Intelligent Data Analysis 21 (2017) 371–383
DOI 10.3233/IDA-150322
IOS Press

Feature bundling in decision tree algorithm

Xu Zhuang[a,*], Yan Zhu[a], Chin-Chen Chang[b,c] and Qiang Peng[a]
[a]*School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, China*
[b]*Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 40724, Taiwan*
[c]*Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan*

Abstract. In empirical data modelling, a model of system is built up from a set of cases that the system has observed. Eventually, the performance of the inducted model is dominated by the quality and quantity of observations. Feature transformation methods are widely used to improve quality of knowledge extracted from observations to build up more accurate and robust model. In the paper, a new feature transformation method named dynamical feature bundling for decision tree algorithm is proposed. Dynamical feature bundling groups a set of features in the tree induction phase and it enables decision tree algorithms to 1) make use of features in one bundle together to make collective judgments in splitting phase; 2) learn more reliable and stable knowledge from feature bundles created based on domain knowledge of experts; 3) embed feature transformation step into tree induction phase, and therefore the extra pre-process step which are necessary for static feature transformation methods is inessential. Our experiments show 2%–9% improvements of AUC value on a very imbalanced dataset. Slight improvements are also obtained on a more balanced data set.

Keywords: Decision tree, feature bundling, feature transformation, web spam detection

1. Introduction

With constant development of information technology in past few decades, people are confronting with ever-increasing amounts of data collected from a variety of sources in real-life: web documents, Internet of things, cash flows, purchase records, etc. Machine learning techniques make it easier for people to find and extract intricate knowledge contained in such large-scale data, on which very complex system can be built. Some successful applications of machine learning technique include decision making in BI (Business Intelligence) systems, spam detection (e.g. e-mail, web and reviews), pattern recognition (e.g. optical character recognition), *etc*.

Since the 50's machine learning research has mostly concentrated on finding relationships in data and studying the processing for extracting such relations [5,10]. Formally, machine learning algorithm aims to find a function or system f so that given a case c we can get the output $L(c)$ by

$$L(c) = S(c). \tag{1}$$

*Corresponding author: Xu Zhuang, School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, China. E-mail: zhuangxu6620362@163.com.

In different machine learning tasks, $L(.)$ has different meanings. For example, $L(c)$ denotes the predicted class of c in classification algorithms; and in clustering, $L(c)$ is the cluster to which c belongs; and $L(c)$ can also be numeric in some value based learning tasks, e.g. regression. A case c is typically represented in the feature vector space,

$$c = (a_1(c), a_2(c), \ldots, a_n(c)), \tag{2}$$

where $a_i(.)$ denotes value of i-th attribute/feature of a case and n is the size of the feature vector. It is assumed that the objective concept (e.g. class in classification problem) of a case is dominated by its attributes in some ways in learning algorithms. Thus it is possible to use scientific methods (statistics, information theory, deep learning, etc.) to analyze relationships among different attributes contributing to certain concept over the training set. As a consequence, the quality of training set has strongly influence on the performance of an algorithm. For example, most conventional classification methods (including SVM, decision tree, random forest, etc.) are broadly based on the hypothesis that the distribution of class labels in the training set is uniform. The performance of such algorithms drops largely when data sets are uneven. Furthermore, the poor performance of algorithms may also be caused by insufficient information and too much noise contained in the data.

Feature transformation [6] is technique used to improve quality of knowledge extracted from data set by mapping data from original feature space into a new created feature space. Most conventional feature transformation methods are value-driven since only knowledge (e.g. relationships among some features) reflected in value changes of features can be caught. However, previous researches showed that the performance of trained model can be improved with assistance of human's domain knowledge [1,8,9] in two different ways. Firstly, domain knowledge can be used to construct higher-level features from the original feature set to enhance performance of model. On the other hand, some special relations among features those not reflect in their values can also be obtained by experts and added to the algorithm using some specific schemes.

In the paper, a new feature transformation method, i.e. dynamical feature bundling, is proposed for C4.5 decision tree algorithm. The novel method aims to explore relations in natures of attributes rather than values. The meaning of nature is twofold. First, it reflects how the feature is extracted or calculated from original data. For example, in some web graph oriented mining tasks, researchers typically use *supporters* (denoted as $sup(d, p)$) at distance d to roughly evaluate the quality and authority of a web page p. $sup(d, p)$ is computed by accumulating pages which directly or indirectly links to p at distances d. A set of $sup(d, p)$s of p can be obtained by changing d from 1 to $n(n > 1)$ and obviously they are inherently related since they are calculated using the same function but with different arguments (different d). Typically, a set of supporters at different distances are used to make collective judgment for a single page, which is more comprehensive and scientific than considering them separately. Second, natures of features are specific task oriented. Some features are close related since they contribute to the target concept apparently if they are taken into account together. For example, researchers take both numbers of in-coming links and out-going links of a web page to evaluate its possibility of being spam. Spam pages are likely to create a large number of out-going links to boost ranking scores (used to rank web pages by search engines) of some other spam pages but with very little number of in-coming links since their low qualities. Thus, taking the numbers of in-coming links and out-going links into account together is meaningful to identify spam in the web.

