Adv Data Anal Classif (2017) 11:353–370
DOI 10.1007/s11634-016-0246-x

REGULAR ARTICLE

T3C: improving a decision tree classification algorithm's interval splits on continuous attributes

Panagiotis Tzirakis[1] · **Christos Tjortjis**[2]

Received: 16 December 2014 / Revised: 4 April 2016 / Accepted: 11 April 2016 /
Published online: 27 April 2016

Abstract This paper proposes, describes and evaluates T3C, a classification algorithm that builds decision trees of depth at most three, and results in high accuracy whilst keeping the size of the tree reasonably small. T3C is an improvement over algorithm T3 in the way it performs splits on continuous attributes. When run against publicly available data sets, T3C achieved lower generalisation error than T3 and the popular C4.5, and competitive results compared to Random Forest and Rotation Forest.

Keywords Data mining · Classification · Decision trees · Interval splits

Mathematics Subject Classification 68T05

1 Introduction

Classification produces a function that maps a data item into one of several predefined classes, by inputting a training data set and building a model of the class attribute, based on the rest of the attributes. Decision Trees is a classification method with intuitive nature which matches the users' conceptual model without loss of accuracy (Berry and Linoff 2004). However, no clear winner exists (Tjortjis and Keane 2002) amongst decision tree classifiers when taking into account tree size, classification and generalisation accuracy. This work focuses on reducing generalisation error, which is the

✉ Christos Tjortjis
c.tjortjis@ihu.edu.gr

[1] Department of Computer Science, University of Crete, Voutes Campus, 700 13 Heraklion, Crete, Greece

[2] School of Science and Technology, International Hellenic University,
14th km Thessaloniki-Moudania, 57001 Thermi, Greece

number of instances that have been misclassified in the test set. We focus on reducing the generalization error because this allows for better prediction, i.e. classification of unseen cases. We are more interested in generalization error because we want to see how the algorithm performs in new, unseen, data, rather than improving classification accuracy by overfitting the data (Murthy and Salzberg 1995).

This paper introduces T3C, which on average outperforms the popular algorithm C4.5 (Quinlan 1993, 1996) as well as T3 (Tjortjis and Keane 2002; Tjortjis et al. 2007), whilst improving T3's performance, even when compared with state-of-the-art meta classifiers such as Rotation Forest (Rodriguez et al. 2006) or advanced decision tree classifiers like Random Forest (Breiman 2001). T3C builds small decision trees with depth at most 3. The main difference between T3C and T3 is that T3C splits continuous attributes with four nodes instead of three.

In the remaining of the paper C4.5, T3, Rotation Forest and Random Forest are briefly described in Sect. 2, including a comparison of their performance, as described in the literature, enriched by new experiments. T3C is detailed in Sect. 3, along with key concepts on how T3C differs from T3. Experimental results are presented in Sect. 4 and discussed/evaluated in Sect. 5. Finally conclusions and directions for future work are given in Sect. 6.

2 Background

Classification is a method aiming at classifying records into one of the many classes which are predefined (Hubert and Van der Veeken 2010; Mozharovskyi et al. 2015). Given a well defined number of classes and a set of pre-classified samples, classification aims at creating a model that can be used for classifying future unknown data. More precisely, classification can be described as a function of two steps (Witten et al. 2011):

- Step 1: Learning. In this step a model is built which describes a predefined set of classified data. The training data are being used by a classification algorithm in order to build the model. The records of the training set are selected using the holdout method; given data are randomly partitioned into two independent sets: the training set (normally 2/3 of the records) for model construction and the test set (normally 1/3 of the records) for accuracy estimation. The model which is determined is known as classifier and is represented by classification rules, decision trees or mathematical formulae.
- Step 2: Classification. In this step, test data belonging to known in advance classes are used in order to calculate the accuracy of the model. There are several methods to assess the accuracy of a classifier. Training data are chosen randomly and independently. The model classifies each one of the training samples. Afterwards, using the test data, the class that data belong to, is compared to the class predicted by the model.

  Note that the classification accuracy of the model is the percentage of the sample data that were classified correctly by the classifier. Generalization accuracy is the number of correct instances divided by the total number of instances in the test set.

Generalization (and similarly classification) error is defined as follows:

$$generalization\ error = 1 - generalization\ accuracy$$

In addition to accuracy, other measures of goodness exist for classification (Han et al. 2010). For instance, we can use sensitivity (i.e. True Positive recognition rate), specificity (i.e. True Negative recognition rate), precision (i.e. what percentage of records that the classifier labelled as positive are actually positive), recall (i.e. completeness what percentage of positive records did the classifier label as positive), F measure or F_1-score (i.e. the harmonic mean of precision and recall). For multiple class datasets the F_1-score can be found with the following formulae:

$$F_1 Score = \frac{2 * precision * recall}{precision + recall}$$

$$precision = \frac{1}{|C|} \sum_{i=1}^{C} \frac{tp_i}{tp_i + fp_i}$$

$$recall = \frac{1}{|C|} \sum_{i=1}^{C} \frac{tp_i}{tp_i + fn_i}$$

where $|C|$ is the number of classes, tp_i are the true positives, fp_i are the false positives and fn_i are the false negatives for class i.

2.1 Decision tree classifiers

Decision trees is one of the most effective and widespread methods for producing classifiers from data (Tjortjis and Keane 2002; Quinlan 1986, 1993; Breiman 2001; Breiman et al. 1984; Aba and Breslow 1998; Auer et al. 1995; Gehrke et al. 1998). There are a large number of decision tree algorithms that have been studied in data mining, machine learning and statistics. Like other classifiers, decision trees grow trees from training data and then their accuracy is measured by using test data. Some of the best known, high performance algorithms are: C4.5 (Quinlan 1993), T3 (Tjortjis and Keane 2002), Rotation Forest (Rodriguez et al. 2006), and Random Forest (Breiman 2001). The following subsections summarise key concepts for these four algorithms.

