# Naive Bayes Classification for Subset Selection in a Multi-label Setting

Luca Mossina and Emmanuel Rachelson
*Departement of Complex Systems Engineering*
*ISAE–SUPAERO, Université de Toulouse*
Toulouse, France
luca.mossina@isae-supaero.fr, emmanuel.rachelson@isae-supaero.fr

***Abstract*—This article introduces a novel probabilistic formulation of multi-label classification based on the Bayes theorem. Under the naive hypothesis of conditional independence of features given the labels, a pseudo-bayesian inference approach is adopted, known as Naive Bayes. The prediction consists of two steps: the estimation of the size of the target label set and the selection of the elements of this set. This approach is implemented in the *NaiBX* algorithm, an extension of naive Bayes into the multi-label domain. Its properties are discussed and evaluated on real-world data.**

***Index Terms*—Naive Bayes, multi-label classification, subset selection.**

## I. Introduction

Multi-label classification (MLC) aims at predicting a set of one or more labels $\mathbf{y} = \{y^1, ..., y^m \mid y^i \in \mathcal{L}\}$, as a function of some input $\mathbf{x}$. Applications cover a diverse range of fields such as text categorization [7], where more than a topic can be associated to a law, or gene function analysis [5], where a gene can be responsible for multiple mechanisms (transcription, cell growth and division, metabolism etc.).

*NaiBX*, our proposed algorithm, works in two steps. First, it learns to predict the number $m$ of labels $y^i \in \mathcal{L}$ to be included in the subset for a given input $\mathbf{x}$. After that, it proceeds predicting the first label $y_{(1)}$ given $\mathbf{x}$ and $m$, the second label $y_{(2)}$ given $\mathbf{x}$, $m$ and $y_{(1)}$, and so on until completion. At the end of the procedure, one will have followed the inference sequence $m \rightarrow y_{(1)} \rightarrow y_{(2)} \rightarrow ... \rightarrow y_{(k)} \rightarrow ... \rightarrow y_{(m)}$. For this purpose, we construct a cascade of naive Bayes classifiers (NBC), where predictor number $k$ predicts the $k$-th element in the subset. Opting for a NBC means assuming conditional independence of features given the labels. This hypothesis induces a dramatic simplification of the predictors computation's complexity (both in time and space), and the overall learning task boils down to training the elementary parameters of a single NBC that can be used for prediction of every element in the subset.

We first introduce the state-of-the-art approaches to MLC in Section II. We recall the principle and properties of Naive Bayes classification in Section III-A. In Section III-B, we introduce a general "cascade of predictors" approach to multi-label classification and propose a Naive Bayes algorithm as the base classifier. We derive a learning algorithm called *NaiBX* and discuss its properties in Section III-D. Experimental results and comparisons are presented in Section IV. Finally, we summarize and conclude in Section V.

## II. Multi-label Classification

Our method addresses the problem of MLC [8][20]; given a discrete set $\mathcal{L}$ of options ("labels") and an arbitrary feature space $\mathcal{X}$, one will look for the most appropriate subset of those options via the classifier $h : \mathcal{X} \longrightarrow \mathcal{P}(\mathcal{L})$.

If the set of available labels is $\mathcal{L} = \{1, 2, ..., L\}$, a target subset of labels $\mathbf{y}$ can be, for example, $\mathbf{y} = \{3, 5, 6, 7, 9, \ldots, L\} \equiv [0, 0, 1, 0, 1, 1, 1, 0, 1, \ldots, 1] \in \{0, 1\}^L$. The latter is often referred to as *one-hot-encoding*.

Existing binary or multi-class classification algorithms have been adapted to the multi-label case, a process known as *algorithm adaption*. ML-kNN [19] and BPMLL [18] are, for example, MLC versions of *k-nearest neighbours* and *back-propagation*. We find, however, that the most efficient approaches are obtained via *problem transformation*, where the multi-label instance is decomposed into simpler binary of multiclass classification problems; the two main paradigms are the *Label Powerset* (LP) and the *Binary Relevance* (BR) [13] methods. The first consists in turning a multi-label problem into a multi-class one, mapping directly an element of the feature space to any of the elements in $\mathcal{P}(\mathcal{L})$. As the number of classes grows exponentially with the number of labels, LP cannot deal with big sets of labels. The second, BR, consists in independently training a binary classifier for each admissible label $l_j \in \mathcal{L}$, obtaining as many classifiers as there are labels.

The *RAndom k-LabEL sets* (RAkEL) algorithm [14], is a particularly interesting and effective variation of LP, where one trains $m$ models whose targets are a subset of $k$ labels from those available. That is, instead of having all the elements in $\mathcal{P}(\mathcal{L})$ as targets, a set of arbitrarily sized label clusters are employed. This makes problems of up to a few dozens of labels manageable.