Feature bundling is a feature transformation method that groups individual features into one or more feature bundles, which are treated as new features in the learning phase. In general, following three kinds of approaches can be used to combine different features [1]:

1) *Feature*1 <logical operator> *Feature*2, where <logical operator> = {AND, OR, NOT, EXOR}.
2) *Feature*1 <arithmetic operator> *Feature*2, where <arithmetic operator> = $\{+, -, x, /\}$.
3) Regression function based on a subset of features.

Kusiak [1] implemented *static* feature bundling (using AND) and his research showed a considerable improvement on a small data set with 86 cases. In the paper, we still only consider operator AND for bundling and implement the bundling decision tree algorithm for several reasons:

1) As one of the top ten data mining algorithms [16], decision tree have shown its stable and effective performance on many research areas. In addition, the result of the decision tree is easy to understand because it can be considered as many sequences which consist of connected features using logic AND.
2) Operator AND has clear meaning. For example, in the case where we want to combine features $f1$ and $f2$ using AND, that means we want the decision tree algorithm to take $f1$ and $f2$ into account together in the tree induction phase. In most cases, we combine features using AND because of their close relationships. For example, features calculated from the same function but with different augments can be bundled together since they actually evaluate the same aspect of object (e.g. the page's supporters are calculated using the same function but with different distances).
3) Operator AND does not break the logical structure of decision tree. Although operator OR also has clear meaning, i.e. at least one of the features in the bundle should be taken into account, it may lead to confusion that a test case can be passed to different branches at the same time, which causes the case may get more than one labels and hence increases the complexity to determine the final label of the case. Operator AND can avoid such weakness since it maintains the logical structure of decision tree.
4) Arithmetic operators and regression functions are always oriented to values, but operator AND is oriented to natures.

In the paper, we proposed dynamical feature bundling methods for C4.5 algorithm. Dynamical feature bundling is embedded in the learning phase of algorithm (i.e. tree induction phase) and different from widely used static feature transformation method which generally needs an extra pre-process step to do the transformation computation. We developed three dynamic bundling algorithms for C4.5 algorithm: abstract bundling, abstract greedy bundling and local greedy bundling. Taking advantage of our domain knowledge in web spam detection, we built up C4.5 decision trees based on some pre-defined feature bundles on two large public data sets WEBSPAM-UK2006 and WEBSPAM-UK2007 [4]. Our experimental results show that the local greedy bundling method can get better improvements than others and it also has minimum time complexity. Compared with original C4.5 algorithm, our method can get 2%–9% improvements on WEBSPAM-UK2007 and very slight improvements (about 1%) on WEBSPAM-UK2006.

2. Related work

2.1. C4.5 decision tree

Divide and conquer method is used to construct C4.5 decision tree. For a given training set D and a set of attributes $A = \{a_1, a_2, \ldots, a_n\}$, the tree induction phase is described as follows:

1) D contains one or more cases that all belong to a single class C. The decision tree of D is a leaf node with the single class C.

2) D is empty. The decision tree of D is a leaf node with the class of its parent.
3) D contains cases belonging to different classes. The data set D is divided into several un-overlapping sub-datasets $\{D_1, D_2, \ldots, D_n\}$ using a test T, which is based on a single attribute with one or more cut points. Then the same tree induction phase is applied to sub-datasets $\{D_1, D_2, \ldots, D_n\}$.

A test T can be denoted as a pair (a_i, cp), where a_i is the test attribute and *cp* is a set of its cut points. For nominal attribute a_i, *cp* of a_i consists of its all distinct values. The dataset is then divided into $|cp|$ sub-datasets that cases in the same sub-dataset have the same value of attribute a_i. *cp* of a numeric attribute a_j consists of only one cut point. The dataset is then divided into two sub-datasets by the cut point. Information gain and gain ratio are common criterion used to decide which test is best available to split the current dataset. The *information* contained in a dataset D is defined as

$$\mathit{Info}(D) = -\sum_{j=1}^{k} \frac{\mathit{freq}(C_j, D)}{|D|} \times \log_2 \left(\frac{\mathit{freq}(C_j, D)}{|D|} \right) \text{bits}, \tag{3}$$

where $\mathit{freq}(C_j, D)$ denotes number of cases in D that belong to label C_j. Information calculated in Eq. (3) measures the average amount of information needed to identify the class of a case in D. If the dataset D is divided into several un-overlapping sub-datasets $D_1, D_2, \ldots, D_n$ using a test T, the expected information requirement is the weighted sum over sub-datasets:

$$\mathit{Info}_T(D) = \sum_{i}^{n} \frac{|D_i|}{|D|} \times \mathit{Info}(D_i). \tag{4}$$

