ARTICLE IN PRESS

Information Sciences xxx (2011) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

The CASH algorithm-cost-sensitive attribute selection using histograms

Yael Weiss, Yuval Elovici, Lior Rokach *

Department of Information Systems Engineering, Ben-Gurion University of the Negev, Be'er Sheva 84105, Israel
Deutsche Telekom Laboratories at Ben-Gurion University, Israel

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Data mining
Cost-sensitive feature selection
Genetic search
Histogram comparison
Misclassification cost
Feature grouping

ABSTRACT

Feature selection is an essential process for machine learning tasks since it improves generalization capabilities, and reduces run-time and a model's complexity. In many applications, the cost of collecting the features must be taken into account. To cope with the cost problem, we developed a new cost-sensitive fitness function based on histogram comparison. This function is integrated with a genetic search method to form a new feature selection algorithm termed CASH (cost-sensitive attribute selection algorithm using histograms). The CASH algorithm takes into account feature collection costs as well as feature grouping and misclassification costs. Our experiments in various domains demonstrated the superiority of CASH over several other cost-sensitive genetic algorithms.

1. Introduction

Feature selection is essential in machine learning tasks, and feature selection algorithms usually select a feature subset for optimizing the performance of the particular task at hand. However, in many domains the cost of obtaining the feature values cannot be neglected and must be taken into account in the feature selection process. For example, in medical diagnostics, the physician may need to consider the costs involved prior to determining which tests to conduct.

In addition to the feature costs, it is also necessary to consider misclassification costs. Again citing the medical domain, given the costs of false positive (FP) and false negative (FN) misclassification costs, physicians, many of whom are working in tightly controlled business frameworks, might consider that further testing is not economically justified. This decision cannot be made if the error misclassification costs are not available.

What these two examples demonstrate is that a cost must be assigned to both the tests and the misclassifications in order to decide on the appropriate course of action. Unfortunately, in the majority of feature selection algorithms, feature and misclassification costs are not considered in practice.

Another important aspect involved in determining the suitability of one course of action rather than another is the question of feature grouping, which arises occasionally when attribute costs vary with the choice of prior tests. For instance, medical tests are generally not performed separately; when certain medical tests are performed together, it is cheaper than when they are performed individually. Therefore, information about the feature grouping must be tailored into the feature selection process.

In this study, we evaluate CASH, a cost-sensitive feature selection algorithm based on a genetic algorithm that considers feature test and misclassification costs as well as feature grouping. Inherent in the algorithm is a new fitness function based on comparing histograms. Extensive experiments were conducted to demonstrate the effectiveness of CASH compared to other methods.

* Corresponding author at: Department of Information Systems Engineering, Ben-Gurion University of the Negev, Be'er Sheva 84105, Israel.
E-mail addresses: wiessy@bgu.ac.il (Y. Weiss), elovici@bgu.ac.il (Y. Elovici), liorrk@bgu.ac.il (L. Rokach).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.01.035

The rest of the paper is organized as follows: Section 2 presents a review of the related literature. In Section 3 we introduce the notation and the problem formulation. Then, in Section 4 we describe our new algorithm with a case study based on a synthetic dataset. In Section 5 we review the experiments that were conducted in this study, the evaluation measures employed and the evaluation of the results of the new algorithm. Then, we discuss the study in Section 6. In Section 7, we discuss directions for future research.

2. Related work

2.1. Genetic algorithm for attribute selection

In attribute selection, the goal is to select, from the original set of attributes, a subset of attributes that are relevant for the target data mining task [26,16,27]. The standard individual representation for attribute selection consists simply of a string of N bits, where N is the number of original attributes and the ith bit, $i = 1,\ldots,N$, can take the value 1 or 0, indicating whether or not, respectively, the ith attribute is selected. This individual representation is simple and traditional crossover and mutation operators can be easily applied. However, this representation has the disadvantage that it does not scale very well with a number of attributes. In applications with many thousands of attributes (such as occurs in text mining and some bio-informatic problems) an individual representation would have many thousands of genes. This slows the runtime of the genetic algorithm (GA).

An alternative individual representation, proposed by Cherkauer and Shavlik [6], consists of M genes (where M is a user-specified parameter), where each gene can contain either the index (id) of an attribute or a flag – say 0 – denoting no attribute. An attribute is considered selected if and only if it occurs in at least one of the M genes of the individual. For instance, the individual "3 0 8 3 0", where $M = 5$, represents a candidate solution where only the 3rd and the 8th attributes are selected. The fact that the 3rd attribute occurs twice in the previous individual is irrelevant for the purpose of decoding the individual into a selected attribute subset. The advantage of this representation is that it scales up better with respect to a large number of original attributes, since the value of M can be much smaller than the number of original attributes. On the other hand, there is a disadvantage: the new parameter, M, that has been introduced is not necessary in cases of standard individual representation.

With respect to the fitness function, the wrapper and filter approaches may be implemented in GAs for attribute selection. In the wrapper approach the GA uses the classification algorithm to compute the fitness of individuals, whereas in the filter approach the GA it does not. The vast majority of GAs for attribute selection follows the wrapper approach [28], and many of these GAs use a fitness function involving two or more criteria to evaluate the quality of the classifier built from the selected attribute subset [29].

GA-based attribute selection focuses mainly on classification tasks. GAs that perform attribute selection for clustering were introduced, for example, by Kim et al. [21] and Jourdan et al. [19]. Moreover Jong et al. [18], used GA for attribute ranking. Once the ranking has been done, one can select a certain number of top-ranked attributes; the number can be specified by the user or computed in a more automated way [34].

Sharpe and Glover [35] and Kudo and Sklansky [23] present an empirical comparisons between GAs and other kinds of attribute selection methods. In general these empirical comparisons show that GAs, with their associated global search in the solution space, usually (though not always) obtain better results than local search-based attribute selection methods.

2.2. Cost-sensitive algorithms

Cost-sensitive learning is an essential task in several real-world applications. Turney [42] presented a taxonomy of the main types of costs involved in inductive concept learning. Two costs, misclassification and test, are particularly relevant to this paper.

Several researches have considered misclassification costs but not test costs [9,10,20]. Other papers focus on the cost of tests, but fail to take into account misclassification costs. Yang and Honavar [45] offer a new subset feature selection method that employs a genetic algorithm as its search method with the aim of finding a feature subset that minimizes the total feature test cost while maximizing accuracy. Iswandy and Koenig [17] present the effectiveness of a feature selection approach that is sensitive to the feature test costs. In particular, they evaluated a genetic algorithm (GA) and particles swarm optimization (PSO). Paclik et al. [31] suggest a new approach for combining the measurement costs and feature grouping in the feature selection process in the product analysis domain. Their method uses a greedy wrapper-based feature selection algorithm that takes into account different groups of features that share computation costs.

In contrast to the abovementioned papers, the new feature selection algorithm that we present in this work is sensitive to both test and misclassification costs. An overview of the most significant papers that take into account both costs is presented in the rest of this section.

Turney [41] was the first to consider both test and misclassification costs. Turney's approach presents the inexpensive classification with an expensive test (ICET) system. ICET uses a method that employs a hybrid approach, combining a greedy search heuristic (decision tree) with a genetic algorithm for building a decision tree that minimizes the total cost objective, which is composed of test and misclassification costs. Furthermore, ICET considers feature grouping. ICET is robust but its execution time is rather long.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Several works use the Markov decision process to handle the cost-sensitive problem. Unfortunately, these methods, like the ICET system, require a time-consuming search procedure. Zubek and Dietterich [46] obtained an optimal solution by executing the search process in a state space until an optimal policy is attained. Taking into account the record values that were achieved for each new record, the optimal policy suggests which action should be carried out to minimize the total cost. In order to reduce computational complexity, Arnt and Zilberstein [2] extended the last model by including the ability to handle time-sensitive utility costs that consider the time needed to obtain the results of a test.

Chai et al. [4] offered csNB, a new cost-sensitive classifier based on a naïve Bayes algorithm. Several researchers, such as Sheng et al. [37] and Freitas et al. [14], use a cost-sensitive decision tree for the classification task. Ling et al. [24] proposed a decision tree algorithm which applies a new splitting criterion, minimal total cost, to training data instead of the well known minimum entropy measurement. Ling et al. [25] subsequently updated their strategy for building cost-sensitive decision trees by incorporating possible discounts when obtaining the values of a group of attributes with missing values in the tree building algorithm. Sheng et al. [38] improve the above-mentioned algorithm by incorporating possible discounts when obtaining values of a group of attributes with missing values in the tree building algorithm.

In another paper, Sheng et al. [38] offered a framework where a decision tree is built for each new test case. Sheng and Ling [36] then suggested a hybrid cost-sensitive decision tree, DTNB, that reduces the minimum total cost by integrating the advantages of a cost-sensitive decision tree and those of the cost-sensitive Naïve Bayes. While it uses the cost-sensitive decision tree in order to decide which tests to choose, for the classification task it uses the cost-sensitive Naïve Bayes. Freitas et al. [14] suggest a new splitting criterion in building decision trees that considers different aspects of test costs. They examine several cost-scale factors that regulate the influence of test costs as it can make trees more sensitive to test costs. They also suggest how to embed the risk cost for performing the test in the new cost-sensitive splitting criterion. The risk cost captures the change in the quality of life due to performing these tests on the patient. However, no experiments were carried out with regard to risk cost. Esmeir and Markovitch [11] introduced a new algorithm, the anytime cost-sensitive tree learner (ACT), which is sensitive to the test and misclassification costs. It constructs a decision tree in a top-down fashion. Moreover, since ACT is an anytime algorithm, it offers a tradeoff between the computation time and classification costs.

Vidrighin et al. [43,44] used a cost-sensitive decision tree to improve the ICET algorithm [41]. The new modified algorithm they developed is termed the ProICET algorithm. There are a number of differences between ICET and ProICET algorithms that mostly derive from the genetic components of the algorithms, such as, as a replacement strategy: while ICET has used the multiple populations' technique, ProICET uses a single population. Moreover, the percentage of training examples in evaluating the fitness score of an individual changed (in ProICET it is set to 70% as opposed to 50% in ICET). Chen et al. [5] also developed a cost-sensitive decision tree, however their algorithm support multiple condition attributes. A cost is paid to obtain the values of the decision attribute and that an instance must be classified without exceeding the spending cost threshold.

3. Notations and problem formulation

We present here common notations, and a real-world application involving heart disease diagnosis to illustrate them. The application was taken from the UCI ML repository [3].

3.1. The classification task

In a typical classification problem, a training set of labelled examples is given. The training set can be described in a variety of languages, most frequently as a collection of records that may contain duplicates. A vector of feature values describes each record. The notation A denotes the set of input features containing n features: $A = \{a_1, \ldots, a_k, \ldots, a_n\}$-and y represents the class variable or the target feature whose domain is a set of K classes: $Y = \{1, 2, \ldots, K\}$.

Table 1
Training dataset.

Record	1	2	3	4	5	6	7
Age	35	32	30	33	64	67	66
Sex	M	F	F	M	F	F	F
Cp	4	4	4	4	1	1	1
Trestbps	94	96	95	97	196	194	195
Restecg	0	0	0	0	2	2	2
Ca	0	0	0	0	3	3	3
Chol	126	500	303	476	205	400	562
Fbs	0	0	0	0	1	1	1
Thalach	71	73	72	75	197	198	196
Thal	3	3	3	3	6	7	7
Exang	0	1	1	1	0	1	0
Oldpeak	0.2	1.6	2.4	1.2	0.6	4.5	5.6
Slope	1	1	1	1	3	3	3
Class	Negative	Negative	Negative	Negative	Positive	Positive	Positive

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

The heart disease diagnosis application presents a learning task: predicting coronary artery disease based on 13 tests carried out on patients. Although there are 303 records in this dataset, we used only 7 artificial records for the sake of simplicity in explaining the algorithm. Table 1 presents the training dataset with the 7 records and the 13 features. Each patient can be classified into two classes: healthy (negative), which indicates a narrowing of less than 50% of the artery, and sick (positive) which indicates more than 50%.