2.1.1 Rotation Forest

In order to create the training data for a base classifier Rotation Forest does the following (Rodriguez et al. 2006):

1. It splits the feature set randomly in K subsets.
2. Principal Component Analysis (PCA) is applied to each subset.

In order to preserve the variability information in the data all principal components are retained. That results in K axis rotations in order to form new features for a base classifier. Diversity is promoted through feature extraction for each base classifier. Decision trees are chosen here because they are sensitive to rotation of the feature axes, hence the name "forest". Accuracy is sought by keeping all principal components and also using the whole data set to train each base classifier.

2.1.2 Random Forest

Random Forest is a combination of decision trees, so that each tree can depend on the values of a random vector that was selected independently from the distribution of a random forest. According to Breiman (2001):

Definition 1 A random forest is a classifier consisting of a collection of tree-structured classifiers $h(x, k), k = 1, \ldots,$ where the k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.

Generalization error of a forest depends on the strength of each tree in the forest and the correlation between them. Choosing a random selection of features to split each node results in comparable error rate of the algorithm AdaBoost (Freund and Schapire 1995) and is more "powerful" when dealing with noise.

2.1.3 C4.5 and C5

C4.5 was introduced by Quinlan (1993), in order to evolve ID3 (Quinlan 1986). C4.5 tries to find small decision trees. In order to choose which attribute to split C4.5 uses the maximum value of the GainRatio defined by:

$$GainRatio = \frac{Gain(A)}{SplitInfo(A)} \tag{1}$$

The $SplitInfo(A)$ is given by the formula:

$$SplitInfo(A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|} \tag{2}$$

where S_1 through S_c are the c subsets of examples resulting from partitioning S by the c-valued attribute A.

Gain(A) formula is:

$$Gain(A) = Entropy(S) - \sum_{u \in Values(A)} \frac{|S_u|}{|S|} Entropy(S_u) \tag{3}$$

where Values(A) is the set of all possible values for attribute A, S_u is the subset of S for which attribute A has value u and entropy is defined as follows

$$Entropy(S) = -\sum_{i=1}^{k} \frac{freq(C_i, S)}{|S|} log_2 \frac{freq(C_i, S)}{|S|} \tag{4}$$

where $freq(C_i, S)$ indicates the number of instances in S that belong to class C_i.

A modern implementation of Quinlan's decision tree algorithm exists in C5.0, which can be found at (RuleQuest 2013). The splitting of discrete attributes is different between C4.5 and C5.0, resulting in C5.0 rulesets having noticeably lower error rates on unseen cases for such datasets.

2.1.4 T2 and T3

T2 (Auer et al. 1995) is a classification algorithm which calculates optimal decision trees up to depth two and uses two kinds of decision nodes:

1. Discrete splits on a discrete attribute, where the node has as many branches as there are possible attribute values, and
2. Interval splits on continuous attributes. A node, which performs an interval split, divides the real axis into intervals and has as many branches as there are intervals. The number of intervals is restricted to be either at most as many as the user specifies, if all the branches of the decision node lead to leaves, or to be at most 2 otherwise. The attribute value unknown is treated as a special attribute value. Each decision node (discrete or continuous) has an additional branch, which takes care of unknown attribute values. T2 builds the decision tree satisfying the above constraints and minimizing the number of misclassifications of cases in the data.

T2 was compared with C4.5 (Tjortjis and Keane 2002). The results have shown that T2 produces smaller trees with approximately the same classification and generalization error as C4.5.

T3 improves T2 by:

1. Introducing the Maximum Acceptable Error (MAE), this allows some classification error (the number of instances that have been misclassified in the training set) at each node, thus reducing overfitting. T2 uses MAE of 0 % as a stopping criterion during tree building, whilst in T3 MAE ranges between 0 and 30 %, and is user specified. If MAE is less than the specified at a given node, then tree building stops and the node is returned.
2. Allowing trees to grow at a maximum depth of three. The user specifies the depth of the tree: the deeper it is the more accurate the classification.

T3 improves T2 in both classification and generalization error. More precisely, in 9 out of 15 datasets T3 has lower generalization error. That is expected because T3 grows bigger trees with less overfitting. It is worth mentioning that T3 did not improve T2 in datasets that contain only continuous attributes.

2.2 Performance comparison

In this section we present experimental results for T3, T3C, C4.5, Random Forest (RaF) and Rotation Forest (RoF). We kept C4.5 as it was shown to produce accurate results in the original paper (Tjortjis and Keane 2002) and included Random Forest and Rotation Forest as these were shown to produce even better results in recent works (Rodriguez et al. 2006; Breiman 2001; Tatsis et al. 2013). We used the same 23 data

Table 1 Generalization error for 23 data sets

Data set	T3	T3C	C4.5	RaF	RoF	Type
Breast-cancer	27.95	27.95	**2.8**	32.6	27.4	D
Chess	6.51	6.51	**0.5**	1.8	0.7	D
Lenses	**30.36**	**30.36**	37.5	37.5	62.5	D
Monk1	**0.0**	**0.0**	23.4	50.0	50.0	D
Monk2	32.99	32.99	34.7	30.3	**10.4**	D
Monk3	**0.0**	**0.0**	2.8	50.0	50.0	D
Mushroom	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	D
Soybean-large	14.71	14.71	10.5	9.7	**4.4**	D
Vote	5.83	5.83	3.0	**0.0**	3.0	D
Lymphography	25.53	25.53	22.6	**0.0**	4.0	C/D
Hypotheroid	1.03	1.03	0.9	**0.7**	1.0	C/D
Hepatitis	17.92	17.92	19.4	19.0	**6.5**	C/D
Crx	**14.24**	**14.24**	15.3	14.8	15.2	C/D
Australian	14.65	**12.92**	14.6	26.7	15.8	C/D
Cleve	22.04	22.04	23.5	32.5	**9.4**	C/D
Med_123	**0.75**	**0.75**	33.4	17.3	15.4	C/D
Iris	4.75	4.75	6.1	6.0	**4.0**	C
Heart	29.32	29.32	19.3	18.9	**16.7**	C
Breast	5.34	5.34	5.9	4.3	**3.9**	C
Diabetes	27.04	**26.24**	27.4	28.5	26.6	C
Pima	24.8	24.8	25.5	20.3	**19.9**	C
Waveform-21	31.26	31.17	23.7	**20.5**	18.1	C
Waveform-40	30.38	30.38	24.4	22.1	**19.0**	C

