Generating datasets with drift

Joana Costa[12]
joana.costa@ipleria.pt,joanamc@dei.uc.pt
Catarina Silva[12]
catarina@ipleiria.pt,catarina@dei.uc.pt
Mário Antunes[13]
mario.antunes@ipleiria.pt,mantunes@dcc.fc.up.pt
Bernardete Ribeiro[2]
bribeiro@dei.uc.pt

[1] School of Technology and Management
Polytechnic Institute of Leiria, Portugal
[2] CISUC - Department of Informatics Engineering
University of Coimbra, Portugal
[3] Center for Research in Advanced Computing Systems
INESC-TEC, University of Porto, Portugal

Abstract

Modern challenges in machine learning include non-stationary environments. Due to their dynamic nature, learning in these environments is not an easy task, as models have to deal both with continuous learning process and also with the acquisition of new concepts. Different types of drift can occur, as concepts can appear and disappear with different patterns, namely sudden, reoccurring, incremental or gradual. Besides striving to propose new techniques to learn in the presence of drift, researchers aim to find appropriate benchmarks to non-stationary scenarios.

In this paper we propose DOTS, a drift oriented tool system, whose main goals are to define and generate datasets with drift for text classification problems. DOTS tries to fill an existing gap in machine learning research for text applications, by making possible to generate benchmark datasets with thoroughly controlled drift characteristics. Because of its ability to export in multiple formats, it can be widely used and in conjunction with well-known classifiers and applications, like SVM Light or WEKA. We will also present a Twitter case study to validate the usefulness of DOTS in a real-word problem scenario.

1 Introduction

Non-stationary environments are characterized by incremental data gathering where the underlying data distribution changes over time without an explicit and known pattern. The enormous growth of the computational power in recent years, along with the popularization of social networks and mobile devices, created a deluge of data and demanded new approaches, as all this information needs to be acquired and treated almost in real-time, as it evolves faster than we were used. There are nowadays multiple real-world applications where this is a given, like image or speak recognition, fraud detection or mining in social networks.

Learning in the presence of drift is not an easy task and requires a distinctive approach. One of the major challenges is posed due to the multitude of drift patterns, and the inability to perceive a priori which ones might be present. Zliobaite[1] identified four drift patterns, as can be seen in Fig. 1: sudden; gradual; incremental; and reoccurring.

- **Sudden drift** occurs when in a given moment a concept appears or disappears in an abrupt way. The speed of the drift is therefore high.
- **Gradual drift** occurs when the probability of a given context to be associated with a concept decreases during a certain period of time, but also the probability to be associated with another context increases proportionally.
- **Incremental drift**, sometimes considered a subgroup of gradual drift, can be considered differently because the change between the two concepts is very slow and only perceived when looking to what is occurring during a larger period of time.
- **Reoccurring drift** occurs when previously active concept reappears after a certain period of time. It is noteworthy to refer that the seasonality of the change must be previously unknown, otherwise the core assumption of the uncertainty about the future would be compromised.

Several recent approaches [2-4] try to deal with such challenges, namely by proposing dynamic techniques that try to detect, or deal, with drifts in different scenarios. But research challenges in learning in non-stationary environments are not only in proposing and deploying learning models able to learn in the presence of different drift patterns, but also to find suited benchmarks to test and compare the proposed approaches. To tackle this issue, we propose the Drift Oriented Tool System (DOTS). DOTS is a drift oriented framework able to define and generate text-based datasets with drift, that can then be used to evaluate and validate learning strategies for dynamic environments.

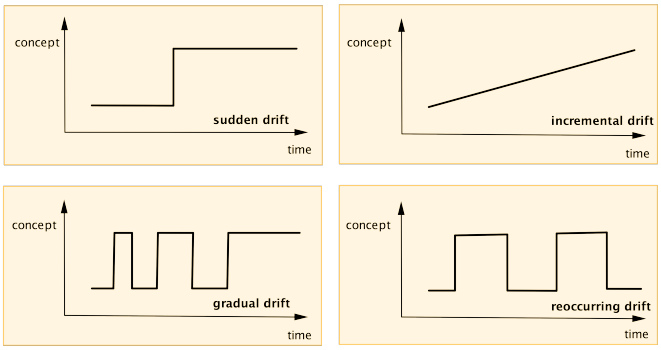

Figure 1: Types of drift.

2 Drift Oriented Tool System

DOTS is a drift oriented framework developed to dynamically create textual datasets with drift. It is a simple and easy to use freeware application with a friendly interface as shown in Fig. 2. It can be download at `http://dotspt.sourceforge.net/`.

