Soft Computing 8 (2003) 102–108 © Springer-Verlag 2003
DOI 10.1007/s00500-002-0250-2

Learning classification rules for telecom customer call data under concept drift

M. Black, R. Hickey

Abstract The application of the CD3 decision tree induction algorithm to telecommunications customer call data to obtain classification rules is described. CD3 is robust against drift in the underlying rules over time (concept drift): it both detects drift and protects the induction process from its effects. Specifically, the task is to data mine customer details and call records to determine whether the profile of customers registering for a '*friends and family*' service is changing over time and to maintain a rule set profiling such customers. CD3 and the rationale behind it are described and experimental results on customer data are presented.

Keywords Concept drift, Decision trees, Adaptive learning, User profiling

1 Introduction

On-line learning systems which receive batches of examples on a continual basis and are required to induce and maintain a model for classification have to deal with two substantial problems: noise and concept drift.

The effects of noise have been studied extensively [3] and this has lead to noise-proofed decision tree learners such as C4.5 [13], C5 [14] and rule induction algorithms, e.g. CN2 [16]. Although considered by Schlimmer and Grainger [6] and again recently by others including Widmer and Kubat [10], Widmer [9] and, from a computational learning theory perspective, by Hembold and Long [7], concept drift [1, 8, 10] has received considerably less attention. There has also been work on time dependency for association rule mining; see, for example, [11] and [12].

By concept drift we mean that rules defining concepts (i.e. classes) are subject to change over time [5, 7–10]. Typically this change may affect some rules but not others. Examples of this phenomenon are common in real world applications such as fraud detection for, say, credit card or mobile phone usage. Here drift may be prompted by advances in technology that make new forms of fraud possible or may be the result of fraudsters altering their behaviour to avoid detection. The consequences of ignoring concept drift when mining for classification models can be catastrophic [1].

Published online: 14 July 2003

M. Black (✉), R. Hickey
Faculty of Informatics, University of Ulster,
Coleraine, BT52 1SA, Northern Ireland
e-mail: mm.black@ulster.ac.uk

As noted by Widmer [10], distinguishing between noise and drift is a difficult task for a learner. The dilemma for the algorithm is to decide between the two: has drift occurred and if so, how are the learned class definitions to be updated? It is common practice to use a time windowing mechanism on the training examples to cope with drift. In the system Flora 4, concept hypotheses description sets are maintained using a statistical evaluation procedure applied when a new example is obtained. Examples deemed to be no longer valid are 'forgotten', that is dropped from the window of examples used for forming generalisations. A window adjustment heuristic (WAH) is employed for this purpose. As is usual with time windowing, the approach assumes that it is the newer examples which are relevant and thus older examples are dropped. Although window size is dynamic throughout learning there is a philosophy of preventing the size from becoming overly large: if after receiving a new example the concepts seem stable then an old example is dropped.

Drift may happen suddenly, referred to as *revolutionary* [1, 2], or may happen gradually over an extended time period, referred to as *evolutionary*; see [1] and [5]. In the former case we refer to the time at which drift occurred as the *drift point*. We can regard evolutionary drift as involving a series of separate drift points with very small drift occurring at each one.

In [1, 2] we proposed a new architecture for a learning system, CD3, to aid detection and correction of concept drift. Like the METAL systems presented by Widmer [9] CD3 utilizes existing learners, in our case tree or rule induction algorithms – the basic induction algorithm is really just a parameter in the system. CD3 provides an alternative strategy to '*windowing*' called '*purging*', as a mechanism for removing examples that are no longer valid. The technique aims to keep the knowledge base more up-to-date by not offering preference to the newer examples and thus retaining older valid information that has not drifted.

2 TSAR and the CD3 algorithm

We assume data will arrive in batches, where batch size may vary from a single example to many. Incoming data may be batched in accordance to time of arrival or in pre-determined numbers of examples. Batch variations may also exist within one induction process.

