are trivially ordered.

From the point of view of the original learning task, transforming $\mathcal{Y}$ into $Dis(\mathcal{Y})$ gives rise to a nondeterministic context. However, we can go even further. We may consider classifiers h_{ND} allowed to predict more than one bin of $\{1, \ldots, k\}$, but always a set of consecutive classes. In these cases, we would like to favor those predictions of h_{ND} that contain the true class, and a smaller rather than a larger number of additional classes.

In other words, we interpret the output $h_{ND}(\boldsymbol{x})$ as an imprecise answer to a query about the right class of an entry $\boldsymbol{x} \in \mathcal{X}$. Thus, the nondeterministic classification can be seen as a kind of *Information Retrieval* task for each entry. This approach allows us to define nondeterministic classifiers by means of an optimization problem; see [9, 10].

Before describing how to learn such nondeterministic classifiers, we need to formalize Information Retrieval measures of performance.

The most frequently used measures in this context are the *Recall* (proportion of all relevant *documents* (now classes) that are found by a search) and *Precision* (proportion of retrieved *documents* that are relevant). In symbols, in a *query* $\boldsymbol{x}$, if $y \in \{1, \ldots, k\}$ is the true (relevant) class, the *Recall* is defined

by

$$R(h(\boldsymbol{x}), y) = 1_{y \in h_{ND}(\boldsymbol{x})}. \tag{7}$$

The *Precision* is given by

$$P(h(\boldsymbol{x}), y) = \frac{1_{y \in h_{ND}(\boldsymbol{x})}}{|h_{ND}(\boldsymbol{x})|}, \tag{8}$$

where $|h_{ND}(\boldsymbol{x})|$ is the *size* of the prediction, i.e., the number of consecutive bins of $\{1, \ldots, k\}$ included in the prediction.

However, it is more informative to measure a tradeoff between *Recall* and *Precision*. The harmonic average of the two amounts is used to capture the goodness of a hypothesis in a single measure. In the weighted case, the measure is called F_β.

$$F_\beta(h_{ND}(\boldsymbol{x}), y) = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{(1+\beta^2)1_{y \in h_{ND}(\boldsymbol{x})}}{\beta^2 + |h_{ND}(\boldsymbol{x})|}. \tag{9}$$

Moreover, since F_β is bounded in $[0, 1]$, the loss of a nondeterministic hypothesis, h_{ND}, may be given by the complementary: $1 - F_\beta$. So, when $S' = \{(\boldsymbol{x'_i}, y'_i) : j = 1, \ldots, m\}$ is a test set of size m, the average loss will be computed by

$$F_\beta loss(h_{ND}, S') = \frac{1}{m} \sum_{j=1}^{m} \left(1 - F_\beta(h_{ND}(\boldsymbol{x'_j}), y'_j)\right). \tag{10}$$

For ease of reference, the *average Recall* of h_{ND} across S' will be computed by

$$Recall(h_{ND}, S') = \frac{1}{m} \sum_{j=1}^{m} 1_{y'_j \in h_{ND}(\boldsymbol{x'_j})}. \tag{11}$$

It is important to realize that for a deterministic hypothesis, h, this amount is the average of "0/1" losses, since all predictions are singletons,

$|h(\boldsymbol{x})| = 1$. Thus, the nondeterministic loss used here is a generalization of the error rate of deterministic classifiers.

These nondeterministic measures can also be applied to interval regressors. Thus, if ϵ is a positive value, the average *Recall* can be interpreted as the proportion of test examples that fall inside a tube of radius ϵ:

$$\begin{aligned} Recall(h_{ND(\epsilon)}, S') &= Inside\_tube(h_{ND(\epsilon)}, S') \\ &= \frac{1}{m}\sum_{j=1}^{m} 1_{y'_j \in h_{ND(\epsilon)}(\boldsymbol{x'_j})}. \end{aligned} \tag{12}$$

Now we are ready to explain our way of building nondeterministic classifiers, see [9, 10]. Let $\boldsymbol{x} \in \mathcal{X}$ be an input, and let us assume that we know the posterior probabilities of classes,

$$Pr(j|\boldsymbol{x}), \forall j \in Dis(\mathcal{Y}) = \{1, \ldots, k\}.$$

Then, the nondeterministic prediction $h_{ND}(\boldsymbol{x})$ optimizing the $F_\beta loss$ for a given β can be computed as follows.

