Classification of Data Streams Applied to Insect Recognition: Initial Results

Vinícius M. A. de Souza, Diego F. Silva, Gustavo E. A. P. A. Batista
Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
São Carlos, SP, Brazil
Email: {vsouza, diegofsilva, gbatista}@icmc.usp.br

Abstract—Applications such as intelligent sensors should be able to collect information about the environment and make decisions based on input data. An example is a low-cost sensor able to detect and classify species of insects using a simple laser and machine learning techniques. This sensor is an important step towards the development of intelligent traps able to attract and selectively capture insect species of interest such as disease vectors or agricultural pests, without affecting non-harmful species. The data gathered by the sensor constitutes a data stream with non-stationary characteristics, since the insects’ metabolisms are influenced by environmental conditions (such as temperature, humidity and atmospheric pressure), circadian rhythm and age. Algorithms that classify data streams often assume that once a prediction is made, the actual labels are provided to update the classifier. In the case of intelligent sensors, these labels are rarely available. The objective of this paper is to evaluate methods that adapt concept drifts by regularly updating the classification models applied to insect recognition in a data stream. We show in our initial results that the philosophy of inserting and removing examples from the training set are of essential importance. We also show that a simple criterion to insert examples with high classification confidence can significantly improve the accuracy.

Keywords—insect classification; data streams; concept drift

I. INTRODUCTION

Novel and relevant applications have emerged with the development of data mining methods. One example is the class of intelligent sensors capable of collecting information about the environment and making decisions based on the input data. A concrete example is a sensor that uses a laser and machine learning techniques to classify species of insects. This sensor is an important step in the development of intelligent traps. Such traps can attract and selectively capture insect species of interest, releasing all other species back into the environment.

Intelligent traps open innumerable possibilities of applications. For public health, the trap can capture mosquitoes belonging to the genera Aedes and Anopheles, known to be vectors of dengue and malaria, respectively. In agriculture and livestock, the trap can capture insect pests such as Diaphorina citri, a vector of the most severe citrus disease currently known [1]. At the same time, the trap releases other insect species that are not pests or disease vectors, limiting the impact of its presence in the environment. This is an important feature, since most insects have an important role in maintaining the ecological balance. For instance, they can be food sources for other animal species, assist in the breeding of plants and agricultural production, since they perform pollination and seed dispersal, or responsible for the production of useful substances for humans such as honey, wax and silk [2].

The scientific challenges are as interesting as the potential applications of this technology. Intelligent sensors must process large amounts of data. These data form a continuous stream of information. In general, sensors have a limited amount of memory and therefore it is not feasible to store the entire data stream for later processing. Therefore, sensors must process the data stream in real-time, identifying events of interest and discarding data consisting other events and background noise.

In the case of a sensor coupled to an intelligent trap, the identification of events is not enough. There is a need to classify all events in real-time. This classification is needed since the trap has to make a decision of capturing or releasing an insect according to its species.

For most applications in intelligent sensors, we cannot assume that the data is generated by a stationary stochastic process. In the case of the currently sensor, this assumption is due to the existence of variations in environmental conditions which can influence the insects’ metabolisms. For example, climate variations affect the behavior of insects, since their metabolism is influenced by temperature [3], air pressure [4] and humidity [5]. Another example are the circadian rhythms [6]. Various insects, including mosquitoes, have periods of high activity at certain times of the day, typically early morning and late afternoon. Thus, an intelligent sensor must adapt to these variations, an idea present in machine learning literature as concept drift [7].

However, it is a very difficult task to measure the influence of all possible combinations of environmental, climate and temporal variations on the behavior of insects. Therefore, the classifier must be able to adapt itself to combinations of conditions not previously known.

Finally, algorithms that classify data streams often assume that once a prediction is made, the actual labels are provided (even if with some delay) to assist in updating the classifier [8]. In the case of intelligent sensors, these labels are rarely available, and the application must adapt to concept drift without assuming that labels from test cases are known.
The objective of this paper is to evaluate those methods which adapt to concept drifts by regularly updating the classification models. We show that the philosophy of inserting and removing training examples is of essential importance to maintain the performance of the classifier over time. We also show that a simple criterion to insert examples with high classification confidence can significantly improve the stream classifier’s accuracy.