sets; 22 publicly available from the UCI repository (Lichman 2013) and one medical set (Tjortjis et al. 2007), as these are used in (Tjortjis and Keane 2002) to conduct experiments.

Table 1 shows the generalization error (%) for each of these five algorithms. The first group of nine data sets contains only discrete attributes. The second group of seven data sets contains both discrete and continuous attributes, and the last group of seven data sets contains only continuous attributes. In Table 1 the last column indicates whether the dataset contains discrete (D), continuous (C) or both discrete and continuous (C/D). These data sets are presented in more detail in Table 2. Table 1 depicts generalisation error for the five algorithms across 23 data sets. We use bold to indicate the best performing algorithm for each data set. All in all, T3 was best in six out of 23 cases, whilst T3C was best in eight out of 23, C4.5 in three, Random Forest in five and Rotation Forest in ten out of 23 data sets, including five out of 7 data sets containing only continuous attributes. From these results we can conclude that T3 performs comparably well against C4.5 as well as against newer classification algorithms like Random Forest and Rotation Forest.

Table 2 The data sets used for experimentation

Data set	Records (training/test)	Classes	Attributes			Total
			Cont.	Discrete		
				<5	≥5	
Breast-cancer	286 (257/29)	2	0	5	4	9
Chess	3196 (2876/320)	2	0	36	0	36
Lenses	24 (21/3)	3	0	4	0	4
Monk1	556 (500/56)	2	0	6	0	6
Monk2	601 (541/60)	2	0	6	0	6
Monk3	554 (499/55)	2	0	6	0	6
Mushroom	8124 (7312/812)	2	0	11	11	22
Soybean	683 (615/68)	19	0	33	2	35
Vote	435 (391/44)	2	0	16	0	16
Lymphography	148 (133/15)	4	3	14	1	18
Hypotheroid	3163 (2847/316)	2	7	18	0	25
Hepatitis	155 (139/16)	2	6	13	0	19
Crx	690 (621/69)	2	6	7	2	15
Australian	690 (621/69)	2	6	6	2	14
Cleve	303 (273/30)	2	6	7	0	13
Med_123	794 (715/79)	2	10	22	0	32
Iris	150 (135/15)	3	4	0	0	4
Heart	270 (243/27)	2	13	0	0	13
Breast	699 (629/70)	2	10	0	0	10
Diabetes	768 (691/77)	2	8	0	0	8
Pima	768 (691/77)	2	8	0	0	8
Waveform-21	5000 (4500/500)	3	21	0	0	21
Waveform-40	5000 (4500/500)	3	40	0	0	40

3 T3C, an improved version of T3

Despite its simplicity and its ability to produce reasonably accurate results, T3 has one deficiency. It does not improve T2's accuracy for datasets that contain only continuous attributes. That was expected because T3 does not interfere in the way T2 splits continuous attributes. This motivated us to change the way T3 splits continuous attributes.

We use pseudo code to present the main functions of T3C. The first function is *BuildTree* with signature:

Tree BuildTree (ItemNo Fp, ItemNo Lp, int Dep, ClassNo PreviousClass)

Algorithm 1 shows the pseudo code for this tree building function.

The second most important function is the function *Build* with signature:

Tree Build (ItemNo Fp, ItemNo Lp, int Dep, Tree Root)

Algorithm 1 Tree BuildTree

```
if Fp..Lp is empty then
    terminate and return a leaf node, which has PreviousClass as the winning class, and relative frequency
        and error rate as 0.
else(i.e. if there remain items for processing)
    calculate the class distribution of this list of items, select the winning class and initialise the relative
        frequency of this class, by dividing the number of occurrences of this class by the total number
        of class occurrences in this set.
    Create a new leaf called Node for this set, using this winning class, and the relative frequency and
        error rate, which have been calculated before.
if error rate ≤ MAE or Dep=0 then
    return this Node and terminate
else
    build BestNode with Node, using the function Build described below.
    Release the Node created
    return BestNode, and terminate
```

Build constructs the optimal decision tree with depth at most Dep for the given list of items.

Algorithm 2 shows the pseudo code.

Algorithm 2 Tree Build

```
1: Create a copy of the Root called BestNode
2: for all the attributes do
3:     if Special Status of the attribute is IGNORE then
4:         continue
5:     if the attribute under examination is continuous then
6:         build a Node using Build2Contin (builds a sub-tree based on a continuous attribute split) for that
               attribute
7:     elseit is discrete
8:         build Node using Build2Discr (builds a sub-tree based on a discrete attribute split)
9:     if the error of Node is less than the error of BestNode (i.e. the root) then
10:        release BestNode
11:        set BestNode := Node
12: return BestNode
```

The Special Status in line 3 is initialized when the attributes are read and determines whether the status of an attribute is discrete or ignore, i.e. missing value.

Two sub-algorithms that are used above are the $Build2Cont$ and $Build2Discr$. The signature for the latter is the following:

Tree Build2Discr(ItemNo Fp, ItemNo Lp, Attribute Att, int Dep, Tree Root)

Algorithm 3 shows the pseudo code.