Stemming from the principles of BR, [10] extended the method by taking into account information about label interdependence. Incrementally, at each step, what was previously predicted is taken into account. Their *Classifier Chains* (CC) meta-algorithm first starts by predicting whether label $l_1$ is to be included in the target vector. It then continues predicting the presence of the second label $l_2$ given the information contained in the data *and* whether $l_1$ was included

in the target vector ($\hat{Y}_1 \in \{0,1\}$). At the $i$-th step we have $\hat{Y}_i = h(X, \hat{Y}_{i-1}, \hat{Y}_{i-2}, ..., \hat{Y}_1)$. The order of evaluation of the binary labels $Y_i \in \{0,1\}$ can affect negatively the performance of the algorithm. The *Ensemble of Classifier Chains* (ECC) [11] extends CC accounting for this limitation: $m$ different predictors are trained on random permutations of the labels, like for example: $h^1(Y_1, Y_4, Y_3, Y_2, \dots)$, $h^2(Y_3, Y_1, Y_2, Y_4, \dots), \dots, h^m(Y_4, Y_3, Y_2, Y_1, \dots)$. Then, one applies a *bagging* [1] step for the selection, via a threshold function, of the best labels. In [4] this method was refined via the *Probabilistic Classifier Chains* (PCC). At each level of the chain they compute a joint distribution, which is the cause of higher computational costs. Because of this, recent advances in CC go towards approximate methods based on Monte Carlo sampling [3] and tree search[2].

## III. Naive Bayes for Subset Selection

Let $\mathbf{X} = (X_1, \dots, X_n)$ be the random vector of observations $\mathbf{x}$, taking its values in $\mathcal{X}$, and let $C$ be the random variable describing the class associated to $\mathbf{X}$ in a classification problem. A probabilistic classifier $f$ assigns the class $c$ to a new observation $\mathbf{x}$ if $c$ maximizes the conditional probability of $C = c$ given that $\mathbf{X} = \mathbf{x}$.

### A. Naive Bayes Classification

As estimating a multivariate conditional distribution can be rather challenging, the Naive Bayes simplification is widely adopted [6] consisting in supposing that, $\forall (i,j) \in [1,n]^2$,

$$\mathbb{P}(X_i \mid C, X_j) = \mathbb{P}(X_i \mid C).$$

While yielding poor probability estimations, the classifications inferred are of good quality [16]. A Naive Bayes Classifier is then deduced as

$$\begin{aligned} f_{NBC}(\mathbf{x}) &= \operatorname*{argmax}_{c \in \mathcal{C}} \mathbb{P}(C = c \mid \mathbf{X} = \mathbf{x}) \\ &= \operatorname*{argmax}_{c \in \mathcal{C}} \frac{\mathbb{P}(C = c)\mathbb{P}(\mathbf{X} = \mathbf{x} \mid C = c)}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \\ &= \operatorname*{argmax}_{c \in \mathcal{C}} \mathbb{P}(C = c)\mathbb{P}(\mathbf{X} = \mathbf{x} \mid C = c) \\ &= \operatorname*{argmax}_{c \in \mathcal{C}} \left[ \mathbb{P}(C = c) \prod_{i=1}^{n} \mathbb{P}(X_i = x_i \mid C = c) \right], \end{aligned} \tag{1}$$

as the denominator does not depend on $c \in \mathcal{C}$.

### B. Cascade of Predictors

In MLC we look for a collection of labels $\mathbf{y} \subset \mathcal{L}$. An intuitive way to proceed is to consider that selecting a given-size subset consists in choosing a first element in $\mathcal{L}$, then a second given the first, then a third given the first and second, and so on until one reaches the appropriate subset size. Selecting a subset of $\mathcal{L}$ can be done by choosing an ordered sequence of values of $\mathcal{L}$ if our selection function at each step effectively re-creates the correct unordered subset. This approach differs from classifier chains since it does not predict in sequence whether the $|\mathcal{L}|$ labels belong or not to the target, but rather picks them incrementally. Notably, the cascade architecture does not rely on an a priori ordering of the labels.

Let $Y$ be the random variable describing the subset of $\mathcal{L}$ that should be associated to $\mathbf{x}$. The target of the classification algorithm is to learn the correct mapping from $\mathbf{x}$ to realizations of $Y$. We write $\bar{y}$ such realizations of $Y$ to avoid confusion with vectors $\mathbf{y}$ of values of $\mathcal{L}$. Then the classifier $f$ we are searching for is

$$f(\mathbf{x}) = \operatorname*{argmax}_{\bar{y} \in \mathcal{P}(\mathcal{L})} \mathbb{P}(Y = \bar{y} \mid \mathbf{X} = \mathbf{x}). \tag{2}$$

In order to sequentially select the elements of the optimal $\bar{y}$, we want to decompose the probability of Equation 2 into elementary probabilities related to each element $y_i$ of $\bar{y}$. Such elementary probabilities are related to the random event "$y_i \in Y$". Let $M$ be the random variable describing the size of $Y$. Then a subset $\bar{y}$ is composed of elements $y_1$ until $y_k$, where $M$ takes the value $k$. Given $\bar{y} = \{y_1, \dots, y_k\} \subset \mathcal{L}$, the following statements hold:

$$\text{“}y_1 \in Y\text{”} \wedge \dots \wedge \text{“}y_k \in Y\text{”} \quad \Leftrightarrow \bar{y} \subset Y \tag{3}$$

$$\text{“}y_1 \in Y\text{”} \wedge \dots \wedge \text{“}y_k \in Y\text{”} \wedge \text{“all others } \notin Y\text{”} \quad \Leftrightarrow \bar{y} = Y \tag{4}$$

$$\text{“}y_1 \in Y\text{”} \wedge \dots \wedge \text{“}y_k \in Y\text{”} \wedge \text{“}M = k\text{”} \quad \Leftrightarrow \bar{y} = Y \tag{5}$$