Then information gained by partitioning D into $D_1, D_2, \ldots, D_n$ using test T is

$$\mathit{gain}(T) = \mathit{Info}(D) - \mathit{Info}_T(D). \tag{5}$$

The necessary information used to split D into $D_1, D_2, \ldots, D_n$ using test T is

$$\mathit{SplitInfo}(T) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right). \tag{6}$$

Then we have

$$\mathit{GainRatio}(T) = \mathit{gain}(T)/\mathit{SplitInfo}(T). \tag{7}$$

Gain ratio measures the portion of useful information for classification gained by splitting D into D_1, $D_2, \ldots, D_n$ using test T.

The test T is selected by a splitting function with the goal that sub-datasets produced by T from dataset D can maximize the gain radio, subject to the constraint that the information gain must be large at least as great as the average gain over all tests examined [12].

Formally, given a dataset D and an numeric attribute A, function $\mathit{Discretize}(D, A)$ outputs the best threshold (cut point) of A that sub-datasets produced by test (A, $\{\mathit{Discretize}(D, A)\}$) can maximize the gain radio and is larger than average gain over all examinations using candidate values of A. The splitting function outputs the best test of dataset D with a set of attributes $A_s = \{a_1, a_2, \ldots, a_n\}$,

$$\mathit{Splitting}(D, A_s) = (A, \mathit{Discretize}(D, A)) = T, \tag{8}$$

where $A \in A_s$. Notation $\mathit{Dataset}(D, T)$ is used to denote the sub-datasets produced by test T.

2.2. Web spam detection

Web spam refers to web pages that use spamming techniques to deceive search engine algorithms for getting higher than their deserved rankings in search engine results. Web spam detection is the research area that focuses on designing automatic algorithms to identify spam *hosts* over the web. A set of link-based and content-based primitive features have been identified to filter spam [2,13]. Our research is totally based on these primitive features and we will provide a brief description about them.

The name of feature in our data set consists of two or three parts separated by notation ".", e.g. PageRank. hp and Neighbors.2.mp. The first part of feature name points out implication of the feature. The second part specifying argument of feature calculation is optional. Finally, the last part indicates which pages are used for feature calculation. For example, the name PageRank.hp means that the feature is the PageRank value of the home page (hp). "Neighbors.2.mp" is explained that Neighbors (a measure) of page with maximum PageRank in the host (mp) using argument 2 in calculation.

3. Feature bundling

We classify feature bundling methods into two groups, static bundling and dynamic bundling, according to moment and independence which the scheme operates in relation to the learning algorithm [14].

Static bundling performs before learning phase and it is independent to the algorithm. Typically static bundling can be only applied to nominal attributes because of the inherent meaning of logical operator. For example, using logical operator AND to combine two nominal values A and B can give a new value denoted as A_B. But it is undefined if we combine two numeric values (what's the result of combining 1 and 2, 1 AND 2 = ?). Thus numeric attributes must be first discretized in the pre-processing step if static bundling is needed. Static bundling is easy to implement, but it is not suitable for some real time systems because of the additional pre-processing step.

Dynamic bundling is embedded in algorithm and therefore it can only make use of partial information contained in the learning phase to provide compact and accurate results with respect to training cases, which in turn means higher possibility of overfitting. Dynamic bundling is efficient and convenient in terms of ability to deal with numeric values and eliminate pre-process step.

3.1. Static bundling

In the paper, ID3 discretization (the static version of the discretization embedded in C4.5 decision tree algorithm) is used to perform static bundling for numeric attributes. Ten cases of our dataset are illustrated in Table 1 to clarify process of static bundling. For convenience, only six attributes and class labels are presented. Results of discretization are shown in Table 2.

We calculate classification quality[1] (CQ) of attributes shown in Table 2 to roughly evaluate their usefulness in identifying web spam, as shown in Table 3.

In our example, we can design two bundles by analyzing relationships of attributes in their natures. The meaning of nature is twofold as discussed in Section 1: 1) it reflects how the attribute is computed from the original data; 2) it is specific task oriented. Thus we can design bundles in these two aspects.

[1]Classification quality of a feature: the percentage of all cases in the set that can be unambiguously identified using the feature [9].