The instance space (the set of all possible examples) is denoted by X while the universal instance space (or the labelled instance space) U is defined as a Cartesian product of the input instance space and the target feature domain, i.e., $U = X \times Y$. The training set consists of a set of m records and is denoted as $S = (\langle x_1, y_1\rangle, \ldots, \langle x_m, y_m\rangle)$ where $x_i \in X$ and $y_i \in Y$. It is assumed that the training set records are generated randomly and independently according to some fixed and unknown joint probability distribution $p(x,y)$ over U.

The notation $I_S : X \rightarrow Y$ represents a classifier (such as classification tree) that was trained using inducer I on the training dataset S.

3.2. Feature subset

Let $B \subseteq A$ be a subset of features. We denote the projection of an instance $x_i \in X$ onto a subset of features B as $\pi_B x_i$. Similarly, the projection of a training set S onto B is denoted as $\pi_B S$. Therefore the notation $I_{\pi_B S} : \pi_B X \rightarrow Y$ stands for a classifier trained on the projection of the training set S using only the features in B.

3.3. Misclassification cost matrix

Let C be the misclassification cost matrix. Each entry $C_{i,j}$ specifies the price to be paid when misclassifying an i-class instance as class j. Usually, $C_{i,i} = 0$. Therefore the expected misclassification cost of classifier $I_{\pi_B S}$ trained on a new example drawn at random from the $p(X,Y)$ distribution is:

$$MC(B) = \sum_{\forall (x,y) \in U} C_{y, I_{\pi_B S}(\pi_B x)} \cdot p(x,y). \tag{1}$$

Table 2 presents the false positive (FP) and false negative (FN) misclassification cost values for the heart disease dataset. The UCI ML repository does not supply the misclassification costs. The costs of the false positive (FP) and false negative (FN) were chosen to be \$600 and \$1000, respectively as proposed by Sheng et al. [38]. These assignments were chosen after consulting a researcher from the Heart-Failure Research Group at the local medical school.

3.4. Test costs

We assume that a certain acquisition cost is associated with each attribute. Following Turney [41], we assume that attributes are partitioned into mutually exclusive groups, such that the attributes that belong to the same group share a common setup cost. Once the common cost is paid by one of the attributes in the group, the remaining attributes get a discount. Therefore the cost of acquiring an attribute in this group is conditional on whether another attribute of the group has already been chosen.

Let G be the group affinity vector. Each entry in the vector G specifies for each feature a_k the group $l (1 \leqslant l \leqslant n)$ to which it belongs. Table 3 presents the features group assignment indication.

Let T_k^b specifies the cost of conducting a test on feature a_k (i.e., acquiring its value), without taking into consideration the cost spent on testing other features that belong to the same group (i.e., without a group discount). Let T_k^a specify the cost of performing a test on feature a_k, while taking into consideration the group discount. The difference $T_k^b - T_k^a$ represents the group discount and is equal for all attributes in the same group.

Table 4 indicates the test costs in dollars (\$), with and without group discount, of each feature in the heart disease diagnosis application. These test costs were obtained from the Ontario Health Insurance Program's fee schedule. The cost of obtaining several of the attributes (age, sex and cp) is as low as \$1. Tests carried out on a group are discounted in terms of costs. For example, the tests in group 7 (chol and fbs) are blood tests that can indicate the presence of coronary artery disease; they share the common cost of \$2.10 for the blood collecting procedure. Additional examples of groups in this application are the tests in groups 8 and 9 that involve heart measurements during exercise. Tests in group 9 (exang, oldpeak, and slope) are electrocardiograph tests which are usually performed simultaneously while the patient runs on a treadmill.

The acquisition cost of all features in subset B is:

Table 2
Misclassification costs.

	Prediction:Negative	Prediction:Positive
Relatiy:Negative	0	FP cost = \$600
Reality:Positive	FN cost = \$1000	0

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Table 3
Features Group.

Feature	Before discount cost	After discount cost
Age	$1	$1
Sex	$1	$1
Cp	$1	$1
Tresbps	$1	$1
Restecg	$15.5	$15.5
Ca	$100.9	$100.9
Chol	$7.27	$5.17
Fbs	$5.2	$3.1
Thal	$102.9	$1
Thalach	$102.9	$1
Exang	$87.3	$1
Oldpeak	$87.3	$1
Slope	$87.3	$1
Age	$1	$1
Sex	$1	$1
Cp	$1	$1
Tresbps	$1	$1
Restecg	$15.5	$15.5
Ca	$100.9	$100.9
Chol	$7.27	$5.17
Fbs	$5.2	$3.1
Thal	$102.9	$1
Thalach	$102.9	$1
Exang	$87.3	$1
Oldpeak	$87.3	$1
Slope	$87.3	$1

Table 4
Test costs.

Group	Feature
1	Age
2	Sex
3	Cp
4	Trestbps
5	Restecg
6	Ca
7	Chol
7	Fbs
8	Thalach
8	Thal
9	Exang
9	Oldpeak
9	Slope

$$TC(B) = \sum_{a_i \in B} T_i^a + \sum_{a_i \in B:(\exists j)(j<i:G(i)=G(j):a_j \in B)} T_i^b - T_i^a. \quad (2)$$

The augend represents the post-discount cost of all features included in subset B. The addend specifies a group's common costs; they are taken into consideration only once for each group (by the group's member with the smallest index).

For example the cost of subset B = {Thalach, Thal, Exang, Oldpeak} in our example is:

$$TC(\{\text{Thalach},\ \text{Thal},\ \text{Exang},\ \text{Oldpeak}\}) = 1 + 1 + 1 + 1 + 102.9 - 1 + 87.3 - 1 = 192.2$$

Following Turney [41], we assume that both types of cost, of the features and of misclassification, are given in the same cost scale (in dollars).Therefore, the summing together of these two cost types in order to get the total cost is feasible.

However, there are many real-world applications where these two costs do not share the same cost scale. Hence, the decision about how to combine the two cost types is a domain-dependent issue. For instance, contrary to our study, Kim and Kim [22] deal with a multi-objective optimization problem instead of a single optimum. Their aim is to minimize two factors: the number of features and the error rate of the classifier. Another possible approach to deal with applications where these costs are given in a different scale was introduced by Qin et al. [32]. They offer a method that sets a maximal budget for one objective while minimizing the other.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

3.5. The goal

Given the abovementioned notations, we are now ready to state our main goal. In this study the problem that we are aiming to solve is how to select a subset of features that minimizes the total of the misclassification and test costs. Consequently, the problem can be formally phrased as follows:

Given an induction Algorithm 1, a training set S with input feature set A, test cost before and after discount vectors T^b and T^a, respectively, and the group affinity vector G, find an optimal features subset $B \subseteq A$ such that $MC(B) + TC(B)$ is minimized.

Since finding an optimal set case is computationally difficult (given n such features, there are 2^n possible feature subsets) and the probability $p(x,y)$ is unknown, we limit ourselves to finding an approximate solution.

4. CASH: Cost-sensitive attribute selection using histograms

In order to solve the problem defined in Section 3, we propose CASH, a cost-sensitive feature selection method which uses a new fitness function based on comparing histograms. This algorithm follows the filter approach, where a certain metric ranks each feature subset. Unlike the wrapper approach, it is not associated with a particular classifier. It is important to note that in the literature most works in this domain combine the feature selection procedure in the process of building the learning model. The proposed algorithm employs a genetic algorithm as a search method. In this research study we consider only the case where the target feature is binary. However, the algorithm can be extended easily to deal with multivariate class variables.

Algorithm 1 presents the proposed CASH algorithm. The CASH algorithm consists of four main steps: preprocessing; creating initial population of individuals; computing the fitness of each individual; and applying a genetic algorithm to the initial population. In the following subsections we describe each one of the above steps, together with a case based on a real application involving the heart disease diagnosis taken from the UCI ML repository mentioned above.

Through the various steps of CASH algorithm we followed a simple cost-sensitive heuristic rule: given a set of records (which might be the entire records in the dataset or only a partial set), classify these records according to the class that brings to a minimum the average total cost from among all the possible classes.

Algoritm 1: The CASH pseudo-code

```
Input:
  D: Training set
  C: misclassification cost matrix
  T^b: test-cost before discount
  T^a: test-cost after discount
  G: group affinity
Output:
  Result_Subset: Near optimal group of features
Step 1: Preprocessing steps.
  Compute the a priori cost of the datasets.
  For each feature:
  Construct the feature's histograms.
    For each record in D
      Compute how it classifies the record
      Estimate the misclassification cost ratio of the record
Step 2: Create initial population of individuals.
Step 3: Compute the fitness of each individual.
Step 4: Apply genetic algorithm on the initial population.
  REPEAT
    Select individuals.
    Apply genetic operators to selected individuals. Create new individuals.
    Compute fitness of each of the new individuals.
    Update the current population (new individuals replace old individuals).
  UNTIL (stopping criteria).
```

4.1. Step 1: Preprocessing steps

The preprocessing performed in step 1 is described below:

4.1.1. Calculate the average a priori cost of the training dataset

By following the cost-sensitive heuristic rule, the average *a priori* cost is computed as follows:

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

For each class i compute the potential misclassification cost when classifying all records in the training dataset to class i as shown in Eq. (3):

$$MCS_i = \sum_{\forall (x,y) \in S} C_{y,i}. \tag{3}$$

The CASH algorithm assumes that the values of the classes are available for free. Most of the related work in this domain is based on this assumption. Therefore, when computing the average cost, since no feature has yet been purchased, the test cost is equal to zero. Consequently, when computing the average *a priori* cost, the misclassification cost is equal to the average total cost.

Then, the minimal potential misclassification cost from the *n potential* average misclassification costs is regarded as the *a priori* cost as shown in Eq. (4):

$$argmin_i[MCS_i]. \tag{4}$$

The average *a priori* cost is subsequently computed by dividing the *a priori* cost by the number of records in the training dataset.

The motivation for computing the average *a priori* cost is that it serves as an indication as to when a features subset should not be obtained. That is to say, if the average *a priori* cost that is computed based on the classes is lower than the average total cost achieved by the features subset, CASH will not move this subset to the next generation.

Using the synthetic training dataset (see Table 1) as an example, we calculate its average *a priori* cost as follows:

First, the algorithm calculates the potential misclassification cost caused by classifying all the records in the dataset as negative. The negative misclassification cost is computed by multiplying the false negative (FN) misclassification cost ($1000) by the number of records in the training dataset with positive class variable values (3). Therefore, the potential misclassification cost is $3000 when classifying all the records in the dataset as negative.

The algorithm calculates the potential misclassification cost caused by classifying all the records in the dataset as positive in the same manner. The positive misclassification cost is computed by multiplying the false positive (FP) misclassification cost ($600) by the number of records in the training dataset with negative class variable values (4). Therefore, the potential misclassification cost when classifying all the records in the dataset as positive is $2400. Since the cost caused by classifying all the records in the dataset as positive ($2400) is smaller than the cost caused by classifying all the records in the dataset as negative ($3000), the algorithm sets the *a priori* cost at $2400. Then, the average *a priori* cost is computed by dividing the *a priori* cost ($2400) by the number of the records in the training dataset (7), which is equal to $342.85.

4.1.2. Compute histograms for each feature in the training dataset

For each feature in the training dataset, compute a histogram for each class value.

Each feature's histogram indicates into which bin the records in the database falls.

The number and size of the bins is determined as follows:

For numerical features, the intervals are determined using Fayyad and Irani [12] MDL discretization. For nominal features, each interval is associated with one value.