The signature for the $Build2Cont$ algorithm is the following:

Tree Build2Cont(ItemNo Fp, ItemNo Lp, Attribute Att, int Dep, Tree Root)

Algorithm 4 shows the pseudo code. The function $SecondSplitContin$ finds the best decision tree when splitting a continuous attribute in the first level and a continuous attribute in the second level. The function $SecondSplitDiscr$ finds the best decision tree when splitting a continuous attribute in the first level and a discrete attribute in the second level.

Algorithm 3 Tree Build2Discr

```
BestNode is a copy of Root
for each possible value $V$ of attribute $Att$ do
    Group data ($Fp...Lp$) according to $V$ and find $Kp$ for the last of them
    Create a branch for the value $V$ using $BuildTree(Fp, Kp, Dep - 1, Root- > BestClass)$
    Compute error for each branch and add it to the total error of $BestNode$
    $Fp = Kp + 1$
return BestNode
```

Algorithm 4 Tree Build2Cont

```
if Dep = 1 then
    Call SecondSplitContin(Fp, Lp, None, Att)
    return IntervalTest
else
    Sort instances according to Att
    Find Kp, the first instance which has a known value for the attribute Att
    if Kp > Lp then
        return Root
    else
        for each attribute $Att1$ do
            if Special Status of the $Att1$ is IGNORE then
                Continue
            if $Att1$ is continuous then
                Call SecondSplitContin(Kp, Lp, Att, Att1)
            if $Att1$ is discrete then
                Call SecondSplitDiscr(Kp, Lp, Att, Att1)
        Find the best split $BestSplit$ between Kp and Lp and
        Split node $BestNode$ according to $BestSplit$
        Create three nodes-children for the $BestNode$
        return BestNode
```

As mentioned before, the main difference between T3 and T3C is the way T3C splits continuous attributes. The following changes were applied to tree building:

1. T3 splits nodes for continuous variables into three nodes: two nodes that occur by splitting the real axis in one point and a third node for unknown variables. T3C splits nodes for continuous variables into four nodes: the node that corresponds to unknown variables and three other nodes that occur by splitting the real axis in two points.
   As mentioned above the real axis is split in two points (assume k_1 and k_2). By doing so, three intervals occur. That is $(-\infty, k_1)$, $[k_1, k_2)$ and $[k_2, +\infty)$ The distance between k_1 and k_2 should be greater than 0.15 in order for the four nodes to be created, otherwise the split becomes just as that of T3. This 0.15 value was calculated empirically via experimental study. This constraint is vital to the algorithm because in some datasets the algorithm would find that the next best split is very close to the k_1 value. This was observed in the training phase. In this case the interval $[k_1, k_2)$ may contain few or no instances and will not generalize very well. While developing T3C we observed that the distance between k_1 and k_2 may be as little as 10^{-4} resulting in trees with high generalization error. We tried different values, including 0.05, 0.1, 0.2, 0.5, 1, and 1.5, for this distance and we selected

a threshold of 0.15 as it produced the best results. If the split does not improve accuracy then T3's split is selected instead.

2. At the lowest level of the tree for continuous variables, T3 produces as many nodes (leaves) as the number of classes plus one (for unknowns). T3C produces for continuous variables as many nodes as the number of classes plus one at the bottom two levels of the tree.

4 Experimental results

We compare T3C's generalisation error to that of T3 as reported in the literature (Tjortjis and Keane 2002). We used the same 23 data sets; 22 publicly available from the UCI repository (Lichman 2013) and one medical set (Tjortjis et al. 2007), as these are used in (Tjortjis and Keane 2002) to conduct experiments. Table 2 lists these sets, along with their number of records (and their split into training and test sets), their number of classes and their number of attributes (and their split into continuous, discrete with less than five distinct values and discrete attributes with five or more distinct values). Nine out of the 23 sets contained only discrete attributes, so no difference in performance was expected in comparison to T3, given that T3C affects performance on data sets including continuous attributes. These datasets have the following characteristics:

1. Different number of attributes. It is important to test the tree size our algorithm creates and how it is related to the number of attributes of the datasets.
2. There are datasets that contain only discrete attributes, only continuous attributes and both continuous and discrete data. These different kinds of attributes will provide us with insights into how our algorithm performs on these situations.

We performed 10 hold-out runs with initial random seed on each run and we report the average Training/Generalization error, Tree size and F-measure along with the standard deviation. Table 3 shows results for T3 and T3C when applied to the 14 out of 23 data sets which include continuous attributes. For each data set, the table shows respectively: tree size, classification and generalization error and F-measure for T3 and for T3C, along with the improvement on generalization error as a percentage:

$$Improv = 1 - \frac{T3CGen.error}{T3Gen.Error} \tag{5}$$

Numbers in bold indicate which algorithm performed better in terms of generalization error, which is the focus of this work. The bottom-line shows the average improvement of T3C over T3: 1.08 %. We observe that T3C improves generalization error for three out of the 14 data sets. The average improvement of T3C over T3 for these three sets is 5 % in terms of generalization error (and 1.1 % in terms of generalization accuracy). We observe that these data sets have either no discrete attributes (Diabetes, Waveform-21), or at least two discrete attributes with five or more distinct values (Australian).