The core idea is that the *expected loss* for a prediction $h_{ND}(\boldsymbol{x}) = [s_0, s_1]$, according to (Eq. 10), can be expressed in terms of the length of the interval,

$$l = s_1 - s_0 + 1, \tag{13}$$

and the probability of the interval.

In fact, with a probability of $1 - Pr([s_0, s_1]|\boldsymbol{x})$, we expect a loss of 1: the *true* class will not be included in $[s_0, s_1]$. On the other hand, with the probability of $[s_0, s_1]$, the *true* class will be in $h_{ND}(\boldsymbol{x})$ and hence the loss will be 1 minus the F_β of the prediction $h_{ND}(\boldsymbol{x}) = [s_0, s_1]$, (Eq. 10, 9). In

symbols,

$$
\begin{aligned}
&E[F_\beta loss, \boldsymbol{x}, [s_0, s_1]] \\
&\quad = \left(1 - \sum_{j=s_0}^{s_1} Pr(j|\boldsymbol{x})\right) + \left(\sum_{j=s_0}^{s_1} Pr(j|\boldsymbol{x})\right)\left(1 - \frac{1+\beta^2}{\beta^2 + l}\right) \\
&\quad = 1 - \frac{1+\beta^2}{\beta^2 + l} \sum_{j=s_0}^{s_1} Pr(j|\boldsymbol{x}). \qquad (14)
\end{aligned}
$$

Therefore, $h_{ND}(\boldsymbol{x})$ is defined as the interval with the lowest expected loss.

$$h_{ND}(\boldsymbol{x}) = arg \min_{[s_0, s_1]} E[F_\beta loss, \boldsymbol{x}, [s_0, s_1]]. \qquad (15)$$

Notice that the width of the prediction is not constant if we translate the intervals $[s_0, s_1]$ into $\mathcal{Y}$. Moreover, the length $l = |h_{ND}(\boldsymbol{x})|$ (Eq. 13) of the intervals in $Dis(\mathcal{Y})$ depends on $\boldsymbol{x}$ too; it is called the *size* of the prediction.

5. Experimental results

In this section we report the results obtained by the regression and classification methods described in this paper. Since the natural approach for coffee rust alarms is regression, we first show the performance of this method. We then move onto the nondeterministic classifiers and compare the performance of both methods. In all cases, the scores presented in Tables and Figures were estimated using a 10-fold cross-validation.

The threshold used to distinguish alarms from non-alarms is $\tau = 4.5$. This number is a usual threshold employed in the application context. Moreover, given that 4.5 represents the 25^{th} percentile in our dataset, the output space was discretized in 8 bins of equal frequency. The split points in $\mathcal{Y} = [0, 100]$ used were:

$$\{1.00, 4.50, 12.00, 18.00, 25.00, 36.85, 66.75\}.$$

5.1. Regression scores

The SVM regressors presented in Section 3 were learned using LibSVM [12], with an *rbf* kernel. The parameters C and σ were adjusted using an *internal grid search* in each training set. The ranges for this search were: $C \in [0.001, 0.01, 0.1, 1, 10, 100, 1000]$, and $\sigma \in [0.01, 0.1, 0.3, 0.5, 0.7]$. The search employed an internal 2-fold cross-validation repeated 3 times, the aim being to optimize the average ϵ-insensitive loss (Eq. 3).

Table 1 reports the regressor scores. It can be seen that the correlations between predicted and actual incidences are quite high, around 0.94. Notice that the value of the radius of the predicted interval, ϵ, has no influence on these results. But, of course, ϵ has a major impact on the proportion of points inside the tube, the *Recall* (Eq. 12). Here, the results range from a tiny 2% to a modest 43% for the regressor h_4, which has $\epsilon = 4$; i.e., a regressor whose predictions are intervals with a width of 8. Evidently, it is easier to include predictions inside wider tubes.