With regard to the synthetic training dataset, for each one of the 13 predictive features the algorithm constructs two histograms: one for the records whose true class is positive and the other for the records whose true class is negative. For example, Fig. 1 presents Oldpeak's histograms. Oldpeak represents the ST depression induced by exercise relative to rest. Oldpeak is a continuous feature whose values range between 0 and 5.6. Since Oldpeak's values are continuously changing, we applied Fayyad and Irani's MDL discretization algorithm to it. The MDL discretization algorithm divides Oldpeak values into two bins: one from minus infinity to 1.9, the other from 1.9 to infinity. In the same manner we constructed the Exang's histograms as illustrated in Fig. 2. Exang represents exercise induced angina. Since Exang is a category feature, it does not go

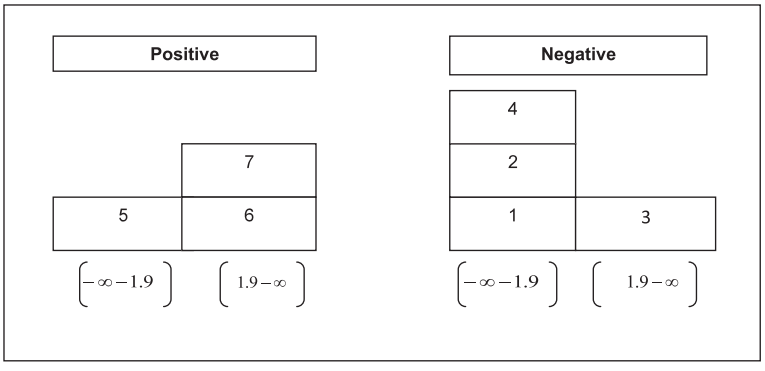

Fig. 1. Oldpeak's constructed histograms.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

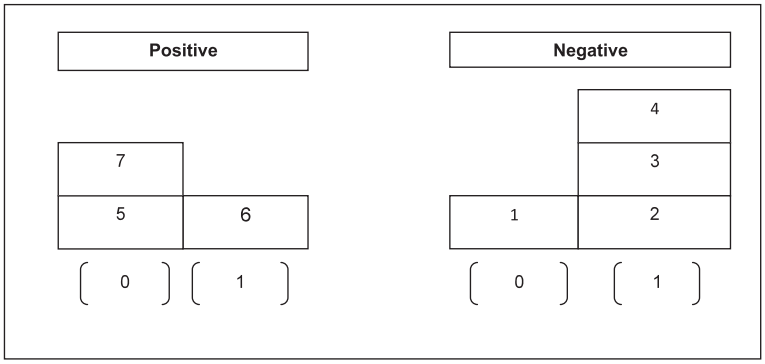

Fig. 2. Exang's constructed histograms.

through the MDL discretization algorithm and its bin number is decided based on its distinct values. Exang has two values, and therefore it has two bins.

4.1.3. Compute for each feature how it classifies the records in the training dataset

For each feature a_k, compute how it classifies the training dataset's records.
For each bin s in the $a'_k s$ histograms:

- Compute for each class i, the misclassification cost that is caused by classifying all the records in the s bin of $a'_k s$ histograms to class i as shown in Eq. (5):

$$MCS_{is} = \sum_{\forall(x,y)\in s} C_{y,i}. \tag{5}$$

- Based on the cost-sensitive majority rule, classify all the records in bin s to the class i which minimizes the misclassification cost.

4.1.4. Calculate for each feature the misclassification cost ratio it assigns to each record in the training dataset

Calculate the misclassification cost ratio of a record, whose true class is i, in the sth bin by feature a_k as shown in Eq. (6). Let $MCSR_{a_k,s,i}$ be the misclassification cost ratio caused by classifying to class i all the records in the s'th bin of feature a_k whose real class is not class i. The denominator of Eq. (6) represents the potential misclassification cost caused by classifying all the records whose class is not i to class$i$. The numerator of Eq. (6) represents the potential misclassification cost caused by not classifying all of the records to their true class. When we averaged the misclassification cost, which was caused by classifying all the records whose real class is i falsely, an equal weight was assigned to each one of the classes. However, a different weight could be assigned to correspond to the class distribution in that bin.

For the binary class problem, the misclassification cost ratio can be also be computed as shown in Eq. (7), where, for the sake of simplicity, we refer to the class that was not picked as class $\bar{y}$.

$$MCSR_{a_k,s,i} = \frac{\sum_{\forall(x,y)\in s} C_{y,i}}{\sum_{\forall(x,y)\in s} C_{y,i} + \left(\sum_{j=1}^{K} \sum_{\forall(x,y)\in s \wedge y=i} C_{y,j}\right) * \frac{1}{K-1}}, \tag{6}$$

$$MCSR_{a_k,s,i} = \left(\frac{\sum_{\forall(x,y)\in s} C_{y,i}}{\sum_{\forall(x,y)\in s} C_{y,i} + \sum_{\forall((x,y)\in s \wedge y=i} C_{y,\bar{y}}}\right). \tag{7}$$

The misclassification cost ratio by feature a_k of each one of the records in the sth bin that has not been classified to its true class is the complementary misclassification cost ratio which is calculated in Eq. (7).

The motivation for calculating the misclassification cost ratio of a certain feature is to supply the algorithm with the knowledge of whether or not the decision is sufficiently distinctive. That is to say, based on the distribution of the classes in a certain bin of an attribute, the CASH algorithm tries to estimate what is the likelihood that the algorithm's classification was correct. For example, if all the classes in a certain bin have the same value (same class), the likelihood that the algorithm has classified these records correctly is high. However, if the classes are distributed equally between all the classes, the likelihood that the algorithm classified these records to their true class is low.

Therefore, CASH prefers to select a feature's classification of a record with a small misclassification cost ratio in order to minimize the misclassification cost.

For example, let's examine how the feature Oldpeak classifies the misclassification cost ratio which it assigns to all the records in the training dataset that fall into the first bin ($[-\infty - 1.9]$).

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

The misclassification cost caused by classifying all the records in that bin as positive is computed by multiplying the false positive (FP) misclassification cost ($600) by the number of records in the bin $[-\infty - 1.9]$ of Oldpeak features with a negative class variable value (4) which is $1800. Then, the algorithm calculates in the same manner the misclassification cost caused by classifying all the records in that bin as negative. The misclassification cost caused by classifying all the records in that bin as negative is computed by multiplying the false negative (FN) misclassification cost ($1000) by the number of records in that bin with a positive class variable value (1), which is $1000. Therefore, the class that is chosen is negative since it leads to lower misclassification costs. Moreover, the misclassification cost ratio of each one of the records whose true class is negative in bin $[-\infty - 1.9]$ is 1000/ (1000 + 1800) = 0.35. Those records whose true class is positive are set to the complementary value, $0.65(1 - 0.35)$.

In a similar manner, we check how the Oldpeak feature classifies all the records in the synthetic training dataset that fall into the second bin ($[1.9 - \infty]$). The misclassification cost caused by classifying all the records in that bin as positive is computed by multiplying the false positive (FP) misclassification cost ($600) by the number of records in the bin $[1.9 - \infty]$ of Oldpeak features with a negative class variable value (1), which is $600. Then, the misclassification cost caused by classifying all the records in that bin as negative is computed by multiplying the false negative (FN) misclassification cost ($1000) by the number of records in the training dataset with a positive class variable value (2), which is $2000. Therefore, the class that is chosen is positive. Moreover, the misclassification cost ratio of each one of the records whose true class is positive in bin $[1.9 - \infty]$ is computed by Eq. (7), and it is 600/ (600 + 2000) = 0.23; those records whose true class is negative are set to $0.77(1 - 0.23)$.

Now let us examine how the feature Exang classifies all the records in the synthetic training dataset that fall into the first bin ([0]).The misclassification cost caused by classifying all the records in that bin as positive is computed by multiplying the false positive (FP) misclassification cost ($600) by the number of records in the bin [0] of the Exang feature with a negative class variable value (1), which is $600. The misclassification cost caused by classifying all the records in that bin as negative is computed by multiplying the false negative (FN) misclassification cost ($1000) by the number of records in the training dataset with a positive class variable value (2), which is $2000. Therefore, the class that is chosen is positive since it leads to a lower misclassification cost. Moreover, the misclassification cost ratio of each one of the records whose true class is positive in bin [0] is 600/(600 + 2000) = 0.23; those records whose true class is negative are $0.77(1 - 0.23)$.

In a similar manner, we check how the feature Exang classifies all the records in the synthetic training dataset that fall into the first bin ([1]). The misclassification cost caused by classifying all the records in that bin as positive is computed by multiplying the false positive (FP) misclassification cost ($600) by the number of records in bin [1] of the Exang feature with a negative class variable value (3), which is $1800. The misclassification cost caused by classifying all the records in that bin as negative is computed by multiplying the false negative (FN) misclassification cost ($1000) by the number of records in the training dataset with a positive class variable value (1), which is $1000. Therefore, the class that is chosen is negative. Moreover, the misclassification cost ratio of each one of the records whose true class is negative in bin [1.9] is 1000/ (1000 + 1800) = 0.357; those records whose true class is positive are set to $0.643(1 - 0.357)$.

4.2. Step 2: Create initial population of individuals

Given a population size of s, construct s chromosomes in a random fashion and insert them into the initial population. Each chromosome is represented as a bit string of dimension n, where n is the number of the features in the training dataset. Each chromosome in the population represents a candidate solution to the feature subset selection problem. If a bit is set to 1, it means that the corresponding feature is selected. However, if the feature's value is set to 0, it indicates that the corresponding feature is not chosen.

We added the following improvement to this phase: each randomly constructed chromosome undergoes an inspection before it is added to the initial population. If the total cost of this chromosome is larger than the average *a priori* cost, it is discarded and a new one created. Continue this process until a chromosome with a total cost that is smaller than the average *a priori* cost is created. It is then inserted into the initial population.

4.3. Step 3: Compute the fitness of each individual

For each feature's subset B in the population, compute the average total cost as follows:

4.3.1. Calculate the average misclassification cost

For each record i in the training dataset, calculate the average misclassification cost assigned by B and add it to $B's$ misclassification cost. Eq. (8) presents the calculation of the misclassification cost of a record by the feature's subset. Let $MCSR_{a_k,i}$ be the misclassification cost ratio which was assigned to the record r when classifying it to class i by feature$a_k \in B$. Again, when averaging the misclassification cost, which was caused by classifying the record falsely whose real class is y, an equal weight was assigned to each one of the classes.

$$\underset{a_k \in B}{minarg} \frac{1}{K-1} \bullet \sum_{j=1}^{K} \left(MCSR_{a_k} \bullet C_{y_i j} \right). \tag{8}$$

Then, the record's misclassification cost is added to the total misclassification cost of subset B.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

The average misclassification cost of the subset is computed by dividing the total misclassification cost of subset B by the number of features in the dataset.

The motivation for this calculation derives from the fact that CASH follows an optimistic approach. Given a feature's subset, CASH assigns to each record the classification with the highest confidence level that was assigned to it by one of the feature's subset.

4.3.2. Calculate the feature test cost of subset B

Calculate the test cost of subset B as presented in Eq. (2).

4.3.3. Compute the average total cost of subset B

The average total cost will be assigned the sum of the average misclassification cost and the features test costs. If the average total cost is larger than the average *a priori* cost of the training dataset, assign M to the total cost where M is a large constant. The motivation to assign a very large number is due to the assumption that we would not want the algorithm to sample any feature in case the average total cost is larger than the average *a priori* cost, and thus the classification decision would be based only on the distribution of the classes. In the selection method, a subset would not be picked to move to the next generation if its average total cost was equal to M.

Using the synthetic database, let us demonstrate this step on the subset which is composed of features Exang and Oldpeak.