Equation 3 expresses the fact that individual properties on the $y_i$ can help characterize the probability that a given subset $\bar{y}$ is included in $Y$. Equation 4 helps expressing that Y is precisely equal to such a subset $\bar{y}$. Its formulation is equivalent to the target of CC algorithms. Finally, Equation 5 is of particular interest to us since it states that the subset that is both included in $Y$ and has the same size as $Y$ is precisely equal to $Y$. For any sequence of values $y_1, \dots, y_i \subset \mathcal{L}$, we introduce the notation

$$\begin{aligned} &p(y_i \mid \mathbf{x}, m, y_1, \dots, y_{i-1}) = \\ &\quad \mathbb{P}(y_i \in Y \mid \mathbf{X} = \mathbf{x}, M = m, y_1 \in Y, \dots, y_{i-1} \in Y). \end{aligned}$$

We then use Equation 5 to decompose the probability estimate of the probabilistic classifier in Equation 2, using the chain rule

$$\begin{aligned} \mathbb{P}(Y = \bar{y} \mid \mathbf{X} = \mathbf{x}) &= \mathbb{P}(M = m \mid \mathbf{X} = \mathbf{x}) \\ &\quad \times \mathbb{P}(\bar{y} \subset Y \mid \mathbf{X} = \mathbf{x}, M = m) \\ &= \mathbb{P}(M = m \mid \mathbf{X} = \mathbf{x}) \\ &\quad \times\ p(y_m \mid \mathbf{x}, m, y_1, \dots, y_{m-1}) \\ &\quad \times\ \ p(y_{m-1} \mid \mathbf{x}, m, y_1, \dots, y_{m-2}) \\ &\quad \times\ \ (\dots) \\ &\quad \times\ \ p(y_2 \mid \mathbf{x}, m, y_1) \\ &\quad \times\ \ p(y_1 \mid \mathbf{x}, m) \\ &= \mathbb{P}(M = m \mid \mathbf{X} = \mathbf{x}) \times \\ &\quad \prod_{i=1}^{m} p(y_i \mid \mathbf{x}, m, y_1, \dots, y_{i-1}). \end{aligned} \tag{6}$$

So, writing $s(\mathbf{x}) = \max_{\bar{y} \in \mathcal{P}(\mathcal{L})} \mathbb{P}(Y = \bar{y} \mid \mathbf{X} = \mathbf{x})$, from Equation 2 we derive

$$s(\mathbf{x}) = \max_{\bar{y} \in \mathcal{P}(\mathcal{L})} \left( \mathbb{P}(M = m \mid \mathbf{X} = \mathbf{x}) \times \prod_{i=1}^{m} p(y_i \mid \mathbf{x}, m, y_1, \ldots, y_{i-1}) \right).$$

In [4] it is argued that the Bayes optimal classifier solves the maximization problem in Equation 2 to optimality. The CC approach, however, exploits Equation 4 and adopts a greedy search heuristic consisting in incrementally picking the most (marginally) probable labels in a predefined (artificial) order. Our cascade architecture somehow falls in between these two extremes. It adopts a greedy, possibly sub-optimal search method that incrementally picks labels in the label set, but does not rely on any predefined ordering of the labels. The cascade architecture searches for a solution to the maximization problem of Equation 2 by computing the heuristic score function

$$\begin{aligned} s(\mathbf{x}) = \max_{y_m \in \mathcal{L}} \Bigg[ & p(y_m \mid \mathbf{x}, m, y_1, \ldots, y_{m-1}) \times \\ & \max_{y_{m-1} \in \mathcal{L}} \Bigg[ p(y_{m-1} \mid \mathbf{x}, m, y_1, \ldots, y_{m-2}) \times \\ & \quad \ldots \times \\ & \quad \max_{y_2 \in \mathcal{L}} \Big[ p(y_2 \mid \mathbf{x}, m, y_1) \times \\ & \quad \max_{y_1 \in \mathcal{L}} \big[ p(y_1 \mid \mathbf{x}, m) \times \\ & \quad \max_{m \in [0,1,\ldots,\lvert\mathcal{L}\rvert]} \mathbb{P}(M = m \mid \mathbf{X} = \mathbf{x}) \big] \Big] \Bigg] \Bigg]. \end{aligned}$$

Each of the $m + 1$ probability estimators in the product above is a classifier in itself. The feature space of $p(y_k \mid \mathbf{x}, m, y_1, \ldots, y_{k-1})$ is $\mathcal{X} \times \mathbb{N} \times \mathcal{L}^{k-1}$. We call such a structure a *cascade* of predictors. The cascade structure unfolds seamlessly from the application of the chain rule (see Figure 1). In a cascade, one predicts the number of elements in the subset, then the first value of the subset, then the second using the results from the computation of the first, etc.