II. LASER INSECT SENSOR

The data used in this paper was obtained from a laser sensor built with low-cost components to remotely capture information about flying insects. Figure 1 shows the general design of the sensor. It consists of a low-powered planar laser source pointed at an array of phototransistors. When a flying insect crosses the laser, its wings partially occlude the light, causing small variations in the light captured by the phototransistors. An electronic circuit board filters and amplifies the signal, and the output is recorded as audio.

Figure 1. The logical design of the sensor. A planar laser light is directed at an array of phototransistors. When an insect flies across the laser, a light variation is registered by the phototransistors as a time series [9].

Figure 1 also shows an example of the data collected by the sensor. That signal was collected from an Aedes aegypti mosquito, a vector of diseases such as dengue and yellow fever. The data consist of, in general, background noise with occasional “events”, resulting in the brief moment that an insect flies across the laser. Note that the signal generated by the passage of the insect has an amplitude that is significantly higher than the amplitude of the background noise. In this way, it is a simple task to identify signal sections in which there is an insect passage. In contrast, the correct classification of each passage according to the insect species that generated the event is a much more elaborate task.

One of the most relevant pieces of information in order to classify events in insect species is the wingbeat frequency. For decades, researchers in entomology have measured and analysed how the wingbeat frequency varies among species [10], [11]. Figure 2 illustrates this variation for seven insect species, including three species of mosquitoes.

III. DATA COLLECTION AND PREPROCESSING

In this section, we briefly describe the procedure used to collect and preprocess the data used to assess the classification strategies presented in this work. We start by describing how we collected the data in the laboratory and later we provide a short description of the feature extraction and a brief description of the resulting dataset.

In this application domain, the use of laboratory data is essential to evaluate and compare classification systems. This is because in order to assess the classification procedures we need a ground-truth, i.e., we need to know the true class labels of each insect passage. In order to achieve that, we have to collect data in a controlled environment in which only one insect species is present in each insectary.

We used five insectaries, each with dozens of specimens of a single specie. All insectaries were placed in the same room and the data were collected at the same time. Since we collected data simultaneously for all species, we are able to simulate a single insectary with multiple species. In brief, we synchronized the recordings and merged the multiple data sources into a single data stream. Figure 3 presents some examples of the insectaries and sensors used to collect the data.

Figure 2. Gaussian curves representing the mean and standard deviation of the wingbeat frequencies of seven species of insects (females only). From left to right, Lucidota atra, Chauliothus marginatus, Oulema melanopus, Drosophila melanogaster, Culex quinquefascitus, Anopheles stephensi, and Aedes aegyptii [9].

Figure 3. Three examples of insectaries adapted to collect data with the sensor.

The data collection was performed during six consecutive days in laboratory conditions in which the temperature varied slightly between 20°C and 22°C (68°F and 71.6°F) and humidity varied between 20% and 35%. We included two species of flies and three species of mosquitoes in the dataset. These species are Drosophila melanogaster (popularly known as fruit flies), Musca domestica (flies) and
the mosquito species *Culex quinquefasciatus* (a vector of lymphatic filariasis), *Culex tarsalis* (a vector of St. Louis Encephalitis and Western Equine Encephalitis) and *Aedes aegypti* (a vector of filariasis, dengue fever, yellow fever, and West Nile virus).

The first step is to design a detector, which has the purpose of detecting data fragments containing events of insects crossing the laser. The general idea of the detector is to apply a sliding window to the data, and calculate the spectrum of the signal inside the window. As most insects have wingbeats which range from 100Hz to 1000Hz, we used the maximum magnitude of the signal spectrum in this range as a confidence detector. Due to lack of space, we redirect the interested reader to [12] for a more detailed description of the detector.

The detector outputs audio fragments of tenths of a second of duration which have at least one insect passage. After the detection stage we end up with 5,325 insect passages, including all five classes, over a six-day period. Table I presents the prior probabilities of each class.