Table 3 Experimental results for T3C and T3

Data set	T3C				T3				Improv
	Tree size	Class. (%)	Gen. (%)	F1 (%)	Tree size	Class. (%)	Gen. (%)	F1 (%)	Gen.
Lymphography	12.9 ± 4.63	18.98 ± 11.95	25.53 ± 7.29	48.03 ± 7.09	12.9 ± 4.63	18.98 ± 11.95	25.53 ± 7.29	48.03 ± 7.09	0.00
Hypotheroid	10.8 ± 0.63	0.61 ± 0.09	1.03 ± 0.26	89.01 ± 2.79	10.8 ± 0.63	0.61 ± 0.09	1.03 ± 0.26	89.01 ± 2.79	0.00
Hepatitis	27.6 ± 1.72	3.6 ± 0.67	17.92 ± 5.43	47.33 ± 5.47	27.6 ± 1.72	3.6 ± 0.67	17.92 ± 5.43	47.33 ± 5.47	0.00
Crx	9.1 ± 5.70	14.18 ± 0.59	14.24 ± 1.90	85.38 ± 2.21	9.1 ± 5.70	14.18 ± 0.59	14.24 ± 1.90	85.38 ± 2.21	0.00
Australian	10.5 ± 6.19	14.11 ± 0.81	**12.92** ± 1.99	86.85 ± 1.83	10.9 ± 6.33	14.11 ± 0.81	14.65 ± 2.18	86.85 ± 1.83	9.12
Cleve	36.1 ± 7.41	11 ± 1.04	22.04 ± 6.29	80.42 ± 5.47	36.1 ± 7.41	11 ± 1.04	22.04 ± 6.29	80.42 ± 5.47	0.00
Med_123	17.4 ± 2.54	0.04 ± 0	0.75 ± 0.58	99.44 ± 0.01	17.4 ± 2.54	0.04 ± 0	0.75 ± 0.58	99.44 ± 0.01	0.00
Iris	11.7 ± 0.95	1.5 ± 0.87	4.75 ± 2.31	95.49 ± 2.31	11.2 ± 0.91	0.9 ± 0.74	4.75 ± 2.31	95.49 ± 2.31	0.00
Heart	4 ± 0	22.4 ± 1.35	29.32 ± 4.07	73.44 ± 2.95	4 ± 0	22.4 ± 1.35	29.32 ± 4.07	73.44 ± 2.95	0.00
Breast	10.2 ± 0.42	3.36 ± 0.49	5.34 ± 1.98	95.8 ± 1.45	10.2 ± 0.42	3.36 ± 0.49	5.34 ± 1.98	95.8 ± 1.45	0.00
Diabetes	11.6 ± 0.31	21.72 ± 1.2	**26.24** ± 2.18	80.96 ± 1.29	4.1 ± 0.14	21.72 ± 1.2	27.04 ± 2.23	80.96 ± 1.29	0.8
Pima	11.1 ± 0.56	21.55 ± 1.04	24.8 ± 2.42	81.87 ± 2.05	11.1 ± 0.56	21.55 ± 1.04	24.8 ± 2.42	81.87 ± 2.05	0.00
Waveform-21	14 ± 0	28.3 ± 0.26	**31.17** ± 0.61	68.93 ± 0.61	14 ± 0	28.3 ± 0.26	31.26 ± 0.68	68.93 ± 0.61	0.00
Waveform-40	14 ± 0	27.92 ± 0.32	30.38 ± 0.91	69.7 ± 0.84	14 ± 0	27.92 ± 0.32	30.38 ± 0.91	69.7 ± 0.84	0.00
Avg improvement									0.71

4.1 T3C vs. C4.5

We performed 10 hold-out runs with initial random seed on each run and we report the average Training/Generalization error, Tree size and F-measure along with the standard deviation. Table 4 shows results for T3 and C4.5 when applied to the 14 out of 23 data sets which include continuous attributes, and has similar format to Table 4. The bottom-line shows the average improvement of T3C over C4.5: 2.29 % in terms of generalization error (and 4.52 % in terms of generalization accuracy). We observe that T3C demonstrates better generalization error than C4.5 for nine data sets, and worse error for five out 14 sets.

It is noted that the tree size of C4.5 is extremely large for the data cases Waveform-21 and -40 (410 in comparison to 0–36 for the other data cases), but not for T3C. These data sets were created using a Data-Generator and all their attributes include noise. It appears that T3C is more resistant to noise than C4.5.

4.2 T3C vs. Rotation Forest

We performed 10 hold-out runs with initial random seed on each run and we report the average Training/Generalization error and F-measure along with the standard deviation. Table 5 shows results for T3 and Rotation Forest when applied to the 14 out of 23 data sets which include continuous attributes, and is formatted similar to Table 4. Numbers in bold indicate which algorithm performed better in terms of generalization error. The bottom-line shows the average deterioration of T3C over Rotation Forest: 21.58 % (and 4.44 % in terms of generalization accuracy). We observe that T3C demonstrates better generalization accuracy than Rotation Forest for four data sets, worse accuracy for the remaining ten out 14 sets. Still T3C improves T3, which was outperformed by Rotation Forest 10 out 14 times and an average deterioration over Rotation Forest at 51.00 % in terms of generalization error (and 4.94 % in terms of generalization accuracy). The results showed that Rotation Forest overwhelms T3C in datasets that contain continuous attributes only. In contrast, T3C outperforms Rotation Forest in four out of seven datasets that contain both discrete and continuous attributes. From those results we can conclude that T3C splits discrete attributes more thriftily than Rotation Forest. The opposite happened in datasets that contain only continuous attributes as Rotation Forest wins T3C.