### C. Cascade of NBCs

Any efficient classification algorithm can be used to predict each level in the cascade. This implies storing in memory $|\mathcal{L}| + 1$ classifiers having increasingly complex feature spaces and predicting values in a class set of size $|\mathcal{L}|$, which may not scale up to large label sets. Furthermore, the feature spaces of the last predictors in the cascade are complex, requiring powerful learning architectures, lots of data and possibly very long training times. Taking NBCs as base classifiers for each level in the cascade induces a dramatic simplification of both training and storage of the multi-label classifier. Let us suppose that each of these estimators is built upon the Naive Bayes assumption. Based on the conclusions of [16][17], although the probability estimates of these classifiers are poor, at each step of the cascade the computed $\operatorname{argmax}$ remains close to optimal. Eventually, we are left with $|\mathcal{L}|+1$ NBCs: one for the subset size prediction and one for each level in the cascade. If we start the numbering at zero, predictor zero estimates $\mathbb{P}(M \mid \mathbf{X})$, then predictor one estimates $p(y_1 \mid \mathbf{x}, m)$, predictor two estimates $p(y_2 \mid \mathbf{x}, m, y_1)$ and so on.

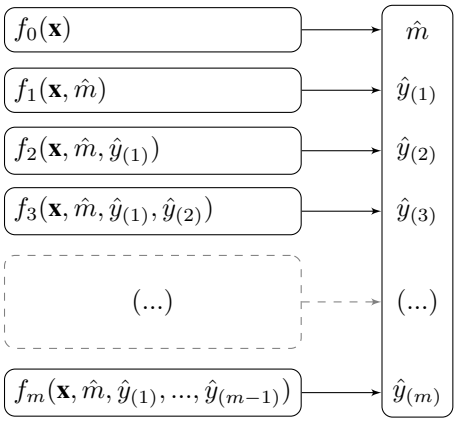


Fig. 1. Illustration of a cascade of predictors

Let $f_k$ be the selection function of predictor number $k$. Since it is a Naive Bayes classifier, according to Equation 1, its selection function decomposes as

$$\begin{aligned} f_k(\mathbf{x}, m, y_1, \ldots, y_{k-1}) = & \operatorname*{argmax}_{y_k \in \mathcal{L}} \mathbb{P}(y_k \in Y) \\ & \times \mathbb{P}(M = m \mid y_k \in Y) \\ & \times \prod_{i=1}^{n} p(X_i = x_i \mid y_k \in Y) \\ & \times \prod_{j=1}^{k-1} \mathbb{P}(y_j \in Y \mid y_k \in Y) \end{aligned} \quad (7)$$

Computing the selection function $f_k$ requires evaluating each factor in Equation 7, which are univariate probability estimators. These are not specific to the $k$-th step in the cascade: take two predictors $f_k$ and $f_{k'}$, both will make use of the same generic estimators $\mathbb{P}(y \in Y)$, $p(X_i \mid y \in Y)$, $\mathbb{P}(M \mid y \in Y)$ and $\mathbb{P}(y' \in Y \mid y \in Y)$. The same univariate probability estimators are simply combined in different fashions at the different stages of the cascade.

The cases of $f_1$ and $f_0$ require different computations. Recall that $f_1(\mathbf{x}, m)$ is the selection function of the first label. Its computation makes use of the same $\mathbb{P}(y \in Y)$, $p(X_i \mid y \in Y)$ and $\mathbb{P}(M \mid y \in Y)$ probability estimators as the rest of the cascade (simply it does not use the $\mathbb{P}(y' \in Y \mid y \in Y)$ estimator). Finally, $f_0(\mathbf{x})$ selects the most probable subset size associated to $\mathbf{x}$ via the relation $\mathbb{P}(M|X) \overset{Bayes}{\propto} \mathbb{P}(M) \times \mathbb{P}(X|M)$, yielding another Naive Bayes Classifier.

Table I summarizes the 6 univariate distributions that are required for the computation of all levels in the cascade, along with the space complexity of their storage (detailed in section

| | Distribution | Space complexity |
|---|---|---|
| $(D1)$ | $\mathbb{P}(M)$ | $\mathcal{O}(\lvert\mathcal{L}\rvert)$ |
| $(D2)$ | $p(X_i \mid M)$ | $\mathcal{O}(n\kappa\lvert\mathcal{L}\rvert)$ |
| $(D3)$ | $\mathbb{P}(y \in Y)$ | $\mathcal{O}(\lvert\mathcal{L}\rvert)$ |
| $(D4)$ | $\mathbb{P}(M \mid y \in Y)$ | $\mathcal{O}(\lvert\mathcal{L}\rvert^2)$ |
| $(D5)$ | $p(X_i \mid y \in Y)$ | $\mathcal{O}(n\kappa\lvert\mathcal{L}\rvert)$ |
| $(D6)$ | $\mathbb{P}(y' \in Y \mid y \in Y)$ | $\mathcal{O}(\lvert\mathcal{L}\rvert^2)$ |

TABLE I
UNIVARIATE DISTRIBUTIONS IN A CASCADE OF NBCS

III-E). Finally, the overall space requirements for the whole cascade is $\mathcal{O}((n\kappa+|\mathcal{L}|)|\mathcal{L}|)$, $\kappa$ being the number of parameters describing a distribution $p(X_i \mid \cdot)$.

### D. The NaiBX Algorithm

*NaiBX* is an online algorithm that combines a training function `add_example` and a prediction function `predict_subset`. The `add_example` procedure computes the statistics describing the 6 probability distributions required by *NaiBX* for future predictions (presented in Table I). In case of continuous features, we estimate the parameters of the distributions $X_i \mid M$ and $X_i \mid Y$. In our numerical experiments, as we assume normality, we store the mean and variance of each distribution, yielding $\kappa = 2$. In case of *bag-of-words* features [9, Chapter 6], we learn just a probability parameter, thus $\kappa = 1$ (see $(D2)$ and $(D5)$ in Table I).