Table 1
Example with ten cases selected from WEBSPAM-UK2007 data set

Neighbors.2.hp	Neighbors.3.hp	Outdegree.hp	Pagerank.hp	Truncatedpagerank.1.hp	TrustRank.hp	Class
488	13063	15	1.21E-07	1.35E-07	3.00E-08	Spam
79	956	39	4.95E-09	4.51E-09	1.23E-09	Spam
142	482	1	5.03E-09	4.61E-09	1.26E-09	Spam
46	203	3	1.21E-08	1.29E-08	3.00E-09	Nonspam
9	3159	0	3.94E-09	2.88E-09	9.78E-10	Nonspam
0	0	1	1.79E-09	3.07E-10	4.42E-10	Nonspam
98	2741	17	7.40E-08	8.53E-08	1.83E-08	Nonspam
1323	16043	9	2.63E-07	2.38E-07	6.53E-08	Nonspam
23233	25429	19	6.44E-06	7.04E-06	1.59E-06	Nonspam
300	40852	9	1.05E-07	1.37E-07	2.66E-08	Nonspam

Table 2
Example data after discretization

Neighbors.2.hp	Neighbors.3.hp	Outdegree.hp	Pagerank.hp	Truncatedpagerank.1.hp	Trustrank.hp	Class
b	a	a	b	a	b	Spam
b	a	b	b	a	b	Spam
b	a	a	b	a	b	Spam
a	a	a	b	a	b	Nonspam
a	a	a	a	a	a	Nonspam
a	a	a	a	a	a	Nonspam
b	a	a	b	a	b	Nonspam
b	b	a	b	b	b	Nonspam
b	b	a	b	b	b	Nonspam
b	b	a	b	b	b	Nonspam

Table 3
Classification quality of data after discretization

Neighbor.2.hp	Neighbors.3.hp	Outdegree.hp	Pagerank.hp	Truncatedpagerank.1.hp	Trustrank.hp
0.3	0.3	0.1	0.2	0.3	0.2

For example, the attributes *Neighbors.*2.*hp* and *neighbors.*3.*hp* are computed using the same function to calculate the number of neighbors in the web graph limited to pre-defined distances, 2 and 3, respectively. So it is natural to group them into one bundle to make collective judgement. Attributes *pagerank.hp* and *truncatedpagerank.1.hp* have the similar meaning that *truncatedpagerank.1.hp* uses the same function to calculate PageRank scores with restriction that pages directly (indicated by the middle part "1" of feature name) pointing to target page can contribute PageRank score to it. Therefore, we group *neighbors.2.hp* and *neighbors.3.hp* into one bundle as well as *pagerank.hp* and *truncatedpagerank.1.hp*. Notation $f1\_f2$ is used to denote the bundle by combining features $f1$ and $f2$. Table 4 presents the data after static bundling.

Because ID3 discretization method is a binary partition approach, at most 2^n values will be constructed when n attributes are grouped by AND. In Table 4, both two bundles have 3 distinct values. The CQs of attributes in Table 4 are shown in Table 5.

Compared with classification qualities in Table 3, the created bundles have higher classification qualities and hence have higher possibility to train a better classifier.

Table 4
Example data after static bundling

Neighbors.2.hp_Neighbors.3.hp	Outdegree.hp	Pagerank.hp_Truncatedpagerank.1.hp	Trustrank.hp	Class
b_a	a	b_a	b	Spam
b_a	b	b_a	b	Spam
b_a	a	b_a	b	Spam
a_a	a	b_a	b	Nonspam
a_a	a	a_a	a	Nonspam
a_a	a	a_a	a	Nonspam
b_a	a	b_a	b	Nonspam
b_b	a	b_b	b	Nonspam
b_b	a	b_b	b	Nonspam
b_b	a	b_b	b	Nonspam

Table 5
Classification quality of features in the example after bundling

Neighbor.2.hp_Neighbors.3.hp	Outdegree.hp	Pagerank.hp_Truncatedpagerank.1.hp	Trustrank.hp
0.6	0.1	0.5	0.2

3.2. Dynamical bundling

Unlike C4.5 tree node splitting approach which tests only one attribute, more than one attributes can be tested in dynamical bundling methods because of inherent meaning of bundling logical operator AND: consider attributes in one bundle together.