Table 1: Regression scores, estimated by a 10-fold cross-validation procedure, for different values of the radius ϵ of the insensitive zone or *tube*

	$\epsilon = 0.1$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$
absolute error	6.74	6.41	6.19	6.12	6.23
$loss_\epsilon$	6.73	6.34	5.94	5.61	5.38
correlation	0.94	0.94	0.94	0.94	0.95
inside_tube	0.02	0.16	0.26	0.36	0.43

In Figure 2 we represent graphically the performance of $h_{ND(4)}$ in a 10-fold cross validation experiment. Notice that the width of the predicted intervals

is $2\epsilon = 2 \times 4 = 8$. To make the figure more clear, we drew only a subset of examples: predictions made one month ahead.

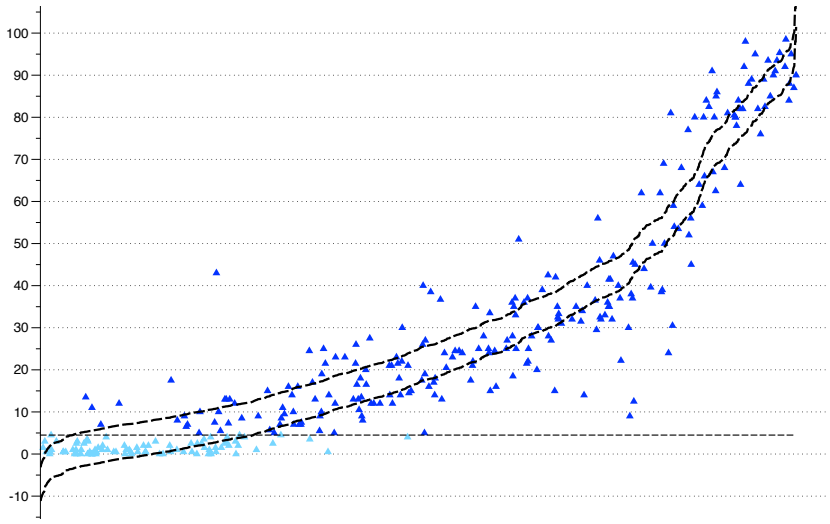

Figure 2: True incidence percentages (▲) and the predicted intervals, in 10-fold cross-validation, using the regressor $h_{ND(4)}$. The horizontal axis represents the indexes of samples ordered according to their predictions. The horizontal dashed line represents the threshold $\tau = 4.5$

5.2. Scores of nondeterministic classifiers

We implemented a couple of nondeterministic classifiers using the procedure described in Section 4, see (Eq. 14, 15). The posterior probabilities of the first one, called nd_1, were provided by a simple multiclass classifier learned by LibSVM [12], with an *rbf* kernel.

For the second classifier, nd_2, we chose a quite different base learner, the implementation of *Logistic Regression* of LibLINEAR [13, 14] with the transformation for ordinal regression developed by Frank and Hall [15]. That is

Table 2: Nondeterministic classifiers. For different values of β, we report the average *size* of predictions, $|h_{ND}(\boldsymbol{x})|$ (Eq 13); as well as the average proportion of classifications that include the true class, *Recall*, (Eq. 7, 11)

	nd_1		nd_2	
β	size	Recall	size	Recall
0.0	1.00	0.56	1.00	0.56
0.5	1.12	0.64	1.05	0.58
1.0	1.43	0.71	1.17	0.62
1.5	1.76	0.77	1.39	0.69
2.0	2.12	0.82	1.57	0.73
2.5	2.45	0.87	1.73	0.76
3.0	2.71	0.90	1.84	0.77
3.5	2.93	0.92	1.93	0.78

to say, the posterior probabilities were defined as follows. To take advantage of the ordering of the set of classes, $\{1, \ldots, k\}$, we trained $k-1$ binary classifiers (using LibLINEAR) to learn posterior probabilities $Pr(y \leq i|\boldsymbol{x})$ for $i = 1, \ldots, k-1$ and $\boldsymbol{x} \in \mathcal{X}$. We then estimate the posterior probabilities of classes using

$$Pr(y = i|\boldsymbol{x}) = Pr(y \leq i|\boldsymbol{x}) - Pr(y \leq i-1|\boldsymbol{x}), \tag{16}$$

assuming that $Pr(y \leq k|\boldsymbol{x}) = 1$ and $Pr(y \leq 0|\boldsymbol{x}) = 0$. Some problematic cases could return negative probabilities for some classes; in these cases, we normalize the values.

