Australian Government
Department of Defence
Science and Technology

Variable Discretisation for Anomaly Detection using Bayesian Networks

Jonathan Legg

National Security and ISR Division
Defence Science and Technology Group

DST-Group–TR–3328

ABSTRACT

Anomaly detection is the process by which low probability events are automatically found against a background of normal activity. By definition there must be many more normal events than anomalous ones. This rare nature of anomalies causes numerical problems for probabilistic methods designed to automatically detect them. This report describes an algorithm that introduces new discretisation levels to support the representation of low probability values in the context of Bayesian network anomaly detection. It is an engineering solution to a problem with an extant discretisation tool that represents a data set's fine structure but fails to capture extreme values or nulls between modes in its probability density. It is demonstrated that the limitations of the extant tool can be overcome using examples of integer and continuous data.

Published by

National Security and ISR Division
Defence Science and Technology Group
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: *1300 333 362*
Facsimile: *(08) 7389 6567*

AR-016-772
January 2017

APPROVED FOR PUBLIC RELEASE

Variable Discretisation for Anomaly Detection using Bayesian Networks

Executive Summary

Many algorithms exist that take data for a continuous variable, such as latitude, and represent it using a finite number of states. Such an approach is common where the data are to be processed using a Bayesian network, for example, and is known as discretisation. A consequence of discretisation is that regions of the state space that are unlikely to be observed can be very coarsely approximated by conventional algorithms. This is a problem for anomaly detection, where detections stem from low probability data. Because outlying or low probability values are mapped to the same states as more highly probable values, anomalies may go undetected.

This report discusses an algorithm that generates a set of states that ensure that low probability data values can be represented. It does this using the states generated by an external algorithm as a first approximation, along with a summary of the data set of interest. This can be in the form of its unique values, or a histogram for continuous variables, and an estimate of the range of expected values. Detection of low probability regions is catered for in two ways: through a list of expected intervals that permits a quick screening of individual variables, and through new states that need to be realised within the Bayesian network.

The approach is demonstrated through examples of whole number data, a realisation of a Gaussian random variable, and real latitude data.

THIS PAGE IS INTENTIONALLY BLANK

Author

Jonathan Legg
NSID

Jonathan has been with DST Group since 1989. He completed his PhD in signal processing at the University of Adelaide in 1997 and joined the Data and Information Fusion Group in 2000. He assesses sensor fusion systems and conducts research into machine learning for applications including anomaly detection.

THIS PAGE IS INTENTIONALLY BLANK

Contents

Figures

Tables

1 Introduction

Bayesian network implementations usually require each variable to take on a finite number of mutually exclusive states, for example [Norsys 2012]. This means that continuous data sets need to be quantised into a discrete number of states, a process commonly known as discretising. In the context of anomaly detection, of interest are the preprocessing that is required for the data, the values of the states, and how continuous data are represented using them.

In addition to their many well known applications (for example, [Korb & Nicholson 2011]), Bayesian networks (BNs) have been used for anomaly detection [Mascaro, Korb & Nicholson 2010]. This involves learning models of normality based on representative data sets, and flagging data under test that have a low joint probability as 'anomalies'. A prototype BN system based on the work of Mascaro, Korb & Nicholson [2010] processes track information such as the AIS data broadcast by maritime vessels [*Automatic Identification System* 2014]. Because no examples of anomalies are provided to the algorithm, this approach can be thought of as one-class classification, or classification based on examples of only one type of data. An advantage of one-class classification over multi-class classification is that it makes few assumptions about the nature of the anomalies that can be detected.

The discretisation algorithm used by the prototype BN system is the classifier 'Snob' [Wallace & Dowe 2000] whose approach is based on the idea of a description of the data that minimises the information theoretic message length required to convey both an encoding method for the data and the data itself when that encoding method is used [Wallace 1968]. The concepts are discussed in a more general context in [Wallace 2005]. Unfortunately, Snob does a poor job at discretising data when anomalies are to be detected. This report discusses an algorithm that overcomes Snob's limitations.

This report is structured as follows. Section 2 provides a brief overview of discretisation issues, and Section 2.1 discusses a number of approaches considered for addressing these problems. Section 3 discusses variables' intervals and expected ranges based on their identifier and representative data, Section 4 describes the proposed algorithm that introduces states in order to represent low probability data, Section 5 shows examples of the algorithm in use, and Section 6 concludes.

2 Discretisation Approaches and Issues

There are a number of well known discretisation approaches; Liu et al. [2002] provides a useful discussion of the topic. Joint distributions should be taken into account if discretisation is to be performed efficiently [Korb & Nicholson 2011, Section 10.3.1.4], and ideally the discretisation and BN learning steps will be combined. Monti & Cooper [1998] model continuous data as a noisy form of some underlying discrete data process, as with an analogue radio tuner seeking to select a radio station, and jointly estimate the discrete states as they build their Bayesian network. However, it is assumed in this report that variables are discretised independently prior to the BN learning stage so that extant discretisation algorithms such as Snob may still be used.

It is further assumed in this report that a continuous value will be discretised by choosing the state with the closest value. The region of values that gets mapped to a particular state is referred to as its corresponding bin, and depends on the locations of adjacent states.

A discretisation problem is illustrated for two continuous variables in Figure 1. Suppose we have

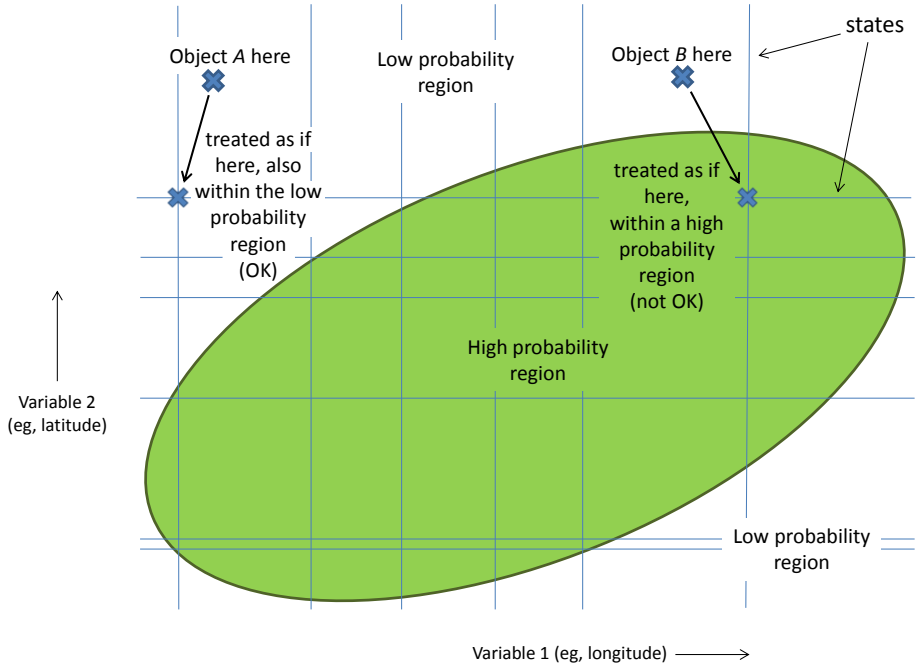

Figure 1: Illustration showing a problem for anomaly detection with discretised data where the states do not allow extreme values to be recognised as such

