AUTOPILOT: Simulating Changing Concepts in Real Data

Patrick Lindstrom[1], Sarah Jane Delany[2], and Brian Mac Namee[1]

[1] School of Computing,
Dublin Institute of Technology, Dublin, Ireland
[2] Digital Media Centre,
Dublin Institute of Technology, Dublin, Ireland
first-name.second-name@dit.ie

Abstract. An increasingly important area in supervised incremental learning is learning in the presence of changing concepts. Research into concept drift is hampered by the lack of availability of controllable 'real life' datasets. In this paper we propose an approach for generating real life data over which we have control of the concept and can generate data exhibiting different types of concept drift. The approach uses a 3-D driving game to produce a data stream of instances describing how to drive around a track. The classification problem is learning the driving technique of the driver, which can be affected by changes in the driving environment causing changes to the concept. The paper gives illustrations of different types of concept drift and how standard concept drift handling techniques can adapt to the concept drift.

1 Introduction

In many real world classification problems the concept being modelled may not be static but can change over time. These changes can be due to external circumstances, hidden contexts or even changes in the underlying data distribution. Examples of changing concepts can be seen in a variety of real world applications. Weather predictions are influenced by seasonal weather variations, customer buying preferences can be influenced by fashion trends or seasonal inclinations and information filtering is dependent on the document content and user interest. Concept drift would undoubtably be present in the above examples, however it would be hard to ascertain when the concept changed, or at what rate it is changing. Without this crucial information evaluating an algorithm for dealing with concept drift can be problematic. Research into handling changing concepts has been hampered by a lack of availability of controllable real world datasets. Researchers use their own specific real world datasets (which are often not made available due to privacy issues) or some publicly available artificial datasets [1]. There is a need to focus on finding real world datasets that show concept drift and where there is some level of control over the concept drift.

The focus of this paper is to simulate concept drift in real data. Our approach uses a 3-D driving game to produce a data stream of instances which describe how

to drive a car around a track. The classification problem is learning the driving technique of the driver. The concept drift is introduced by changing the driving environment. For example, after the weather worsens the track can become wet from rain and the friction of the car alters and affects the driving style of the driver. The classifier which has modelled the driver's driving technique is then inappropriate for the new environment and has to adapt to the new environment.

The paper is organised as follows. Section 2 discusses existing research into concept drift concentrating on the types of drift, the data used and the approaches to handling the drift. Section 3 describes how the driving game has been used to generate datasets exhibiting different types of concept drift. An illustration of handling this concept drift is included in Section 4, with discussion and directions for future work in Section 5.

2 Existing Work

Existing research into handling concept drift generally has two sources of data, real world datasets and artificial datasets. Real world data sets are collected from a naturally occurring process. Examples include financial [2], biological [3] and spam filtering [4] data. With real concept drift data it can often be hard to ascertain when and why the concept drift occurred. Some research has involved tweaking real world datasets by deleting classes or including data from new classes [5] or by manipulating class labels [6]. Datasets based on confidential data such as financial data are generally not made available to other researchers.

Another approach is to use artificially created data sets, such as STAGGER [7], the moving hyperplane [8] and Narasimhamurthy's framework [1]. In this paper we present a technique for creating real world data where the point at which drift occurs and how quickly it occurs can be controlled.

2.1 Types of Concept Drift

Concept drift can be broadly categorised into three categories.

Sudden shift is when the concept changes abruptly. Tsymbal [9] uses the example that someone graduating from college might suddenly have completely different monetary concerns.

Gradual drift is when the concept gradually changes from one concept to another. Widmer and Kubat [10] used the example of a device that gradually begins to malfunction to illustrate the idea of gradual drift. Stanley [11] categorised drift into moderate and slow drift, depending on the rate of change.

Recurring trends or contexts is when trends or patterns can be found to repeat themselves at intervals. Recurring trends are commonly found in seasonal data [12].

When concept drift or shift occurs due to a change in the data distribution it is known as virtual concept drift [10] or population shift [13]. Virtual concept drift may manifest itself as any type of concept drift.

A change in concept may also occur when all features relevant to a concept are not identified (also known as hidden context), for example seasonal changes causing regular cyclical transformations in many natural phenomena [12].

2.2 Handling Concept Drift

Tsymbal [9] categorises the approaches for handling concept drift under the headings of instance selection, instance weighting and ensembles.

