Computing Optimal Hypotheses Efficiently for Boosting

Shinichi Morishita

University of Tokyo
moris@k.u-tokyo.ac.jp
http://gi.k.u-tokyo.ac.jp/~moris/

Abstract. This paper sheds light on a strong connection between AdaBoost and several optimization algorithms for data mining. AdaBoost has been the subject of much interests as an effective methodology for classification task. AdaBoost repeatedly generates one hypothesis in each round, and finally it is able to make a highly accurate prediction by taking a weighted majority vote on the resulting hypotheses. Freund and Schapire have remarked that the use of simple hypotheses such as single-test decision trees instead of huge trees would be promising for achieving high accuracy and avoiding overfitting to the training data. One major drawback of this approach however is that accuracies of simple individual hypotheses may not always be high, hence demanding a way of computing more accurate (or, the most accurate) simple hypotheses efficiently. In this paper, we consider several classes of simple but expressive hypotheses such as ranges and regions for numeric attributes, subsets of categorical values, and conjunctions of Boolean tests. For each class, we develop an efficient algorithm for choosing the optimal hypothesis.

1 Introduction

Classification has been a prominent subject of study in the machine learning and data mining literature. Let $\mathbf{x}_i$ denote a vector of values for attributes, which is usually called a *record* or a *tuple* in the database community. Let y_i denote the objective Boolean value that is either 1 or 0. We call a record $(\mathbf{x}_i, y_i)$ *positive* (resp. *negative*) if $y_i = 1$ ($y_i = 0$). Given $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ as a training dataset, classification aims at deriving rules that are capable of predicting the objective value of y from $\mathbf{x}$ with a high probability.

For classification problems, decision trees are used mostly in practical applications. Recently, to further improve the prediction accuracy of existing classifiers, boosting techniques have received much interest among the machine learning and data mining communities [14]. A classifier is called a *weak hypothesis* if its predication accuracy regarding the training dataset is at least better than 1/2. A boosting algorithm tries to generate some weak hypotheses so that it makes it possible to perform a highly accurate prediction by combining those weak hypotheses. There have been many proposals for such boosting algorithms [14,

7]. Freund and Schapire presented the most successful algorithm, named "AdaBoost", that solved many of the practical difficulties of the earlier boosting algorithms [9].

2 AdaBoost

The key idea behind AdaBoost is to maintain the record weights in the training dataset. AdaBoost assumes the existence of a *weak learner* that is able to output a weak hypothesis in a finite number of steps, though it is not always the case in practice. To overcome this problem, we will propose efficient weak learners that manage to output optimal hypotheses, but for the purpose of explanation, we continue the discussion by assuming that weak hypotheses can always be generated.

In each iteration AdaBoost calls on a weak learner to generate one weak hypothesis by considering the weighted records as the training dataset and updates the weights of the records to force the next call of the weak learner focus on the mis-predicted records. In this way, we prepare a set of voters with different characteristics. In the final step, we define the weight of each voter according to its prediction accuracy in the training dataset, and we generate the final hypothesis using a weighted majority vote.

2.1 Pseudo-code

We now present a pseudo-code for AdaBoost. First, the inputs to AdaBoost are a training dataset $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, the initial weights $w_i^1 = 1/N$ $(i = 1, \ldots, N)$, a weak learner named `WeakLearn`, and the integer T specifying number of iterations. AdaBoost repeats the following three steps for each $t = 1, 2, \ldots, T$:

1. Calculate the distribution p_i^t of each record $(\mathbf{x}_i, y_i)$ by normalizing weights w_i^t; namely,
$$p_i^t = \frac{w_i^t}{\sum_{i=1}^N w_i^t}.$$
2. Invoke `WeakLearn` to produce such a weak hypothesis $h_t : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \to \{1, 0\}$ that the error ϵ_t of h_t is less than $1/2$, where ϵ_t is defined:
$$\epsilon_t = \sum_{i=1}^N p_i^t \, |h_t(\mathbf{x}_i) - y_i|.$$
Observe that $|h_t(\mathbf{x}_i) - y_i| = 1$ if h_t mis-predicts the objective value y_i. Otherwise, $|h_t(\mathbf{x}_i) - y_i| = 0$.
3. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$. Note that $\beta_t < 1$ since $\epsilon_t < 1/2$. We then set the new weight w_i^{t+1} for each $i = 1, \ldots, N$ according to the formula:
$$w_i^{t+1} = w_i^t \; \beta_t^{1-|h_t(\mathbf{x}_i) - y_i|}.$$

If h_t mis-predicts the objective value y_i of the i-th records (x_i, y_i), observe that

$$\beta_t^{1-|h_t(\mathbf{x}_i) - y_i|} = 1,$$

and hence the weight does not change; namely $w_i^{t+1} = w_i^t$. Otherwise the weight decreases, because $w_i^{t+1} < w_i^t \beta_t$. Put another way, the weights of incorrectly predicted records relatively increase so that the weak learner can focus on these "hard" records in the next step.

