Concept Drift Detection Using *Online* Histogram-Based Bayesian Classifiers

César A. Astudillo[1], Javier I. González[1], B. John Oommen[2(✉)], and Anis Yazidi[3]

[1] Department of Computer Science, Universidad de Talca, Km. 1 Camino a Los Niches, Curicó, Chile
castudillo@utalca.cl
[2] School of Computer Science, Carleton University, Ottawa, Canada
oommen@scs.carleton.ca
[3] Department of Computer Science, Oslo and Akershus University College of Applied Sciences, Oslo, Norway
Anis.Yazidi@hioa.no

Abstract. In this paper, we present a novel algorithm that performs online *histogram*-based classification, i.e., specifically designed for the case when the data is dynamic and its distribution is non-stationary. Our method, called the Online *Histogram*-based Naïve Bayes Classifier (OHNBC) involves a statistical classifier based on the well-established Bayesian theory, but which makes some assumptions with respect to the independence of the attributes. Moreover, this classifier generates a prediction model using uni-dimensional *histograms*, whose segments or buckets are fixed in terms of their cardinalities but dynamic in terms of their widths. Additionally, our algorithm invokes the principles of information theory to automatically identify changes in the performance of the classifier, and consequently, forces the reconstruction of the classification model in run-time as and when it is needed. These properties have been confirmed experimentally over numerous data sets (In the interest of space and brevity, we present here only a subset of the available results. More detailed results are found in [2].) from different domains. As far as we know, our histogram-based Naïve Bayes classification paradigm for time-varying datasets is both novel and of a pioneering sort.

Keywords: Online Naïve Bayes Classifier · Online learning · Concept drift · Dynamic Histograms

C.A. Astudillo—This work was partially supported by the FONDECYT Grant No. 11121350, Chile.

B.J. Oommen—*Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway. The work of this author was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

B.H. Kang and Q. Bai (Eds.): AI 2016, LNAI 9992, pp. 175–182, 2016.
DOI: 10.1007/978-3-319-50127-7_14

1 Introduction

In the fields of machine learning and statistical learning, supervised classification is a well-known problem that consists of identifying the category (or class) to which a new observation belongs, based on a sample data set that contains instances for which their respective categories are known. This task can be achieved using a range of methods, including linear classifiers, Support Vector Machines (SVMs), decision trees, neural networks, etc. Most of these families of supervised classifiers operate in an *offline* manner, i.e., the data set (training data) used for building the learning model is known in advance. Such algorithms operate in three phases: First, a learning model is constructed using the training data in which the instances are all labeled. In the second phase, the model that has been built is used to classify data instances from a test set, and a performance measure is obtained. Finally, in the third phase, the model is deployed and used to predict the category of an unlabeled data instance data [1].

It is possible to find a wide range of applications where the data arrives in the form of a stream, consisting of a (theoretically, infinite) sequence of instances that may become available at a very rapid rate. These applications include telecommunications data management, financial applications, sensory analysis, web history logs, etc. The analysis of dynamic and real-time data streams poses several challenges when compared to processing the data in an offline manner. In fact, *offline* classifiers assume that the training records can be examined and accessed several times. This is consistent with the three previously-mentioned phases of any supervised classification algorithm.

Unfortunately, in many real life situations, the rate of arrival of the data instances is so rapid that it is infeasible to store them for post-processing, forcing the algorithm to achieve the processing of a single instance at any given time instant. The work reported in [5] describes another difficulty with classifying data streams with respect to the dynamic nature of the data in the following terms: "Even if all the available examples can be handled by the system, the patterns discovered by an algorithm in the data from the past, may be hardly valid and useful for the new data obtained hours or even minutes later." *Online* algorithms [1] deal with such data streams, in which the labeled and unlabeled records are mixed. In this context, the phases of training, testing and deployment are interleaved. This is precisely the domain of this paper, and the results we contribute involve *histogram*-based Bayesian classifiers.