Instance selection is a broad term that can be used to describe concept drift algorithms that select which instances to train on, based on their perceived relevance to the current concept. The most common technique is a sliding window based technique where the classifier is retrained periodically on a 'window' of new training data . Windowing techniques can be broken down into fixed size algorithms which learn from the last n instances [14] and adaptive size algorithms which uses some heuristic to adjust the window size [15, 16].

Instance weighting creates a bias towards certain instances by assigning high weights to them. Instances can be weighed according to different features, such as age or competence with regard to the current concept. Klinkenberg found that when evaluated against a sliding window approach instance weighting compared unfavourably [17].

An ensemble is a collection of classifiers whose predictions are combined to classify new instances. Kuncheva [18] presents the ensemble approach to learning in changing environments as online learning with forgetting. Online learning is achieved by adding new ensemble members trained with the most recent data to the ensemble and forgetting is achieved by deleting old or less-useful members.

3 Description of Approach

The approach taken to produce the real-world dataset is to use a 3-D driving computer game to generate a data stream of instances that reflect how the driver has driven around the track. The game was built on top of the Microsoft XNA Racing Car Starter Kit[3]. Functionality to collect data about how the player drives the car around the track and to change the driving conditions was added to the game.

3.1 Generating the Datasets

To generate the dataset a player drives the car around the track. An instance is saved every frame of the game. The data collected at each frame reflects the car's position at that point in the game and the action that the driver took. The data collected for each instance, listed in Table 1, can be categorised as follows;

(i) classification features, labelled F_i, which reflect the position and driving characteristics of the car such as acceleration and speed at that point.

[3] http://creators.xna.com/en-us/starterkit/racinggame

(ii) the action the driver was making at that point, labelled $Class$.
(iii) certain meta data used for analysing the results, labelled M_i.

Table 1. Car Data Instance

Feature	**Description**
F_{DTLL}	Distance from top left corner of car to left guard rail
F_{DTLR}	Distance from top left corner of car to right guard rail
F_{DTRR}	Distance from top right corner of car to left guard rail
F_{DTRL}	Distance from top right corner of car to right guard rail
F_{DBLL}	Distance from bottom left corner of car to left guard rail
F_{DBLR}	Distance from bottom left corner of car to right guard rail
F_{DBRR}	Distance from bottom right corner of car to left guard rail
F_{DBRL}	Distance from bottom right corner of car to right guard rail
F_{ACC}	Current acceleration of car
F_{SP}	Current speed of car
F_{LA}	The action (class) of the last instance
M_{LN}	The lap number
M_{TS}	A timestamp of when the instance was taken
M_{CP}	The x, y & z co-ordinates of the car
M_{CD}	Car details, information about the car such as mass and max speed
$Class$	The action (class) of the instance

The class of the instance is the action the user took at that sample point. The possible actions include $Accelerate$, $Decelerate$, $TurnLeft$ and $TurnRight$ and all possible combinations of these four actions. Some combinations are prevented by the game logic such as $TurnLeft$ AND $TurnRight$. This leaves nine valid classes which are detailed in Table 2.

Table 2. Classes used

Class	**Dec value**	**Binary value**	**Description**
C_{NO}	0	0000	No operation taken by the user
C_{AC}	1	0001	Accelerate
C_{DC}	2	0010	Decelerate
C_{TL}	4	0100	Turn Left
C_{TR}	8	1000	Turn Right
C_{TLAC}	5	0101	Turn Left AND Accelerate
C_{TLDC}	6	0110	Turn Left AND Decelerate
C_{TRAC}	9	1001	Turn Left AND Accelerate
C_{TRAC}	10	1010	Turn Left AND Decelerate

Preliminary experiments show there is some redundancy among the positioning features but we expect that with more complex tracks (including varying road widths sharp bends etc.) these features may be useful so we have chosen to include them. The meta data is not used for classification, but can be used to analyse the results. For example M_{CP} can be examined to establish the place the car was on the track when, for example, there was a change in accuracy. This may indicate certain parts of the track causing problems for the classifier.

As instances are saved each frame, the size of the dataset generated depends on the frame per second (fps) rate of the game which in turn depends on the current hardware usage. The fps rate and therefore the sampling rate usually varies between 30 and 60 fps. Each lap of the track takes about 2 minutes to complete and results in, on average, about 8000 instances being sampled.