a high probability geographical region in which the training data are likely to fall (shown in green) and that the variables are highly correlated, meaning that the position of the high probability region along one axis depends on the value along the other axis. Suppose the variables are independently discretised into states represented by the blue lines. If an anomaly occurs such that the data under test fall outside the region spanned by the high probability training data, the discretisation is likely to be acceptable if they are mapped into a region that also has a low joint probability, such as on the left (A). However, when the states do not extend beyond the high probability data set, the anomaly can be mapped into a region with a high joint probability, and thereby be indistinguishable from normal data, and so not be detected as an anomaly (B). This is unacceptable.

Figure 2 shows a histogram of a data set using 50 equally spaced bins (represented by transparent bars) to contrast it with the histogram that results when the same data set is discretised using the states represented by vertical red lines. These latter states were chosen by an available classification algorithm (Snob). Note that the tails and the low probability region between -34.0 and -33.85, evident when 50 bins are used, are not evident in the histogram that results from the Snob discretisation.

An extract of the time series data corresponding to Figure 2 is shown in Figure 3. Although the extant discretisation algorithm chose a set of discretisation states that capture the structure of the high probability data, it failed to reasonably discretise the data using them. The original data are shown in blue, Snob's discretisation states are shown as grey horizontal lines, Snob's discretisation is shown in red and the closest states are shown in green. This issue of failing to correctly use discretisation bins is also addressed by the implemented algorithm.

Complicating discretisation issues include

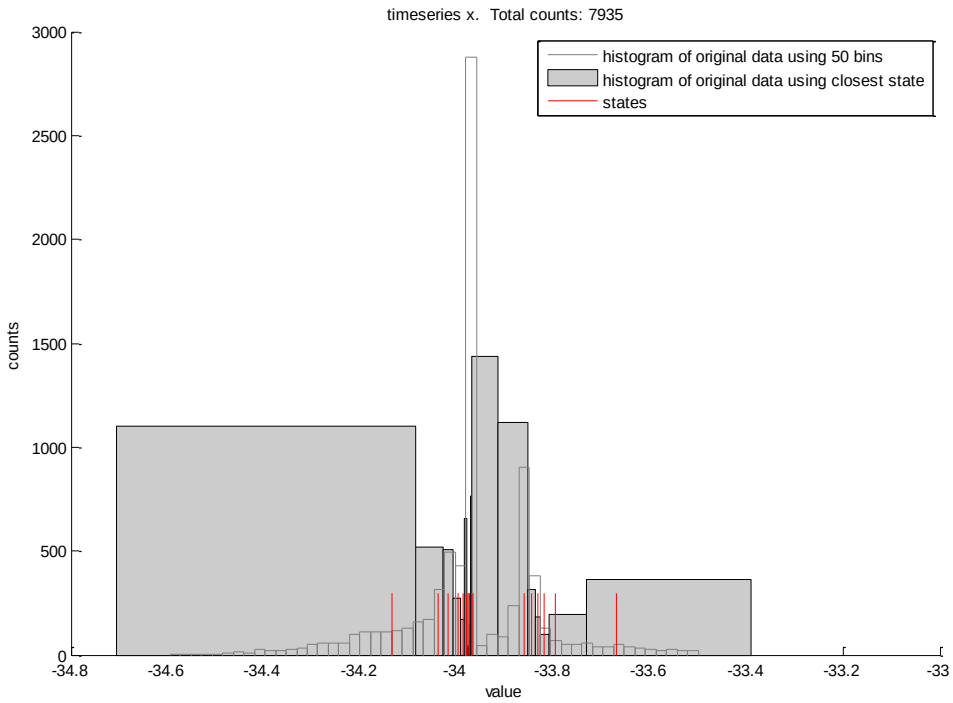

Figure 2: Example of a continuous variable poorly discretised for anomaly detection purposes

1. **The domain of the continuous data.** Discretisation approaches generally assume that the domain of the continuous data is limitless. This is inappropriate for many variables, including circular variables such as days of the week, hours of the day and angles [Pyle 1999], and nonnegative variables such as speed. This means that there are opportunities for efficient discretisation being lost.

2. **Allowing for missing and 'empty' data.** This is an issue where the algorithm that takes the discrete data to build a Bayesian network requires that all entries in the data set have an assigned value.

   Data may be missing because it isn't provided. The prototype BN system uses the number 511 for unknown vessel headings, for example (which is a fast but poor approach, as this will alter the heading statistics; alternatives are discussed by Pyle [1999]).

   An entry may be 'empty' when it isn't applicable, for example where a vessel type doesn't match any of the definitions. This case is also discussed by Pyle [1999].

3. **Low probability discrete values.** We assume that data such as days of the week or the names of teams involved in a sporting competition are arbitrarily enumerated. (The specific ordering may affect the quantitative performance, but should not affect the qualitative performance.) We need to guarantee that low probability discrete data are captured in order to detect anomalies. For example, suppose that the data used to represent normality only incorporated weekday measurements taken on Wednesday (3) and Thursday (4) with equal probability. An extant discretisation algorithm may represent the day of the week using a single state with value 3.5. (After all, it has seen no examples of low probability data that need to be distinguished from