As we know, pattern classification is the discipline of building machines for classifying patterns based on prior knowledge or on statistical information extracted from the patterns [4]. However, as opposed to what we shall call traditional classification, we are interested in data that is generated in real time. Examples of sources of such streamed data are sensor networks on Mars, underwater sensors in the deep ocean, atmospheric measurements, etc. These potentially infinite sequences of information are usually known as "data streams" [1]. The data stream serves as an appropriate model when a large amount of data arrives for processing and where it is impractical to store it all, implying that a model that attempts to learn from it must achieve the task by processing each

pattern at a time. Devising classifiers that work with data streams poses new challenges when compared to the standard classification algorithms, since the latter algorithms are able to examine the patterns repeatedly.

Algorithms for data streams are more complicated to design because they must be able to extract all the information needed with just a single examination of the patterns. According to the author of [1], algorithms capable of learning from streaming data must process the patterns in amortized $O(1)$ time. From this we can say that online learners are induction models that are trained – one instance at a time. The goal is to predict the classes of novel patterns as accurately as possible. The key feature that defines online learning is that shortly after the prediction is made, the actual class of the instance is discovered. This information can then be used to refine the algorithm's prediction hypothesis [6].

More formally, the classification in an online algorithm proceeds as follows. The sequence of operations can be decomposed into three phases. First of all, the algorithm receives an instance. Secondly, the algorithm predicts the class of the instance. Thirdly, the algorithm receives the true class of the instance [6]. The third phase is the most crucial one, since the algorithm can use this information to update the classification model.

Various algorithms that possess online learning properties have been reported in the literature. An example of one such algorithm, is the Online Random Forest [7]. In [7], the authors propose an online learning scheme based on the properties of Random Forests previously described in [3]. The main idea is to build a sequence of decision trees which learn from the data in an independent manner, and a final decision is made based on a function that depends on the output of all these trees. The Online Random Forest starts with a single node as the root of the tree, and systematically adds new nodes depending on a predefined criteria. When a new decision node is added to the tree, a series of functions of the form $g(x) < \theta$ divide the data. Both the functions and the constant θ are defined randomly. These functions produce a division of the feature space, and according to a performance measure, the best division is selected.

Due to space limitations, a more detailed survey of the field and the details of the background material is omitted. It is found in [2]. However, we shall concentrate on our contribution, namely, the formulation of our histogram-based OHNBC algorithm.

2 The OHNBC Algorithm

Based on the phenomena described above, we now detail the algorithmic characteristics and functionality of our Online *Histogram*-based Naïve Bayes Classifier (OHNBC). We first define the general structure of OHNBC. We then specify some of the algorithm's functionality in greater detail, and then describe how the components fit together. The general structure of the OHNBC algorithm is formalized in Algorithm 1, and also explained in [2] in a textual manner. It is omitted here in the interest of brevity and due to space limitations.

The two main interleaving phases of the algorithm's classification and training are obvious.

Algorithm 1. `OHNBC` ($\mathcal{X}$, λ, τ, θ)

Input:
i) $\mathcal{X}$: Stream of instances for classification.
ii) λ: Minimum number of training instances needed for classification.
iii) τ: Size of the reference window.
iv) θ: Threshold for the threshold.

Output:
i) Confusion matrix with the classification results.
ii) The histograms of the classes for the OHNBC.

Method:

while an instance $\mathbf{x} \in \mathcal{X}$ **do**
  if number of trained instances is less than λ **then**
    $\omega \leftarrow$Original class of $\mathbf{x}$
    for all attribute $d \in \mathbf{x}$ **do**
      Get the histogram for the attribute d and the class ω.
      Add the value of the attribute d to the histogram.
    end for
    Update *prior* probabilities $\mathcal{P}$ associated with the class ω.
  else
    $P \leftarrow$ *prior* probabilities.
    for all attribute $d \in \mathbf{x}$ **do**
      for all ω_i class **do**
        $H(d, \omega_i) \leftarrow$ uni-dimensional histogram for the attribute d ω_i.
        if standard deviation of $(H(d, \omega_i))$ equals 0 **then**
          Adjust all the $H(d, \omega_i)$.
        end if
        Compute the probability density function for $(H(d, \omega_i))$
        Multiply the the density functions over d with their *prior* values
      end for
    end for
    if original class of $\mathbf{x}$ equals the OHNBC's predicted class **then**
      Update the size of the windows.
      Update the classification model with the current correct prediction.
    end if
    Calculate the difference in entropy of the windows.
    if the If this difference is larger than θ **then**
      Reset the values of the windows.
      Discard the current classification model.
    end if
  end if
end while

End Algorithm

3 Experimental Design and Results

The solution that we propose here was tested for various artificial data sets[1] and three synthetic data which emulate conditions under which concept drift occurs [1]. These files were generated using the data generation tool DatGen proposed by Melli[2].