3.2 The Classification Problem

The player drives the car around the track for a number of laps. The instances generated in one lap of the game can be used as training data and the data generated from other laps can be presented in a data stream for classification as test data. Support Vector Machines (SVMs) [19] using a linear kernel was chosen as the classifier. We chose to use a linear kernel as preliminary experiments showed that its performance was comparable to the more complex kernels and did not require parameter tuning.

The following illustrative example shows that it is possible to learn the driving technique of a player. Consider a player that drives a number of laps around the track. The car starts off stationary. Due to its mass and acceleration capability it takes a certain amount of time to gather speed once the player starts driving. The driver also needs to accelerate often to overcome friction. This is reflected in the class distribution of the first lap shown in the second column in Table 3. There are no left turns (C_{TL}) or right turns (C_{TR}) in the first lap. Both turns are coupled with an accelerate action (C_{TLAC} and C_{TRAC}).

The instances saved for the first lap are used as training data and an SVM classifier is built on this data. The data from the subsequent lap is then used as test data. Each instance from this test lap is classified and the actual action taken by the driver is compared against the predicted action from the classifier and accuracy calculated as the proportion of the total number of test instances that were correctly classified. The classification resulted in an overall 97.57% accuracy and the results, broken down into the percentage frequency of each class and the precision and recall, are presented in Table 3.

The class distribution in the data is very skewed with very high frequencies for the *Accelerate* class and low frequencies for the other classes. It is worth noting that not all classes are used for this driver on this track.

3.3 Changing the Concept

To change the concept the game code was modified to allow the mass and acceleration of the car to be adjusted while a person was playing the game. It

Table 3. Static concept class distribution details

	Training Lap	Test Lap		
	Freq (%)	Freq (%)	Precision	Recall
C_{NO}	0.39	1.60	0.979	0.979
C_{AC}	92.37	90.42	0.987	0.987
C_{TL}	0.00	0.08	0	0
C_{TLAC}	5.23	5.61	0.833	0.845
C_{TR}	0.00	0.00	0	0
C_{TRAC}	2.01	2.29	0.88	0.88

was expected that this change in the game behaviour would result in a different driving style by the player and therefore a change in the concept. The change in mass and acceleration was designed to simulate a change in ground friction, with a high mass car having high friction and a low mass car a lower friction. In a real world scenario, events such as changes in weather conditions may significantly alter the road conditions and consequently the ground friction. This would in turn affect the way in which a driver drives the car. Future work may include a more reality grounded approach such as introducing strong winds into the game.

To try to handle the change in concept a straightforward technique for handling concept drift was used, a fixed-size sliding window approach. After 500 instances had been presented for classification to the classifier, the classifier was rebuilt on updated training data. This new 'window' of training data included the new 500 instances just processed while the oldest 500 instances were removed.

4 Handling the Drift

Two experiments were run to test the data generated to see if the expected concept drift was present and if it could be handled. The experiments were designed to generate a *concept shift* and a *concept drift* dataset.

4.1 Sudden concept shift

In order to simulate a sudden shift in the concept the car was driven four laps around the track. The first two laps were driven using a heavy car with slow acceleration. At the start of lap three the mass and acceleration of the car was changed simulating a change in the driving conditions. The last two laps were driven with a light car with fast acceleration. The first lap was used as training data and the remaining three laps were presented as a data stream for classification against the resulting classifier. The distribution details and the classification results are detailed in Table 4. Precision and recall figures are included for each class for each lap.

It is evident that the classifier trained on lap one is successful on lap two and fails for the subsequent laps once the concept changes. The concept change in

Table 4. Dataset 1: Concept shift dataset class distribution

	Lap 1	Lap 2			Lap 3			Lap 4		
	Freq (%)	Freq (%)	Pr	Rec	Freq (%)	Pr	Rec	Freq (%)	Pr	Rec
C_{NO}	0.15	1.03	0	0	84.86	0	0	85.17	0	0
C_{AC}	92.53	90.31	0.98	0.99	5.67	0.06	0.99	5.88	0.06	0.99
C_{TL}	0	0.22	0	0	6.66	0	0	6.51	0	0
C_{TLAC}	5.26	5.90	0.81	0.84	0.03	0.01	1	0	0	0
C_{TR}	0	0	0	0	2.77	0	0	2.37	0	0
C_{TRAC}	2.06	2.54	0.87	0.88	0	0	0	0.07	0.03	1

this data can be considered as being caused by a dramatic distribution change with the driver accelerating much more significantly with the heavy car and freewheeling more significantly with the lighter car, as would be expected. For laps one and two the class distribution is heavily skewed towards *Accelerate*. After the change in concept the car only needs to accelerate every few seconds to combat friction and keep a constant speed. This results in the data for laps three and four being heavily skewed towards *NoOperation*. The precision and recall figures reveal that classes with few or no instances in the training data were very hard for the classifier to correctly classify.