The central idea is that a time-stamp is associated with examples and treated as an attribute, *ts*, during the induction process. In effect the learning algorithm is assessing the relevance of the time-stamp attribute. This is referred to as the *time-stamp attribute relevance* (TSAR) principle, the implication being that if it turns out to be relevant then drift has occurred.

Any tree or rule induction algorithm which effectively handles noise can be used at the heart of a TSAR learning system. In particular, if ID3 with post-pruning is used we call the resulting system CD3 (CD = concept drift). We have implemented a version of CD3 which uses the well-known Niblett–Bratko post-pruning algorithm [17].

Specifically, the system maintains a set of examples (the *example base*) deemed to be valid and time-stamps these as *ts=current*. When a new batch arrives, the examples in it are stamped *ts=new*. Induction then takes place using a noise-proofed tree or rule-building algorithm – referred to as the *base algorithm*. The pseudocode for CD3 is presented in Table 1.

Following the induction step, CD3 will provide a pruned induced tree structure that may or may not have the *ts* attribute present. This induced tree could be used for classification of unseen examples by merely having their descriptions augmented with the *ts* attribute and setting its value to '*new*'. It will be assumed that all unseen examples presented following an induction step are generated from the current model in force when the most recent batch arrived. This assumption is held until another new batch of training examples arrives invoking an update of the tree.

CD3, however, offers another method for classification which does not require unseen examples having their description altered. From the pruned induced tree CD3 will extract rules, the rules representing all the concept paths from the root to the leaves. In the case of *ts* being present in some or all of the paths, CD3 will look upon those having a *ts* value '*current*' as being out-of-date and now an *invalid* rule. Paths with the ts value '*new*' will be viewed as up-to-date and thus *valid* rules. Finally, paths in which *ts* is not instantiated correspond to rules which were valid prior to the new batch and are still so, that is, unchanged *valid* rules.

Table 1. Pseudocode for CD3 induction algorithm

```
CD3 (Header_file, Batch_parameters, Data_file, Output_file,
Purger_parameter, No_of_trials)
   repeat for No_of_trials
   load file specification(Header_file);
   load training data(Data_file, Trial);
   begin
      extract first batch and mark as
                    'current'(Batch_parameters);
      while more batches
         extract next batch and mark as 'new';
         call ID3_Induce_tree;
         prune induced tree;
         extract rules;
         separate drifted rules;
         purge invalid training examples(Purger_parameter);
         append 'new' examples to 'current';
         test rules;
         output results(Output_file);
      end while
   end repeat
end CD3
```

These rules can now be separated into valid and invalid, and the *ts* attribute dropped from the rule conditions. The TSAR methodology uses the *ts* attribute to differentiate between those parts of the data which have been affected by drift from those that have not. Once the invalid and valid rules have been highlighted and separated, the *ts* attribute has fulfilled its purpose and can now be removed.

Classification requests can be ongoing, but since its goal is to provide the most up-to-date classifier, CD3 must always be ready to accept these updates. This requires maintaining a correct and up-to-date database of training examples. Following the recent induction step and rule conversion, the CD3 database is out of date with regards to its knowledge structure. Before CD3 can accept these new updates of training examples it must remove existing examples in the knowledge base that are covered by the most recently identified invalid rules. As presented in [1, 2] CD3 provides a removal technique called *purging* which can be applied to the knowledge base following the concept drift detection phase to extract the now out of date examples thus maintaining an up-to-date version of the database. In purging examples that are no longer relevant, the TSAR approach does not take account of age as windowing mechanisms tend to do. Rather the view is that an example should be removed if it is believed that the underlying rule, of which the example is an instance, has drifted, namely matches that of an invalid rule.

The invalid rules can be discarded or stored for trend analysis. This is a separate activity not catered for by CD3. The valid rules are used for classification with an unseen example description being matched against one of the rule conditions to obtain its classification.

A TSAR system implements incremental learning as batch re-learning, as such the knowledge is induced again from scratch every time a new batch arrives with the new batch being added to the existing example base. This is in contrast to FLORA4 which incrementally learns on receipt of *each* new example and without re-learning from the existing examples. Bearing in mind that, depending on the extent and frequency (over time) of drift, many examples could be purged from the example base this is not as inefficient as it might appear. It may be possible to produce a more genuinely incremental and hence more computationally efficient implementation by exploiting the techniques used by [15] in the ITI algorithm.