There are many situations that deal with the various scenarios involving concept drift [1]. The following describes each of the proposed scenarios:

- **Scenario 0:** This corresponds to that case when there is no concept drift. Here, instances are taken early in the stream, using which the learning model for the classifier is built, and it is then used to classify the instances.
- **Scenario 1:** In this scenario, we observe concept drift. Here, the data stream arrives in the form of data blocks. The data within each block possesses the same probability distribution. However, contiguous blocks have different data distributions. The cardinality of the blocks within this scenario is large enough to allow for the proper training of the classification model.
- **Scenario 2 and 3:** These scenarios inherit the essential properties of Scenario 1. The fundamental difference is that the sizes of the blocks are not large enough to allow the proper construction of the learning models. This corresponds to a complex environment where the instances available are not sufficient to correctly ensure the update of the classification model, especially at the boundaries between the blocks. In Scenario 2, the blocks have fixed sizes, while in Scenario 3, which is the more realistic one, the blocks themselves have random sizes and frequencies.

For the case of artificially generated data sets, we emulated scenarios 1, 2 and 3, which contained concept drift within their characteristics.

The first synthetic data set follows the characteristics of Scenario 1 and contains a total of three million instances, divided into three blocks of a million instances each, respectively. We consider three different categories ("C1", "C2" and "C3") in a five dimensional space (denoted by "A", "B", "C", "D" and "E").

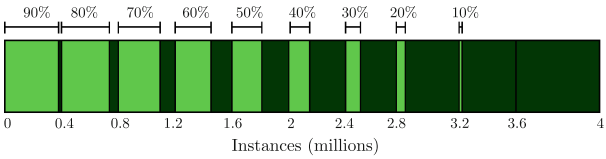
Figure 1a is a graphical representation of the distribution of blocks for the synthetic data set '1'. The various distributions are represented using gray scales. The figure shows that for the first block of one million instances the distribution corresponds to a unique distribution, followed by a block of one million instances corresponding to the second distribution, and eventually the last block of one million instances correspond to the third distribution.

The second data set of 4 million instances, emulates the characteristics of Scenario 2. Figure 1b is a graphical representation of the data set, specifying the data distribution associated with a block at any given time. The set is divided into ten blocks of 400,000 instances each. The first block is composed of 90%

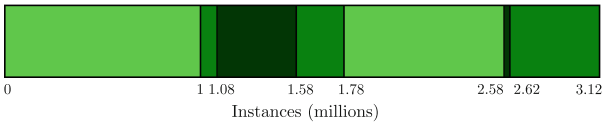
[1] As mentioned earlier, in the interest of space and brevity, we present here only a subset of the available results. More detailed results are found in [2].

[2] The data generation tool DatGen is publicly available at the following URL: http://www.datasetgenerator.com.

(a) *Synthetic data set 1.*



(b) *Synthetic data set 2.*



(c) *Synthetic data set 3.*

Fig. 1. Graphical representation of the synthetic data sets representing the three scenarios.

of instances belonging to the first distribution, while 10% corresponds to the second distribution. The next block has a ratio of 80% of instances of the first distribution and 20% of instances from the second distribution. This change in proportions is successively applied to the remaining blocks up to the tenth block, which contains 100% of instances of the second distribution.

Finally, the third set of data follows the characteristics of Scenario 3 and contains a total of 3 million and twelve hundred instances divided into several blocks of different sizes and frequencies. Figure 1c illustrates the distribution of the blocks for this synthetic data set. Analogous to the above sets, gray scales are used to differentiate the distributions.