Table 5 shows how each one of these two features has classified each one of the records in the training dataset. Additionally, Table 5 presents the misclassification cost ratio that was assigned in step 4.1.4 to each one of the records by each feature. The first column represents the features in the subset. The second column represents the records in the training dataset. The third and fourth represent the classification and the misclassification cost ratio that the feature assigned to the record.

First, the algorithm computes the average misclassification cost of this subset. In order to demonstrate the average misclassification cost calculation, first let us see how it operates on record 1. While the misclassification cost ratio for Oldpeak is 0.35, for Exang it is 0.77. Therefore, the estimated misclassification cost of this subset is calculated by multiplying the minimum misclassification cost ratio of these two features (0.35) with the FP misclassification cost ($600) {since record 1's real class is negative}, which is equal to $350. Then, the misclassification cost of all the records in the training dataset will be calculated in the same manner as follows:

Since the real class of records 2–4 is negative, their misclassification cost is the FP cost ($600). Since the minimum misclassification cost of these three records is 0.35, the misclassification cost that this subset assigns to them is the product of the FP cost ($600) and the minimum misclassification cost ratio (0.35), which is 350$.

Now let's examine how this subset classifies records 5–7. Since the true class of these records is positive, their misclassification cost is the FN cost ($1000). Since the minimal misclassification cost ratio of these three records is 0.23, the misclassification cost that this subset assigns to each of these records is the multiplication of the FN cost ($1000) and the misclassification cost ratio (0.23) which is $230.

The sum of all the misclassification costs of all the records assigned by this subset is:

$\$210 * 4 + \$230 * 3 = \$1530$. However, since the CASH algorithm uses the average total cost as an evaluation measurement, we need the average misclassification cost. Hence, we divide the misclassification cost that was assigned by this subset to all the training dataset's records ($1530) by the number of records in the training dataset (7), which is equal to $218.57.

After calculating the average misclassification cost, the test cost of this subset is calculated. By following Eq. (2) we can see that this feature's subset test cost is computed as follows: $1 + 1 + 87.3 - 1 = 88.3$. In other words, since Oldpeak and Exang belong to the same group, the test cost before discount will only be assigned to Oldpeak ($87.3), while the test cost after

Table 5
Oldpeak and Exang: classification and misclassification cost ratio of the record in the training dataset.

Feature	Record	Classification class	Misclassification cost ratio
Oldpeak	1	Negative	0.35
	2	Negative	0.35
	3	Positive	0.77
	4	Negative	0.35
	5	Negative	0.65
	6	Positive	0.23
	7	Positive	0.23
Exang	1	Positive	0.77
	2	Negative	0.35
	3	Negative	0.35
	4	Negative	0.35
	5	Positive	0.23
	6	Negative	0.54
	7	Positive	0.23

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

discount will be assigned to Exang ($1).Therefore, the test cost which is assigned by this subset is the sum of these two test costs, which is equal to $88.3($87.3 + $1).

Consequently, the average total cost of this subset is set to the summation of the average misclassification cost and the costs of the subset's features, which is $306.87($218.57 + $88.3).

In this above example, the contribution of the group cost is significant. Let's compute the average total cost of this subset while ignoring the group cost, and assume therefore that these two features have no common cost. This means that Exang's cost value does not depend on whether the Oldpeak test is in the same subset with it, and vice versa. In this scenario, the test costs of this subset will be the sum of the "before discount costs" of both of these features, which is $174.6 (the sum of Oldpeak's cost ($87.3) and Exang's cost ($87.3)). Therefore, the average total cost of this subset, when the group cost was not considered, is the sum of the average misclassification cost ($218.57) and the test cost (which is now equal to $174.6), which is $393.17. Since the database's average *a priori* cost ($342.85) is smaller than the average total cost of this subset ($393.17), the algorithm will set the average misclassification cost of this subset to a big number, M. Moreover, the algorithm assigns zero probability to be picked in the selection method of the genetic algorithm for the next generation.

Therefore, as this example demonstrates, when group cost is not taken into account, the result of the CASH algorithm is different.

4.4. Step 4: Apply genetic algorithm to the initial population

The traditional crossover and mutation operators have be applied. (See Section 5.1 for more details of the implementation setting.).

5. Experiments

A variety of experiments were conducted to compare the overall performance of the CASH algorithm with existing state-of-the-art algorithms. Table 6 shows a summary of all of the runs that were executed in these experiments.

In this section we first discuss the general implementation and the evaluation measurements that were used in the experiments. Subsequently, we describe the methodology we implemented in the experiments. Finally, we present and discuss the results of the experiments..

5.1. Implementation setting

CASH was implemented in Java using the Weka framework. We used Fayyad and Irani's MDL descretization algorithm (1993) for the feature discretization assignment in its Weka implementation [13]. Since we used the filter approach, which is not associated with any particular classifier, we had to choose a classifier for the classification assignment. We chose a decision tree as the classifier since cost-sensitive decision tree classifiers are used by the majority of the related studies (see Section 2.2 for the literature survey of cost-sensitive algorithms). Specifically, decision tree classifiers were used in the research studies to which we compare our algorithm's performance [14,43,44]. We used the J48 classifier, a Java implementation in WEKA data mining applications of the C4.5 decision tree algorithm that Quinlan [33] introduced. In order to make our classifier cost-sensitive to misclassification costs, we used a meta-learner implemented in Weka MetaCostClassifier [9] since the J48 Weka implementation does not take into account the misclassification cost in the classification task.

Another important decision determined the genetic algorithm's configuration. We decided to base the majority of the parameter settings on those that were used by Vidrighin et al. [43,44]. The motivation for choosing nearly the same configuration was to show that the superiority of the new algorithm's performance over previous algorithms was due to its new fitness function rather than the genetic algorithm's parameters. Table 7 presents the parameters that were used by all the algorithms that utilized a genetic algorithm as their search method.

5.2. Experiment setting

Typically, machine learning researchers use datasets from the UCI repository. Table 8 presents the 11 datasets from the UCI datasets that were used in previous research studies for comparing cost-sensitive algorithms. Following previous research studies, the cost of each attribute was uniformly distributed between $0 and $100. We compared the six algorithms (CASH, ProICET GA + META, GA + META + CS − ID3, csDT:csf = 0 and csDT:csf = 1) on the training set. It should be noted that the group discount of attributes is not considered since the ProICET algorithm does not support this type of cost. Missing values were replaced with the most frequent attribute's value of the instance's class. The datasets are originally binary or converted to binary classification problems in a "one against all" fashion, such that the value "positive" is assigned to the

Table 6
Summary of the experiments that were conducted.

Number of algorithms	Dataset	Number of iterations	Cost matrix	Total number of runs
6	11	10	17	11,220

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Table 7
The parameters used by all the algorithms that implemented the genetic algorithm as their search method.

Parameter	Value
Replacement strategy	Single population technique
Selection method	Elitism selection and rank selection
Population size	50
Number of generations	200
Probability of crossover	0.6
Probability of mutation	0.2
Number of crossover points	4
Size of elitism	4

Table 8
The properties of the datasets.

Dataset	Application area	Number of attribute	Number of records	Class dist. (N/P)	Missing values
Pima	Life	8 + 1	768	500/268	None
Bupa	Life	5 + 1	345	169/176	None
Thyroid	Life	24 + 1	3726	3488/238	None
Hepatitis	Life	20 + 1	155	32/123	Yes
Breast	Life	10 + 1	699	458/241	Yes
SPECT	Life	22 + 1	267	55/212	None
Kr-Vs-Kp	Games	36 + 1	3196	1534/ 1662	None
Cars	Manufacturing	6 + 1	1728	1210/384/69/65	None
Voting	Social	16 + 1	435	267/168	288
Tic-Tac-Toc	Games	9 + 1	958	333/625	None
Ecoli	Life	7 + 1	336	230/102	None

most frequent class and "negative" to all other classes. Additionally, we compare the algorithms' performances on 17 cost matrices. We assume that the diagonal entries of the misclassification cost matrices are zeros and the FP and FN costs vary in the different experiments. Specifically, we differentiate between two cases. In the first case, the FP cost is equal to the FN cost and it varies from 10 to 1000. In the second case we examine the following FP to FN ratios: 1/2, and 1/3.

In this experiment, we used the 2×5 cross validation method as proposed by Dietterich [8]. The 2×5 CV procedure is known to have a smaller type I error than the 10 CV procedure. Each split was conducted as follows: we randomly chose 50% of the records in the database as the training set; the remaining 50% of the records were added to the testing set. For each split, we switched between the training set and the testing set. Therefore, each dataset was examined a total of 10 times.

5.3. Algorithms used for comparison

In order to evaluate the performance of CASH, we compared it to three algorithms:

1. csDTy [14] is sensitive to both test and misclassification costs. It employs a cost-sensitive decision tree to obtain a setting for the cost-scale factor (*csf*) that adjusts the strength of the bias towards lower cost attributes. The cost-scale factor will usually assume a value between 0.0 and 1.0. With the cost-scale factor increasing, the impact of the test costs on the features that will be picked by the decision tree will also increase. When the cost-scale factor is set to zero, the costs are ignored and the model selection is equivalent to the model selection of the (non cost-sensitive) C4.5. In this study we compared CASH's performance to csDT with a cost-scale factor of 0.0 and 1.0. The reason for not considering all the other values in the range is that, from the results that were published by Freitas [14], the best result for all the misclassification cost matrices was always obtained when the cost-scale factor was set to either 0.0 or 1.0. Since csDT's code is not available to the public, and we could not get it, we implemented it. In order to verify the correctness of our implementation, we compared our results with those originally published by Freitas [14]. It is important to note that we used the same dataset (Pima dataset) and cost matrices as well as the same cost and group files. Moreover, the same cross-validation folds are implemented. The results were similar to the published results. Unfortunately, we do not have access to the original implementation of the csDT_csf1 and csDT_csf0 or to their detailed accuracy performance in each fold. Therefore, we cannot use paired *t*-test. Alternatively, we could use the reported confidence interval as a basis to verify our implementation. However, simply checking that the two 95% confidence intervals do not overlap is not sufficient. Instead, we use the procedure developed by Afshartous and Preston [1] and conclude that, with reference to the csDT_csf1 algorithm, in 10 out of 13 cases the null hypothesis that the two implementations are identical cannot be rejected. In the remaining cases our implementation reports better results, i.e., the mean of our algorithm is at least as good as the originally reported mean. With reference to the csDT_csf0 algorithm, in 11 out of 13 cases the null hypothesis that the two imple-

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

mentations are identical cannot be rejected. In the remaining cases our implementation reports better results. Using this conservative approach ensures that we do not unfairly downgrade the reference algorithms. Thus any results that indicate that our new algorithm outperforms the reference algorithm will still hold even if we used the original implementation.

2. ProICET [43,44] was used in the second sub-experiment. Although it is also sensitive to both test and misclassification costs, it does not take into account the group cost. ProICET is a modified version of the ICET algorithm [41]. As noted previously, ICET is a cost-sensitive algorithm that uses a genetic algorithm as its search method. Also, it uses the EG2 algorithm [30], which is a modified version of C4.5. EG2 uses the information cost function (ICF) as its splitting criterion. Moreover, in ICET, the n costs, Ci, are bias parameters and do not represent the true test cost of the attribute. The *n* cost's role is to prevent the decision tree classifier from becoming trapped in local optimum by assigning a different cost to each feature based on past trial performances. There are a number of differences between the ICET and ProICET algorithms, such as replacement strategy, that stem mainly from the genetic components of the algorithms. For example, while ICET uses the multiple population technique, ProICET uses a single population. Moreover, the percentage of training examples in evaluating the fitness score of an individual was changed (in ProICET it was set to 70% as opposed to 50% in ICET).
3. GA + META is a wrapper feature selection method that uses the decision tree as the fitness function. The decision tree that is employed is a J4.8 classifier, which is a Java implementation of the C4.5 DT algorithm introduced by Quinlan. As a search method it uses a genetic algorithm. However, while GA + META is sensitive to misclassification costs, it is not sensitive to test costs.
4. GA + META + CS-ID3 is similar to the GA + META algorithm since it is also a wrapper feature selection method that uses a decision tree algorithm as the fitness function and a genetic algorithm as a search method. However, while the splitting criterion used by GA + META is the traditional information gain, the splitting criterion of GA + META + CS was modified according to the CS-ID3 algorithm, which makes this algorithm also sensitive to test costs. CS-ID3 [39,40] is a cost-sensitive decision tree algorithm with the following splitting criterion, $\frac{(\Delta I_i)^2}{C_i}$, where ΔI_i is the information gain (or gain ratio) for attribute i and C_i is the cost of attribute i.