It is assumed that data in Fig. 1 consist of five attributes ($f1$, $f2$, $f3$, $f4$ and $f5$) and two classes ($c1$ and $c2$). Figure 1(a) shows a normal c4.5 decision tree without using attribute $f5$ and Fig. 1(b) shows the decision tree created when attributes $f2$ and $f3$ are grouped by logical operator AND. Suppose there are m distinct values for attribute $f2$ and n for $f3$ in the dataset, we need $(m \times n)$ tests to determine the cut points f and g when we want the algorithm to take into $f2$ and $f3$ account together. Consequently, the time complexity to find cut points of grouped K attributes is $O(n^k)$, where n denotes the size of the data in the node. Obviously, it is too expensive to implement such an algorithm. To maintain the effectiveness of original c4.5 algorithm, we proposed three dynamical bundling methods, namely, *abstract bundling*, *abstract greedy bundling* and *local greedy bundling*. The proposed dynamical bundling algorithms treat each bundle as one attribute in tree induction phase, and provide different ways to find cut points for attributes in the same bundle.

We redefine the splitting function (Eq. (8)) and discretization function for generalization. The function *Discretize*(D, A) no longer outputs a single value to denote the cut point of an numeric attribute because A may be a feature bundle. The *Discretize* function for feature bundle is defined as:

$$\textit{Discretize}(D, A) = \{cpl_1, cpl_2, \ldots, cpl_n\}, \tag{9}$$

where A is a feature bundle represented as $f1\_\ f2\_\ldots\_\ f_n$, cpl_i is the list of cut points for attribute f_i. Without losing generality, definition in Eq. (9) is also applied to case where A is a single attribute. Accordingly, splitting function in Eq. (8) is extended to feature bundles. Next we propose three dynamical bundling algorithms to provide different discretization functions for calculating cut points of features in bundle.

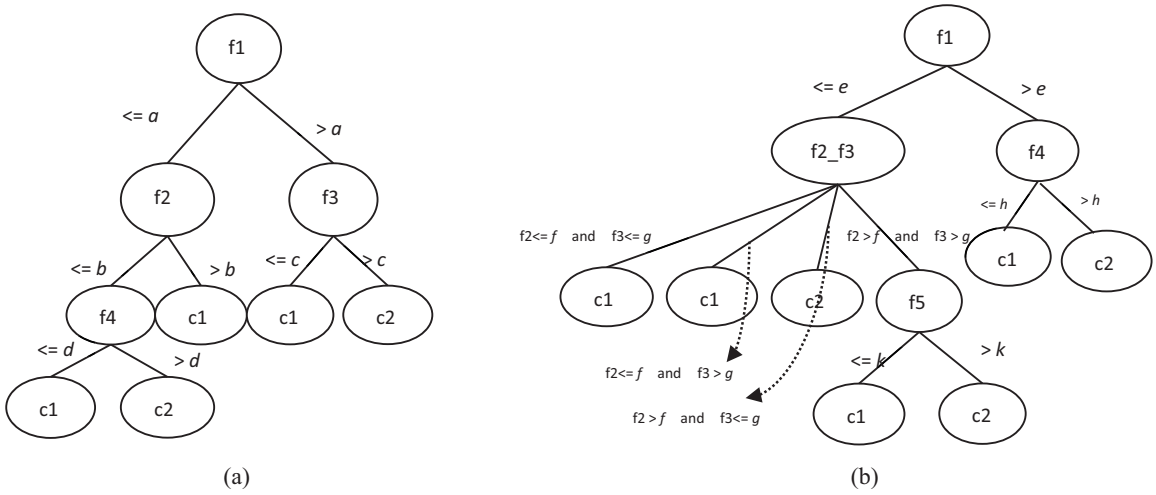

Fig. 1. Two examples of decision trees: (a) a normal c4.5 decision tree; (b) a c4.5 decision tree based on bundle $f2\_f3$.

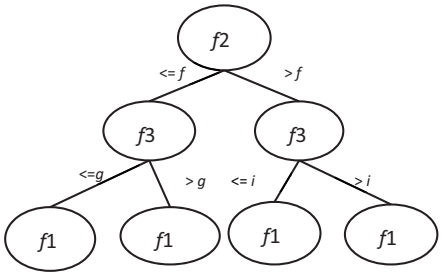

Fig. 2. Example of abstract bundling.