We also tested other combinations of these base learners using or not the Frank and Hall approach. The results were quite similar. Thus, we

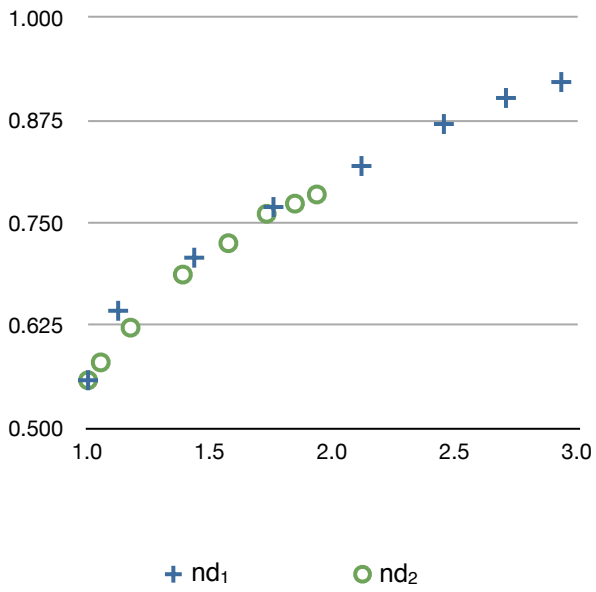

Figure 3: Average *size* (horizontal axis) and *Recall* of predictions with $\beta \in \{0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$ for the nondeterministic classifiers nd_1 and nd_2. The optimum point would be $(1, 1)$

report only the scores achieved by nd_1 and nd_2 to show the possibilities of nondeterministic classifiers in this context.

Table 2 gathers the average *size* and *Recall* of predictions of nd_1 and nd_2 computed for different β values; see Section 4. As anticipated, both scores increase with increasing values of β. Thus, for $\beta = 3.5$, the predictions of nd_1 include almost 3 bins on average; therefore, the *Recall* reaches 0.92.

To compare the performance of the two nondeterministic classifiers, Figure 3 shows pairs of *size* and *Recall* obtained for each β and each classifier. The theoretically perfect pair would be the point $(1, 1)$ corresponding to a classifier with a perfect *Recall* (1) obtained with predictions of only 1 class for

all inputs. We can see that the sequence of points representing the scores of both classifiers perform similarly. Quite acceptable results can be obtained with nd_1 and nd_2, although the values of β may be different for each classifiers. For instance, with nd_1 and $\beta = 1.5$ it is possible to obtain an average *Recall* of 0.77 with predictions of an average size of 1.76; while nd_2 needs a $\beta = 2.5$ to obtain similar scores: 0.76 and 1.73; see Table 2.

On the other hand, notice that Table 2, for $\beta = 0$, shows also the scores of the *deterministic* classifiers obtained with LibSVM (multiclass) and LibLINEAR (Frank and Hall approach). In both cases the *Recall* (usually called *accuracy* for deterministic classifiers) is 0.56.

As we did for regressors, in Figure 4 we represent graphically the predictions and true values for the classifier nd_1 with $\beta = 1.5$. Again we used only predictions made one month ahead. Notice that the inputs are arranged in different way in Figures 2 and 4, since the ordering of samples in the vertical axis is given by the predictions in the two cases.

5.3. Comparison of regression and classification alarms

In practice, we assume that the monitoring system for coffee rust will be used with certain frequency. To compare learners simulating this behavior, in this section, we proceed as follows. In the dataset, for each target day there is a group of 5 examples: one for each $t \in 10, 15, 20, 25, 30$; see Section 2 and Figure 1. Thus, during the experiments reported in this section, each group of examples was dealt as a unit. In other words, the examples of the group were never separated into training and test splits in the cross-validations.