Figure 1 shows the accuracy over time for this dataset, Dataset 1, with accuracy reported for each batch of 500 instances presented. The solid line labelled *Static Model* represents the static model, the accuracy based on the initial classifier trained on the data from lap one. This shows the accuracy initially fluctuating between 81% and 97% and then dropping dramatically at lap three when the concept changed consistent with the results in Table 4.

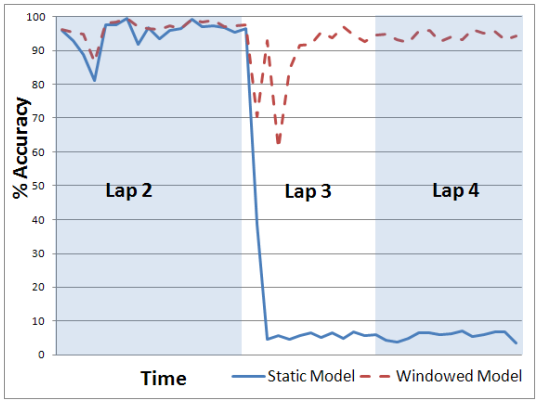

Fig. 1. Classifier accuracy plotted over time showing the sudden concept shift in Dataset 1 (*Static Model*) and how it can be handled (*Windowed Model*).

Figure 1 also shows the results of applying the fixed-size windowing approach to the data in Dataset 1 to handle the changes in concept. The dashed line labelled *Windowed Model* shows the results of updating the initial classifier created from the training data with this windowing approach. As can be seen from the graph the accuracy is slightly better on lap two and significantly better on laps three and four.

4.2 Gradual concept drift

To simulate gradual concept drift four laps were driven around the same track. For the first and second lap the heavy car with slow acceleration was used. On the third lap the mass and acceleration transitioned from the values of the heavy car to the values of the light car over 60 seconds. The fourth lap was driven with the light car. The classifier was trained using the data gathered from the first lap and tested with the remaining data. This resulted in the dataset shown in Table 5.

Figure 2 shows a graph of the accuracy of this dataset, Dataset 2, over time. The solid line, labelled *Static Model* shows reasonable consistent high accuracy on lap two. However when the concept starts changing at the beginning of lap three the accuracy declines but more gradually than that of Dataset 1, the concept shift data. The dashed line, labelled *Windowed Model*, showing the accuracy achieved by the updating the classifier using the windowing approach, demonstrates the successful handling of the concept drift. The behaviour of the dashed lines at the start of lap three on both graphs suggests that it is easier to learn a gradual change in concept than a sudden change in concept.

Table 5. Dataset 2: Concept drift dataset class distribution

	Lap 1	Lap 2			Lap 3			Lap 4		
	Freq (%)	Freq (%)	Pr	Rec	Freq (%)	Pr	Rec	Freq (%)	Pr	Rec
C_{NO}	0.67	2.60	0	0	74.65	0.96	0.34	84.73	0.97	0.30
C_{AC}	91.48	89.02	0.96	0.99	15.48	0.23	0.97	6.11	0.09	0.94
C_{TL}	0	0.08	0	0	6.44	0	0	6.60	0	0
C_{TLAC}	5.53	5.84	0.83	0.84	0.48	0.06	0.75	0	0	0
C_{TR}	0	0.07	0	0	2.60	0	0	2.55	0	0
C_{TRAC}	2.31	2.39	0.86	0.87	0.34	0.10	0.6	0	0	0

There are some other interesting observations that can be made from the data. Both graphs of the static model in Figs 1 and 2 both show a dip in accuracy at the start of the second lap. This can be explained by the fact that the car starts from a stationary position in the training data but is moving at speed at the start of the second lap. Also, lap two in both graphs takes longer than laps three or four due to the slower acceleration capabilities of the heavy car.