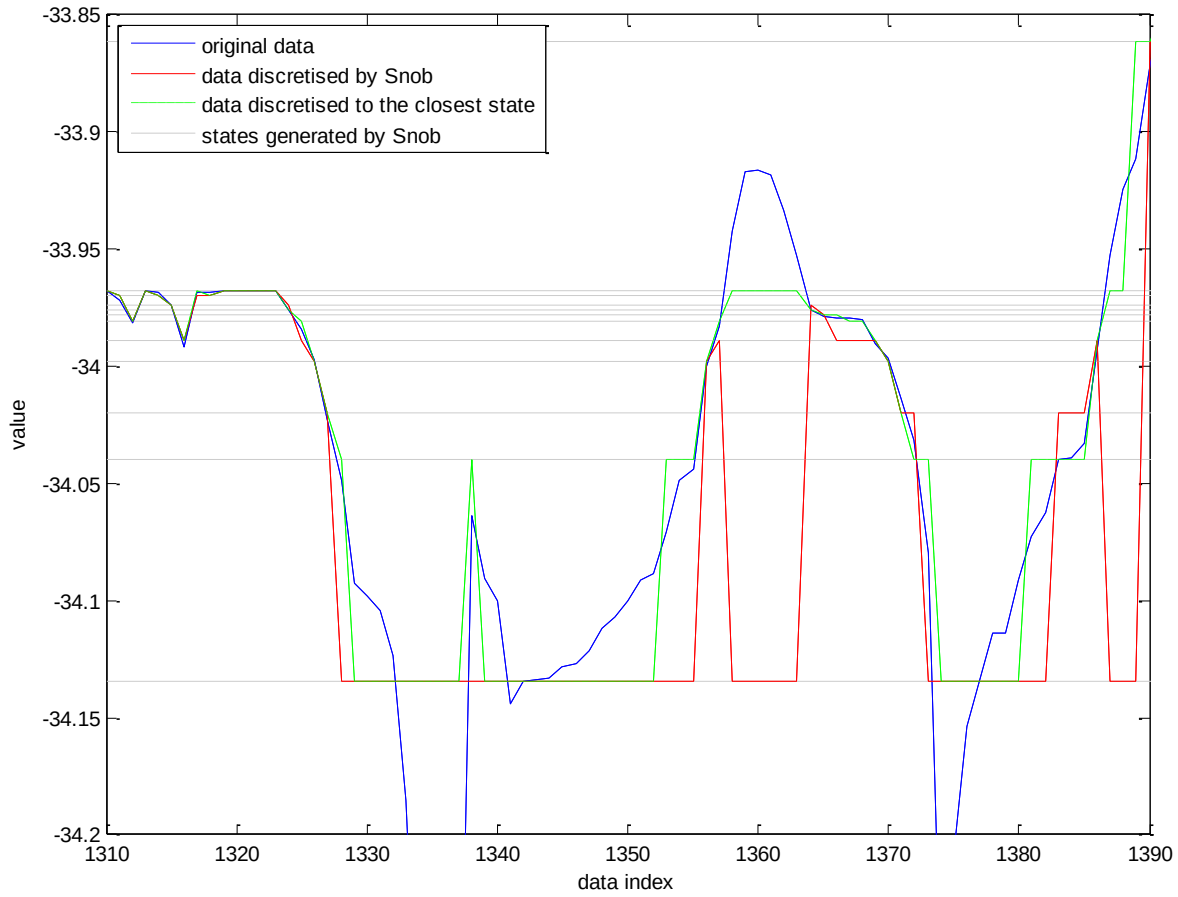

Figure 3: Demonstration of a discretisation bug where data are not being quantised according to the closest state. The red and green lines should always coincide

these high probability days.) This means that test data corresponding to Friday (5) are mapped to 3.5, exactly the same state as used for the data corresponding to Wednesday, and the data are not flagged as anomalous. A mechanism to distinguish the days prior to Wednesday and later than Thursday from the days Wednesday and Thursday is required[1].

2.1 Potential Solutions

A number of approaches have been identified to address the discretisation issues outlined in Section 2. Progressing from the simplest to the most complex solution, these are:

1. **Add states to capture extreme values.** That is, add two more states in most cases.

   The disadvantage with this approach is the additional size of the BN potentials for no reason other than to capture extremely low singleton probabilities. BNs are appropriate when they use the relationships between variables to model joint probabilities. Thus this is not an efficient use of computing power.

2. **Test each variable outside the BN.** Because the power of a BN to efficiently work with joint distributions is not needed when individual variables are tested for anomalies, the approach here is to use the existing discretisation and BN, and have a separate test for each variable.

   This is a computationally efficient solution that keeps the discretiser unchanged.

3. **Discretise with explicit bin limits.** Rather than having every continuous value map to the nearest state, with this approach the bins have explicitly specifed, mutually exclusive limits. Values that have low probability fall outside all bins and are discretised using an 'other' state. Thus the additional complexity is in the discretisation process rather than the complexity of the BN. This would be appropriate for distributions that have many modes, each of which is well localised. For example, although a vessel may spend some time tied up at a dock, it also spends time approaching it or leaving it. If the dock were represented using a bin, the anomaly detection process would need to avoid labelling the vessel an anomaly when it is legitimately transiting, perhaps through the use of the joint distribution. If it were to berth in a different dock, that may be anomalous.

4. **Add states that capture nulls between modes.** For a multimodal distribution, such as the one represented by the histogram in Figure 2, states can be introduced that guarantee that low probability values are separated from high probability values.

5. **Discretise with an anomaly detection BN in mind.** This is the optimal single variable discretisation solution. It is probable that there are gains if the discretisation criterion incorporates constraints or weights associated with anomaly detection. For a given detection performance there should be fewer states than using an extant discretising algorithm and then combining options 1 and 4. This is an advantage when learning BN structure and potentials from data: the fewer the number of unknowns to be estimated, the less data that is required in order to achieve a given uncertainty in the result.

   An appealing approach for large data sets is that of a Dirichlet process (for example, [Barber 2011, Section 20.5], [Murphy 2012, Section 25.2.2]), where new discretisation levels are added

[1] This example is shown in row 1 of Table 3.

as required in order to adequately represent the data, although a modification may be required to accommodate the representation of low probability data.

The solution adopted here is a combination of approaches 2 and 4: it explicitly checks the limits of individual variables (Section 3) and leverages the output from an extant discretisation algorithm such as Snob, only providing new states where they are necessary to distinguish anomalies (Section 4).

3 Expected Intervals and Ranges

Expected intervals are used to check if variables are outside their normal values. Table 1 has a representative selection of variables from an example data set and their expected ranges. $\min(x)$ and $\max(x)$ are the minimum and maximum data values respectively, $\mathrm{minRange}$ is defined as

$$\mathrm{minRange}(x) = \mathrm{mean}(x) - [\mathrm{mean}(x) - \min(x)] \times \mathit{interval\ gain}$$

and $\mathrm{maxRange}$ as

$$\mathrm{maxRange}(x) = \mathrm{mean}(x) + [\max(x) - \mathrm{mean}(x)] \times \mathit{interval\ gain}$$

where $\mathit{interval\ gain}$ allows for $\mathrm{minRange}$ and $\mathrm{maxRange}$ to confidently span the likely values for the variable. The interval gain was chosen to be 1.25.

The issue of out-of-range values is discussed by Pyle [1999] in the contexts of training, testing and executing a machine learning approach where data are required to be in the range 0–1, say. One mechanism for accommodating unusual values is to scale the data nonlinearly so large values do not cause saturation: although the majority of the range is used to accommodate expected values, larger values prior to scaling always remain larger values following scaling.