We implemented GA + META + CS-ID3 by modifying C4.5 so that it selects the attribute that maximizes the CS-ID3's splitting criterion. As CASH, all of the five algorithms in this study consider the misclassification costs by using a meta-learner implemented in Weka MetaCostClassifier [9].

5.4. Evaluation measurement

In order to compare the different feature selection algorithms, we employed as the evaluation measurement the one commonly used in this domain: the average total cost, which is composed of the sum of the average misclassification cost and the test cost of the tests. We decided to average the calculation for each instance in the database since the evaluation measurement, used by the majority of the related studies, is the average total cost. In particular, the average total cost measurements were used in the research studies [14,43,44], to which we compared CASH's performance.

The lower the evaluation measure value is, the better the algorithm performs. Based on the training set, the feature selection algorithms select a subset of features. Then, features that were not selected were eliminated from the corresponding training and testing set. Afterward, a cost-sensitive decision tree was induced on each of the training sets and its performance was evaluated on the corresponding test set.

Additionally, we compared the execution time of each algorithm.

5.5. Statistical analysis

We used hypothesis tests in order to examine if CASH performs best over multiple datasets. As García et al. [15] have suggested, we used the Friedman aligned ranks for testing the differences among the total cost means that were obtained by each algorithm. A value of location is computed as the average performance achieved by all algorithms in each dataset and misclassification cost matrix. The null-hypothesis is that all of the algorithms perform the same and the observed differences are merely random. Once the Friedman aligned rank test rejects the hypothesis of equivalent algorithms, the specific differences among the algorithms are examined using Finner's step-down procedure. In the case of only two classifiers, we use the Wilcoxon signed ranks test in order to reject the null hypothesis as proposed by Demsar [7].

5.6. Experiment results

Table 9 presents the average total cost obtained in all of the runs. The first column represents the datasets that were examined. The second column represents the algorithms that were compared. Then, each of the next 17 columns shows the average total cost on different cost matrices. As can be seen from Table 9, the average total cost of the CASH algorithm tends to be better than that of all the other algorithms. There are 14 misclassification cost matrices in which the CASH algorithm outperforms ProICET and GA + META on *all* datasets in terms of total cost. There are only two datasets (Breast and Bupa) in which ProICET or GA + META achieved lower average total costs in three out of 17 misclassification cost matrices (200–600, 600–200 and 400–200). In addition, we can see from Table 9 that in comparison to GA + META + CS-ID3 algorithm,

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Table 9
Comparing cost-sensitive algorithms: summary of experimental results.

Dataset	Algorithms	Misclassification Cost Matrices (FP cost – FN cost)																
		10 – 10	20 – 20	50 – 50	100 – 100	200 – 200	400 – 400	500 – 500	10 – 100	100 – 10	50 – 150	150 – 50	100 – 200	200 – 100	200 – 400	400 – 200	200 – 600	600 – 200
Pima	csDT:csf = 0	232.5	234.8	241.8	253.3	276.4	322.7	345.8	207.1	3.6	251.3	120.0	282.7	168.3	319.3	194.8	315.6	162.5
	csDT:csf = 1	3.2	6.4	16.0	33.6	69.2	176.0	159.7	6.7	3.5	18.7	33.5	68.1	31.9	136.1	63.9	136.1	79.9
	ProICET	136.0	145.0	153.0	136.0	149.2	195.0	263.1	153.0	120.0	146.0	122.0	132.0	148.4	195.0	194.0	186.9	221.4
	CASH	0.9	1.8	4.6	33.6	69.2	176.0	145.0	6.7	3.3	17.0	33.5	56.3	33.0	93.2	79.0	110.0	69.8
	GA + META	141.0	145.0	153.0	165.0	189.5	239.0	280.9	93.0	37.3	151.0	166.0	201.0	167.0	237.0	269.0	234.9	210.7
	GA + CS-ID3	3.2	6.4	16.0	33.6	69.2	176.0	159.7	6.7	3.3	17.0	33.5	84.0	49.1	133.5	81.1	136.1	73.9
Bupa	csDT:csf = 0	121.7	125.8	138.2	158.8	199.9	282.2	323.4	67.1	48.0	164.1	110.3	180.3	171.6	261.3	252.8	345.9	315.8
	csDT:csf = 1	5.0	9.8	24.6	49.2	98.5	197.0	246.2	52.0	48.0	78.0	72.0	104.0	96.0	208.1	196.0	312.1	287.9
	ProICET	101.0	114.0	123.0	139.0	187.6	244.0	274.5	121.0	125.0	147.0	125.0	153.0	169.8	199.0	136.0	260.1	246.4
	CASH	3.5	7.0	17.4	34.9	69.8	146.0	192.8	6.5	3.5	17.0	32.6	66.8	34.9	138.0	196.0	306.1	293.9
	GA + META	112.0	115.0	121.0	133.0	155.4	201.0	223.3	44.0	26.8	55.0	100.0	138.0	104.3	183.0	152.0	216.2	169.9
	GA + CS-ID3	5.0	10.1	24.6	50.5	101.0	201.9	252.4	52.0	48.0	78.0	72.0	104.0	96.0	208.1	196.0	312.1	287.9
Thyroid	csDT:csf = 0	112.6	112.6	112.7	113.0	113.4	114.3	114.8	102.5	145.3	106.0	128.9	108.7	121.6	109.4	122.8	108.0	131.5
	csDT:csf = 1	105.3	105.4	105.6	106.0	1.0	108.0	108.6	7.5	137.7	109.4	124.2	104.2	108.0	105.4	118.2	111.8	127.7
	ProICET	116.0	111.0	110.0	110.0	112.0	112.0	112.3	112.0	110.0	111.0	111.0	110.0	110.5	118.0	89.0	110.5	112.2
	CASH	0.8	1.5	3.8	7.5	15.1	30.1	37.6	7.5	0.8	3.8	11.3	15.1	7.5	30.1	15.0	45.2	15.1
	GA + META	111.0	111.0	111.0	112.0	112.2	113.0	114.1	111.0	107.0	108.0	108.0	111.0	109.9	112.0	112.0	113.1	110.5
	GA + CS-ID3	109.9	110.6	113.5	111.1	111.8	112.2	112.3	101.9	132.9	104.9	122.8	108.1	117.2	109.1	118.6	107.0	125.8
Hepatitis	csDT:csf = 0	105.7	106.8	110.1	115.6	126.5	148.3	159.2	14.0	140.0	73.1	150.1	109.6	150.7	127.6	172.0	114.6	190.9
	csDT:csf = 1	106.4	108.0	113.0	121.2	137.5	170.3	186.6	14.0	211.7	29.7	210.5	74.2	189.5	101.5	216.6	92.4	280.1
	ProICET	95.8	91.3	97.6	96.4	97.5	115.0	108.1	98.0	104.0	107.0	119.0	107.0	102.9	109.0	109.0	106.3	104.8
	CASH	1.4	2.8	7.0	9.2	18.5	36.9	55.8	7.5	8.5	12.0	8.4	13.1	16.2	26.3	33.0	42.3	46.2
	GA + META	102.0	104.0	111.0	123.0	146.0	193.0	215.7	115.0	133.0	132.0	111.0	136.0	139.7	173.0	177.0	214.1	219.2
	GA + CS-ID3	105.1	108.1	112.9	122.6	142.0	195.9	200.4	27.8	117.8	112.7	150.3	121.0	162.7	146.0	197.5	162.7	225.3
Voting	csDT:csf = 0	137.7	140.7	149.9	165.0	195.4	256.2	286.6	2.8	62.1	86.0	13.9	27.9	120.6	55.7	152.9	58.8	167.7
	csDT:csf = 1	104.1	107.1	116.2	131.3	161.6	222.1	252.3	2.8	33.9	78.9	13.9	27.9	111.4	55.7	145.6	55.7	135.1
	ProICET	91.5	98.5	107.0	121.0	132.4	191.0	218.8	111.0	95.2	107.0	112.0	116.0	132.3	139.0	169.0	208.4	208.4
	CASH	2.7	5.6	13.9	27.9	55.7	111.0	139.3	2.8	7.2	38.0	13.9	27.9	55.7	55.7	116.0	55.7	144.8
	GA + META	65.4	68.4	77.3	92.2	122.0	182.0	211.4	163.0	30.6	74.0	22.3	44.5	91.8	72.9	128.0	73.5	129.1
	GA + CS-ID3	93.2	98.5	100.6	115.8	146.2	207.0	242.2	2.8	52.2	38.0	75.6	27.9	93.4	55.7	127.4	60.0	167.5
Car	csDT:csf = 0	216.1	216.4	217.1	218.3	220.7	225.5	227.9	153.7	284.5	196.7	253.8	253.8	196.7	256.2	202.0	194.9	256.2