Algorithm 1 presents the incremental learning process of the `add_example` function, where $update\_parameters$ is a generic updating step of the parameters of distributions in Table I. For $(D2)$ and $(D5)$ we update the mean and variance, for $(D1), (D3), (D4)$ and $(D6)$ we update their univariate probability estimates. Algorithm 2 presents the operations performed when a new sample $\mathbf{x}$ requires the prediction of the associated subset of labels. The `predict_subset` function receives a new observation $\mathbf{x}_{new}$ as an input and predicts a vector of labels $\mathbf{y}_{pred}$ in a two-step process. In the first one, it estimates the size of the target vector via the `predict_size` function. In the second step it proceeds by estimating the elements of the vector through the cascade of predictors. At each iteration the function `predict_label` is called and fed as an input the size $\hat{m}$ and the labels estimated so far. Note that *NaiBX* was thought as the natural extension of Naive Bayes Classifiers to the multi-label case. If one trains *NaiBX* on a data set with targets $\mathbf{y}_{obs}$ of size $m = 1$ with values from a target set of size $|\mathcal{L}| = 2$, that is $f_{classifier} : \mathcal{X} \longrightarrow \mathcal{L} = \{c_1, c_2\}$, then *NaiBX* will act as a traditional binary Naive Bayes Classifier. Furthermore, allowing $|\mathcal{L}| > 2$ will return a multi-class classifier.

**Algorithm 1:** *NaiBX*—Generic Learning Step

```
add_example(x, y)
m = length(y)
update_parameters(ℙ(M))
for x_i in x do
    update_parameters(p(X_i | M))
for label y in y do
    update_parameters(ℙ(y))
    update_parameters(ℙ(M | y))
    for x_i in x do
        update_parameters(p(X_i | y))
    for each label y' ∈ y, y' ≠ y do
        update_parameters(ℙ(y | y'))
    y ← y \ y
```

**Algorithm 2:** *NaiBX*, Prediction Step

```
predict_subset(x_new):
    m̂ ← pred_size(x_new)
    y_pred ← ∅
    while length(y_pred) ≤ m̂ do
        y_pred ← y_pred ∪ pred_label(x_new, m̂, y_pred)
    return y_pred

pred_size(x_new):
    m̂ ← argmax_{d∈{0,1,...,L}} ℙ(m_d) × ∏_{i=1}^{n} ℙ(X_i|m_d)
    return m̂

pred_label(x_new, m̂, {y_1, y_2, ..., y_i}):
    y_{i+1} ← argmax_{y_{i+1}∈{ℒ}} ℙ(y_{i+1}) × ℙ(m̂|y_{i+1})
                × ∏_{i=1}^{n} ℙ(x_i|y_{i+1})
                × ∏_{j=1}^{i} ℙ(y_j|y_{i+1})
    return {y_1, y_2, ..., y_i} ∪ {y_{i+1}}
```

$\mathbf{x_{new}}$ is the features vector, $n$ is the number of features.

### E. Complexity Analysis

Storing the cascade of predictors during the training phase boils down to storing the parameters of the six probability distributions presented in Table I. The space complexity of storing these parameters are recalled in the above tables. The space requirement for the whole cascade of classifiers is in $\mathcal{O}(|\mathcal{L}|(\kappa n + |\mathcal{L}|))$. It is also relevant to note that if one specializes the previous approach to the prediction of fixed-length subsets of size $m$, then the analysis above still holds with the simplification that there is no need for predictor number zero. The time complexity of the training and prediction phases unfolds straightforwardly from the presentation in Algorithms 1 and 2. These remarks are summarized in Table II.

| | Training | Prediction |
|---|---|---|
| Time | $\mathcal{O}(\lvert\mathcal{L}\rvert(\kappa n + \lvert\mathcal{L}\rvert))$ | $\mathcal{O}(\lvert\mathcal{L}\rvert^2(n + \lvert\mathcal{L}\rvert))$ |
| Space | $\mathcal{O}(\lvert\mathcal{L}\rvert(\lvert\mathcal{L}\rvert + n))$ | $\mathcal{O}(\lvert\mathcal{L}\rvert)$ |

TABLE II
*NaiBX*— TIME AND SPACE COMPLEXITY

| Data | $N$ | $dim(\mathcal{X})$ | $Labels$ | $LCard$ | $LDens$ |
|---|---|---|---|---|---|
| (1) *Continuous Features* | | | | | |
| *CAL500* | 502 | 68 | 174 | 26.043 | 15.0 % |
| *Emotions* | 593 | 72 | 6 | 1.869 | 31.1 % |
| *Mediamill* | 43902 | 120 | 101 | 4.376 | 4.3 % |
| *NUS-WIDE* | 269648 | 128 | 81 | 1.873 | 2.3 % |
| *Scene* | 2407 | 294 | 6 | 1.074 | 17.9 % |
| *Yeast* | 2417 | 103 | 14 | 4.237 | 30.2 % |
| (2) *Bag-of-Words Features* | | | | | |
| Bibtex | 7395 | 1836 | 159 | 2.402 | 1.5 % |
| Enron | 1702 | 1001 | 53 | 3.378 | 6.4 % |
| LLog | 1460 | 1004 | 75 | 1.180 | 1.6 % |
| Slashdot | 3782 | 1079 | 22 | 1.181 | 5.4 % |

TABLE III
EXPERIMENTS DATA SETS

## IV. EXPERIMENTAL RESULTS

The computational experiments were carried out on data from a set of standard MLC data sets commonly adopted in the literature [15][12]. In Table III are the data used in the experiments[1], including sets of continuous features and sets of binary *bag-of-words* encoding.