Table 1: Examples of representative ranges

Variable name identifier	Type	Lower limit	Upper limit
ends in ‘pc’, i.e. percentage	real	0.0	100.0
ends in ‘sd’, i.e. standard deviation	real	0.0	maxRange
ends in ‘rate’, e.g. ‘headingChangeRate’	real	minRange	maxRange
starts with ‘is’, e.g. ‘isWeekDay’	boolean	0	1
starts with ‘num’, e.g. ‘numLocalInteractions’	whole	0	maxRange
ends with ‘id’, i.e. identifier	whole	minimum	maximum
‘day’ and ‘week’, e.g. ‘daysInAWeek’	whole	0	6
‘month’	whole	0	11

4 Algorithms to Introduce States that Capture Probability Transitions

The proposed algorithm is discussed in the context of whole number variables, but it is also applicable to continuous variables as described in Section 4.1. The first stage of the algorithm is of the 'splitting' variety [Liu et al. 2002], where a set of data is divided up through the introduction of new states.

Possible values are a set of values that span the expected range of a variable. Where the variable has discrete possibilties, such as the vessel types 'liner', 'tug' and 'yacht', the possible values are the unique data in the training set. Where the variable values are whole numbers, the possible values are given by all values in the range. It is useful for data in known sequences to be represented as ordinal variables [Barber 2011, Section 8.1.2] rather than strings; for example, using day numbering such as Monday $= 1$ rather than naming the days of the week. Where the variable is continuous, the possible values may be given by the locations of an appropriate number of histogram bins, for example.

Table 2 illustrates the getStatePass algorithm using a day of the week example. The possible values (PV) for the day of the week are 0–6 (see Table 1). Suppose the data only uses the values 2, 3 and 5 (column 2) and we are provided with the states 0, 2.5, 4 and 5 by an extant discretising algorithm such as Snob. The algorithm is shown in Algorithm 1 (parts 1 and 2).

1. For each possible value the algorithm asks, 'does this possible value have a corresponding data point?' and records the result in column 3 of Table 2. This is shown in Algorithm 1, lines 5–8.
2. It then maps the possible values to the nearest of the states it is working with, as shown in column 5 (Algorithm 1, lines 9–12).
3. For each state, the closest of the possible values is then recorded (lines 13–15). This is to ensure that transitions across the high/low probability threshold occur in the correct positions for continuous variables (see Section 4.1).
4. It then notes the consistency with the presence of data at each of the states' mapped possible values (final column in Table 2; Algorithm 1, lines 16–54). If there is data at each, or there is no data at each, then the state's bin is doing its job of keeping together data with similar probabilties. If there is data at any but not all, then a new state is required to separate the corresponding mapped possible values. In this way, unseen data will be treated differently from data in the training set, assisting with the detection of anomalies. The approach used to generate

Table 2: Day of the week example. Monday is represented by 1, and data are present for Tuesday, Wednesday and Friday

Possible values	Unique data points	Data at the PVs	Extant states	PVs mapped to states	Data at the possible values for these states
0		no	0	0,1	no, no: same – OK
1		no			
2	2	yes	2.5	2,3	yes, yes: same – OK
3	3	yes			
4		no	4	4	only one – OK
5	5	yes	5	5,6	yes, no: different – not OK
6		no			

Algorithm 1 The getStatePass algorithm, part 1

```
1:  procedure GETSTATEPASS(uniqueData, possibleVals, currentStates)
2:      if |currentStates| = 0 then          ▷ Trivial case: provide first possible value as a state
3:          return false, possibleVals_0
4:      end if

5:      PVsWithData ← ∅                      ▷ Occurrence of data at the possible values
6:      for all u ∈ uniqueData do
7:          PVsWithData ← PVsWithData ∪ closestValue(u, possibleVals)
8:      end for

9:      mpv[b] ← ∅ ∀b ∈ currentStates        ▷ Map the possible values to the current states
10:     for all v ∈ possibleVals do
11:         mpv[closestValue(v, currentStates)] ← mpv[closestValue(v, currentStates)] ∪ v
12:     end for
13:     for all b ∈ mpv do                   ▷ Add the closest possible value
14:         mpv[b] ← mpv[b] ∪ closestValue(b, possibleVals)
15:     end for
```

a new state is discussed below.

Within Algorithm 1 part 2 we loop around the mapped possible values using pv_l and pv_r for 'possible value – left' and 'possible value – right' respectively. When a state is added the procedure returns in preparation for another iteration.

The $\mathrm{minDistance}(v, \mathbf{x})$ function trivially returns the minimum distance between v and the closest value in $\mathbf{x}$, and $\mathrm{closestValue}(v, \mathbf{x})$ returns the corresponding value in $\mathbf{x}$.

We generate a new state as shown in Figure 4. In order for the new set of states to separate pv_l and pv_r, the midpoint m between the closest existing state c and the new state n must lie between pv_l and pv_r. This is satisfied when

$$pv_l < m = \frac{c+n}{2} < pv_r,$$

or

$$pv_l + (pv_l - c) < n < pv_r + (pv_r - c).$$

We could choose

$$n = pv_m + (pv_m - c)$$

where pv_m is the midpoint between pv_l and pv_r, except that this often creates states with ambiguous distances (for example, 3 and 5 are equally distant from 4). Instead of the midpoint we can interpolate between pv_l and pv_r by fraction f according to

$$pv_f = pv_l + f(pv_r - pv_l).$$

$f = 0.51$ tends to result in rounding problems (for example, 3.01999999 instead of 3.02), but $f = 0.625$ appears to work well most of the time.

Unfortunately there is a finite probability that we will choose a new state that corresponds to an existing state: it is possible that $pv_l \leq c \leq pv_r$, although $pv_l \neq pv_r$. To accommodate this we adopt

Algorithm 1 cont.d The getStatePass algorithm, part 2

```
16:     OK ← true
17:     newStates ← sort(currentStates)
18:     for all b ∈ newStates do
19:         firstStatePV ← true
20:         for all pv_r ∈ sort(mpv[b]) do                    ▷ Working to the right
21:             if pv_r ∈ PVsWithData then
22:                 dataHere_r ← true
23:             else
24:                 dataHere_r ← false
25:             end if
26:             if firstStatePV then                          ▷ This will happen for the first pass:
27:                 firstStatePV ← false
28:                 dataHere_l ← dataHere_r                   ▷ initialises dataHere_l
29:             else
30:                 if dataHere_r ≠ dataHere_l then           ▷ Data presence changed for these values?
31:                     newStateStrategy ← 'aggressive'
32:                     while true do                         ▷ Need to create a state
33:                         n ← CreateNewState(pv_l, pv_r, newStates, newStateStrategy)
34:                         if minDistance(n, newStates) = 0 then     ▷ Check for existing state
35:                             if newStateStrategy = 'aggressive' then
36:                                 newStateStrategy ← 'moderate'
37:                             else if newStateStrategy = 'moderate' then
38:                                 newStateStrategy ← 'conservative'
39:                             end if
40:                         else
41:                             newStates ← newStates ∪ n     ▷ Accept new state
42:                             break
43:                         end if
44:                     end while
                                ▷ A new state has been placed: exit procedure and try again
45:                     OK ← false                ▷ Indicate that currentStates are unacceptable
46:                     break
47:                 end if
48:             end if
49:             pv_l ← pv_r
50:         end for
51:         if OK = false then
52:             break
53:         end if
54:     end for
55:     return OK, newStates
56: end procedure
```