The definition of alarms was also modified from (Eq. 2) so as to consider

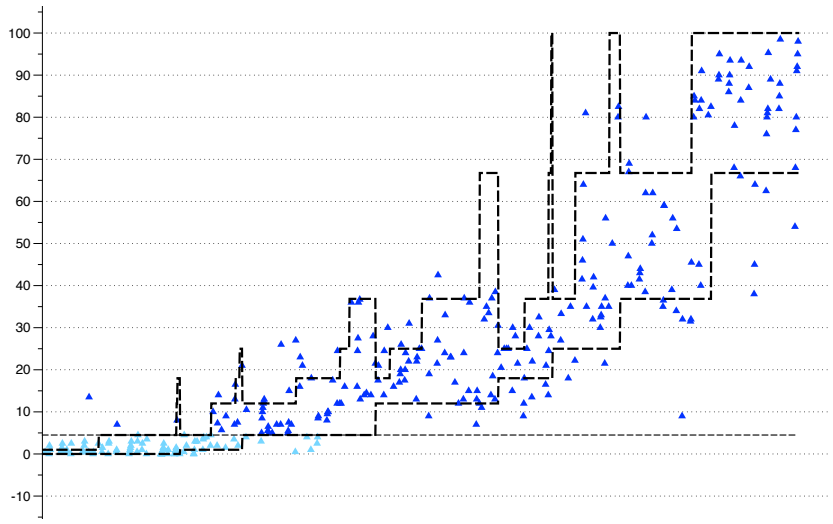

Figure 4: True incidence percentages (▲) and the predicted intervals in 10-fold cross-validation using the classifier learned with nd_1 for $\beta = 1.5$

the time dimension of the data. We consider the predictions of a group of examples with the same target to be an alarm if any of the predictions for $t \in \{10, 15, 20, 25, 30\}$ is an alarm. On the other hand, if for all t, the predictions are non-alarms, then we assume that the prediction for the group is a non-alarm. In any other case, the prediction will be considered a *warning*.

Table 3 shows the confusion matrices both for regressors and nondeterministic classifiers using the previous definition. The lower part of the table shows the results for regressors. As expected, we observe that the value of the radius ϵ of predictions is absolutely crucial. Thus, for *deterministic* regression (the default value of the insensitive zone used was $\epsilon = 0.1$), there are no doubtful classifications. No warnings appear in the corresponding columns of Table 3 for this regressor. Unfortunately, the consequence is that the number of errors is too high: 14 false non-alarms, and 16 false alarms.

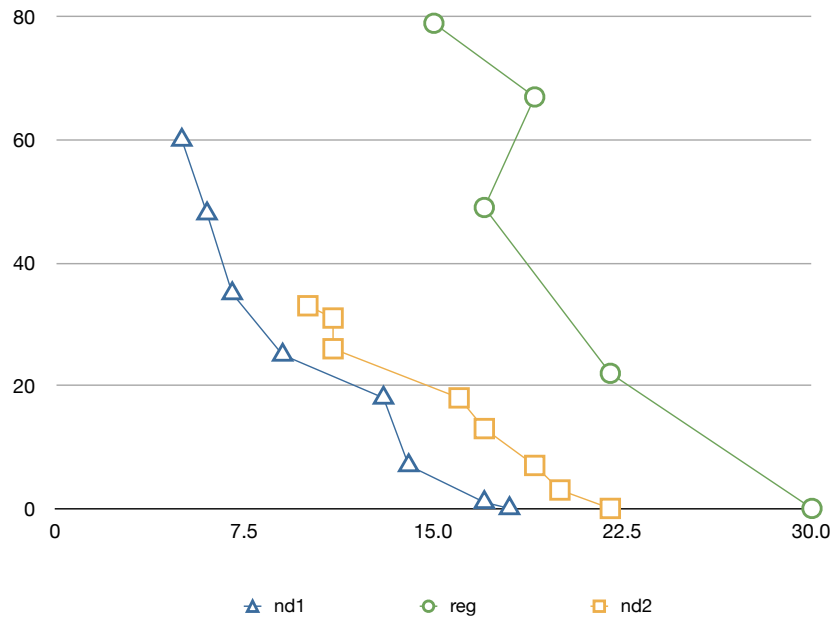

Figure 5: Performance of the nondeterministic hypotheses represented in the product space of *number of errors* (horizontal axis) and *warnings* (vertical). The optimum point would be $(0, 0)$.

For wider prediction intervals ($\epsilon \geq 1$), the number of errors decreases dramatically, but it is at the cost of an increase in warning predictions. Thus, for $\epsilon = 1$ the number of false non-alarms is only 5 with 22 warnings (6.5% of all cases), and 17 false alarms (5%). With $\epsilon \geq 2$, the number of false non-alarms is zero, but the warnings rise to $14.4\%, 19.7\%$ and 23.2% respectively for $\epsilon = 2, 3, 4$.