### A. Evaluation Metrics

We adopt a fairly standard pool of metrics [13] to get a sense of the performance of the algorithm. Let us denote by $\mathbf{y}^i$ and $\hat{\mathbf{y}}^i$ respectively the observed and predicted target vectors for the $i$-th entry in a collection of data. The *label cardinality*, given by $LCard = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{y}^i\right|$, allows to quantify the *multi-labelness* of data, yielding the average number of labels per target across the dataset. The *label density* $LDens = \frac{LCard}{|\mathcal{L}|} \times 100 = \frac{1}{N|\mathcal{L}|}\sum_{i=1}^{N}\left|\mathbf{y}^i\right| \times 100$, expresses what proportion of the available labels are, on average, associated to a data point $\mathbf{x}$.

Let $\mathbb{I}(\mathbf{y}^i = \hat{\mathbf{y}}^i)$ take value one if the two vectors are exactly equivalent and zero otherwise. Then the *zero-one* loss metric $L_{01}$ and complementary *zero-one* score $Z_S$ are given by $L_{01} = 1 - Z_S = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\mathbf{y}^i = \hat{\mathbf{y}}^i)$ respectively. For large target vectors, $Z_S$ becomes less meaningful as even a single mistake will invalidate an otherwise good prediction. More forgiving is the Hamming Loss (with its complement, the Hamming score), measuring the average number of operations it would take to turn the predicted vector $\mathbf{y}$ into the correct one.

$$H_L = 1 - H_S = 1 - \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{L}\sum_{k=1}^{L}\mathbb{I}\left(y_k^i = \widehat{y}_k^i\right)\right).$$

We also report the Accuracy, Precision and Recall [13] of our predictions, given respectively by

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\mathbf{y}^i \cap \widehat{\mathbf{y}}^i\right|}{\left|\mathbf{y}^i \cup \widehat{\mathbf{y}}^i\right|}, Pre = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\mathbf{y}^i \cap \widehat{\mathbf{y}}^i\right|}{\left|\widehat{\mathbf{y}}^i\right|},$$

$$Rec = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\mathbf{y}^i \cap \widehat{\mathbf{y}}^i\right|}{|\mathbf{y}^i|}.$$

[1]Due to lack of space, only the most relevant data sets are presented here.

| Data | ALGO | $\Delta Card$† | $H_s$ | $Z_s$ | $Acc$ | $Pre$ | $Rec$ | $Time\ [s]$ $Train$ | $Time\ [s]$ $Pred$ |
|---|---|---|---|---|---|---|---|---|---|
| (1) *Continuous Features* | | | | | | | | | |
| CAL500 | *ECC* | -12.472 | 0.855 | 0.000 | 0.228 | 0.531 | 0.277 | 29.265 | 3.427 |
| | *RAkEL* | -16.024 | **0.863** | 0.000 | 0.209 | **0.610** | 0.235 | 24.480 | 0.142 |
| | *NaiBX* | **-0.325** | 0.817 | **0.002** | **0.249** | 0.379 | **0.372** | 0.031 | 6.235 |
| Emotions | *ECC* | **0.066** | **0.806** | **0.320** | 0.578 | **0.682** | 0.706 | 0.695 | 0.006 |
| | *RAkEL* | 0.377 | 0.795 | 0.317 | **0.592** | 0.642 | **0.771** | 0.759 | 0.003 |
| | *NaiBX* | 0.079 | 0.771 | 0.275 | 0.528 | 0.639 | 0.648 | 0.000 | 0.017 |
| Mediamill | *ECC* | | | | | | | >12 hrs | |
| (CV 66%) | *RAkEL* ‡ | -4.121 | 0.957 | 0.055 | 0.092 | 0.658 | 0.043 | 17486.1 | 3.4 |
| | *NaiBX* | 3.627 | 0.909 | 0.014 | 0.134 | 0.174 | 0.339 | 0.1 | 44.1 |
| NUS-WIDE | *ECC* | | | | | | | >12 hrs | |
| (CV 66%) | *RAkEL* | | | | | | | >12 hrs | |
| | *NaiBX* | -0.236 | 0.968 | 0.186 | 0.164 | 0.223 | 0.201 | 0.2 | 67.3 |
| Scene | *ECC* | **-0.061** | **0.906** | **0.642** | **0.694** | **0.752** | 0.710 | 20.154 | 0.078 |
| | *RAkEL* | 0.295 | 0.885 | 0.538 | 0.684 | 0.640 | **0.816** | 23.808 | 0.028 |
| | *NaiBX* | 0.334 | 0.866 | 0.453 | 0.623 | 0.631 | 0.784 | 0.050 | 0.277 |
| Yeast | *ECC* | -0.288 | **0.799** | **0.203** | 0.536 | **0.680** | 0.633 | 30.826 | 0.138 |
| | *RAkEL* | **0.007** | 0.796 | 0.180 | **0.541** | 0.663 | 0.664 | 12.848 | 0.020 |
| | *NaiBX* | 0.314 | 0.705 | 0.115 | 0.405 | 0.541 | 0.555 | 0.041 | 0.629 |
| (2) *Bag-of-Words Features* | | | | | | | | | |
| Bibtex | *ECC* | **0.063** | 0.982 | **0.148** | **0.348** | 0.426 | **0.451** | 1328.649 | 304.649 |
| (CV 66%) | *RAkEL* | -1.28 | **0.984** | 0.022 | 0.050 | 0.369 | 0.043 | 76.862 | 10.585 |
| | *NaiBX* | -1.122 | **0.984** | 0.147 | 0.278 | **0.461** | 0.301 | 0.244 | 53.126 |
| Enron | *ECC* | -0.42 | **0.947** | **0.143** | **0.448** | **0.600** | **0.525** | 156.391 | 3.401 |
| | *RAkEL* | -0.326 | 0.938 | 0.068 | 0.354 | 0.512 | 0.463 | 234.868 | 1.183 |
| | *NaiBX* | **-0.109** | 0.923 | 0.016 | 0.267 | 0.397 | 0.431 | 0.897 | 18.638 |
| LLog | *ECC* ‡ | -0.824 | **0.984** | 0.234 | 0.265 | **0.473** | 0.143 | 135.000 | 5.893 |
| | *RAkEL* | -0.501 | 0.981 | 0.221 | 0.268 | 0.293 | 0.168 | 140.102 | 1.390 |
| | *NaiBX* | **-0.321** | **0.984** | **0.420** | **0.341** | 0.422 | **0.341** | 0.565 | 7.382 |
| SlashDot | *ECC* | **-0.240** | 0.957 | 0.411 | 0.489 | **0.629** | 0.502 | 116.427 | 2.351 |
| | *RAkEL* | -0.476 | 0.946 | 0.191 | 0.257 | 0.490 | 0.293 | 30.051 | 0.059 |
| | *NaiBX* | -0.276 | **0.961** | **0.424** | **0.497** | 0.535 | **0.533** | 0.290 | 4.445 |