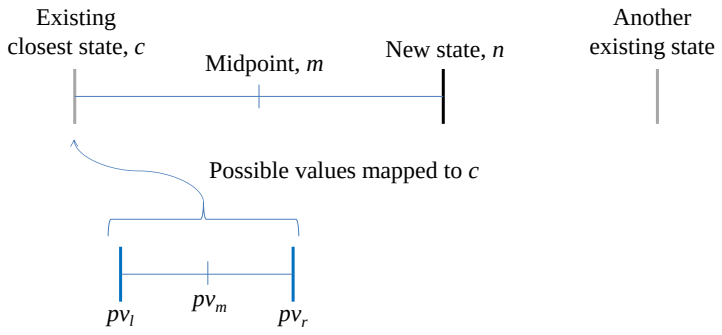

Figure 4: A new state is created to separate pv_l from pv_r

a three stage state assignment strategy that varies from 'aggressive' ($f = 0.625$) through 'moderate' ($f = 0.5001$) to 'conservative', where f is drawn from a uniform distribution.

The new state generation process is shown in Algorithm 2.

Once a new state is created, getStatePass (Algorithm 1) returns, and is called again by getNewStates until no more new states are required. This higher level function is shown in Algorithm 3.

The createNewState function (Algorithm 2) is a local solution that does not take the global set of possible values or existing states into account. Consequently the set of states available from line 7 in Algorithm 3 may contain redundancy. The pruning functions referred to in Algorithm 3 lines 8–10 remove states that are redundant given the existance of other states, but will not remove any states in the original set. 'Pruning' in this context is analogous to 'merging', or a bottom-up discretisation approach, as discussed in [Liu et al. 2002]. pruneLeftBins, shown in Algorithm 4, starts from the minimum state value, removing states that result in getStatePass returning OK. It stops trying states when getStatePass returns 'false'. pruneRightBins and pruneOtherBins, not explicitly shown, try

Algorithm 2 The createNewState function

1: **function** CREATENEWSTATE($pv_l, pv_r, states, newStateStrategy$)
2: **if** $newStateStrategy =$ 'aggressive' **then** ▷ Need the fraction offset; see text
3: $f \leftarrow 0.625$
4: **else if** $newStateStrategy =$ 'moderate' **then**
5: $f \leftarrow 0.5001$
6: **else**
7: $f \overset{\text{draw}}{\leftarrow} U(0, 1)$
8: **end if**
9: $pv_f \leftarrow \text{interpolate}(pv_l, pv_r, f)$
10: $c \leftarrow \text{closestValue}(\text{mean}(pv_l, pv_r), states)$
11: **return** $pv_f + (pv_f - c)$
12: **end function**

Algorithm 3 The getNewStates algorithm

```
1:  function GETNEWSTATES(uniqueData, possibleVals, currentStates, allowStateDeletion)
2:      originalStates ← currentStates

3:      OK ← false
4:      while OK = false do
5:          OK, currentStates ← getStatePass(uniqueData, possibleVals, currentStates)
6:      end while
7:      newStates ← sort(currentStates)

8:      newStates ← pruneLeftBins(uniqueData, possibleVals, newStates, originalStates)
9:      newStates ← pruneRightBins(uniqueData, possibleVals, newStates, originalStates)
10:     newStates ← pruneOtherBins(uniqueData, possibleVals, newStates, originalStates)

11:     if allowStateDeletion & (|newStates| > |possibleVals|) then
12:         newStates ← possibleVals          ▷ Guaranteed to distinuish between the possible values
13:     end if

14:     return newStates
15: end function
```

removing states from the right, and then all of the rest, respectively.

If it is acceptable to delete states in the extant set, and the current number of states is at least equal to the number of possible values, then the possible values themselves may be used as the discretising states (Algorithm 3, lines 11–13).

Algorithm 4 The pruneLeftBins algorithm; pruneRightBins and pruneOtherBins are similar

```
1:  function PRUNELEFTBINS(uniqueData, possibleVals, currentStates, originalStates)
2:      OK ← true
3:      while OK = true & (newStates_0 ∉ originalStates) do
4:          testStates ← newStates \ newStates_0
5:          if |testStates| > 0 then
6:              OK, dummyStates ← getStatePass(uniqueData, possibleVals, testStates)
7:          else
8:              break
9:          end if
10:         if OK = true then
11:             newStates ← newStates \ newStates_0
12:         end if
13:     end while
14:     return newStates
15: end function
```

4.1 Continuous Data

To use this algorithm in the context of continuous data sets, a histogram with an appropriate number of bins is used (although other approaches are possible). The range of the bins is the expected range of values as discussed in Section 3. The possible values are given by the bin locations, and the unique data values are given by the locations of the histogram bins that exceed a chosen threshold. For anomaly detection, a threshold of $0.05 \times$ the number of histogram samples may be appropriate (that is, corresponding to the lower 5% probability). An example of this is shown in Figure 5, where there were no initial states.

At values of the histogram near the threshold there is a transition interval where the values may or may not exceed the threshold. This distinction is captured by the new states. The numbers above each of the states in the figure show the number of calls to $\mathrm{getStatePass}$ needed to generate the corresponding state. Most of the introduced states were removed during the pruning process. The most aggressive interpolation strategy was used throughout this process.

Note that lines 13–15 in Algorithm 1 add the appropriate closest possible value to the list of PVs mapped to each of the states. Without this, new states may not be added to correctly capture all high/low probability transitions when there is more than one state close to a PV. An example of this