4.3 T3C vs. Random Forest

As mentioned before, Random Forest demonstrated better results than T3C. We performed 10 hold-out runs with initial random seed on each run and we report the average Training/Generalization error and F-measure along with the standard deviation. Table 6 shows results for T3 and Random Forest when applied to the 14 out of 23 data sets which include continuous attributes, and the format is similar to Table 4. Numbers in bold indicate which algorithm performed better in terms of generalization error. More precisely, in ten out of 14 datasets Random Forest had lower generalization error than T3C, and in four out of 14 T3C gave lower generalization error. T3C demonstrated an

Table 4 Experimental results for T3C and C4.5

Data set	T3C				C4.5				Improv
	Size	Class. (%)	Gen. (%)	F1 (%)	Size	Class. (%)	Gen. (%)	F1 (%)	Gen.
Lymphography	12.9 ± 4.63	18.98 ± 11.95	25.53 ± 7.29	48.03 ± 7.09	18.8 ± 5.29	9.23 ± 2.5	**22.6** ± 5.2	64.55 ± 15.83	−12.96
Hypotheroid	10.8 ± 0.63	0.61 ± 0.09	1.03 ± 0.26	89.01 ± 2.79	8.2 ± 1.03	0.62 ± 0.08	**0.9** ± 0.17	90.16 ± 2.12	−14.44
Hepatitis	27.6 ± 1.72	3.6 ± 0.67	**17.92** ± 5.43	47.33 ± 5.47	11.4 ± 6.17	8.53 ± 3.10	19.4 ± 5.27	39.81 ± 12.83	7.53
Crx	9.1 ± 5.70	14.18 ± 0.59	**14.24** ± 1.90	85.38 ± 2.21	31.8 ± 12.75	9.42 ± 1.79	15.1 ± 3.65	83.61 ± 3.77	5.7
Australian	10.5 ± 6.19	14.11 ± 0.81	**12.92** ± 1.99	86.85 ± 1.83	35.9 ± 17.3	9.36 ± 2.11	14.6 ± 2.47	87.12 ± 2.25	11.51
Cleve	36.1 ± 7.41	11 ± 1.04	**22.04** ± 6.29	80.42 ± 5.47	28.2 ± 6.14	8.83 ± 1.37	23.5 ± 3.72	79.16 ± 3.17	6.21
Med_123	17.4 ± 2.54	0.04 ± 0	**0.75** ± 0.58	99.44 ± 0.01	1 ± 0	33.28 ± 1.11	33.4 ± 2.15	79.9 ± 1.59	97.75
Iris	11.7 ± 0.91	1.5 ± 0.87	**4.75** ± 2.31	95.49 ± 2.31	6.2 ± 1.03	2.01 ± 1.49	6.1 ± 3.03	94.03 ± 3.05	22.13
Heart	4 ± 0	22.4 ± 1.35	29.32 ± 4.07	73.44 ± 2.95	27.4 ± 7.22	7.47 ± 2.80	**19.3** ± 3.52	80.59 ± 3.32	−51.92
Breast	10.2 ± 0.42	3.36 ± 0.49	**5.34** ± 1.98	95.8 ± 1.45	19 ± 6.18	1.8 ± 0.58	5.9 ± 0.86	95.41 ± 0.73	9.49
Diabetes	11.6 ± 0.31	21.72 ± 1.2	**26.24** ± 2.18	80.96 ± 1.29	36.2 ± 15.64	15.58 ± 2.97	27.4 ± 3.06	79.2 ± 2.18	4.23
Pima	11.1 ± 0.56	21.55 ± 1.04	**24.8** ± 2.42	81.87 ± 2.05	34.2 ± 13.03	16.2 ± 3.33	25.5 ± 1.71	80.77 ± 1.74	2.75
Waveform-21	14 ± 0	28.3 ± 0.26	31.17 ± 0.61	68.93 ± 0.61	413 ± 24.27	4.11 ± 0.88	**23.7** ± 1.27	76.38 ± 1.24	−31.52
Waveform-40	14 ± 0	27.92 ± 0.32	30.38 ± 0.91	69.7 ± 0.84	443.6 ± 18.83	2.56 ± 0.70	**24.4** ± 1.13	75.62 ± 1.14	−24.51
Avg improvement									2.29

Table 5 Experimental results for T3C and Rotation Forest

Data set	T3C			Rotation Forest			Improv
	Class. (%)	Gen. (%)	F1 (%)	Class. (%)	Gen. (%)	F1 (%)	Gen.
Lymphography	18.98 ± 11.95	25.53 ± 7.29	48.03 ± 7.09	1.74 ± 0.53	**17.7** ± 6.64	59.71 ± 16.33	−44.24
Hypotheroid	10.8 ± 0.63	**0.61** ± 0.09	1.03 ± 0.26	89.01 ± 2.79	1.2 ± 0.19	87.15 ± 2.62	14.17
Hepatitis	3.6 ± 0.67	17.92 ± 5.43	47.33 ± 5.47	2.06 ± 1.56	**15.8** ± 4.46	55.8 ± 10.0	−13.42
Crx	14.18 ± 0.59	**14.24** ± 1.90	85.38 ± 2.21	2.97 ± 0.76	14.4 ± 3.05	84.45 ± 2.95	1.11
Australian	14.11 ± 0.81	**12.92** ± 1.99	86.85 ± 1.83	3.34 ± 0.94	13.8 ± 1.76	87.67 ± 1.52	6.38
Cleve	11 ± 1.04	22.04 ± 6.29	80.42 ± 5.47	1.1 ± 0.52	**18.5** ± 4.89	83.5 ± 4.2	−19.14
Med_123	0.04 ± 0	**0.75** ± 0.58	99.44 ± 0.01	3.3 ± 1.08	8.6 ± 1.56	93.74 ± 1.19	91.28
Iris	1.5 ± 0.87	4.75 ± 2.31	95.49 ± 2.31	0.2 ± 0.42	**4.1** ± 2.15	96.1 ± 1.84	−15.85
Heart	22.4 ± 1.35	29.32 ± 4.07	73.44 ± 2.95	1.52 ± 0.75	**18.1** ± 3.33	83.6 ± 2.99	−61.99
Breast	3.36 ± 0.49	5.34 ± 1.98	95.8 ± 1.45	1.08 ± 0.39	**3.4** ± 1.31	97.3 ± 1.06	−57.06
Diabetes	21.72 ± 1.2	26.24 ± 2.18	80.96 ± 1.29	11.32 ± 1.80	**24.5** ± 1.49	82.04 ± 1.58	−7.10
Pima	21.55 ± 1.04	24.8 ± 2.42	81.87 ± 2.05	12.67 ± 1.97	**23.8** ± 2.66	82.37 ± 2.35	−4.20
Waveform-21	28.3 ± 0.26	31.17 ± 0.61	68.93 ± 0.61	0.39 ± 0.08	**15.7** ± 0.82	84.27 ± 0.83	−98.54
Waveform-40	27.92 ± 0.32	30.38 ± 0.91	69.7 ± 0.84	0.02 ± 0.03	**15.7** ± 0.44	84.33 ± 0.45	−93.50
Avg improvement							−21.58