Lastly, AdaBoost outputs the final hypothesis h_f that is a weighted majority vote of T weak hypotheses where the weight $-\ln \beta_t$ is associated with hypothesis h_t:

$$h_f(\mathbf{x}) = \begin{cases} 1 \text{ if } \sum_{t=1}^{T} (-\ln \beta_t) h_t(\mathbf{x}) \geq \sum_{t=1}^{T} (-\ln \beta_t) \frac{1}{2} \\ 0 \text{ otherwise.} \end{cases}$$

2.2 Boosting Property

Freund and Schapire proved the following theorem, which is called the *boosting property*.

Theorem 1. [9] Let ϵ denote $(\sum_{i=1}^{N} |h_f(\mathbf{x}_i) - y_i|)/N$, the error of the final hypothesis. Then,

$$\epsilon \leq \Pi_{t=1}^{T} 2(\epsilon_t(1-\epsilon_t))^{\frac{1}{2}} \quad \blacksquare$$

The above theorem indicates that the error of the final hypothesis adapts to the error of individual weak hypothesis. Since the error ϵ_t is less than $1/2$, $2(\epsilon_t(1-\epsilon_t))^{\frac{1}{2}}$ is strictly less than 1, and it approaches to 0 when the error ϵ_t gets to be closer to 0. Thus, if most weak hypotheses are moderately accurate, the error of the final hypothesis drops exponentially fast, minimizing the number of iterations T efficiently.

2.3 Decision Stumps and Optimization

To design the weak learner of AdaBoost, in the literature, C4.5 has been frequently used as a weak learner [3, 8]. In [8] Freund and Schapire employed C4.5 to generate large decision trees, but they remarked that large decision trees were a kind of overly complex weak hypothesis and thereby failed to perform well against unseen test datasets.

This result led them to consider the other extreme class of decision trees, single-test decision trees named *decision stumps*. Decision stumps, on the other hand, are simple and hence may not be subject to overfitting to the training datasets. However, Domingo and Watanabe [6] reported that in later rounds of iterations in AdaBoost, the resulting decision stump is often too weak, yielding a large number of very weak hypotheses for improving the prediction accuracy of the final hypothesis.

One solution to overcome this problem is to select among decision stumps the optimal one that minimizes the error, because Theorem 1 indicates that

the choice of highly accurate hypothesis enables sharp reduction of the error of the final hypothesis. To further investigate this optimization problem, we here introduce some new terms. Minimization of the error ϵ_t of the t-th hypothesis h_t is equivalent to maximization of the prediction accuracy $1 - \epsilon_t$.

$$\begin{aligned}
1 - \epsilon_t &= \sum_{\{i|h_t(\mathbf{x}_i)=y_i\}} p_i^t \\
&= \sum_{\{i|h_t(\mathbf{x}_i)=y_i=1\}} p_i^t + \sum_{\{i|h_t(\mathbf{x}_i)=y_i=0\}} p_i^t \\
&= \sum_{\{i|h_t(\mathbf{x}_i)=y_i=1\}} p_i^t + (\sum_{\{i|y_i=0\}} p_i^t - \sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=0\}} p_i^t) \\
&= (\sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=1\}} p_i^t - \sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=0\}} p_i^t) + \sum_{\{i|y_i=0\}} p_i^t.
\end{aligned}$$

Since $\sum_{\{i|y_i=0\}} p_i^t$ is independent of the choice of h_t, our goal is to maximize the first term enclosed in the parentheses, which we will simplify using g_i^t defined:

$$g_i^t = \begin{cases} p_i^t & \text{if } h_t(\mathbf{x}_i) = 1 \text{ and } y_i = 1 \\ -p_i^t & \text{if } h_t(\mathbf{x}_i) = 1 \text{ and } y_i = 0 \end{cases}$$

Then,

$$(\sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=1\}} p_i^t - \sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=0\}} p_i^t) = \sum_{\{i|h_t(\mathbf{x}_i)=1\}} g_i^t$$

If h_t outputs the correct answer to y_i, g_i^t is positive, thereby adding gain p_i^t to the prediction accuracy $1 - \epsilon_t$. Otherwise, p_i^t is deducted from the accuracy. We therefore call g_i^t the *accuracy gain* (or *gain*, for short) of prediction when h_t outputs 1. Consequently, maximization of the accuracy $1 - \epsilon_t$ is equivalent to maximization of the sum of gains, $\sum_{\{i|h_t(\mathbf{x}_i)=1\}} g_i^t$.