The TSAR approach does not require the user to set parameter values manually other than those that may be required by the base algorithm. Noise and the presence of pure noise attributes, namely those that are never useful for classification, will interfere with the ability of CD3 to decide whether drift has occurred. The *ts* attribute may be retained in a pruned tree even if there has been no drift – a 'false positive' – and, as a consequence, some examples will be wrongfully purged (*false purging*). Conversely *ts* may be pruned away when drift has taken place with the result that invalid examples will remain in the example base and contaminate learning.

Nevertheless, we believe CD3 to be quite robust – defects in purging should have a minimal effect on the algorithm even in a very noisy domain. Refinements to the purging process are possible, however, and are the subject of ongoing research.

2.1 Refinement of the TSAR methodology

CD3 uses a very simple form of time stamping relying on just two values: 'current' and 'new'. This is sufficient to allow it to separate valid and invalid rules and to maintain good classification rates using the former. Nevertheless, it is worth considering how the mining process could be allowed to review and maybe revoke earlier decisions. In [2] we used two versions of time stamp refinement.

The first refinement of the time stamps as *batch identifiers* resulted in the CD4 algorithm. With this algorithm, each batch is assigned and retains indefinitely its own unique batch identifier. At each round of mining, all the data from the previous batches is used together with the new batch. Thus there is no purging process.

As with CD3, CD4 is able to distinguish valid and invalid rules by appropriate instantiation of the time stamp attribute possibly revising decisions made in a previous induction. As explained in [2], drift will be identified as having occurred after a particular batch and this will be represented within the knowledge structure as a binary split at the corresponding *ts* value of the batch identifier. This procedure will result in a set of invalid and valid rules. The valid rules can then be used for online classification.

The second refinement procedure removes the effects of batches on mining by using continuous time stamping in which each training example has its own unique time stamp – now a numeric attribute. The learning algorithm is now free to form binary splits as for CD4 but without regard to batch. Thus it can place a split point within a batch (either the new or any of the previous batches) and review these decisions at each new round of mining. Again the procedures for extraction of valid and invalid rules and maintenance of a database of currently valid examples are as described above. The resulting algorithm is CD5. As with CD4, there is no purging.

As demonstrated in [2] both the extension to individual batch time stamps and further to individual example time stamps in algorithms CD4 and CD5, respectively, appear to produce results comparable to, and possibly, slightly superior to those obtained from CD3.

It was also highlighted in [2] that the simple strategy of dealing with drift by ignoring all previously received data and just using the new batch of data is effective only immediately after drift. Elsewhere it prevents increase in classification rate through accumulation of data. It also, of course, denies us the opportunity to detect drift should it occur.

With the benefit from the enhanced time stamps turning out to be marginal, the choice of which algorithm to deploy may depend on the characteristics of the domain used and the nature of the on-line performance task.

It should also be noted that, without a purging process, CD4 and CD5 will produce considerably larger trees than CD3 and may take longer to learn.

3 An experimental trial with call data

Until now we have developed and experimented with artificial data [1, 2]. We are now able to extend this to an experimental trial of real world call data acquired from British telecom (BT). The main task was to ascertain whether the profiles of customers registering for the services Friends and Family, Premier Line and Option 15 were changing over time.

3.1 Data and pre-processing

Customer and call data was provided on all landline calls for a sample of 1000 customers over a period of of twenty seven months. The attributes are described in Tables 2, 3 and 4. The data was presented in five batches that reflected BT's belief that drift is most likely in March and October.

The *F&F* (for Family and Friends), *P_L* (for Premier Line) and *O15* (for Option 15) indicators, as shown in Table 2, had five separate indicators, one for each of the five time periods. These were translated into one indicator for each field, within individual batches, highlighting if the customer was now using the service in that time period. This also allowed for customers to register for a service and, within one of the successive time points, de-register for the service. These indicators are referred to as *ffind*, *plind* and *O15ind*.