![Table I](https://example.com/table.png)

<table>
<thead>
<tr>
<th>Species of insect</th>
<th>Examples</th>
<th>Distribution (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Musca domestica</em></td>
<td>917</td>
<td>17.22</td>
</tr>
<tr>
<td><em>Culex quinquefasciatus</em></td>
<td>1,285</td>
<td>24.13</td>
</tr>
<tr>
<td><em>Culex tarsalis</em></td>
<td>1,265</td>
<td>23.76</td>
</tr>
<tr>
<td><em>Drosophila melanogaster</em></td>
<td>954</td>
<td>17.91</td>
</tr>
<tr>
<td><em>Aedes aegypti</em></td>
<td>904</td>
<td>16.98</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>5,325</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

Although very short (most recordings have less than two tenths of a second) the recordings are sampled at 16000 Hz, resulting in time series with approximately 3,000 observations. The next step is to extract features from the signals, an approach widely used in the signal processing community [13]. Feature extraction performs a representational change of the original audio data, from a high-dimensional weak-feature domain to a low-dimensional strong-feature domain. Since there are literally hundreds of signal processing features available in the literature, we performed several experiments to investigate which features would provide the best results for our domain. We experimented with features used in similar applications such as spoken digit recognition [14], music instrument recognition [15] and recognition of species of animals by their calls [16]. Due to lack of space, we will reserve these results for a future publication. In this paper, we will use the Mel-Frequency Coefficients (MFCC) as features. We note here that MFCC provided some of the best mean results in our preliminary experiments.

In short, MFCCs are calculated by taking the magnitudes of frequency components using an acoustically-defined scale called *mel*, which originated from the study of Stevens et al. [17]. This scale relates physical frequencies to the frequencies perceived by the human auditory system. Equation 1 shows the conversion from frequency \(f\) to mel-frequency \(m\). Next, we apply a Discrete Cosine Transform. The MFCC are the cepstrum coefficients obtained from this operation. We direct the interested readers to [13] for a detailed explanation.

\[
m = 2595 \times \log_{10}(1 + \frac{f}{700})
\] (1)

### IV. Stream Learning Approaches

In general, two approaches can be used for dealing with concept drifts in data streams [18]: i) approaches that detect changes and adapt the classification model; and ii) approaches that adapt the classification model at regular time intervals without considering whether in fact a change occurred.

The first approach makes use of change detection methods to assist in deciding the best time for updating the classification model. In general, the detection methods assume the premise that the classifier error in non-stationary processes increases over time. The identification of changes is given according to a constantly monitored indicator such as error rate. As soon as the indicator achieves a particular threshold, the classification model is updated. The need to know the true class labels of the newly classified examples, even if with some delay, for monitoring the indicators is a major difficulty of using a change detection method. In several applications, such as the insect identification sensor discussed in this paper, the true labels of the newly processed examples are unknown. For this reason, we do not use this approach in this paper.

The second approach assumes that the information obtained from the most recent data is the most relevant, and the data becomes less relevant as the time passes. This approach uses window techniques that keep the most recent data or assign weights to the examples according to their age. The main difference between the window and weight approaches is that window techniques use only the data inside of the window. In contrast, weight techniques use all the data, with old data being assigned to smaller weights. Due to its simplicity and since there is no need to know the true labels of the test examples, we decided to use this approach in this work.

We evaluate five different settings through the use of fixed-size and landmark windows to form the training set:

1) **Fixed window at the beginning of stream.** The training set is composed by a percentage of data from the beginning of the stream and not updated over time. Thus, it is possible to observe whether the data in the beginning of stream are sufficiently representative so that the classifier does not need to adapt over time;

2) **Landmark window updated with predicted labels.** The training set is initially composed by a certain percentage of data from the beginning of the stream. When a new instance is classified, it becomes part
of the training set together with the predicted label, so that the training set increases incrementally with each new classified instance. Therefore, incorrectly labeled examples are inserted in the training set as the algorithm makes incorrect predictions;

3) Landmark window updated with correct labels. Similar to the previous setting, with the difference being that the training set is always updated with the correct labels. Note that this approach is not possible in practice, but we included it here as a reference performance;

4) Sliding window updated with predicted labels. It uses a sliding window of fixed size as a training set. In this setting, the training set is constantly updated whenever a test instance is processed, so that the oldest instance is dropped off from the window and a new classified instance is added with its predicted label. Thus, we can observe whether the recent data are more representative than the old data and analyze the influence of the removal of old instances;

5) Sliding window updated with correct labels. Similar to the previous setting, with the difference that the training set is always updated with the correct label of the last instance processed. Once again, this approach is not feasible in practice, but it is included here as a point of reference.