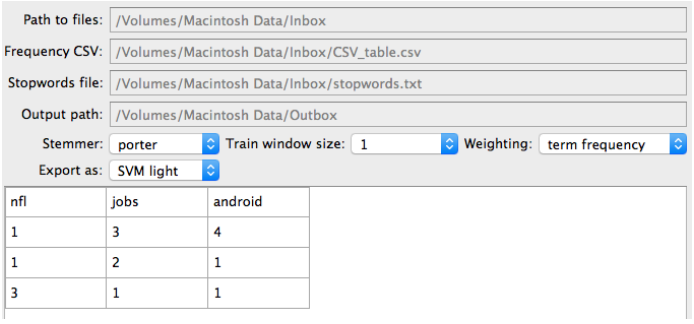

Figure 2: DOTS interface.

The main purpose of DOTS framework is to represent drift patterns in a text-based dataset. Therefore, the input of the DOTS framework is two fold and composed by text documents and a frequency table, as depicted in Fig. 3. Each text document file represents the documents of the same class and the frequency table is used to define the drift patterns of the scenario. A major characteristic of DOTS is the possibility of defining the time window where each document appears, being thus possible to define time drifts. This is done by the frequency table that is a mandatory input of the DOTS framework (see Fig. 3).

The frequency table must be in the CSV format and each row corresponds to a time instance. It is not important if a time instance represents a minute, an hour, or a day, but it is assumed that all of them correspond to the same amount of time. The first row contains the name of the class, and each of the above cells contain the number of documents of that

given class in the subsequent time windows. Fig. 2 depicts a task being added to the framework, in which the first time-window has one, three and four documents of the classes `nfl`, `jobs` and `android` respectively. The DOTS framework represents each document in a vector space

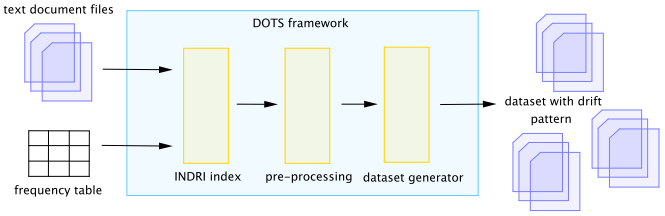

Figure 3: The DOTS framework.

model, also known as *bag of words*, a commonly used text document representation. Two problems arise when using the *bag of words*: high-dimensionality feature space and overfitting. High dimensional space can cause computational problems and overfitting can easily prevent the classifier from generalizing and thus endagered the prediction process. To tackle both problems pre-processing methods were also integrated in the DOTS framework and are the second phase of the DOTS processing. In order to use stopword removal a text file containing stopwords must be defined. These words, considered non-informative words, will not be included in documents representation. Besides stopwords removal, DOTS also allows stemming. This pre-processing method consists in removing case and inflection information of a word, reducing it to the word stem. Stemming does not alter significantly the information included, but it does avoid feature expansion. We have included two important stemming algorithms: the Porter algorithm[5] and Krovetz algorithm[6].

It is also possible to define the weighting scheme used to represent each word of a document, that is the weight of each feature of a document. Two weighting schemes were defined, namely term frequency (*tf*) and term frequency-inverse document frequency, commonly known as *tf-idf*. Considering the defined input, DOTS will create a word index, the INDRI index, provided by INDRI API, from the Lemur Project[1]. By outputting an INDRI index, DOTS provides all the features presented by the INDRI project, a powerful query interface, that provides state-of-the-art text search, field retrieval and text annotation.

It is also possible to define multiple training window sizes and multiple export file formats. The training window size will define in each time-window how many previous time-windows will be considered, as this is important to test learning models with memory capabilities. For instance, to perceive for how long it is relevant to keep previously gathered information and how that can affect the learning model capabilities, which can be seen as the importance of previously seen examples in future classifications. By exporting in multiple file formats, DOTS allows for creating datasets that can be used in different classification frameworks, like `SVM Light` and `Weka`. Three output formats were implemented: Comma-Separated Values (CSV) file format, the Attribute-Relation File Format (ARFF) used in the widely used `WEKA` software, and the `SVM Light` file format.

As it is often relevant to define various testing scenarios, DOTS permits adding tasks using INI files. These are structured files with "`key=value`" pairs, that contain the definition of multiple tasks. By using an INI file as input, users are able to define more than one task at a single time, thus optimizing the time spent on task setup. A complete tutorial can be download at `http://dotspt.sourceforge.net/`.