Other attributes of interest were: life stage indicator (*LSIND*), acorn code (*acorn*) and single line indicator (*SLIND*).

For the attributes in Table 3 marked with an asterisk there exist 13 sets of summarised data under the following sub headings shown in Table 4. These summarised subgroups occur within the call data in the order shown resulting in $79 = 1 + 6*13$ attributes in total: the first, encoded telephone number, is unique to the customer and then there are 13 batches of six re-occurring attributes.

Only those attributes of particular interest for the problem stated above were selected for the data mining process.

It became clear from early analysis that there was a clear shift within these customers from non F&F users to F&F users over the twenty seven months period. Our initial fears come from the first batch where the proportion of the F&F users was quite small. We had concerns regarding classes with small coverage. Another concern was that this might be a population shift problem [4] and not that of concept drift. However, the results of the experiments would prove or disprove this.

Initially the acorn field had 55 values. These values are derived from the postcode and categorise communities with respect to their location and the sub-groups of the population within that community. These can then be grouped into seven higher classification groups. These higher seven bands were used for our experiments.

Similarly the life stage indicator had ten values spanning from 1: representing young people, to 10: representing retired couples. These were regrouped to five values, combining 1 and 2 to give 'ls_a', 3 and 4 to give 'ls_b' and

Table 2. Customer attributes

Field	Description	Type
encode (telno)	ecrypted telephone number	char
Distcode	district code (27 unique)	char
Startdat	customer started using no.	dd-mon-ccyy time
Acorn	residental codes from postcodes	integer
Ffind	friends and family indicator	Y/N
Ffdate	first got service	dd-mon-ccyy time
F&F: 6 attributes = Oct 1995, Mar/Oct 1996, Mar/Oct 1997, Mar 1998	had F&F service at this time	Y/N
Plind	premier line indicator	Y/N
Pldate	first got service	dd-mon-ccyy time
P&L: 6 attributes = Oct 1995, Mar/Oct 1996, Mar/Oct 1997, Mar 1998	had P&L service at this time	Y/N
O15ind	option15 ind.(fixed call amount)	Y/N
O15date	first got service	dd-mon-ccyy time
O15: 6 attributes = Oct 1995, Mar/Oct 1996, Mar/Oct 1997, Mar 1998	had O15 service at this time	Y/N
O15 in Mar 1998	"	Y/N
Xdir	Xdirectory	Y/N
Mps	Mailing preference scheme	Y/N
Tps	Telephone preference scheme	Y/N
Dontmail	Don't mail marketing data	Y/N
Lusind	Low user scheme – Code form	X/H
Ccind	Charge card indicator	Y/N
Hwind	Hard wired indicator	Y/N
Lsind	Life stage indicator (from postcode, 1..10)	integer
Slind	More than one line	Y/N
Postcode	Post code	POSTCODE

so on. The resulting attributes and their final values are shown below in Table 5.

A new attribute, *revenue*, was derived as:

$$revenue = number\ of\ calls * average\ cost\ of\ calls$$

This was discretised into six bands. For example 'a_12' represents all customers with revenue less than £12,000. The value 'b_28' represents all customers with revenue values from £12,000 but less than £28,000 and so on.

3.2
Analysis of experimental results

Following data preparation, the remaining records within each batch were split into training and test batches as shown in Table 6. Recall that the flexibility in update regime of CD3 permits the batches to be of various sizes.

The *mark2* purger regime was used for this trial due to its prior success in [1]. The first batch *Oct_95* was applied by CD3 as its current batch. The consecutive batches were applied one by one in order of date. The algorithm recorded the classification rate, *CR*, of CD3 based on the test set provided, the percentage purged on each iteration, and the highest position of the *ts* attribute within the tree. Our initial hopes were that by analysing the position of the *ts* attribute and the percentage purged we would be able to identify drift between consecutive batches clearly.