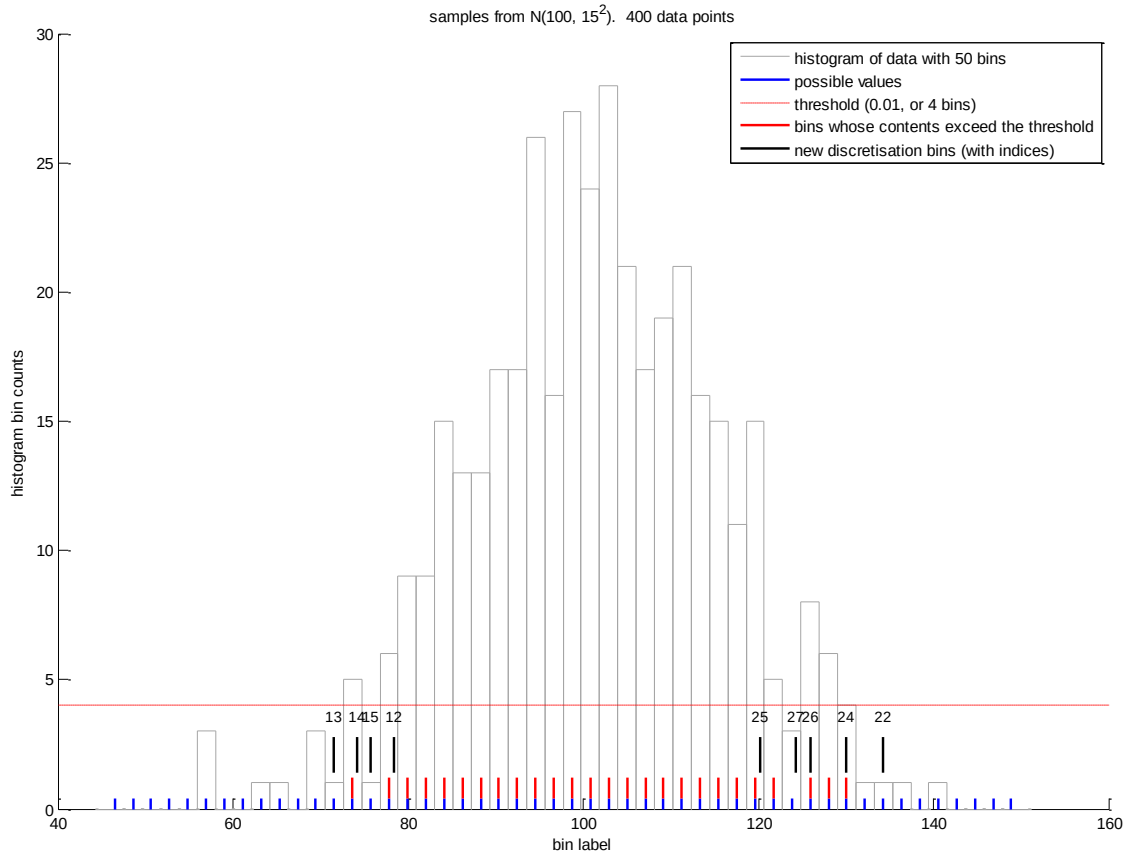

Figure 5: Demonstration of the state generation algorithm operating on data drawn from a Gaussian distribution

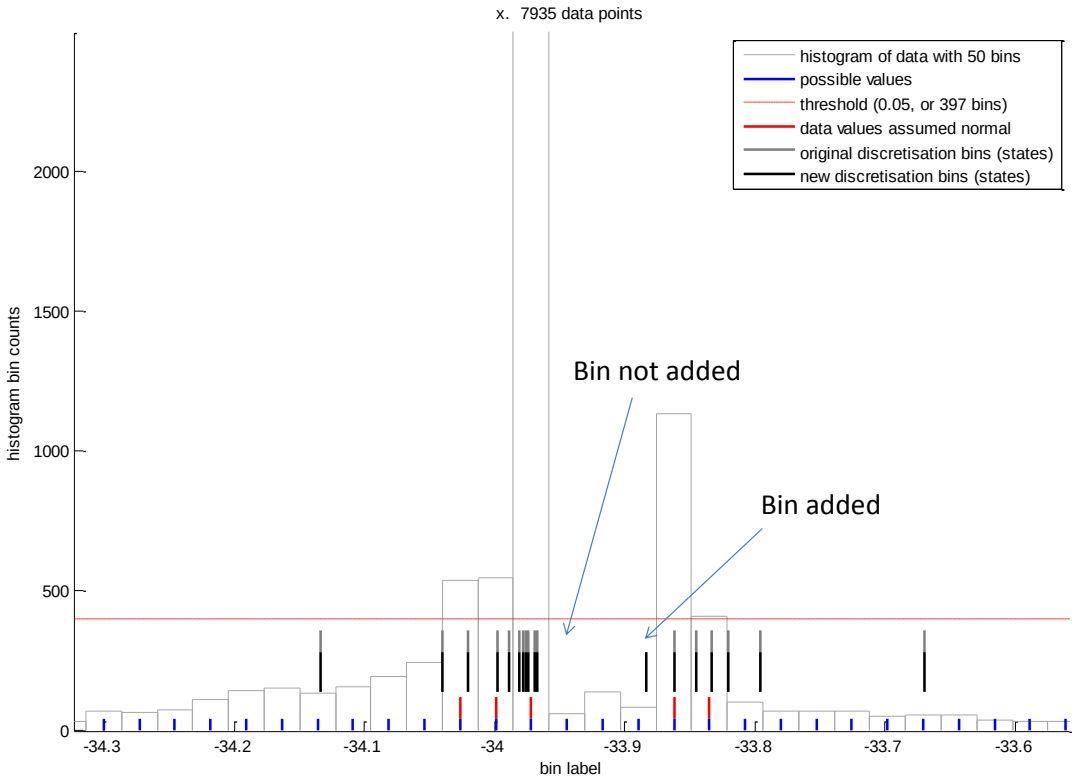

Figure 6: Demonstration of a high/low probability transition that is not captured with a bin

is shown in Figure 6 using the continuous data introduced in Figure 2.

5 Examples

See Table 3 for examples of suitable states for representing different unique data values for the variable 'daysInAWeek'. The range of possible values is 0–6. In the first example, the data represents Wednesday and Thursday and there is a state between them. If anomalous data occurred, such as a Tuesday or Friday, these would be treated as if they were a Wednesday or Thursday and the anomaly would go undetected. This problem is addressed through the introduction of the new states which distinguish Tuesday from Wednesday and Friday from Thursday. The original state is maintained. The other examples behave similarly.

Figure 7 shows the histogram from Figure 2 with states added that successfully capture the region of low probability. The leftmost added state is difficult to distinguish from one of the extant states in the figure. The histogram of the original continuous data set discretised using the new set of states is given in Figure 8. The low probability region is clearly visible, but was missed in Figure 2.

Table 3: Examples of algorithm output for whole number variable 'daysInAWeek'.