† $\Delta Card = \widehat{LCard} - LCard$, the smaller (in absolute value) the better.
‡ most of the predictions were empty sets, losses yield no valuable meaning.

TABLE IV
METRICS—EXPERIMENTAL RESULTS

None of the aforementioned metrics can be considered flawless. In general, measuring the performance in MLC can be a problem in itself, as some models can perform better than others given a specific metric.

### B. Methods

Among the options available, ECC seems to be one of the most efficient variations on BR, while *RAkEL* is among the most interesting variations on LP. For both cases, we opted for a Support Vector Machine [6] as base classifier as reported in the literature.

We are interested in methods granting running times in the order of seconds or minutes. Nonetheless, for the sake of comparing predictive performances, we allowed running times (including handling data, training and testing) of up to 12 hours with 4 GB of memory reserved to the task. When some method failed to deliver a result, its corresponding line was left blank. For *NaiBX* we ran our implementation while for *RAkEL* and *ECC* we used MEKA [12] and adopted the default parameters.

A 10-fold cross-validation [6] was applied to estimate losses as $\bar{\ell} = \frac{1}{10}\sum_{k=1}^{10}\ell_k$, where $l_k$ is the loss measure in the $k$-th test fold. For bigger data sets (indicated in the tables by "CV 66%") we split the data into training and testing partitions, reserving 66% of the data to training. Computing times (in seconds), $Train$ and $Pred$, are included to assess the impact of the algorithms' complexity on performances.

| Data | ALGO | $\Delta Card\dagger$ | $H_s$ | $Z_s$ | $Acc$ | $Pre$ | $Rec$ |
|---|---|---|---|---|---|---|---|
| | | (1) *Continuous Features* | | | | | |
| CAL500 | *Best* | -0.325 | 0.863 | 0.002 | 0.249 | 0.610 | 0.372 |
| | $NBX_{TrueM}$ | - | 0.833 | 0.012 | 0.299 | 0.423 | 0.423 |
| Emotions | *Best* | 0.066 | 0.806 | 0.320 | 0.592 | 0.682 | 0.771 |
| | $NBX_{TrueM}$ | - | 0.814 | 0.526 | 0.623 | 0.669 | 0.669 |
| Mediamill | *Best* | -0.236 | 0.968 | 0.186 | 0.164 | 0.223 | 0.201 |
| | $NBX_{TrueM}$ | - | 0.970 | 0.356 | 0.223 | 0.263 | 0.264 |
| Scene | *Best* | -0.061 | 0.906 | 0.642 | 0.694 | 0.752 | 0.816 |
| | $NBX_{TrueM}$ | - | 0.903 | 0.712 | 0.721 | 0.725 | 0.725 |
| Yeast | *Best* | 0.007 | 0.799 | 0.203 | 0.541 | 0.680 | 0.633 |
| | $NBX_{TrueM}$ | - | 0.757 | 0.218 | 0.481 | 0.579 | 0.579 |
| | | (2) *Bag-of-Words Features* | | | | | |
| Bibtex | *Best* | 0.063 | 0.984 | 0.148 | 0.348 | 0.461 | 0.451 |
| | $NBX_{TrueM}$ | - | 0.983 | 0.212 | 0.346 | 0.406 | 0.406 |
| Enron | *Best* | -0.109 | 0.947 | 0.143 | 0.448 | 0.600 | 0.525 |
| | $NBX_{TrueM}$ | | 0.928 | 0.134 | 0.357 | 0.460 | 0.460 |
| LLog | *Best* | -0.321 | 0.984 | 0.420 | 0.341 | 0.422 | 0.341 |
| | $NBX_{TrueM}$ | - | 0.983 | 0.479 | 0.375 | 0.392 | 0.392 |
| SlashDot | *Best* | -0.240 | 0.961 | 0.424 | 0.497 | 0.629 | 0.533 |
| | $NBX_{TrueM}$ | - | 0.970 | 0.680 | 0.702 | 0.713 | 0.713 |