Table 6 Experimental results for T3C and Rotation Forest

Data set	T3C			Random Forest			Improv
	Class. (%)	Gen. (%)	F1 (%)	Class. (%)	Gen. (%)	F1 (%)	Gen.
Lymphography	18.98 ± 11.95	25.53 ± 7.29	48.03 ± 7.09	0 ± 0	**18** ± 5.97	53.38 ± 11.76	−41.83
Hypotheroid	0.61 ± 0.09	**1.03** ± 0.26	89.01 ± 2.79	0.03 ± 0	1.2 ± 0.17	86.84 ± 2.26	14.17
Hepatitis	3.6 ± 0.67	17.92 ± 5.43	47.33 ± 5.47	0.1 ± 0.31	**15.5** ± 3.54	50.56 ± 12.76	−15.61
Crx	14.18 ± 0.59	14.24 ± 1.90	85.38 ± 2.21	0.06 ± 0.09	**13.8** ± 3.06	84.77 ± 3.24	−3.19
Australian	14.11 ± 0.81	**12.92** ± 1.99	86.85 ± 1.83	0 ± 0	13.3 ± 1.28	88.4 ± 1.54	2.86
Cleve	11 ± 1.04	22.04 ± 6.29	80.42 ± 5.47	1.1 ± 0	**18.5** ± 3.56	83.5 ± 3.37	−19.14
Med 123	0.04 ± 0	**0.75** ± 0.58	99.44 ± 0.01	13.07 ± 0.68	33.4 ± 2.19	79.71 ± 1.64	97.75
Iris	1.5 ± 0.87	**4.75** ± 2.31	95.49 ± 2.31	0 ± 0	5.1 ± 1.68	95.04 ± 1.69	6.86
Heart	22.4 ± 1.35	29.32 ± 4.07	73.44 ± 2.95	0 ± 0	**18.4** ± 3.47	83.03 ± 3.32	−59.35
Breast	3.36 ± 0.49	5.34 ± 1.98	95.8 ± 1.45	0 ± 0	**3.82** ± 1.39	97.0 ± 1.10	−39.79
Diabetes	21.72 ± 1.2	26.24 ± 2.18	80.96 ± 1.29	0 ± 0	**24** ± 2.60	82.04 ± 2.10	−9.33
Pima	21.55 ± 1.04	24.8 ± 2.42	81.87 ± 2.05	0 ± 0	**24.2** ± 2.39	81.79 ± 2.02	−2.48
Waveform-21	28.3 ± 0.26	31.17 ± 0.61	68.93 ± 0.61	0 ± 0	**15.2** ± 0.51	84.81 ± 0.50	−107.07
Waveform-40	27.92 ± 0.32	30.38 ± 0.91	69.7 ± 0.84	0 ± 0	**15** ± 0.70	85.01 ± 0.70	−102.53
Avg improvement							−19.16

Table 7 Experimental results for datasets with only discrete attributes

Data set	T3C	C4.5	Random Forest	Random Forest	C5.0
Breast-cancer	**27.95**	28.15	32.17	28.67	28.16
Chess	6.51	0.94	1.06	**0.88**	0.94
Lenses	30.36	**21.61**	33.39	34.29	45.89
Monk1	**0**	6.21	0.89	8.59	**0**
Monk2	32.99	30.74	53.69	**6.64**	33.67
Monk3	**0**	**0**	**0**	**0**	**0**
Mushroom	**0**	**0**	**0**	**0**	0.25
Soybean-large	14.71	5.02	**0.13**	1.15	12.69
Vote	5.83	3.27	**0**	1.29	5.28

average deterioration over Random Forest at 19.16 % in terms of generalization error (and an average deterioration of 1.49 % in terms of generalization accuracy).

It is also notable that in datasets that contain both discrete and continuous attributes T3C gives lower generalization error in three out of seven datasets and Random Forest gives lower generalization error in four out of seven datasets. That occurred due to the high ability of T3C in splitting discrete attributes.

4.4 Discrete attributes

Observing the above results we conclude that T3C is doing well on data sets containing discrete variables. For that reason, it is appropriate to compare T3C on data sets containing only discrete variables to see the performance in these data sets. The results of the comparison are shown in the Table 7. For each data set, the table shows the generalization error for T3C, C4.5, Random Forest, Rotation Forest and for C5, respectively.

From Table 7, we conclude that T3C gave better results in three out of nine cases, which demonstrates the strength of the algorithm in relation to the others. C4.5 gave better results in four out of nine cases, Random Forest gave better results in four out of nine cases, Rotation Forest in three out of nine cases and C5 in only two cases. From this, we can conclude that T3C gave good results compared to the other algorithms. In particular, on dataset Monk1, T3C has a much smaller generalization error than the other algorithms, apart from C5.

On these nine data sets Rotation Forest gives an average generalization error of 6.50 %. C4.5 comes second with 8.36 %, whilst Random Forest and T3C follow with average generalization error of 11.01 and 11.03 %, respectively. Finally C5 appears to be slightly over-fitting the data with an average generalization error of 12.34 %.

5 Performance evaluation

From the comparison of T3 and T3C we can conclude that T3C performed better than T3. In particular, T3C improved T3 on generalization error by 1.08 % on average for all datasets including continuous attributes. This was expected because T3C has a greedier approach when the tree decides to split continuous attributes. As it was discussed in Sect. 3, one more node is used when a continuous attribute is being split. Despite this T3C does not produce much bigger trees because of the other two changes we introduced. Actually for 14 data sets it produced an average 0.55 nodes more than T3.