Data	Initial states	New states
[3, 4]	[3.5]	[1.75, 3.5, 5.75]
[2, 3, 5]	[0,2.5,4,5]	[0, 2.5, 4, 5, 6.25]
[2, 3, 5]	[]	[0, 3.25, 4, 5.25, 6]
[0]	[0]	[0, 1.25]
[1, 4, 6]	[1, 4, 6]	[0.25, 1, 2.25, 4, 5.25, 6]
[1, 4, 5, 6]	[1, 4, 6]	[0.25, 1, 2.25, 4, 6]

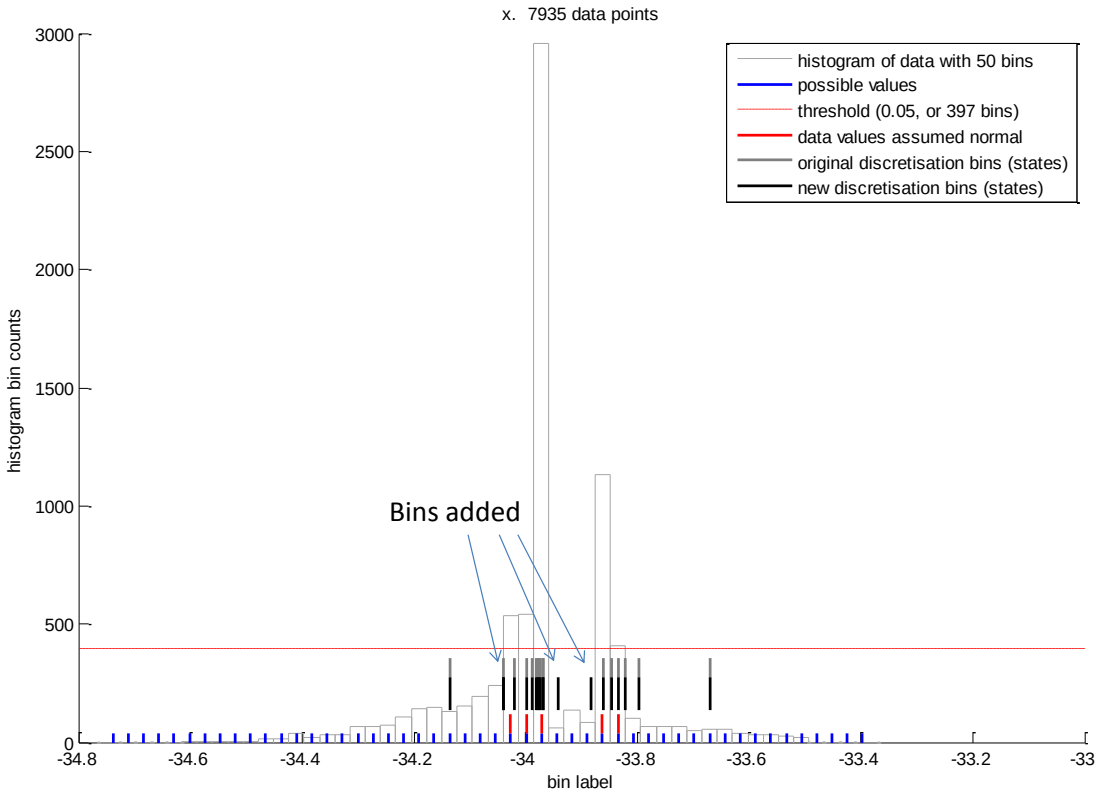

Figure 7: Demonstration of the state generation algorithm operating on the x data

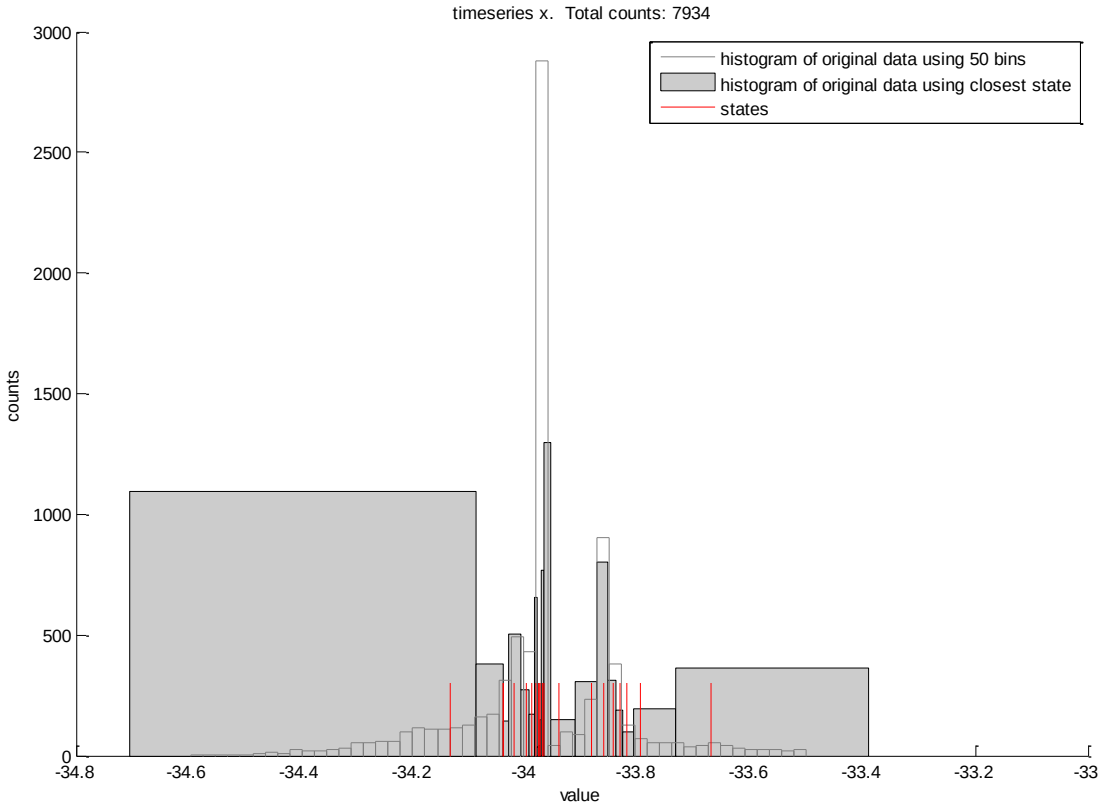

Figure 8: The x data discretised according to the new states. The low point in the histogram has been captured