3.2.1. Abstract bundling

Abstract bundling separates examinations of attributes in one bundle for keeping the logical structure of original C4.5 decision tree algorithm that non-leaf node tests only one attribute. Given dataset D and a bundle $f1\_\ f2\_\ldots\_\ f_n$ consisting of a set of attributes $F_s = \{f1, f2, \ldots, f_n\}$, the set of total permutations of F_s is denoted as *Per*(F_s) and $|Per(F_s)| = n!$. To find cut points of attributes in the bundle, a decision tree for the bundle $f1\_\ f2\_\ldots\_\ f_n$ is constructed as follows:

1) The height of the tree is equal to $|F_s|$ and nodes with the same depth use the same attribute to divide the data;
2) The decision tree is constructed for each permutation p in *Per*(F_s) by testing the features in F_s from top to down with the same order in p. For example, a permutation of feature set {$f1$, $f2$, $f3$} is ($f2$, $f3$, $f1$) and decision tree based on ($f2$, $f3$, $f1$) is shown in Fig. 2. The set of datasets produced by division using last feature of permutation p is denoted as $FD_s(p)$. In Fig. 2, $f1$ is the last testing feature and $|FD_s(f2, f3, f1)| = 8$. The *information gain* and *information gain radio* caused by splitting dataset D into datasets $FD_s(p)$ is denoted as *IG*($FD_s(p)$) and *GR*($FD_s(p)$), respectively. Because every FD_s corresponds to a permutation p, *IG*($FD_s(p)$) and *GR*($FD_s(p)$) can be simplified as *IG*(p) and *GR*(p), respectively.
3) The decision tree based on permutation p which causes the "best" *IG*(p) and *GR*(p) is chosen. The "best" in C4.5 means *GR*(p) should be maximum with the constraint that *IG*(p) is larger than

average value of all examinations.

In summary, the goal of function *Discretize*(D, A) in abstract bundling is to find the permutation which performs best in terms of information gain and gain radio. The result of *Discretize*(D, A) of the example in Fig. 2 is *Discretize*$(D, f1\_f2\_f3) = \{(k, m, l, n)_{f1}, (f)_{f2}, (g, i)_{f3}\}$. If there are k features in one bundle, $k!$ permutations will be examined. For each permutation, we need at most $(k \times n)$ tests to determine cut points of all nodes, where n denotes the number of cases in the dataset. So totally, the time complexity of abstract bundling is $O(k! \times k \times n)$ for bundle consisting of k features. Compared with bundling method in Fig. 1(b) in which the time complexity of a bundle with k features is $O(k^n)$, the abstract bundling is more applicable since k is small (< 5) in most cases.

3.2.2. Abstract greedy bundling

Greedy algorithm is used in C4.5 to calculate the best attribute for every node. Using the similar idea, we propose another effective bundling method named abstract greedy bundling.

Abstract greedy bundling has similar idea with abstract bundling that examinations are actually separated. Abstract bundling uses exhaustive search method to find best partition over all possible permutations in one bundle. However, if the number of attributes in one bundle is large enough, performance of the algorithm will drop off precipitously. To avoid this flaw, attributes are selected from top to bottom without backtracking in abstract greedy bundling. Given dataset D and a bundle $f1\_f2\_\ldots\_f_n$ consisting of a set of features $F_s = \{f1, f2, \ldots, f_n\}$, the tree constructed in abstract greedy bundling is based on rules:

1) The height of the tree is smaller than or equal to $|F_s|$ and nodes having the same depth use the same attribute to divide the data;
2) A local test is performed for node(s) with the same depth. Attributes in candidate attribute set CF_s ($CF_s = F_s$ for root node in the tree constructed for the bundle) will be tested and the feature (denoted as *BF*) which causes the best *IG*((*BF*)) and *GR*((*BF*)) is chosen. Update the candidate feature set CF_s by removing the used feature $CF_s = CF_s - BF$.
3) Construct the tree using rule (2) until all features are used or there is no partition which can lead greater than 0 information gain.

Steps of calculating cut points of example in Fig. 2 are:

1) Information gain and gain ratio are calculated for features $f1$, $f2$ and $f3$. The feature leading best *IG* and *GR* is chosen. In Fig. 2, $f2$ is selected and then removed from candidate feature set: $CF_s = \{f1, f3\}$.
2) $f1$ and $f3$ are tested for nodes with depth 1 (root has depth 0). Note that two cut points for each of them will be calculated and the *IG* and *GR* are computed over all four datasets produced. Then $f3$ is selected and $CF_s = \{f1\}$.
3) Use $f1$ to partition the data.

One exception of above steps is that if there is no partition can lead information gain greater then 0, the tree stops growing.

For all nodes with depth i in the tree (root has depth 0), we need $((k-i) \times n)$ tests. Furthermore, since nodes in the same depth use the same attribute to partition data, the number of total tests in one bundle with k attributes is

$$k \times n + (k-1) \times n + \ldots + n = \frac{(1+k) \times k}{2} \times n. \tag{10}$$

Obviously, $(1 + k)/2 < k!$ when $k \geqslant 2$. So the greedy bundling is more effective than the abstract bundling in the terms of runtime cost. On the other hand, the greedy bundling is more compatible with C4.5 (no backtracking) and it can be easily implemented based on the original C4.5 decision tree algorithm.