The upper part of Table 3, on the other hand, shows the scores of non-deterministic classifiers for different values of β. Once again, the increase in the degree of nondeterminism, in this case the value of β, marks a transfer from errors to warnings.

In Figure 5 we represented the sequences of errors versus warnings for regressors and nondeterministic classifiers. Since the point $(0, 0)$ would be the optimum, we observe that the sequence of nd_1 scores seem to be better than the sequence of nd_2. The scores due to regressors would be the worst of the three in the sense of being further from $(0, 0)$.

Notice that these curves are somehow similar to REC curves [11], the generalization of ROC curves for regression learning tasks. In fact the curves are the error-reject tradeoff curves of Chow [5].

5.4. Considering asymmetric costs

In the curves of Figure 5, the misclassifications (errors) are summed together. Then it is not possible to consider that the costs of false positives and false negatives may be different as it is the case in this alarm context. If we like to consider asymmetric costs for both kinds of errors and even for warnings, we must consider somehow 3D representations of the performance of nondeterministic learners.

Table 4 reports for each learner, the percentages of false negatives (e^- in the table), false positives (e^+), and warnings (w). In fact, this part is just another way to present the information collected in Table 3.

We assume that costs are computed by a linear function

$$cost(e^-, e^+, w) = e^- C_{e^-} + e^+ C_{e^+} + w C_w. \tag{17}$$

According to the introduction, the costs of false negatives is higher than the cost of false positives. Additionally, we suppose that the cost of warnings is not null, although small and of course lower than the cost of false positives.

In symbols,

$$C_{e^-} > C_{e^+} \gg C_w. \tag{18}$$

Notice that it is very hard to quantify some of the costs involved in these kinds of errors. This is the case of the ecological costs, or the decrease in coffee quality. Thus, we prefer to express the costs by means of intervals that follow the general idea and that may be discussed in the application field.

We have identified two different settings that roughly speaking grasp the relationships of the costs of the defects of coffee rust alarms. First we consider that the cost of warnings (C_w) is just 1% of the cost of false negatives (C_{e^-}). In this setting we have that the predictions of learner nd_1 with $\beta = 3.5$ has the lowest cost for a wide range of values for the cost of false positives, C_{e^+}, whenever

$$\frac{15}{100} C_{e^-} \leqslant C_{e^+} \leqslant \frac{90}{100} C_{e^-}.$$

In Table 4 the best predictions have costs between 6.47 and 15.29 when the cost for false negatives (C_{e^-}) is set arbitrarily to 1000 to ease the reading of the results.

The second setting assumes that C_w is 5% of C_{e^-}. In that case, the learner nd_1 has again the cheapest costs with $\beta = 2.5$ for a narrower range now:

$$\frac{15}{100} C_{e^-} \leqslant C_{e^+} \leqslant \frac{60}{100} C_{e^-}.$$

To finish this section, it is important to make a remark about the method employed to handle costs. If we have in advance a fixed list of costs, we can introduce them in the procedure described in Section 4 to build nondeterministic classifiers. However, if we only have approximate costs that have to be expressed by ranges, as it is our case (and probably many others), the

approach presented in this section is a reliable method to decide the best degree of nondeterminism and the best learner.

6. Conclusion

We have discussed how to learn a predictor to obtain an alarm system for coffee rust, the main coffee crop disease in the world. In this case, the aim is to employ chemical prevention of the disease only when necessary so as to achieve healthier quality products and reductions in costs and environmental impact. In this context, the costs of misclassifications are not symmetrical. False negative predictions would lead to not preventing a severe increase in the incidence of the disease, and may lead to the loss of coffee crops.

From a formal point of view, the aim is to predict the value of future incidence of the disease on the basis of previously measured values and weather records. If the incidence exceeds a certain threshold, we have an alarm situation.

We have presented a collection of predictors able to forecast an interval of possibilities instead of a single value. They can be implemented using many different approaches. In this paper, we considered nondeterministic classifiers after discretization of the target value, and an straightforward implementation for interval predictors obtained by Regression Support Vector Machines using the so-called ϵ-insensitive zone.