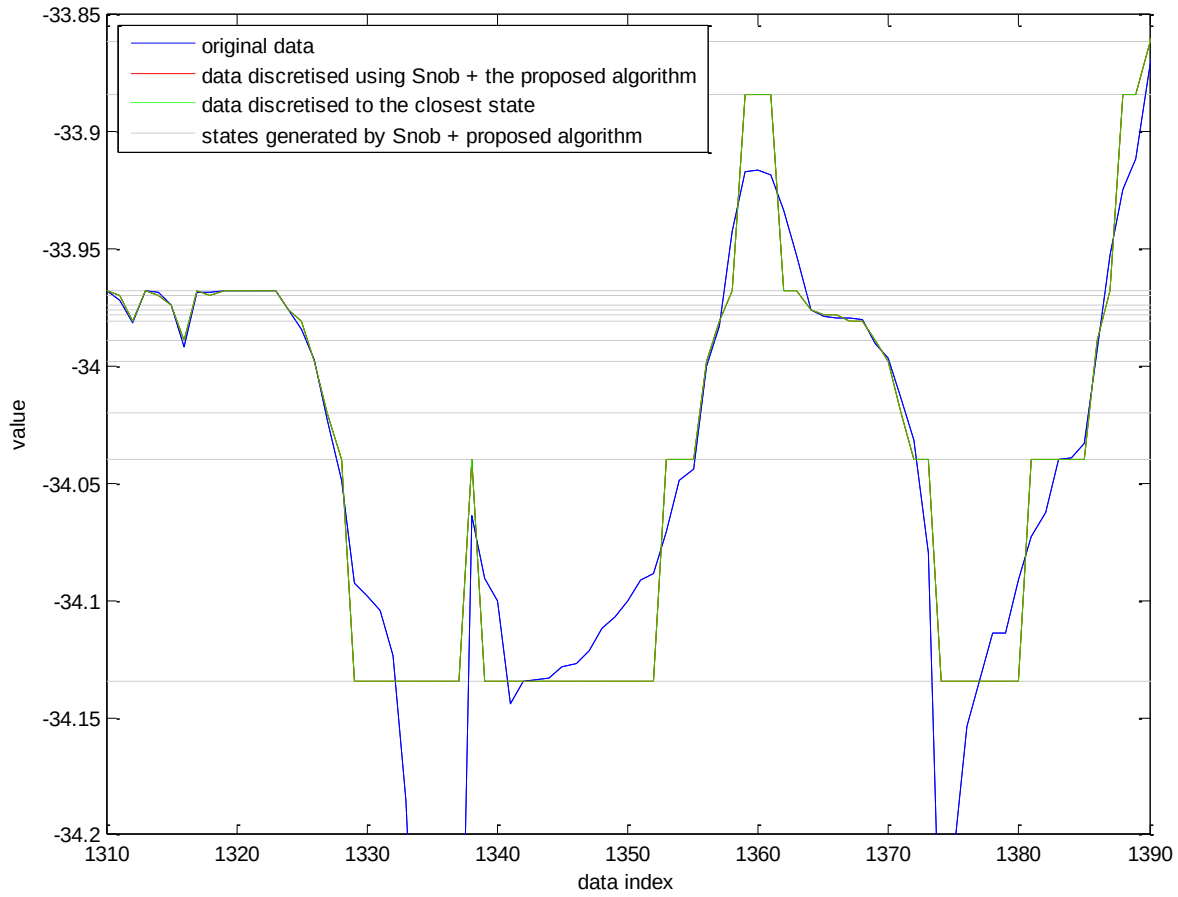

Figure 9: The x data discretised correctly. The green lines are on top of the red lines

Figure 9 shows the x data rediscretised according to the original Snob states and new states as generated by the proposed algorithm. The 'data discretised using Snob + the proposed algorithm' and 'data discretised to the closest state' lines are coincident, unlike in Figure 3, and utilise the additional state shown at the top of the figure, which is absent in Figure 3.

6 Conclusion

An engineering solution to the problem of variable discretisation for anomaly detection has been described that augments extant discretisation approaches by adding new discretisation bins as required. This permits anomaly detection algorithms to detect low probability events that would otherwise have been confused with normal activities. The solution was demonstrated using examples of synthetic and real data.

Acknowledgements

The author acknowledges the advice provided by Prof. Murray Loew of George Washington University, Dr. Sam Davey and other members of Data and Information Fusion Group.

7 References

Automatic Identification System (2014) Wikipedia. URL – http://en.wikipedia.org/wiki/Auto-matic_Identification_System. Accessed 22 Sep 2014.

Barber, D. (2011) *Bayesian Reasoning and Machine Learning*, Cambridge University Press. In press.

Korb, K. B. & Nicholson, A. E. (2011) *Bayesian Artificial Intelligence*, second edn, Chapman and Hall/CRC, London EC2A 4BQ.

Liu, H., Hussain, F., Tan, C. L. & Dash, M. (2002) Discretization: An enabling technique, *Data Mining and Knowledge Discovery* **6**(4), 393–423.

Mascaro, S., Korb, K. & Nicholson, A. (2010) *Learning Abnormal Vessel Behaviour from AIS Data with Bayesian Networks at Two Time Scales*, Technical report 2010/4, Bayesian Intelligence, Clayton, Victoria, Australia. URL – http://bayesian-intelligence.com/publications.php.

Monti, S. & Cooper, G. F. (1998) A multivariate discretization method for learning Bayesian networks from mixed data, *in UAI98 — Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.

Murphy, K. P. (2012) *Machine Learning: A Probabilistic Perspective*, The MIT Press, Cambridge, Massachusetts, USA, London, England.

Norsys (2012) Netica, Norsys Software Corporation. URL – http://www.norsys.com.

Pyle, D. (1999) *Data Preparation for Data Mining*, Morgan Kaufmann Publishers, Inc, 340 Pine Street, Sixth Floor, San Francisco, CA, USA.

Wallace, C. (1968) An information measure for classification, *Computer Journal* **11**(2), 185–194. URL – http://www.allisons.org/ll/MML/Structured/1968-WB-CJ/.

Wallace, C. (2005) *Statistical and Inductive Inference by Minimum Message Length*, Springer, USA.

Wallace, C. S. & Dowe, D. L. (2000) MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions, *Statistics and Computing* **10**(1), 73–83. URL – http://www.springerlink.com/content/j5875v19k11941v8/.

THIS PAGE IS INTENTIONALLY BLANK

DEFENCE SCIENCE AND TECHNOLOGY GROUP DOCUMENT CONTROL DATA	1. DLM/CAVEAT (OF DOCUMENT)

2. TITLE	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Variable Discretisation for Anomaly Detection using Bayesian Networks	Document (U) Title (U) Abstract (U)
4. AUTHOR	5. CORPORATE AUTHOR
Jonathan Legg	Defence Science and Technology Group PO Box 1500 Edinburgh, South Australia 5111, Australia

6a. DST Group NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DST-Group–TR–3328	016-772	Technical Report	January 2017

8. Objective ID	9. TASK NUMBER	10. TASK SPONSOR
fAV1169775	AIR 07/324	CDRSRG

13. DST Group Publications Repository	14. RELEASE AUTHORITY
http://dspace.dsto.defence.gov.au/dspace/	Chief, National Security and ISR Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

No limitations. Approved for public release.

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111

16. DELIBERATE ANNOUNCEMENT

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. RESEARCH LIBRARY THESAURUS

Bayesian networks, machine learning

19. ABSTRACT

Anomaly detection is the process by which low probability events are automatically found against a background of normal activity. By definition there must be many more normal events than anomalous ones. This rare nature of anomalies causes numerical problems for probabilistic methods designed to automatically detect them. This report describes an algorithm that introduces new discretisation levels to support the representation of low probability values in the context of Bayesian network anomaly detection. It is an engineering solution to a problem with an extant discretisation tool that represents a data set's fine structure but fails to capture extreme values or nulls between modes in its probability density. It is demonstrated that the limitations of the extant tool can be overcome using examples of integer and continuous data.