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

	csDT:csf = 1	297.6	298.3	300.4	303.9	310.8	324.6	331.5	66.4	293.9	253.7	299.2	273.0	313.7	286.7	316.3	273.0	303.5
	ProICET	136.0	131.0	136.0	142.0	152.3	177.0	185.1	141.0	138.0	140.0	146.0	155.0	144.3	173.0	173.0	160.1	185.3
	CASH	3.0	6.0	15.0	30.0	60.0	120.0	149.9	3.0	7.0	35.0	15.0	30.0	7.0	60.0	120.0	60.0	140.0
	GA + META	176.0	177.0	182.0	189.0	203.8	234.0	248.3	59.0	152.0	183.0	187.0	194.0	192.5	211.0	210.0	216.8	211.3
	GA + CS-ID3	191.6	187.7	197.6	205.0	219.9	249.7	264.6	114.2	223.0	175.2	238.4	195.9	239.3	222.3	245.8	235.5	250.6
Kr-Vs-Kp	csDT:csf = 0	264.7	264.7	265.0	265.3	266.1	267.6	268.3	220.9	256.0	263.3	227.3	268.5	251.7	269.9	252.9	266.3	230.6
	csDT:csf = 1	493.7	494.2	495.8	498.4	493.2	506.8	493.2	244.1	378.7	459.1	423.0	496.0	442.6	449.4	503.8	473.9	435.0
	ProICET	217.0	222.0	223.0	205.0	200.7	217.0	221.8	226.0	225.0	221.0	216.0	212.0	213.7	229.0	229.0	222.0	223.9
	CASH	4.8	9.6	23.9	47.8	95.6	190.0	239.2	4.8	5.2	26.0	23.9	47.8	52.2	95.6	104.0	95.6	104.4
	GA + META	260.0	259.0	259.0	260.0	260.3	262.0	262.5	260.0	236.0	260.0	251.0	258.0	260.3	260.0	262.0	253.7	262.7
	GA + CS-ID3	238.6	258.1	259.4	261.7	266.3	275.4	257.8	186.1	206.2	197.7	212.3	239.1	236.7	246.5	247.1	210.5	218.1
Tic-Tac-Toc	csDT:csf = 0	229.2	229.8	231.6	234.7	240.7	252.8	258.9	118.9	171.8	217.1	237.6	224.7	239.8	233.6	249.9	20.6	257.7
	csDT:csf = 1	267.2	267.2	277.3	289.9	315.2	365.7	391.0	3.5	53.9	35.0	288.0	121.5	333.6	156.1	370.3	107.2	354.7
	ProICET	127.0	125.0	120.0	137.0	172.6	222.0	240.8	134.0	134.0	140.0	132.0	156.0	166.0	212.0	186.0	194.1	193.1
	CASH	3.5	6.9	17.3	34.7	69.3	139.0	173.3	3.5	6.5	33.0	17.3	34.7	65.3	69.3	129.0	69.8	130.2
	GA + META	191.0	193.0	199.0	209.0	228.4	268.0	287.3	122.0	122.0	224.0	153.0	198.0	224.6	221.0	256.0	190.5	283.3
	GA + CS-ID3	196.9	202.8	206.6	218.7	242.9	291.4	315.6	3.5	114.2	77.0	220.5	155.4	252.9	183.9	287.1	123.9	288.9
E. coli	csDT:csf = 0	130.2	130.6	131.8	133.8	137.9	146	184.5	127.1	88.6	134.4	146.6	140.6	151.7	145	156.8	143.3	155.6
	csDT:csf = 1	4.3	8.7	21.3	43.6	87.1	174.3	217.9	5.7	4.3	21	28.7	56	44.6	112.9	87.1	112.9	87.1
	ProICET	128	120	129	129	126.9	137	136.8	122	128	124	118	130	122.5	130	123	129.8	138.5
	CASH	4.3	5.7	21.3	41	80.5	136	182	5.7	4.3	21	28.7	56	42.6	100	85	108.6	85.4
	GA + META	136	136	139	141	147	158	164	123	124	133	131	143	150	151	160	146.1	150.6
	GA + CS-ID3	84.3	85.7	89.7	96.5	98.5	136	150.5	85.7	86.2	92.9	91.7	101.8	98.5	120.7	114	122.7	117.9
Spect	csDT:csf = 0	241.7	243.1	247.5	254.8	319.1	298.3	312.8	1.5	355.8	42.3	351.6	187.5	337.7	187.5	337.7	68.1	399.8
	csDT:csf = 1	1.5	3.1	8	18	30.7	61.4	76.8	1.5	8.5	24.9	8	15.4	30.7	30.7	69.4	30.7	91.7
	ProICET	154	139	124	133	162.5	161	173.2	153	147	146	122	157	176.1	177	140	173.5	149
	CASH	1.5	3.1	8	15.7	41.3	101	130.5	1.5	8.5	23	8	15.4	30.7	30.7	61	54.6	102.4
	GA + META	178	177	185	193	209.6	239	262.5	59	252	241	189	198	245.6	217	270	224.5	292.2
	GA + CS-ID3	84.2	85.3	89.4	95.4	107.9	137.9	145.2	85.6	85.2	92.5	91.8	98.2	101.8	120.6	133.5	120.8	118.1
Breast	csDT:csf = 0	8	168.9	170.7	173.8	180	192.3	198.5	93.9	153.2	166.3	150.3	152.1	152.7	161.7	163.3	191.6	176.7
	csDT:csf = 1	31	32.1	35.2	40.4	70.9	71.8	82.2	36.9	96	244	58.2	44.2	48.9	106.7	48.9	106.7	85.4
	ProICET	116	110	114	105	131	115	128.1	110	120	120	123	97.5	112.5	106	116	117.5	126
	CASH	3.4	6.9	16.4	65.5	65.2	114	122.4	35	65.5	90	47.7	41.9	101.7	73.5	166	166.8	196.4
	GA + META	132	133	134	137	143.5	156	161.7	126	138	144	136	137	142	146	152	154.1	165
	GA + CS-ID3	150.1	105.7	152.6	155.7	161.7	173.5	179.9	98.3	87.6	137.2	142.9	44.2	83.5	150	61.8	169.9	180

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

there are 168 cases out of 187 cases where CASH outperformed the GA + META + CS-ID3 algorithm and 14 cases when the two algorithms achieved the same average total cost. Also, CASH outperformed csDT with csf = 1 in 141 cases; in 29 cases they showed the same performances. Moreover, compared to csDT where csf = 0, CASH outperformed in all of the cases except 3 where csDT with csf = 0 outperformed CASH. In 5 cases they both showed the same performances.

Additionally, Figs. 3 and 4 represent the performance of each algorithm in each of the datasets, given misclassification matrix cost when FP cost and FN cost are equal to 200 and when the FP cost is equal to 200 and FN cost is equal to 100, respectively. We can see visually from these graphs that the CASH algorithm outperforms the rest of the algorithms with significant differences in the majority of the datasets.

In the statistical tests, the null-hypothesis that all algorithms perform the same and the observed differences are merely random was rejected with $T(5) = 365.2$ and $p < 1\%$. Once the Friedman aligned ranks test rejects the hypothesis of equivalent algorithms, the specific differences among the algorithms are examined using Finner's step-down procedure. The results indicated that CASH outperforms all other five algorithms with $p < 1\%$.

Analysis of the number of features in each chosen subset indicates that the CASH algorithm tends to pick smaller subsets than the other algorithms. Since CASH is sensitive to the ratio between the misclassifications and tests costs, when the misclassification costs are lower compared to the test costs, the CASH algorithm does not pick features at all. And as the ratio gets higher, the number of chosen features increases. These results are consistent with those obtained in the first set of experiments.

5.7. The effect of the average a priori cost

In this section we compare two variations of the proposed algorithm. We explored the effect of the average *a priori* cost on CASH's performance by removing all of the average *a priori* cost considerations in the algorithm. CASH usage of the average *a priori* cost is twofold, firstly, because the average *a priori* cost serves as an indication of when a feature's subset should not be obtained. That is to say, if the average *a priori* cost is lower than the average total cost achieved by a feature's subset, CASH will not be interested in moving this subset to the next generation. Secondly, in generating the initial population, each ran-

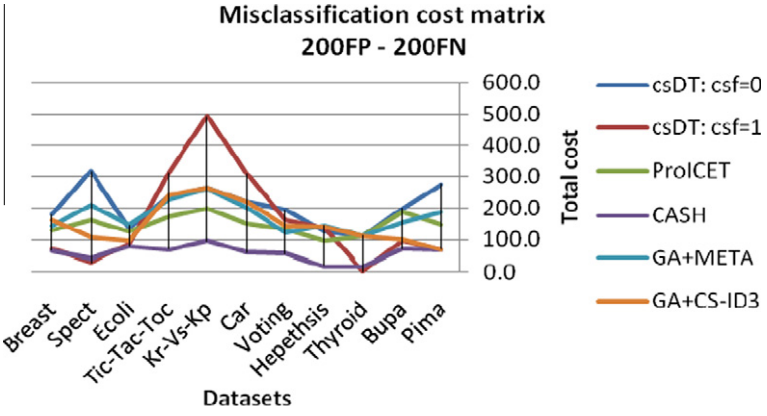

Fig. 3. The total cost that was obtained for each algorithm in each dataset, given a misclassification cost matrix of FP cost = FN cost = 10.

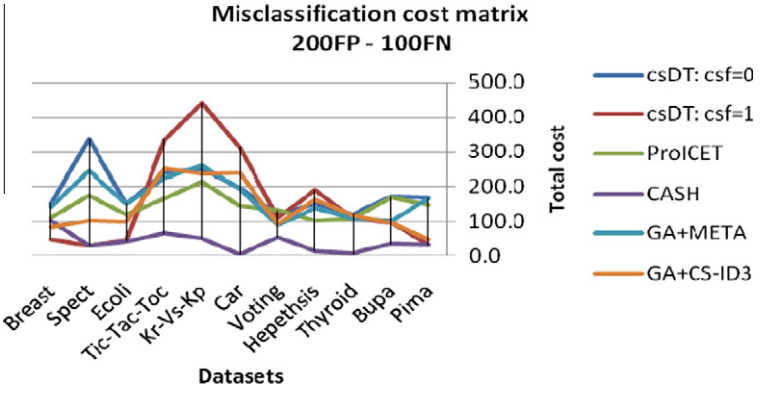

Fig. 4. The total cost that was obtained for each algorithm in each dataset, given a misclassification cost matrix where FP = 200 and FN = 10.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Table 10
Comparing the CASH algorithm with two different average total cost computation methods: summary of experimental results.

Dataset	Algorithms	Misclassification cost matrices (FP cost – FN cost)																
		10 – 10	20 – 20	50 – 50	100 – 100	200 – 200	400 – 400	500 – 500	10 – 100	100 – 10	50 – 150	150 – 50	100 – 200	200 – 100	200 – 400	400 – 200	200 – 600	600 – 200
Pima	CASH – original	0.9	1.8	4.6	33.6	69.2	176.0	145.0	6.7	3.3	17.0	33.5	56.3	33.0	93.2	79.0	110.0	69.8
	CASH –modified	21.2	20.3	29.1	17.6	73.4	170.3	205.5	2.9	29.5	16.0	34.0	65.9	31.9	137.9	68.9	161.6	63.9
Bupa	CASH – original	3.5	7.0	17.4	34.9	69.8	146.0	192.8	6.5	3.5	17.0	32.6	66.8	34.9	138.0	196.0	306.1	293.9
	CASH – modified	78.0	82.1	96.7	117.4	176.5	258.8	300.0	58.0	47.0	77.0	95.9	141.0	135.7	242.1	227.3	335.8	288.7
Thyroid	CASH – original	0.8	1.5	3.8	7.5	15.1	30.1	37.6	7.5	0.8	3.8	11.3	15.1	7.5	30.1	15.0	45.2	15.1
	CASH – modified	0.8	1.6	3.9	27.7	93.6	102.3	105.6	7.8	0.8	3.9	31.5	99.6	28.0	96.9	105.5	105.5	100.5
Hepatitis	CASH – original	1.4	2.8	7.0	9.2	18.5	36.9	55.8	7.5	8.5	12.0	8.4	13.1	16.2	26.3	33.0	42.3	46.2
	CASH – modified	45.4	46.9	86.4	111.2	123.6	133.0	145.2	58.8	78.9	92.3	96.5	108.3	117.0	123.3	139.6	137.5	151.5
Voting	CASH – original	2.7	5.6	13.9	27.9	55.7	111.0	139.3	2.8	7.2	38.0	13.9	27.9	55.7	55.7	116.0	55.7	144.8
	CASH – modified	2.7	5.6	13.9	27.9	55.7	111.4	139.3	2.8	7.2	37.5	13.9	27.9	67.0	55.7	114.1	55.7	133.5
Car	CASH – original	3.0	6.0	15.0	30.0	60.0	120.0	149.9	3.0	7.0	35.0	15.0	30.0	7.0	60.0	120.0	60.0	140.0
	CASH – modified	3.0	6.0	15.0	30.0	60.0	119.9	149.9	3.0	7.0	35.0	15.0	30.0	63.9	60.0	127.5	60.0	138.6
Kr-Vs-Kp	CASH – original	4.8	9.6	23.9	47.8	95.6	190.0	239.2	4.8	5.2	26.0	23.9	47.8	52.2	95.6	104.0	95.6	104.4
	CASH – modified	17.7	57.7	65.9	68.5	166.9	303.2	273.8	22.5	10.0	81.5	226.2	118.7	83.8	217.4	202.6	202.7	138.6
Tic-Tac-Toc	CASH – original	3.5	6.9	17.3	34.7	69.3	139.0	173.3	3.5	6.5	33.0	17.3	34.7	65.3	69.3	129.0	69.8	130.2
	CASH – modified	3.5	7.0	17.5	34.9	69.8	169.9	240.2	3.5	6.5	32.5	17.5	34.9	66.0	69.8	131.0	69.8	145.1
E. coli	CASH – original	4.3	5.7	21.3	41.0	80.5	136.0	182.0	5.7	4.3	21.0	28.7	56.0	42.6	100.0	85.0	108.6	85.4
	CASH – modified	4.4	8.7	24.0	50.1	91.1	166.8	196.4	5.6	4.4	21.8	28.2	58.5	43.6	108.8	91.9	112.0	92.9
Spect	CASH – original	1.5	3.1	8.0	15.7	41.3	101.0	130.5	1.5	8.5	23.0	8.0	15.4	30.7	30.7	61.0	54.6	102.4
	CASH – modified	3.5	3.1	7.7	15.7	41.4	83.6	87.6	1.5	8.5	23.0	47.3	15.4	34.9	34.9	91.0	35.3	112.5
Breast	CASH – original	3.4	6.9	16.4	65.5	65.2	114.0	122.4	35.0	65.5	90.0	47.7	41.9	101.7	73.5	166.0	166.8	196.4
	CASH – modified	7.4	8.4	11.6	16.9	39.1	75.7	84.0	7.4	10.4	89.9	48.0	23.6	21.6	50.6	62.9	62.1	80.8

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Table 11
Comparing the CASH algorithm with two different average total cost computation methods: summary of experimental results.