3.2.3. Local greedy bundling

Attributes in the same bundle will be tested together in local greedy bundling method. As discussed above, the time complexity of finding the best cut points for all k attributes in one node with n cases is $O(n^k)$. To reduce the time consumption of the algorithm, we calculate the best cut point for each attribute of a bundle separately. The method breaks the rule that only one attribute can be tested in one node.

Given dataset D and a bundle $f1\_\ f2\_\ldots\_\ f_n$ consisting of a set of features $F_s = \{f1, f2, \ldots, f_n\}$,

$$\textit{Discretize}(D, f_1\_f_2\_\ldots\_f_n) = \{cpl_1, cpl_2, \ldots, cpl_n\}, \tag{11}$$

where $cpl_i = \textit{Discretize}(D, f_i)$.

Consequently, if k attributes are in the node, there should be 2^k branches after splitting. This method is effective because only $n \times k$ tests are needed.

4. Experimental results

4.1. Datasets

To make bundles reliable and reasonable, we use two datasets to which we are familiar in the experiments, WEBSPAM-UK2006 [3] and WEBSPAM-UK2007 [17]. These two datasets consist of more than 100 M HTMLs collected from about 10,000 UK domain hosts. In addition to original HTMLs of pages, the datasets also present a set of pre-computed features (139 features), web spam labels (spam or non-spam), host level web graph and page level web graph. Typically, these datasets are used for several research areas including web spam detection [18], web spam demotion [15], natural language analysis [11], imbalanced learning [7] and so on. In our experiments, the web spam detection task is conducted.

4.2. Feature bundles

Feature bundling explores potential relationships among different features. Features in one bundle should perform close relationships by their natures rather than values. On the basis of pre-computed features presented in WEBSPAM-UK2006/2007 datasets, we propose three kinds of bundles for web spam detection.

1) *hp_mp* and *hp_mp_average* bundles. In many hosts, the page with maximum pagerank value is exactly the home page. So it is natural to group these two versions of features into one bundle. However for *hp_mp_average* bundle, it is really very difficult to analyze relationships among the three versions of features. If the host has very few pages, its *hp* and *mp* can reflect its most information and thus they may have significant impacts on the average value. However for hosts with many pages, the impacts may be smaller.

Table 6
Experimental result on WEBSPAM-UK2007

Algorithm	Bundles				
	No bundling	hp_mp	hp_mp_average	Content related	Link related
Static bundling	0.657	0.703	0.691	0.711	0.688
Abstract bundling	0.599	0.622	0.623	0.631	0.614
Abstract greedy bundling	0.599	0.620	0.611	0.615	0.623
Local greedy bundling	0.599	0.687	0.665	0.678	0.658

Table 7
Experimental result on WEBSPAM-UK2006

Algorithm	Bundles				
	No bundling	hp_mp	hp_mp_average	Content related	Link related
Static bundling	0.886	0.891	0.892	0.889	0.891
Abstract bundling	0.878	0.884	0.882	0.885	0.892
Abstract greedy bundling	0.878	0.888	0.884	0.892	0.895
Local greedy bundling	0.878	0.880	0.886	0.881	0.880

2) Bundles based on variable based features. Features using the same function but with different parameters are close related. Researchers developed these features to make a collective judgment and the motivation is also can be applied to algorithms. Take these features into account together, algorithm can find more reliable and stable information to build classifier.
3) Ad-hoc bundles. The above two types of bundles are designed for their naturality. The ad-hoc bundles are more related to a specific research area. For example, a host with many out-links but very few in-links are quite suspected because it may be an artificial node in the link farm. In such case, it is valuable to consider in-links and out-links at the same time.

4.3. Experiments

In this section, we use four bundling algorithms and five different bundles to test effectiveness of our methods. *hp_mp* bundle and *hp_mp_average* bundle are explained in Section 4.2. The content related bundle is only based on content features. In such a bundle, *hp_mp* bundle and *hp_mp_average* bundle are not used again. Instead, variable based features and ad-hoc bundles are used. The link related bundle is dealt with in similar way.

We use AUC (Area Under ROC Curve) to evaluate the results because these two data sets are quite imbalanced. Using precision, recall and F1 measure cannot tell us enough information about the classifiers since all of them may get very good values due to imbalanced data. AUC can comprehensively evaluate the usefulness of classifiers to correctly identify cases in each class.

From Table 6, we see that the static bundling performs best. By using additional bundles, about 3%–6% improvements are gained. In dynamical bundling, the local greedy bundling outperforms others. The best improvement is near to 9%. In Table 7, results are much better than in Table 6 because of the more balanced data. Slight improvements are obtained on this dataset.