In any case, the use of interval predictors allows us to distinguish a third type of situation between alarms and non-alarms: when the predictions include both alarm and non-alarm situations; we call these *warnings*. As ex-

pected, we found that the number of warnings is higher as we increase the radius of predicted intervals, while the number of wrong predictions decreases. In other words, predictions are more reliable when they assert alarms or non-alarms, but further analysis may be required to decide what to do in uncertain situations (warnings).

To compare the performance of these alarm predictors, we presented a framework where the costs of false negatives are higher than that of false positives, and both are higher than the cost of warning predictions. Under mild conditions, we identified ranges of costs where one of the nondeterministic classifiers outperforms the other learners.

From a practical point of view, the implementation of any of these predictors is very affordable; the only requirement is a cheap weather station able to register the data described in Section 2.

We believe that the use of nondeterministic predictors could be successfully generalizable to other prediction tasks where the target values are not easily predictable by conventional classifiers or regressors. Moreover, the comparison method deployed in the previous section can be useful for other situations with different costs of false positives and false negatives.

7. Acknowledgements

The authors are grateful to the Brazilian Fundação Procafé for providing the data used in this paper. The research reported here has been supported in part under grant TIN2008-06247 from the MICINN (Ministerio de Ciencia e Innovación, of Spain), and grant 2009/07366-5 from the FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo, Brazil).

References

[1] L. Japiassú, A. Garcia, A. Miguel, C. Carvalho, R. Ferreira, L. Padilha, J. Matiello, Influência da carga pendente, do espaçamento e de fatores climáticos no desenvolvimento da ferrugem do cafeeiro, in: Simpósio de Pesquisa dos Cafés do Brasil (2007), Águas de Lindóia, SP, Brasil.

[2] C. A. A. Meira, L. H. A. Rodrigues, S. A. de Moraes, Análise da epidemia da ferrugem do cafeeiro com árvore de decisão, Tropical Plant Pathology 33 (2008) 114–124.

[3] C. A. A. Meira, L. H. A. Rodrigues, S. A. de Moraes, Modelos de alerta para o controle da ferrugem-do-cafeeiro em lavouras com alta carga pendente, Pesq. agropec. bras 44 (2009) 233–242.

[4] P. Bartlett, M. Wegkamp, Classification with a reject option using a hinge loss, Journal of Machine Learning Research 9 (2008) 1823–1840.

[5] C. Chow, On optimum recognition error and reject tradeoff, IEEE Transactions on Information Theory 16 (1970) 41–46.

[6] H. Papadopoulos, K. Proedrou, V. Vovk, A. Gammerman, Inductive confidence machines for regression, in: T. Elomaa, H. Mannila, H. Toivonen (Eds.), Proceedings of the European Conference on Machine Learning (ECML), Springer, 2002, pp. 345–356.

[7] E. Hullermeier, Credible Case-Based Inference Using Similarity Profiles, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 847–858.

[8] O. Luaces, L. H. A. Rodrigues, C. A. A. Meira, J. R. Quevedo, A. Bahamonde, Viability of an alarm predictor for coffee rust disease using regression, in: Proceedings of the IEA-AIE 2010, Córdoba, Spain.

[9] J. Alonso, J. J. del Coz, J. Díez, O. Luaces, A. Bahamonde, Learning to predict one or more ranks in ordinal regression tasks, in: W. Daelemans, B. Goethals, K. Morik (Eds.), Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), LNAI 5211, Springer, 2008, pp. 39–54.

[10] J. J. del Coz, J. Díez, A. Bahamonde, Learning nondeterministic classifiers, Journal of Machine Learning Research 10 (2009) 2273–2293.

[11] J. Bi, K. P. Bennett, Regression error characteristic curves, in: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, pp. 43 – 50.

[12] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2001. Software available at url http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[13] C.-J. Lin, R. C. Weng, S. S. Keerthi, Trust region Newton method for logistic regression, Journal of Machine Learning Research 9 (2008) 627–650.

[14] R. Fan, K. Chang, C. Hsieh, X. Wang, C. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[15] E. Frank, M. Hall, A simple approach to ordinal classification, in: Proceedings of ECML 2001, volume 2167/2001 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 145–156.