The setting fixed window at the beginning of stream simulates the use of a static model applied in a data streams scenario. The result obtained by this configuration can be considered as a “lower” reference to the other settings, in the sense that other settings are able to adapt to recent data and should be able to outperform this setting. The settings landmark window updated with predicted labels and sliding window updated with predicted labels are the closest to the intended scenario in this work. The settings landmark window updated with correct labels and sliding window updated with correct labels represent the best case of the two previous configurations and can be used as an “upper” reference for them.

V. RESULTS AND DISCUSSION

We present our empirical results divided in two sections. Section V-A provides the results for batch learning, i.e., we ignore the time attribute of our data and evaluate the entire dataset using a train and test approach. Although batch learning is not suitable for our application, it will give us the best classifiers for our data. Section V-B presents and discusses the results of using these classifiers as base classifiers with the window settings discussed in Section IV.

A. Batch Results

It is clear that batch learning is not suitable for data generated from a non-stationary process that potentially has an unlimited size. However, batch learning is a simple setting in which we can evaluate a large number of classifiers and select the best to be used in a stream setting. As a collateral result, the performance obtained by the batch classifiers are also “upper” reference performances. This “upper” adjective is used due to the fact that the batch classifier is able to see all the data (past and future) and therefore will probably achieve better performance than any stream classifier.

As mentioned before, we use the Mel-Frequency Cepstrum Coefficients as attributes. More specifically, 50 coefficients. The batch experiments were carried out using Weka [19]. We searched the parameters space of the classification algorithms in order to optimize them, this procedure is especially important for the Support Vector Machines. We assess our results using the leave-one-out cross-validation procedure. Due to space restrictions, we report the results of the five classifiers with best performance (and its respective parameters values) in terms of accuracy in Table II.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter values</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine</td>
<td>Puk kernel with $\omega$, $\sigma = 1$</td>
<td>93.95</td>
</tr>
<tr>
<td>Random Forest</td>
<td>Trees = 15</td>
<td>92.09</td>
</tr>
<tr>
<td>$k$-Nearest Neighbors</td>
<td>Neighbors = 1, Euclidean dist.</td>
<td>92.08</td>
</tr>
<tr>
<td>Gaussian Mixture Models</td>
<td>Gaussians = 10</td>
<td>91.56</td>
</tr>
<tr>
<td>Radial Basis Function Network</td>
<td>Clusters = 5</td>
<td>89.92</td>
</tr>
</tbody>
</table>

We note that the accuracy rates reported in Table II are around 90% and no learning paradigm stands over the others by a large margin. We decided to select $k$-Nearest Neighbors ($k$-NN) and Gaussian Mixture Models (GMM) as base classifiers for our data stream experiments. These two classifiers were ranked among the top five, besides having some interesting properties. $k$-NN is simple and naturally incremental and GMM naturally outputs class membership probabilities, which will prove useful when incorporating new instances into the training set, as we discuss next.

B. Stream Results

Given the settings of the data stream classification described in Section IV and the base classifiers selected in Section V-A, we present in this section the results achieved with insect stream data.

We performed experiments with window sizes of 5%, 10%, 15% and 20% of the entire stream. Due to the similar behavior of the results, we only present the results for an intermediate window size (15% or 799 examples) here.

The results for the five stream settings with $k$-Nearest Neighbor as base classifier are illustrated in Figure 4.a. In this figure, the horizontal axis represents the temporal order of instances in the stream and the vertical axis represents the precquential accuracy [20] over time. Precquential accuracy is the classification accuracy obtained by the cumulative sample of all classified samples.