3 Main Results

We here present some optimization algorithms for selecting an optimal hypothesis h_t minimizing the sum of gains from a class of simple hypotheses. Classes of our interest include ranges and regions for numeric attributes, subsets of categorical values, and conjunctions of Boolean tests.

3.1 Optimal Ranges

We here consider hypotheses h_t such that

$$h_t(x_i) = 1 \quad \text{iff} \quad x_i \in [l, h].$$

Our goal is to compute an optimal range $[l, h]$ that maximizes the accuracy gain of h_t; namely

$$\max \sum_{\{i|x_i \in [l,h]\}} g_i^t.$$

Without loss of generality, we assume that x_i are sorted in an ascending order $x_1 \leq x_2 \leq x_3 \leq \ldots$, which requires $O(N \log N)$-time sorting cost though. If some $x_i, \ldots, x_{i+k}$ are equal, we merge them in the sense that we take the sum of gains $g_i^t + \ldots + g_{i+k}^t$, assign the sum to x_i, and rename the indexes so that all indexes are consecutive; namely, $x_1 < x_2 < \ldots$. Then, input the sequence of gains $g_1^t, g_2^t, \ldots$ to Kadena's algorithm [4]. Given a sequence of M reals $g_1, g_2, \ldots, g_M$, Kadena's algorithm computes an optimal range $[s, t]$ that maximizes $\sum_{i \in [s,t]} g_i$ in $O(M)$-time. M is at most the number of records N but is typically much smaller than N, and hence Kadena's algorithm works in $O(N)$-time.

It is natural to consider the use of two or more disjoint ranges for maximizing accuracy gain. Brin, Rastogi, and Shim presented an efficient way of computing the optimal set of at most k ranges in $O(kM)$-time [5], which is a highly non-trivial extension of Kadena's algorithm.

3.2 Optimal Regions

We have discussed the advantage of using region splitting hypotheses of the form:

$$h_t((x_{i1}, x_{i2})) = 1 \quad \text{iff} \quad (x_{i1}, x_{i2}) \in [v_1, v_2] \times [w_1, w_2].$$

We here present how to efficiently compute an optimal rectangle R that maximizes the accuracy gain; namely,

$$\max \sum \{g_i^t \mid (x_{i1}, x_{i2}) \in R\}.$$

In order to limit the number of rectangles to a moderate number, we first divide the domain of x_{i1} (also, x_{i2}) into M non-overlapping buckets such that their union is equal to the original domain. M is at most the number of records N but is typically much smaller than N. Using those buckets, we divide the two dimensional plane into M^2 pixels, and we represent a rectangle as a union of those pixels. Although the number of rectangles is $O(M^4)$, it is straightforward to design an $O(M^3)$-time (namely, $O(N^3)$-time) algorithm by using Kadena's algorithm. The idea is that for each of ${}_M C_2$ pairs of rows, we apply Kadena's algorithm to calculate the optimal range of columns. Also, a subcubic time algorithm that uses funny matrix multiplication [15] is also available.

We are then interested in the design of efficient algorithm for computing more than one rectangles for maximizing the accuracy gain. However, Khanna, Muthukrishnan and Paterson remark that the problem is NP-hard [11]. Brin, Rastogi, and Shim present an approximation algorithm for this problem [5], however the approximation is within a factor of $\frac{1}{4}$ of the optimal solution. Thus computing the optimal set of more than one rectangles is computationally intractable.

So far we have been focusing on rectangles. In general there have been developed efficient algorithms for computing the optimized gain region among various classes of two dimensional connected regions whose boundaries are more flexible than those of rectangles. For instance, an *x-monotone* region is such a connected

region that the intersection with any column is undivided, and a *rectilinear convex* region is such an x-monotone region that the intersection with any row is also undivided. The optimized gain x-monotone (rectilinear convex, respectively) region can be computed in $O(M^2)$-time [10] (in $O(M^3)$-time [16]). Since $M \leq N$, M in $O(M^2)$ and $O(M^3)$ can be replaced with N. The use of these regions is expected to further improve the prediction accuracy of the final hypothesis.

3.3 Optimal Conjunctions

Given a dataset $\{(x_{i1}, x_{i2}, \ldots, x_{in}, y_i) \mid i = 1, \ldots, N\}$ such that x_{ij} is Boolean valued; that is, $x_{ij} \in \{0, 1\}$. In this section, we consider hypotheses h_t that are conjunctions of simple tests on each attribute; that is;

$$h_t((x_{i1}, x_{i2}, \ldots, x_{in})) = 1 \quad \text{iff} \quad x_{ij_1} = 1 \wedge \ldots \wedge x_{ij_M} = 1. \tag{1}$$

We are then interested in computing the optimal conjunction that maximizes the gain. The number of conjunctions of the form (1) is ${}_nC_M$. If we treat M as a variable, we can prove that the problem is NP-hard by reducing the problem to the NP-completeness of the minimum set cover problem according to the line suggested in [12, 13].