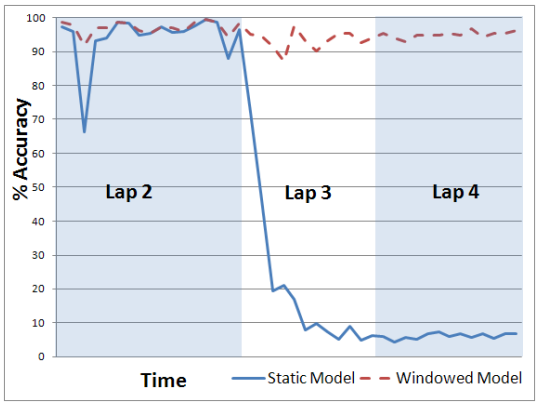

Fig. 2. Classifier accuracy plotted over time showing the gradual concept drift in Dataset 2 (*Static Model*) and how it can be handled (*Windowed Model*).

5 Conclusions and Future Work

This paper presents an approach to generating a real world dataset for incremental learning of changing concepts. The advantages offered by such a dataset include the opportunity to control the timing and extent of the concept changes introduced. We have shown that we can introduce and handle changes to the concept, both *concept shift*, a dramatic immediate change and *concept drift*, a more gradual change.

However, research into handling concept drift in real world situations has to consider a number of important factors. It is important to be able to identify when the concept changes and to have the capability of providing the classifier with 'correctly' labelled data from which to learn the changing concept. Spam filtering is an example of a real world application which handles concept drift where the capability of providing correctly labelled data is relatively straightforward. The user of the system can indicate errors by the classifier, which may indicate concept changes, by highlighting missed spam emails or by retrieving incorrectly labelled legitimate emails.

In order to simulate a real-world situation, the scenario offered by the driving game should take into consideration factors such as these. Our experiments in this paper have assumed that 'correctly' labelled instances are available from which to learn. We envisage that future work will consider how we can simulate such real world factors. Rather than waiting for the car to crash, changes in the concept could be triggered by monitoring the position of the car on the track with respect to the guard rails. Correctly labelled data could be provided by requesting the player to take over the controls when changes in the concept are anticipated. Consideration of such issues will allow a more realistic simulation of a real world scenario.

References

1. Narasimhamurthy, A., Kuncheva, L.I.: A framework for generating data to simulate changing environments. Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications table of contents (2007) 384–389
2. Abdullah, M., Ganapathy, V.: Neural network ensemble for financial trend prediction. TENCON 2000. Proceedings **3** (2000)
3. Tsymbal, A., Tsymbal, A., Pechenizkiy, M., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections. In: Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on. (2006) 679–684
4. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems **18**(4–5) (2005) 187–195
5. Klinkenberg, R.: Using labeled and unlabeled data to learn drifting concepts. Workshop notes of the IJCAI-01 Workshop on Learning from Temporal and Spatial Data (2001) 16–24
6. Black, M., Hickey, R.: Maintaining the performance of a learned classifier under concept drift. Intelligent Data Analysis **3**(6) (1999) 453–474
7. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning **1** (1986) 317–354
8. Kolter, J., Maloof, M.: Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. (2003) 123–130
9. Tsymbal, A.: The problem of concept drift: definitions and related work. Informe técnico: TCD-CS-2004-15, Departament of Computer Science Trinity College **4** (2004) 2004–15
10. Widmer, Kubat: Learning in the presence of concept drift and hidden contexts. Machine Learning **23** (1996) 69–101
11. Stanley, K.O.: Learning concept drift with a committee of decision trees. Computer Science Department, University of Texas-Austin (2001)
12. Widmer, G., Kubat, M.: Special issue on context sensitivity and concept drift. Machine Learning **32** (1998) 83?201
13. Kelly, M.G., Hand, D.J., Adams, N.M.: The impact of changing populations on classifier performance, San Diego, California, United States, ACM (1999) 367–371
14. Kubat, M.: Floating approximation in time-varying knowledge bases. Pattern recognition letters **10** (1989) 223–227
15. Widmer, G., Kubat, M.: Learning flexible concepts from streams of examples: Flora 2. (1992) 463–467
16. Klinkenberg, R., Joachims, T.: Detecting concept drift with support vector machines. Proceedings of the Seventeenth International Conference on Machine Learning (ICML) (2000) 11
17. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intell. Data Anal **8** (2004) 281–300
18. Kuncheva, L.I.: Classifier ensembles for changing environments. In Roli, F., Kittler, J., Windeatt, T., eds.: 5th International Workshop on Multiple Classifier Systems (MCS 2004), Springer (2004) 1–15
19. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc (1995)