Figure 4.a shows that the only settings able to outperform the “lower” reference classifier fixed window at the
Beginning of stream are the “upper” reference classifiers landmark window updated with correct labels and sliding window updated with correct labels. The best non-reference setting was the landmark window updated with predicted labels that obtained results slightly worse than the “lower” reference.

There are two major observations that we can make from the results in Figure 4.a. First, the best performance was achieved by the setting that increases the training set incrementally and does not dispose older instances. Also, note that the disposal of the training set instances by their age does not seem to be the best philosophy, since the sliding window settings achieved worse results than the landmark window settings. One explanation for this is the presence of recurrent changes in the data, since the disposal of instances can remove the oldest concepts in the training set that could be used again. Recurrent changes are common in biological data since living beings frequently present circadian rhythms. We will explore the disposal of instances in future research.

Secondly, the settings that add new (correct) instances outperformed the fixed training set setting. This is evidence that incorporating the most recent data of the stream has a beneficial effect for the classification. In contrast, adding (indiscriminately) data with predicted labels seems to degrade the classification. However, we can soften this situation by using a probabilistic classifier such as a simple Naive Bayes or Gaussian Mixture Models (or a classifier that outputs a probability of belonging to a given class is higher than the threshold (95% in the case of our experiments). In this way, we expect to reduce the error propagation in the training set.

The results for the five proposed settings using the GMM algorithm are illustrated in Figure 4.b. Just the classifier that uses a sliding window with predicted labels lies below the “lower” reference fixed window at the beginning of stream.

We can also calculate the mean accuracy obtained by each of the five settings with k-NN and GMM algorithms. These mean accuracy rates are calculated over the entire stream test set. Table III presents the results. GMM outperformed the k-NN algorithm with the same setting, while comparing the mean accuracy for the entire test set. GMM obtained 77.93% and k-NN 69.85%.

Table III

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed window at the beginning of stream</td>
<td>87.79 80.70</td>
</tr>
<tr>
<td>Landmark window updated with correct labels</td>
<td>90.82 87.92</td>
</tr>
<tr>
<td>Landmark window updated with predicted labels</td>
<td>87.60 84.22</td>
</tr>
<tr>
<td>Sliding window updated with correct labels</td>
<td>88.91 86.06</td>
</tr>
<tr>
<td>Sliding window updated with predicted labels</td>
<td>69.85 77.93</td>
</tr>
</tbody>
</table>

Although the performance of the GMM algorithm is lower than the k-NN algorithm, probabilistic algorithms are interesting since they allow for the exploration of different philosophies for disposal and insertion of instances in the training set. Furthermore, the sensor location in space and time will change the a priori probabilities of passage of certain species. This sort of information can be easily incorporated in generative models such as GMM.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we present the evaluation of five settings for data stream learning using two standard classifiers. Our results show that k-NN obtained the best overall results when
compared to GMM. However, GMM obtained significant performance improvements for the classifiers updated with predicted labels.

It is interesting to note that the approach that uses a fixed training set in the beginning of the stream obtained very competitive results. However, the performance of this setting is likely to be optimistic. Due to our necessity of class labels, we collected data in laboratory conditions which are more stable than field conditions. Therefore, concept drifts caused by climate changes are likely to be more gradual than in real conditions. We note, however, these drifts are still present in laboratory conditions due to minor temperature changes, insect circadian rhythm and aging.

There are several directions in which this work will be extended. We are building a climate controlled chamber in our laboratory so that we can collect laboratory data with controlled concept drifts caused by climate changes. We believe this dataset will be useful for the stream data research community. Until now, the community has used completely artificial datasets to evaluate concept drift algorithms. Artificial datasets are currently the only resource available to evaluate stream learning algorithms when one needs to know the exact moment a drift occurred.

We will also evaluate several philosophies of discarding training examples, for instance: by age, by number of training samples of each class, etc.

ACKNOWLEDGMENT

This work was funded by FAPESP awards 2011/17698-5, 2011/04054-2, 2012/50714-7 and 2012/07295-3.

REFERENCES