TABLE V
METRICS—$NBX_{TrueM}$ VS TO THE BEST ALTERNATIVE.

### C. Results

The good overall performance of *NaiBX* is very close on average to its best competitor for $H_s$, $Z_s$ and $Acc$, when it is not the best itself. If time is not a determinant factor, then highly engineered algorithms can be employed, otherwise a compromise has to be made. If a trade-off between rapidity of learning/prediction and quality of estimations has to be taken into account, *NaiBX*, with its simplicity and agile structure, is an interesting option worth considering. In particular, *NaiBX* will naturally scale up to large label sets, where methods derived from BR that need training a model for each label might not be exploitable.

### D. Prediction of the Target Size

The prediction of the target size $m$ is the peculiar feature of *NaiBX* and we think that it deserves some attention.

We ran an experiment supposing the size of the target was known and reported the results as $NBX_{TrueM}$, skipping the size prediction phase. As reported in Table V, on most of the instances $NBX_{TrueM}$ is as good as or better than the best performing algorithm on all metrics, prompting great interest for further work on the specific topic of size estimation. Estimating $m$ with more refined methods at the cost of increased computational time is an option worth exploring.

## V. CONCLUSION

The proposed algorithm showed significant advantages in terms of computation costs and proved to be competitive in terms of predictive performance, thus offering a viable alternative for tasks requiring a more agile computational footprint. Our approach allows to see the problem from a different perspective than the current literature, notably thanks to the prediction of the target size ($m$) independently from the prediction of the labels. In future research we will further address this aspect, not excluding the possibility of mixing different prediction paradigms for the two tasks.

Overall, we introduced *NaiBX*, a computationally light and efficient multi-label classification method, that proved to be both scalable to large and complex data sets and competitive with state-of-the-art algorithms in terms of predictive performance. This opens up new perspectives for its application on large scale, real-world data.

## ACKNOWLEDGEMENTS

This research benefited from the support of the "FMJH Program Gaspard Monge in optimization and operation research", and from the support to this program from EDF.

## REFERENCES

[1] Breiman, L.: Bagging predictors. In: Machine Learning, pp. 123–140 (1996)
[2] Dembczyński, K., Kotłowski, W., Waegeman, W., Busa-Fekete, R., Hüllermeier, E.: Consistency of probabilistic classifier trees. In: ECML—PKDD 2016, pp. 511–526. Springer (2016)
[3] Dembczyński, K., Waegeman, W., Hüllermeier, E.: An analysis of chaining in multi-label classification. In: Proceedings of the 20th ECAI, pp. 294–299. IOS Press (2012)
[4] Dembczynski, K.J., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML 2010, pp. 279–286 (2010)
[5] Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in neural information processing systems, pp. 681–687 (2001)
[6] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc. (2001)
[7] Loza Mencía, E., Fürnkranz, J.: Efficient pairwise multilabel classification for large-scale problems in the legal domain. Machine Learning and Knowledge Discovery in Databases pp. 50–65 (2008)
[8] Madjarov, G., Kocev, D., Gjorgjevikj, D., DÅ¿eroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition **45**(9), 3084 – 3104 (2012). Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011)
[9] Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
[10] Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: ECML—PKDD 2009, pp. 254–269. Springer (2009)
[11] Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333 (2011)
[12] Read, J., Reutemann, P., Pfahringer, B., Holmes, G.: MEKA: A multi-label/multi-target extension to Weka. Journal of Machine Learning Research **17**(21), 1–5 (2016). URL http://meka.sourceforge.net
[13] Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. Int J Data Warehousing and Mining **3**(3), 1–13 (2007)
[14] Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multi-label classification. IEEE Transactions on Knowledge and Data Engineering **23**(7), 1079–1089 (2011)
[15] Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: Mulan: A java library for multi-label learning. JMLR **12**, 2411–2414 (2011). URL http://mulan.sourceforge.net/data sets-mlc.html
[16] Zhang, H.: The optimality of naive bayes. In: Proc. of FLAIRS (2004)
[17] Zhang, H.: Exploring conditions of the optimality of naive bayes. International Journal of Pattern Recognition and Artificial Intelligence **19**(2), 183–198 (2005)
[18] Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering **18**(10), 1338–1351 (2006)
[19] Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7), 2038 – 2048 (2007)
[20] Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1819–1837 (2014)