Dataset	Algorithms	Misclassification cost matrices (FP cost – FN cost)																
		10 – 10	20 – 20	50 – 50	100 – 100	200 – 200	400 – 400	500 – 500	10 – 100	100 – 10	50 – 150	150 – 50	100 – 200	200 – 100	200 – 400	400 – 200	200 – 600	600 – 200
Pima	CASH original	0.9	1.8	4.6	33.6	69.2	176.0	145.0	6.7	3.3	17.0	33.5	56.3	33.0	93.2	79.0	110.0	69.8
	CASH modified	6.8	6.4	19.5	19.4	74.9	133.6	163.0	3.2	29.4	16.0	34.0	89.3	31.9	150.8	63.9	168.3	63.9
Bupa	CASH original	3.5	7.0	17.4	34.9	69.8	146.0	192.8	6.5	3.5	17.0	32.6	66.8	34.9	138.0	196.0	306.1	293.9
	CASH modified	5.0	44.8	67.5	89.5	138.8	225.0	268.1	52.0	48.0	72.0	78.0	104.0	117.6	208.1	212.4	312.1	287.9
Thyroid	CASH original	0.8	1.5	3.8	7.5	15.1	30.1	37.6	7.5	0.8	3.8	11.3	15.1	7.5	30.1	15.0	45.2	15.1
	CASH modified	0.8	1.6	3.9	7.8	15.5	31.0	105.6	7.8	0.8	3.9	11.6	15.5	7.8	31.0	15.5	46.6	15.5
Hepatitis	CASH original	1.4	2.8	7.0	9.2	18.5	36.9	55.8	7.5	8.5	12.0	8.4	13.1	16.2	26.3	33.0	42.3	46.2
	CASH modified	1.4	2.8	7.0	14.0	28.0	55.9	69.9	30.9	78.2	18.5	21.0	28.0	14.0	55.9	28.0	83.9	50.4
Voting	CASH original	2.7	5.6	13.9	27.9	55.7	111.0	139.3	2.8	7.2	38.0	13.9	27.9	55.7	55.7	116.0	55.7	144.8
	CASH modified	2.7	5.6	13.9	27.9	55.7	111.4	139.3	2.8	7.2	37.5	13.9	27.9	55.7	55.7	114.1	55.7	150.0
Car	CASH original	3.0	6.0	15.0	30.0	60.0	120.0	149.9	3.0	7.0	35.0	15.0	30.0	7.0	60.0	120.0	60.0	140.0
	CASH modified	3.0	6.0	15.0	30.0	60.0	119.9	167.6	30.0	7.0	35.0	15.0	30.0	68.5	60.0	125.4	60.0	139.5
Kr-Vs-Kp	CASH original	4.8	9.6	23.9	47.8	95.6	190.0	239.2	4.8	5.2	26.0	23.9	47.8	52.2	95.6	104.0	95.6	104.4
	CASH modified	4.8	13.8	27.0	49.1	99.2	191.4	255.5	4.8	5.2	29.5	23.9	48.3	59.8	98.6	108.0	95.6	104.2
Tic-Tac-Toc	CASH original	3.5	6.9	17.3	34.7	69.3	139.0	173.3	3.5	6.5	33.0	17.3	34.7	65.3	69.3	129.0	69.8	130.2
	CASH modified	3.5	7.0	17.5	34.9	87.0	201.8	233.7	3.5	8.5	32.5	17.5	34.9	66.9	69.8	155.4	69.8	130.2
E. coli	CASH original	4.3	5.7	21.3	41.0	80.5	136.0	182.0	5.7	4.3	21.0	28.7	56.0	42.6	100.0	85.0	108.6	85.4
	CASH modified	4.3	8.7	21.8	43.6	87.1	174.3	217.9	5.6	4.4	21.8	28.2	58.0	43.6	107.9	87.1	110.6	87.1
Spect	CASH original	1.5	3.1	8.0	15.7	41.3	101.0	130.5	1.5	8.5	23.0	8.0	15.4	30.7	30.7	61.0	54.6	102.4
	CASH modified	1.5	3.1	8.0	15.7	30.7	61.4	76.8	1.5	8.5	23.0	8.0	15.4	30.7	30.7	61.4	30.7	108.8
Breast	CASH original	3.4	6.9	16.4	65.5	65.2	114.0	122.4	35.0	65.5	90.0	47.7	41.9	101.7	73.5	166.0	166.8	196.4
	CASH modified	4.0	8.1	8.2	13.4	23.9	44.8	55.2	34.5	10.4	15.2	48.0	17.7	20.5	32.4	37.9	119.2	51.9

domly constructed chromosome undergoes an inspection before it is added to the initial population. If the average total cost of this chromosome is larger than the average *a priori* cost, it is discarded and a new one created.

Table 10 presents the average total cost obtained in all of the runs. The first column represents the datasets for which we examined the average total cost computation method of chromosomes. The second column represents the two versions of the CASH algorithm that were compared: a CASH algorithm when the average *a priori* cost was taken into account and a CASH algorithm when the average *a priori* cost was not considered. Each of the next 17 columns shows the average total cost on different cost matrices of these two versions of CASH.

As can be seen from Table 10, the average total cost of the CASH algorithm when the average *a priori* cost was taken into account tends to be better than when it was not. In 100 out of the 187 cases, CASH showed a better performance when the average *a priori* cost was considered than when it was not. In 51 cases there was no difference in the performance and in only 36 cases did a superior performance occur as a result of not considering the average *a priori* cost. Moreover, the null-hypothesis that the two alternatives perform the same is rejected using the Wilcoxon test with $z = -6.9$ and $p < 1\%$. Thus, we conclude that the average *a priori* cost feature is required.

5.8. The effect of the average total cost computation of chromosomes

We explored the effect of the average total cost computation method of chromosomes on CASH's performance by altering the average total cost computation of the chromosomes. CASH computes the average total cost of chromosomes in the following way: given a features' subset, CASH assigns for each record the classification with the highest confidence level that was assigned to it by one of the features in the chromosomes. In order to investigate the effect of the average total cost computation method of chromosomes, we modified this computation as follows: given a feature's subset, to each record the algorithm assigns the classification which was assigned by the majority of the features in the chromosome.

Table 11 presents the average total cost obtained in all of the runs. The first column represents the datasets where we examined the average total cost computation method of chromosomes on them. The second column represents the two versions of CASH that were compared: "CASH original" represents the algorithm with the average total cost computation before the alteration, while "CASH modified" represents with the modified average total cost computation method. Each of the next 17 columns shows the average total cost on different cost matrices of these two versions of CASH. As can be seen from Table 11, the results indicate that in the majority of cases, CASH with the original average total cost computation method has a lower average total cost than CASH with the modified method. In 116 of the 187 cases, CASH performed better with the original computation method than CASH with the modified computation method. In 37 cases the two versions performed the same; in only 34 cases was CASH with the modified computation superior to CASH with the original computation method. Moreover, the null-hypothesis that the two alternatives perform the same is rejected using the Wilcoxon test with $z = -4.8$ and $p < 1\%$. Thus, we conclude that the proposed total cost computation method should be.

5.9. Analysis of computational cost

The aim of this section is to compare the computational cost of the various methods by measuring the running time. Table 12 presents the actual time (in milliseconds) required for the feature selection and classification task. We conducted all of our experiments with the following hardware configuration: a desktop computer implementing a Microsoft Windows Server 2003 operating system with Intel (R) Core (TM) 2 Quad CPU Q6600 @ 2.40 GHz 2.39 GHz, and 7.70 GB of physical memory.

As we can see from Table 12, CASH is consistently faster than all of the genetic search-based algorithms (ProICET, GA + -META and GE + META + CS-ID3). The savings in time becomes more significant when the data dimensionality increases. These results might be due to two different properties of the CASH algorithm. First, instead of using the wrapper approach, which requires several repetitions of the decision tree training, we used the filter approach. Second, most of the processing time of the algorithm is in the preprocessing step of the algorithm, which is performed only once. In all the generation and the evaluation of each subset only a simple calculation is conducted.

Table 12
Comparing the execution time (in ms).

Datasets	csDT	ProICET	CASH	GA + META + CS-ID3	GA + META
Pima	453	2,796,497	2034	1,851,003	1,849,345
Bupa	172	2,613,000	759	56,978	56,128
Thyroid	1375	3,676,000	2892	2,993,444	2,981,315
Hepatitis	285	2,618,009	1066	5,852,784	5,851,617
Voting	188	2,816,510	956	7,568,562	7,567,410
Car	255	3,512,691	1014	61,777	60,987
Tic-Tac-Toc	265	2,801,360	1013	1,963,509	1,962,079
E. coli	188	2,808,469	635	261,731	260,937
Spect	250	3,091,197	3765	17,021,432	17,019,530
Breast	187	2,796,000	1683	17,282,630	17,280,467
Kr-Vs-Kp	656	3,460,278	5652	50,567,486	50,565,860

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

Since CASH is a filter-based feature selection, we regarded its execution time as the summation of the execution time in the feature selection phase and the execution time of the classifier construction and classification phase. On the other hand, csDT is a classification algorithm, which does not use a feature selection algorithm. Therefore, as expected, csDT is faster than CASH. However, when CASH's feature selection algorithm is applied to the dataset before the construction and classification phase, this phase is 25–50% faster compared to the construction and classification phase execution time of csDT. The reason for this is that CASH removes features from the dataset and therefore the classification task is performed on a dataset with fewer features. Therefore, if the feature selection is conducted offline, it can more rapidly conduct the classifier construction and classification phase after CASH has been applied to the dataset.