If we combine findings presented in Tables 4, 5, 6 and 7 we can conclude that T3C performs well regarding generalization error.

For instance, T3C also improves T3 by 0.71 % on average for all datasets (Tables 3, 7), and 1.05 % better generalization error on data sets including continuous attributes. It should be also noted that T3C improves T3 also in terms of classification error by 0.74 % on average for all datasets (Tables 3, 7), and 1.16 % better classification error on data sets including continuous attributes (and 0,42 % better F measure). Also, regarding generalization error, T3C is better than C4.5 by 5.87 % on average for all datasets including continuous attributes (Table 4), and produces trees with an average 65.28 nodes less than C4.5. It is worth mentioning that T3C improved C4.5 in datasets that contain both discrete and continuous attributes (better in 5 out of 7 sets), whilst achieving better generalization error in 4 out of 7 data sets with only continuous attributes. On the other hand, C4.5 achieved better generalization accuracy in 5 out of 9 data sets with only discrete attributes. T3C had also comparable performance to Random Forest. More specifically, T3C improved Random Forest for datasets that contain both discrete and continuous attributes by 17.96 %, but was worse than Random Forest by 43.78 % for datasets that contain only continuous attributes as well as worse by 1.39 % for datasets that contain only discrete attributes(Tables 6, 7).

As for the comparison between T3C and Rotation Forest, T3C demonstrated an average 31.78 % worse performance than Random Forest in terms of generalization accuracy, for all data sets (Tables 5, 7). As with Rotation Forest results, T3C performed better in datasets that contain both discrete and continuous attributes. More precisely, in four out of seven datasets T3C had lower generalization error than Random Forest. That did not occur in datasets that contain only continuous attributes, as Random Forest performed better in these datasets.

As we can see T3C performed better than T3, C4.5, comparably to Random Forest and worse than Rotation Forest, in terms of generalization error.

6 Conclusions and future work

Experimental results have shown that T3C improves T3 in terms of generalization accuracy without producing big trees and without overfitting. Specifically, T3C improved T3 on datasets that contain continuous attributes. Moreover, T3C improves C4.5 in terms of generalization error (and tree size). When comparing T3C with Random Forest, T3C yields comparable generalization error, and worse generalization error compared to Rotation Forest. T3C has the advantage of producing small trees and performs well when only discrete attributes are present (best performer in 4 out of 9 such data sets, better even than C5). Concluding, although T3C improves on T3, and C4.5, further improvements can be made. For instance, T3C can further improve the way it splits continuous attributes. It would be challenging to consider a split on continuous attributes with more than five nodes and with trees of depth more than three. Also T3C can be parameterized it terms of which goodness measure we require to improve on. Increasing depth improves classification accuracy; reducing splits could improve generalization accuracy. Moreover, further work can be done focusing on area under ROC curve, sensitivity/specificity or precision/recall improvements. Finally, further experiments involving other decision tree classification algorithms could help establish guideline on algorithm selection depending on the nature and the characteristics of the dataset at hand.

References

Aba DW, Breslow LA (1998) Comparing simplification procedures for decision trees on an economics classification. Technical report, DTIC Document

Auer P, Holte RC, Maass W (1995) Theory and applications of agnostic PAC-learning with small decision trees. In: Theory and applications of agnostic PAC-learning with small decision trees. Morgan Kaufmann, San Francisco, pp 21–29

Berry MJ, Linoff GS (2004) Data mining techniques: for marketing, sales, and customer relationship management. Wiley, New York

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and regression trees. Wadsworth and Brooks, Monterey

Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: Computational learning theory. Springer, Berlin, pp 23–37 (1995)

Gehrke J, Ramakrishnan R, Ganti V (1998) Rainforest-a framework for fast decision tree construction of large datasets. In: VLDB, vol 98, pp 416–427

Han J, Kamber M, Pei J (2010) Data mining: concepts and techniques. Morgan Kaufmann, San Francisco

Hubert M, Van der Veeken S (2010) Robust classification for skewed data. Adv Data Anal Classif 4(4):239–254

Lichman M (2013) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine, CA. http://archive.ics.uci.edu/ml

Mozharovskyi P, Mosler K, Lange T (2015) Classifying real-world data with the DDα-procedure. Adv Data Anal Classif, 9(3):287–314

Murthy S, Salzberg S (1995) Decision tree induction: how effective is the greedy heuristic? In: Proceedings of the first international conference on knowledge discovery and data mining. Morgan Kaufmann, San Francisco, pp 222–227

Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106

Quinlan JR (1993) C4.5: programs for machine learning, vol 1. Morgan Kaufmann, San Francisco

Quinlan JR (1996) Improved use of continuous attributes in c4.5. J Artif Intell Res 4:77–90

Rodriguez J, Kuncheva L, Alonso C (2006) Rotation forest: a new classifier ensemble method. IEEE Trans Pattern Anal Mach Intell 28(10):1619–1630

RuleQuest (2013). http://www.rulequest.com. Last Accessed April 2016

Tatsis VA, Tjortjis C, Tzirakis P (2013) Evaluating data mining algorithms using molecular dynamics trajectories. Int J Data Min Bioinform 8(2):169–187

Tjortjis C, Keane JA (2002) T3: an improved classification algorithm for data mining. Lect Notes Comput Sci 2412:50–55

Tjortjis C, Saraee M, Theodoulidis B, Keane JA (2007) Using t3, an improved decision tree classifier, for mining stroke related medical data. Methods Inf Med 46(5):523–529

Witten I, Frank E, Hall M (2011) Data mining: practical machine learning tools and techniques. Morgan Kaufmann, San Francisco