The classification performance of CD3 starts well at a peak of 85% on application of the second batch: *Mar_96* as shown in Fig. 1. It would seem that there is little change between *Oct_95* and *Mar_96*. The position of the *ts* attribute should confirm this. Application of the next two batches: *Oct_96* and *Mar_97*, show a slight decline in performance. Could this be an indication of drift?

Figure 2 confirms our initial suspicions. The *ts* attribute only reaches level 2 on the application of the second batch: *Mar_96* (0 is the root position: top, 1 is second top and so on). This could be false purging occurring or it could be

Table 3. Call attributes

Field	Type
Encode (telno)	Char
**no. of calls*	Int
**average duration of calls*	Real
**variance of duration of calls*	Real
**average cost of calls*	Real
**variance of cost of calls*	Real
**no. of distinct destinations phoned*	Int

Table 4. Call summarised sub-groups

All calls	*Low- call calls*
Day-time calls	*Mobile calls*
Directory Enquiry calls	*National calls*
International calls	*Premium Rate calls*
ISP calls	*Short calls*
Local calls	*Week-end calls*
Long calls	

Table 5. Selected and processed fields for experimentation

	Fields						
	Acorn	*F&F*	*P_L*	*O15*	*SLIND*	*LSIND*	*Revenue*
Values	acorn_a, acorn_b, acorn_c, acorn_d, acorn_e, acorn_f	y,n	y,n	y,n	y,n	ls_a, ls_b, ls_c, ls_d, ls_e,	a_12, b_28, c_40, d_52, e_70, f_high

Table 6. Training and test batches for experimental run

Month	Total Examples	Training examples	Test examples
October 1995	840	840	
March 1996	837	558	279
October 1996	848	566	282
March 1997	823	549	274
October 1997	793	529	264

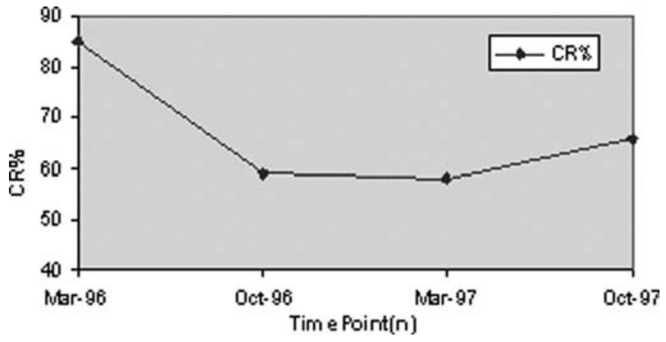

Fig. 1. CR (%) for CD3 with BT data

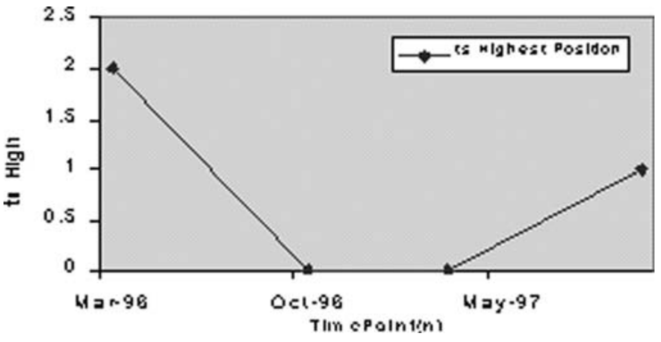

Fig. 2. Highest *ts* position for CD3 with BT data

the beginning of concept drift. CD3 still achieves a good classification rate.

The tree in Table 7 shows that, at this early stage, the *revenue* attribute is the most informative and in some cases the only attribute required in determining 'F&F' users. The acorn attribute then becomes informative. The drift occurs within acorn value acorn_d highlighting a change in the relevance of the *lsind* as highlighted in Table 7.