Table 3: Confusion matrices obtained by nondeterministic classifiers (top) and regressors (bottom). From left to right there is an increase in the degree of nondeterminism represented by the values of β and the radius ϵ of the insensitive zone or *tube* respectively. Columns represent true classes: alarm (a), and non-alarm ($\neg a$). Rows report the occurrences of each possible prediction *(Pre)* (alarm, warning (w), non-alarm)

		True Classes							
		$\neg a$	a	$\neg a$	a	$\neg a$	a	$\neg a$	a
	Pre.	$\beta = 0.0$		$\beta = 0.5$		$\beta = 1.0$		$\beta = 1.5$	
nd_1	$\neg a$	76	9	76	8	74	5	68	4
	w	0	0	0	1	2	5	8	10
	a	9	246	9	246	9	245	9	241
nd_2	$\neg a$	72	9	72	8	71	7	69	5
	w	0	0	1	2	2	5	4	9
	a	13	246	12	245	12	243	12	241

		True Classes							
		$\neg a$	a	$\neg a$	a	$\neg a$	a	$\neg a$	a
	Pre.	$\beta = 2.0$		$\beta = 2.5$		$\beta = 3.0$		$\beta = 3.5$	
nd_1	$\neg a$	66	2	61	1	51	1	42	1
	w	12	13	18	17	29	19	39	21
	a	7	240	6	237	5	235	4	233
nd_2	$\neg a$	68	4	68	1	66	1	66	1
	w	5	13	7	19	9	22	10	23
	a	12	238	10	235	10	232	9	231

		True Classes									
		$\neg a$	a	$\neg a$	a	$\neg a$	a	$\neg a$	a	$\neg a$	a
	Pre.	$\epsilon = 0.1$		$\epsilon = 1.0$		$\epsilon = 2.0$		$\epsilon = 3.0$		$\epsilon = 4.0$	
regres	$\neg a$	69	14	55	5	34	0	10	0	5	0
	w	0	0	13	9	34	15	56	11	65	14
	a	16	241	17	241	17	240	19	244	15	241

Table 4: Costs for nondeterministic classification in 4 different sets of cost values for the percentages of errors (false negatives (e^-), and false positives (e^+)) and warnings (w). In bold we highlighted the best scores

				costs	C_{e^-} C_{e^+} C_w	1000 150 10	1000 900 10	1000 150 50	1000 600 50
	β	e^-	e^+	w					
nd_1	0.0	2.65%	2.65%	0.00%		30.44	50.29	30.44	42.35
	0.5	2.35%	2.65%	0.29%		27.53	47.38	27.65	39.56
	1.0	1.47%	2.65%	2.06%		18.88	38.74	19.71	31.62
	1.5	1.18%	2.65%	5.29%		16.26	36.12	18.38	30.29
	2.0	0.59%	2.06%	7.35%		9.71	25.15	12.65	21.91
	2.5	0.29%	1.76%	10.29%		6.62	19.85	**10.74**	**18.68**
	3.0	0.29%	1.47%	14.12%		6.56	17.59	12.21	18.82
	3.5	0.29%	1.18%	17.65%		**6.47**	**15.29**	13.53	18.82
	β	e^-	e^+	w					
nd_2	0.0	2.65%	3.82%	0.00%		32.21	60.88	32.21	49.41
	0.5	2.35%	3.53%	0.88%		28.91	55.38	29.26	45.15
	1.0	2.06%	3.53%	2.06%		26.09	52.56	26.91	42.79
	1.5	1.47%	3.53%	3.82%		20.38	46.85	21.91	37.79
	2.0	1.18%	3.53%	5.29%		17.59	44.06	19.71	35.59
	2.5	0.29%	2.94%	7.65%		8.12	30.18	11.18	24.41
	3.0	0.29%	2.94%	9.12%		8.26	30.32	11.91	25.15
	3.5	0.29%	2.65%	9.71%		7.88	27.74	11.76	23.68
	ϵ	e^-	e^+	w					
reg	0.1	4.12%	4.71%	0.00%		48.24	83.53	48.24	69.41
	1.0	1.47%	5.00%	6.47%		22.85	60.35	25.44	47.94
	2.0	0.00%	5.00%	14.41%		8.94	46.44	14.71	37.21
	3.0	0.00%	5.59%	19.71%		10.35	52.26	18.24	43.38
	4.0	0.00%	4.41%	22.94%		8.91	42.00	18.09	37.94