3 Case study: Twitter stream

To validate the usefulness of DOTS in a real-world problem, we will present a Twitter stream case study. It constitutes a paradigmatic example of a text-based scenario where drift phenomena occur commonly. *Twitter* is a micro-blogging service where users post text-based messages up to 140 characters, also known as *tweets*. *Twitter* is also responsible for the popularization of the concept of *hashtag*. An *hashtag* is a single word started by the symbol "#" that is used to classify the message content and to improve search capabilities. We have used a classification strategy previously introduced in [7], where the Twitter message *hashtag* is used to label the content of the message.

[1] `http://www.lemurproject.org/`

In this particular case, DOTS is used to create datasets able to test multiple learning scenarios. DOTS receives a document set for each class of tweets containing the same hashtag. A CSV table with different drift patterns was also defined, reproducing artificial drifts, like sudden, gradual, incremental, reoccuring and normal. As an example, a sudden drift might be represented by tweets from a given hashtag that in a given temporal moment start to appear with a significant frequency. Each tweet was represented by DOTS as a vector space model and pre-processing methods were applied, like stopword removal and stemming. We have exported, for our convenience, in SVMLight format, as is required to use Support Vector Machines (SVM) as the learning model of this case study. Two different weighting scheme in document representation were tested, term-frequency and *tf-idf*. Table 1 presents the obtained results using both weighting schemes. To evaluate the possible outcomes of the classification, we used the van Rijsbergen F_β measure: $F_\beta = \frac{(\beta^2+1)P \times R}{\beta^2 P + R}$ with $\beta = 1$. The obtained results showed that using term-frequency to represent document features improves the overall classification performance even in the presence of different drift patterns.

Drift	Hashtag	F_1 using tf	F_1 using tf-idf
Sudden	#syrisa	79.37%	75.32%
Gradual	#airasia	57.88%	56.08%
Incremental	#isis	83.49%	81.75%
Reoccurring	#android	60.66%	59.49%
Normal	#sex	74.88%	74.56%

Table 1: Performance measure for the results obtained with Twitter stream.

4 Conclusions and Future Work

In this paper we have presented the Drift Oriented Tool System. DOTS is designed for text-based problems and can produce datasets with multiple drift patterns and in multiple file formats, which is of major importance as it can be used with different classification frameworks. It aims to fulfil the existing gap in machine learning of tools able to reproduce benchmarks in dynamic environments.

We have also presented a case study based in a Twitter scenario, to validate the usefulness of using DOTS in real-world problems. We have generated multiple datasets and have tested different classification strategies, to define the best characteristic of a learning model in this particular scenario. All of these demonstrate the convenience of DOTS to create text-based datasets with drift and thus be used to evaluate and validate learning strategies in dynamic environments.

Regarding future work we will look at including new features in the DOTS framework, namely, the possibility of noise addition to the datasets and new file exporting types.

Acknowledgment

We acknowledge the iCIS project (CENTRO-07-ST24-FEDER - 107002003).

References

[1] Indre Zliobaite. Learning under Concept Drift: an Overview. Tech. Report, Vilnius University, Faculty of Mathematics and Informatic, 2010.

[2] Joana Costa, Catarina Silva, Mário Antunes, and Bernardete Ribeiro. Concept Drift Awareness in Twitter Streams. In *Proc. of the 13th International Conference on Machine Learning and Applications*, pages 294-299, 2014.

[3] D. Mejri, R. Khanchel, and M. Limam. An Ensemble Method for Concept Drift in Nonstationary Environment. In *Journal of Statistical Computation and Simulation, vol. 83, no. 6*, pages 1115-1128, 2013.

[4] G. Ditzler and R. Polikar. Incremental Learning of Concept Drift from Streaming Imbalanced Data. In *IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 10*, pages 2283-2301, 2013.

[5] P. Willett. The Porter Stemming Algorithm: Then and Now. In *Program, vol.40, no. 3*, pages 219-223, 2006.

[6] R. Krovetz. Viewing morphology as an Inference Process. In *Proc. of the 16th Annual International ACM SIGIR conference on Research and development in information retrieval*, pages 191-202, 1993.

[7] Joana Costa, Catarina Silva, Mário Antunes, and Bernardete Ribeiro. Defining Semantic Meta-Hashtags for Twitter Classification. In *Proc. of the 11th International Conference on Adaptive and Natural Computing Algorithms*, pages 226-235, 2013.