Table 7. A section of the pruned tree from CD3 after applying second batch

```
revenue
        a12 → n - [40, 735] / 775
        b28 → n - [37, 211] / 248
        c40 → n - [5, 43] / 48
        d52
             acorn
                  acorn_a → n - [1, 2] / 3
                  acorn_b → def – n
                  acorn_c → y - [1, 1] / 2
                  acorn_d
                       ts
                            curr
                                 lsind
                                      ls_a → def – n
                                      ls_b → y - [1, 0] / 1
                                      ls_c → def – n
                                      ls_d → n - [0, 3] / 3
                                      ls_e → def – n
                            new → y - [2, 0] / 2
```

Again with reference to Fig. 2, the position of the *ts* attribute after applying the next two batches: *Oct_96* and *May_97* confirms our suspicions about drift. The *ts* attribute climbs to the top of the tree indicating that all of the data has drifted. This drift continues into the fourth batch: *May_97*.

The percentage of examples being purged as shown in Fig. 3 is measured as a percentage of the current number of examples. This indicates that for the second batch, i.e. *Mar_96*, the drift has not really started, resulting in a very low percentage purged of 0.54%. Again this could be false purging or the beginning of the change. However, the situation becomes clearer with the next two batches. After applying the third batch we see an increase in the percentage being purged to 13%. This increase accelerates to the highest rate of the experiment between the third and fourth batches: *Oct_96* and *May_97* from 13% to 34%.

Following this as with the other findings above, the drift seems to begin to decline after applying the final batch. Although the percentage being purged still increases, it is doing so at a slower rate. We also see that the *ts* attribute moves down to position one in Fig. 2 accordingly.

From the tree in Table 8 it is clear at this final stage that the *revenue* is once again the most informative attribute and that drift is only applicable to the lowest band of revenue, 'a12'. Customers within this *revenue* band and *acorn* categories of either 'acorn_d' or 'acorn_f' previously

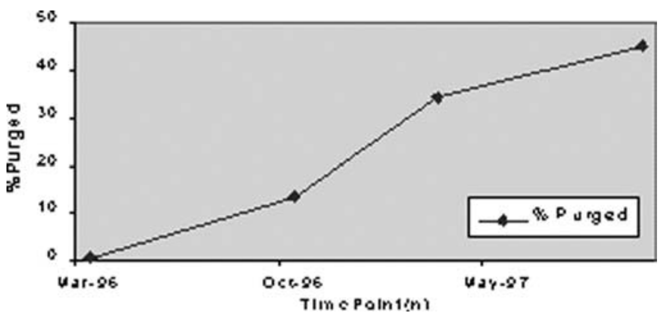

Fig. 3. % Purged for CD3 with BT data

Table 8. A section of the pruned tree output from CD3 after applying final batch

```
revenue
  a12
    ts
      curr
        acorn
          acorn_a
            lsind
              ls_a → def - n
              ls_b → y - [3, 2] / 5
              ls_c → y - [9, 3] / 12
              ls_d
                p_l
                  y → y - [1, 0] / 1
                  n → n - [23, 117] / 140
               ls_e
                o15
                  y → y - [1, 0] / 1
                  n
                    p_l
                      y → y - [1, 0] / 1
                      n → n - [17, 69] / 86
          acorn_d → n - [77, 322] / 399
          acorn_f → n - [39, 205] / 244
      new
       p_l
        y → y - [27, 1] / 28
        n
       o15
        y → y - [8, 1] / 9
        n
        acorn
          acorn_a
            lsind
              ls_a → def - n
              ls_b → y - [2, 2] / 4
              ls_c → y - [5, 3] / 8
          acorn_d
            lsind
              ls_a → n - [2, 4] / 6
              ls_b → y - [2, 1] / 3
              ls_c → n - [6, 7] / 13
              ls_d → y - [39, 31] / 70
              ls_e → y - [6, 5] / 11
          acorn_f
            lsind
              ls_a → n - [0, 2] / 2
              ls_b → y - [8, 7] / 15
              ls_c → n - [2, 3] / 5
              ls_d → y - [8, 3] / 11
              ls_e → y - [11, 10] / 21
```

all had an F&F indicator of 'n': non F&F users (Table 7 has had some of the binary splits of the acorn attribute under the revenue value 'a12' removed to aid readability). However, after applying the final batch: *Oct_97*, these two bands go under major change with almost all becoming F&F users. The reduction in the percentage purged, and presumably the drift is also highlighted by the increase in the classification rate in Fig. 1 to 66%.