6. Discussion

The advantages of the new CASH algorithm, as the experimental study indicated, can be summarized as follows:

1. Compared to state-of-the-art cost-sensitive algorithms, CASH displayed lower average total costs in processing the majority of misclassification cost matrices and datasets. Moreover, CASH chooses fewer features which is considered to be a benefit since users generally regard smaller decision trees as more comprehensible. As we can see from the experimental results, CASH' performance is superior in most of the cases to the other five algorithms to which it was compared. Let's now try to explore why CASH is superior to other algorithms
   1. GA + META and csDT (when the csf equals zero): While CASH takes into account both misclassification and test costs, GA + META takes only the misclassification cost into consideration. Therefore, we expect that, if GA + META and csDT (when the csf equals zero) will be modified to consider the tests' cost too, their performances will be improved.
   2. GA + META + CsId3 algorithm follows the wrapper feature selection method. Therefore, it uses a classifier to assess the features' subsets. It is important to mention that the classifier that is used in the feature selection process **is not** the same classifier as is used in the classification task. The genetic search algorithm's configurations were set to be the same in GA + META + CsId3 and CASH algorithms. Moreover, the classifier that is used in the classification was the same chosen to perform the same task (the C4.5 decision tree) in both of the algorithms. Therefore, the major difference between CASH and GA + META + CsId3 algorithms lie in their different fitness function. Therefore, since CASH's performance is superior to GA + META + CsId3's and the major difference between the two algorithms lies in their fitness function, we believe that CASH's fitness function balances better between the two types of costs units than GA + META + CsId3's fitness function.
   3. ProICET and csDT algorithms (when the csf equals 1): ProICET and csDT (when the csf equals 1) algorithms follow the embedded approach since it incorporates the feature selection as part of the classifier construction in the training process. Since CASH algorithm shows a better performance compared to these two algorithms, we can assume that this is due to the superiority of CASH's fitness function to the splitting criterion of ProICET and csDT algorithms, respectively. Moreover, in Section 5.7 we examine the effect of the consideration of the average *a priori* cost on CASH's performance. The *a priori* cost serves as an indication of when a features subset should not be obtained. The results of the examination have shown that the average total cost of the CASH algorithm, when the average *a priori* cost was taken into account, tends to be better than when CASH did not consider the average *a priori* cost in a significant way. Moreover, in many cases csDT (when csf is equal 1) tends to be better than CASH's performance, when it does not take into consideration the average *a priori* cost. Therefore, we believe that CASH's superiority in this study to the other algorithms, and particularly to csDT (when csf is equal to one), is due to the superiority of the method of indicating when a feature's subset should not be obtained.
2. Since CASH follows the filter approach, it can be used in conjunction with any classification algorithm and not only decision tree classifiers. Possibly there might be domains where other classifiers will dramatically reduce the average total cost.
3. The new algorithm is faster than existing cost-sensitive methods, which use a genetic algorithm as their search algorithm, mainly because we use a simple histogram-based fitness function that is faster than the wrapper estimation. Moreover, most of the histogram computations are performed only once as a preprocessing step. The GA algorithm then uses these pre-computed values to calculate the fitness function. Consequently, during the GA generations we perform a relatively low volume of computations.
4. Although CASH, ProICET and GA + META use the same genetic representation, genetic operators, population size and number of generations, CASH nevertheless outperforms ProICET and GA + META. We conclude from this observation that the proposed fitness function selects offspring better than existing GA algorithms.

The CASH algorithm also has a major drawback. The current implementation of the algorithm deals only with binary class problems. This drawback can be alleviated by converting the multi-class task into a set of binary classification tasks. There are several alternatives for performing this transformation, such as "one-against-all", "one-against-one" and the error-correcting output codes (ECOC) methods.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

7. Conclusion and future research

This study presents a novel cost-sensitive fitness function based on histogram comparison. It is integrated together a genetic search method to form a new feature selection algorithm termed CASH (cost-sensitive attribute selection algorithm using histograms). CASH takes into account both test and error misclassification costs as well as feature grouping. This paper examines whether the CASH algorithm outperforms other cost-sensitive algorithms in terms of average total costs and execution time.

The algorithm was evaluated on a wide range of standard datasets. The results show that CASH outperforms other cost-sensitive algorithms in that it has a lower average total cost. Moreover, it is also consistently faster than the other genetic search-based algorithms..

With regard to future research, several possible enhancements come to mind. First, the current implementation of the algorithm deals only with binary class problems. However, extending the algorithm to cope with multivariate class problems is not a complicated task. Secondly, in this study we assumed that, while the feature value measurement process of the instances in the testing set has a cost, the feature value measurement process of the records in the training set is available for free. It would be interesting to integrate into the algorithm sensitivity to the feature value cost during the learning phase similarly to active learning algorithms. A third point to be considered in future research is the application of the algorithm to the security domain in order to deploy host-based intrusion detection systems (HIDS) for mobile phones. By using the combination of the new cost-sensitive feature selection algorithm with the HIDS, we can supply a suitable security solution for mobile devices that takes into consideration the mobile phone's resource limitations, such as: CPU processing power and battery power. The new feature selection algorithm would be able to decide which features the agent would collect from the mobile device in relation to the mobile device's resources.

Acknowledgments

The authors gratefully thank the action editor and the anonymous reviewers whose constructive comments helped in improving the quality and accuracy of this paper.

References

[1] D. Afshartous, R.A. Preston, Confidence intervals for dependent data: equating non-overlap with statistical significance, Computational Statistics and Data Analysis 54 (10) (2010) 2296–2305.
[2] Arnt, S. Zilberstein, Attribute measurement policies for cost-effective classification, in: Workshop Data Mining in Resource Constrained Environments, 4th International Conference on Data Mining, 2004.
[3] Blakem, C.J. Merz, UCI Repository of Machine Learning Databases, 1998. Available from: <http://www.ics.uci.edu/~mlearn/MLRepository.html> (accessed 19.01.10).
[4] X. Chai, L. Deng, Q. Yang, C.X. Ling, Test-cost sensitive Naive Bayes classification, in: Proceedings of 4th International Conference on Data Mining, 2004, pp. 51–58.
[5] Y. Chen, C.C. Wu, K. Tang, Building a cost-constrained decision tree with multiple condition attributes, Information Sciences 179 (7) (2009) 967–979.
[6] K. Cherkauer, J.W. Shavlik, Growing simpler decision trees to facilitate knowledge discovery, in: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996, pp. 315–318.
[7] J. Demšar, Statistical comparison of classifiers over multiple datasets, Journal of Machine Learning Research 7 (2006) 1–30.
[8] T. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation 10 (7) (1998) 1895–1992.
[9] P. Domingos, MetaCost: a general method for making classifiers cost-sensitive, Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (1999) 155–164.
[10] P. Domingos, Metacost: a general method for making classifiers cost-sensitive, Knowledge Discovery and Data Mining (1999) 155–164.
[11] S. Esmeir, S. Markovitch, Anytime learning of decision trees, Journal of Machine Learning Research (JMLR) 8 (2007) 891–933.
[12] U. Fayyad, K.B. Irani, Multi-interval discretization of continuous valued attributes for classification learning, in: Thirteenth International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[13] E. Frank, M.A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I.H. Witten, Weka: a machine learning workbench for data mining, in: O.Z. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, Springer, 2005, pp. 1305–1314.
[14] Freitas, A. da Costa-Pereia, P. Brazdil, Cost-sensitive decision trees applied to medical data, in: DaWak, Lecture Notes in Computer Science, 2007, pp. 303–312.
[15] S. García, A. Fernandez, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (10) (2010) 2044–2064.
[16] Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2005) 1157–1182.
[17] K. Iswandy, A. Koenig, Feature selection with acquisition cost for optimizing sensor system design, Advances in Radio Science 4 (2006) 135–141.
[18] K. Jong, E. Marchiori, M. Sebag, Ensemble learning with evolutionary computation: application to feature ranking, Proceedings of Parallel Problem Solving from Nature VIII (PPSN-2004), LNCS 3242, Springer, 2004, pp. 1133–1142.
[19] L. Jourdan, C. Dhaenens-Flipo, E.G. Talbi, Discovery of genetic and environmental interactions in disease data using evolutionary computation, in: G.B. Fogel, D.W. Corne (Eds.), Evolutionary Computation in Bioinformatics, Morgan Kaufman, 2003, pp. 297–316.
[20] M. Kai, Inducing cost-sensitive trees via instance weighting, in: Principles of Data Mining and Knowledge Discovery, Second European Symp, 1998, pp. 139–147.
[21] Y. Kim, W.N. Street, F. Menczer, Feature selection in unsupervised learning via evolutionary search, in: Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000), 2000, pp. 365–369.
[22] G. Kim, S. Kim, Feature selection using genetic algorithms for handwritten character recognition, in: Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, 2000, pp. 103–112.
[23] M. Kudo, J. Sklansky, Comparison of algorithms that select features for pattern classifiers, Pattern Recognition 33 (2000) 25–41.
[24] C. Ling, Q. Yang, J. Wang, S. Zhang, Decision trees with minimal costs, in: Proceedings of 21st International Conference Machine Learning, 2004, p. 69.
[25] C. Ling, V.S. Sheng, Q. Yang, Test strategies for cost-sensitive decision trees, IEEE Transactions on Knowledge and Data Engineering 18 (8) (2006) 1055–1067.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035

[26] H. Liu, H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic, Dordrecht, 1998.
[27] S. Maldonado, R. Weber, J. Basak, Simultaneous feature selection and classification using kernel-penalized support vector machines, Information Sciences 181 (1) (2011) 115–128.
[28] S. Maldonado, R. Weber, A wrapper method for feature selection using support vector machines, Information Sciences 179 (13) (2009) 2208–2217.
[29] N. Matatov, L. Rokach, O. Maimon, Privacy-preserving data mining: a feature set partitioning approach, Information Sciences 180 (14) (2010) 2696–2720.
[30] M. Núñez, The use of background knowledge in decision tree induction, Machine Learning 6 (1991) 231–250.
[31] P. Paclik, R.P.W. Duin, G.M.P. van Kempen, R. Kohlus, On feature selection with measurement cost and grouped features, Lecture Notes in Computer Science 2396 (2002) 461–469.
[32] S. Qin, S. Zhang, C. Zhang, Cost-sensitive decision trees with multiple cost scales, Lecture Notes in Computer Science, AI 3339 (2004) 80–390.
[33] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, San Francisco, 1993.
[34] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (1) (2010) 1–39.
[35] P. Sharpe, R.P. Glover, Efficient GA based techniques for classification, Applied Intelligence 11 (1999) 277–284.
[36] S. Sheng, C.X. Ling, Hybrid cost-sensitive decision tree, PKDD (2005) 274–284.
[37] S. Sheng, C.X. Ling, Q. Yang, Simple test strategies for cost-sensitive decision trees, in: Proceedings 16th European Conference Machine Learning, 2005, pp. 365–376.
[38] S. Sheng, C.X. Ling, A. Ni, and S. Zhang, Cost-sensitive test strategies, in: Proceedings of 21st National Conference on Artificial Intelligence (2006).
[39] M. Tan, J. Schlimmer, Cost-sensitive concept learning of sensor use in approach and recognition, in: Proceedings of the Sixth International Workshop on Machine Learning, 1989, ML-89, pp. 392–395.
[40] M. Tan, Cost-sensitive learning of classification knowledge and its applications in robotics, Machine Learning 13 (1993) 7–33.
[41] P. Turney, Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm, Journal on Artificial Intelligence Research 2 (1995) 369–409.
[42] P. Turney, Types of cost in inductive concept learning, Proc. Workshop Cost-Sensitive Learning, in: 17th International Conference on Machine Learning, 2000, pp. 15–21.
[43] C. Vidrighin, C. Savin, R. Potolea, A hybrid algorithm for medical diagnosis, in: Proceedings of the International Conference on Computer as a Tool, (EUROCON 2007), Warsaw, 2007, pp. 668–673.
[44] C. Vidrighin, R. Potolea, I. Giurgiu, M. Cuibus, ProICET case study on prostate cancer data, in: Proceedings of the 12th International Symposium of Health Information Management Research, 2007, pp. 237–244.
[45] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, IEEE Transactions on Intelligent Systems 13 (1998) 44–49.
[46] V. Zubek, T.G. Dietterich, Pruning improves heuristic search for cost-sensitive learning, in: Proceedings of 19th International Conference on Machine Learning, 2002, pp. 27–34.

Please cite this article in press as: Y. Weiss et al., The CASH algorithm-cost-sensitive attribute selection using histograms, Inform. Sci. (2011), doi:10.1016/j.ins.2011.01.035