We showed that our methods can get some improvements on the two datasets. Especially when the dataset is quite imbalanced (like WEBSPAM-UK2007), the relationships in natures of attributes are more reliable and stable than only in values. Our experiments imply that the natural relationships among features are important as in values.

5. Conclusions

Feature bundling is a feature transformation method that focuses on exploring relationships among different attributes by their natures rather than values. Relationships based on values are largely affected by the quality of training set while relationships in nature are more reliable and stable.

To lead algorithms "understand" such natural relationships, we developed both static bundling and dynamical bundling algorithms. The static bundling adopts logical operator AND to combine attributes using their nominal values. This method is easy for implementing and experiments showed it get best results on the two data sets. However, an additional pre-processing step cause it is not suitable for some real time environments. To overcome such flaw, we design three different dynamical bundling algorithms based on C4.5 decision tree algorithm. From our experiments, the local greedy bundling method can get better improvements than others and it also has minimum time complexity. Compared with original C4.5 algorithm, our method can get 2%–9% improvements on WEBSPAM-UK2007 and very slight improvements (about 1%) on WEBSPAM-UK2006.

This study focuses on evaluating the effectiveness of bundling method rather than building a good classifier for web spam detection. Much better results can be obtained if some other methods are used to deal with special problems in the field. For example, sampling method can be used to solve the imbalanced problem. Our research shows that the natural relationships are important and can largely improve the performance of classifiers especially when the quality of training data is not good enough.

References

[1] A. Kusiak, Feature transformation methods in data mining, *IEEE Transactions on Electronics Packaging Manufacturing* **24** (2001), 214–221.
[2] A. Ntoulas, M. Najork, M. Manasse and D. Fetterly, Detecting spam web pages through content analysis, in: *Proceedings of the World Wide Web Conference*, Scotland (2006), 83–92.
[3] C. Castillo, D. Donato, L. Becchetti, P. Boldi, M. Santini and S. Vigna, A reference collection for web spam, *SIGIR Forum* **40** (2006), 11–24.
[4] C. Castillo, K. Chellapilla and L. Denoyer, Web spam challenge 2008, in: *Proceedings of the 4th International Workshop on Adversary Information Retrieval on the Web (AIRWeb)* (2008).
[5] C. Matheus, The need for constructive induction, in: *Machine Learning – Proceedings of the Eighth International Workshop* (1991), 173–177.
[6] H. Abdi and L.J. Williams, Principal component analysis, *Wiley Interdisciplinary Reviews: Computational Statistics* **2** (2010), 433–459.
[7] H. He and E.A. Garcia, Learning from imbalanced data, *IEEE Transactions on Knowledge and Data Engineering* **21** (2009), 1263–1284.
[8] H. Liu and H. Motoda, Feature transformation and subset selection, *IEEE Intell Syst Their Appl* **13** (1998), 26–28.
[9] H. Zhao, A.P. Sinha and W. Ge, Effects of feature construction on classification performance: An empirical study in bank failure prediction, *Expert Systems with Application* **36** (2009), 2633–2644.
[10] I. Guyou and A. Elisseeff, An introduction to feature extraction, Springer Berlin Heidelberg, 2006, pp. 1–25.
[11] J. Martinez-Romo and L. Araujo, Web spam identification through language model analysis, in: *Proceedings of AIRWeb '09*, Spain (2009), 21–28.
[12] J.R. Quinlan, *C4.5: Programs For Machine Learning*, Morgan Kaufmann, San Meteo, CA, 1993.
[13] L. Becchetti, C. Castillo, D. Donato, S. Leonardi and R. Baeze-Yates, Link-based characterization and detection of web spam, in: *AIRWeb* (2006), 1–8.
[14] S. Garcia, J. Luengo, J.A. Saez, V. Lopez and F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, *IEEE Transactions on Knowledge and Data Engineering* **25** (2013), 734–750.
[15] X. Liu, Y. Wang, S. Zhu and H. Lin, Combating web spam through trust-distrust propagation with confidence, *Pattern Recognition Letters* **34** (2013), 1462–1469.
[16] X. Wu, V. Kumar, J.R. Quinlan et al., Top 10 algorithms in data mining, *Knowledge and Information Systems* **14** (2008), 1–37.

[17] Yahoo! Research: Web Spam Collections. http://barcelona.research.yahoo.net/webspam/datasets/Crawled by the Laboratory of Web Algorithmics, University of Milan, http://law.dsi.unimi.it/.
[18] Z. Gyongyi and H. Garcia-Molina, Web spam taxonomy, in: *Proceedings of the 1st International Conference on Very Large Data Bases* (2005).
[19] Z. Pawlak, Rough set: Theoretical aspects of reasoning about data, Boston, 1991.