4 Conclusion

The experiments confirmed the company's suspicions. Within the twenty-seven week period the profile of customers using the service has changed. The TSAR methodology allowed CD3 to locate the drift and highlight the changing properties within the customer profile.

Looking closely at the percentage being purged in Fig. 3, we can see that, towards the end of the trial, CD3 is purging almost 50% of the data. At this stage 2085 examples out of a total of 2762 have been retained.

It would be interesting is to follow this trial with another few batches after October 1997 to determine if the drift reduces and if so where. Over a longer period it may reduce and then reappear.

By using the TSAR approach the user can analyse the drift at each stage. It is very interesting to study the differences in the knowledge structure between what was current and what is now new. The tree structure offers a very clear and readable interpretation of the drift.

References

1. **Black M, Hickey RJ** (1999) Maintaining the performance of a learned classifier under concept drift, Intelligent Data Analysis **3**: 453–474
2. **Hickey RJ, Black M** (2001) Refined time stamps for concept drift detection during mining for classification rules. In: Roddick K, Hornsby JF (Eds) Spatio-Temporal Data Mining (TSDM 2000). Lecture Notes in Artificial Intelligence 2007. Springer, Berlin Heidelberg New York, pp. 20–30
3. **Hickey RJ** (1996) Noise modelling and evaluating learning from examples, Artificial Intelligence **82**: 157–179
4. **Kelly MG, Hand DJ, Adams NM** (1999) The impact of changing populations on classifier performance. In: Chaudhuri S, Madigan D (Eds) Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA: ACM Press, pp. 367–371
5. **Klenner M, Hahn U** (1994) Concept versioning: A methodology for tracking evolutionary concept drift in dynamic concept systems. In: Cohn AG (Eds) Proceedings of Eleventh European Conference on Artificial Intelligence. Amsterdam, Nertherlands, Wiley, Chichester, England, pp. 473–477
6. **Schlimmer JC, Granger RH** (1986) Incremental learning from noisy data, Machine Learning **1**: 317–354
7. **Hembold DP, Long PM** (1994) Tracking drifting concepts by minimising disagreements, Machine Learning **14**: 27–45
8. **Hulten G, Spencer L, Domingos P** (2001) Mining time-changing data streams. In: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining. San Francisco, CA: ACM Press, pp. 97–106
9. **Widmer G** (1997) Tracking changes through meta-learning, Machine Learning **27**: 259–286
10. **Widmer G, Kubat M** (1996) Learning in the presence of concept drift and hidden contexts, Machine Learning **23**: 69–101
11. **Chakrabarti S, Sarawagi S, Dom B** (1998) Mining surprising patterns using temporal description length. In: Gupta A, Shmueli O, Widom J (Eds) Proceedings of the Twenty-Fourth International Conference on Very Large databases. Morgan Kaufmann, San Mateo, California, pp. 606–661
12. **Chen X, Petrounias I** (1999) Mining temporal features in association rules. In: Zytkow J, Rauch J (Eds) Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases. Lecture Notes in Artificial Intelligence, Vol. 1704. Springer, Berlin Heidelberg New York, pp. 295–300
13. **Quinlan JR** (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California
14. **Quinlan JR** (1998) See5. www.rulequest.com/

15. **Utgoff PE** (1997) Decision tree induction based on efficient tree restructuring, Machine Learning **29**: 5–44
16. **Clark P, Boswell R** (1991) Rule induction with CN2: some recent improvements. In: Kodratof (Eds) Proceedings of the European Workshop on Learning (EWSL-91). Lecture Notes in Artificial Intelligence. Springer, Berlin Heidelberg New York, pp. 151–163
17. **Bratko I** (2001) Prolog programming for artificial intelligence, 3rd edn. Addison-Wesley, Wokingham, England