3.1 Parameter Optimization

The OHNBC classifier requires three parameters: λ, which is the minimum number of instances are required to build the learning model, τ, which is the size of the reference window, and θ, which is the entropy threshold for identifying that a concept drift has occurred. The values chosen for each of these parameters are presented below:

- $\lambda \in \{2000, 5000, 10000, 20000, 50000, 100000\}$
- $\tau \in \{500, 1000, 2000, 5000, 10000\}$
- $\theta \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
- $\lambda \in \{500, 1000, 2000, 5000\}$
- $\tau \in \{50, 100, 200, 500, 1000\}$

Table 1. Final parameters for the OHNBC for each of the synthetic data sets.

Data sets	Parameters		
	OHNBC		
	λ	τ	θ
Synthetic-data-set1	2000	1000	0.10
Synthetic-data-set2	2000	500	0.10
Synthetic-data-set3	2000	2000	0.10

Table 1 lists the values found by applying a *Grid Search* to determine the best suitable parameters for the real and synthetic data sets. Each row indicates a specific data set, while in each column we list the values of the optimum parameters for each data set, respectively.

3.2 Results

This section presents an overview of the performance of the proposed algorithm, when compared to the Naïve Bayes algorithm (NB). Subsequently, we also performed a detailed analysis of the OHNBC algorithm relative to the data's concept drift. The results presented here are those obtained by averaging over ten runs of the classifier.

Table 2 presents the results for synthetic data sets. It summarizes the number of instances correctly classified ("Certainty") and the proportion of instances used for training the model ("Training") for the various synthetic data sets (rows). As one can see, in all these three scenarios, the accuracy of the OHNBC far surpasses that of the NB. We attribute the improved performance to the ability of the OHNBC to detect the changes in the corresponding distributions, which the NB clearly lacks. Given the artificial construction of these sets, we already know that there are concept changes in the stream, and the exact times when they occur in the data stream. Additionally, the way by which these schemes have been devised make them ideal for situations with a large number of instances, as in this case. Indeed, from the column titled "Training", we observe that it was not necessary to use a large percentage of training instances to build the learning models that achieved such high accuracy rates.

Table 2. Results obtained for the runs of the synthetic data sets.

Data sets	Accuracy		Fraction of training data set	
	NB	OHNBC	Total	Ratio
Synthetic-data-set1	0.824	0.991	3,000,000	0.002
Synthetic-data-set2	0.699	0.988	4,000,000	0.009
Synthetic-data-set3	0.746	0.984	3,120,000	0.005

Finally, in relation to the amount of concept drifts detected for each of the synthetic data sets, our algorithm is able to identify all the concept changes that were artificially embedded in the stream.

4 Conclusion

This article has tackled the problem of classification in environments where the instances are sequentially arriving in the form of a data stream, and whose statistical distribution potentially varies over time. We have proposed a novel method to identify these changes using Bayesian theory and the principles of Shannon's information theory. Our proposed classification algorithm, called the Online *Histogram*-based Naïve Bayes Classifier (OHNBC) follows the paradigm of online training outlined by three key phases: First, the algorithm receives an instance. Next, the algorithm predicts the class of the instance based on a histogram-based representation. Finally, the algorithm receives the true class of the instance and uses it to update the classification model. The results that we have obtained on synthetic and real data clearly demonstrate its power in classifying data streams characterized by non-stationary distributions.

References

1. Abdulsalam, H., Skillicorn, D., Martin, P.: Classification using streaming random forests. IEEE Trans. Knowl. Data Eng. **23**(1), 22–36 (2011)
2. Astudillo, C.A., Gonzalez, J., Oommen, B.J., Yazidi, A.: Concept drift detection using classifiers that are online, Bayesian and histogram-based. Unabridged Version of this paper. (In Preparation)
3. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
4. García-Laencina, P., Sancho-Gómez, J., Figueiras-Vidal, A.: Pattern classification with missing data: a review. Neural Comput. Appl. **19**, 263–282 (2010). doi:10.1007/s00521-009-0295-6
5. Last, M.: Online classification of nonstationary data streams. Intell. Data Anal. **6**(2), 129–147 (2002)
6. Littlestone, N.: Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. Mach. Learn. **2**(4), 285–318 (1988)
7. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H.: On-line random forests. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1393–1400, October 2009