In practice, it would be reasonable to limit the number M of conjuncts to a small constant, say five. Then, the problem becomes to be tractable, but the problem still demands an efficient way of computing the optimal conjunction especially when the number of attributes n is fairly large.

Connection with Itemset Enumeration Problem We first remark that the problem has a strong connection with itemset enumeration problem [1]. We then generalize the idea of Apriori algorithm [2] for computing the optimal conjunction efficiently.

Let $a_1, a_2, \ldots, a_n$ be n items. We will identify a record with an itemset according to the mapping ϕ:

$$\phi : (x_{i1}, \ldots, x_{in}) \ \rightarrow \ \{a_j \mid x_{ij} = 1\}.$$

Example 1. $\phi((1, 0, 1, 1, 0)) = \{a_1, a_3, a_4\}$. ∎

We also regard the conjunction $x_{ij_1} = 1 \wedge \ldots \wedge x_{ij_M} = 1$ as $\{a_{j_1}, \ldots, a_{j_M}\}$.

Example 2. We associate $x_{i3} = 1 \wedge x_{i4} = 1$ with $\{a_3, a_4\}$. The property that the record $(1, 0, 1, 1, 0)$ satisfies $x_{i3} = 1 \wedge x_{i4} = 1$ can be rephrased by using words of itemsets; namely, $\{a_1, a_3, a_4\} \supseteq \{a_3, a_4\}$. ∎

In general, we have the following equivalence:

$$\begin{aligned} & h_t((x_{i1}, x_{i2}, \ldots, x_{in})) = 1 \\ \text{iff} \quad & x_{ij_1} = 1 \wedge \ldots \wedge x_{ij_M} = 1 \\ \text{iff} \quad & \phi((x_{i1}, x_{i2}, \ldots, x_{in})) \supseteq \{a_{j_1}, \ldots, a_{j_M}\}. \end{aligned}$$

In what follows, let $\mathbf{x}_i$ denote $(x_{i1}, x_{i2}, \ldots, x_{in})$, for simplicity. Now finding the optimal conjunction of the form (1) that maximizes the sum of gains is equivalent to the computation of the itemset I that maximizes

$$\sum_{\{i|\phi(\mathbf{x}_i)\supseteq I\}} g_i^t.$$

Let us call the above sum of gains the *gain* of I, and let $gain(I)$ denote the sum. The gain of I may appear to be similar to the so-called *support* of I, which is defined as $|\{i \mid \phi(\mathbf{x}_i) \supseteq I\}|/N$, where N is the number of all records. Thus one may consider the possibility of applying the Apriori algorithm to the calculation of the optimal itemset I that maximizes $gain(I)$. To this end, however, Apriori needs a major conversion.

Extending the Idea of Apriori Algorithm Let $support(I)$ denote the support of I. The support is *anti-monotone* with respect to set-inclusion of itemsets; that is, for any $J \supseteq I$, $support(J) \leq support(I)$. The Apriori algorithm uses this property to effectively prune away a substantial number of unproductive itemsets from its search space. However, the gain is not anti-monotone; namely, $J \supseteq I$ does not always imply $gain(J) \leq gain(I)$, because some g_i^t could be negative.

We solve this problem by modifying the Apriori algorithm so that it is capable of handling the anti-monotone gain function. Suppose that during the scan of the lattice of itemsets beginning with smaller itemsets and continuing to larger ones, we visit an itemset I. The following theorem presents a way of computing a tight upper bound on $gain(J)$ for any superset J of I.

Theorem 2. For any $J \supseteq I$,

$$gain(J) \leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} g_i^t.$$

Proof. Recall

$$g_i^t = \begin{cases} p_i^t & \text{if } y_i = 1 \\ -p_i^t & \text{if } y_i = 0. \end{cases}$$

Then,

$$gain(J) = \sum_{\{i|\phi(\mathbf{x}_i)\supseteq J\}} g_i^t = \sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=1\}} p_i^t - \sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=0\}} p_i^t$$

Since $p_i^t \geq 0$ and $\{i \mid \phi(\mathbf{x}_i) \supseteq J, y_i = 1\} \subseteq \{i \mid \phi(\mathbf{x}_i) \supseteq I, y_i = 1\}$,

$$\sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=1\}} p_i^t \leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} p_i^t.$$

Then,

$$\begin{aligned} gain(J) &= \sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=1\}} p_i^t - \sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=0\}} p_i^t \\ &\leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=1\}} p_i^t \leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} p_i^t. \end{aligned}$$

∎

Definition 1. Let $u(I)$ denote the upper bound

$$\sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} g_i^t.$$

∎

During the scan of the itemset lattice, we always maintain the temporarily maximum gain among all the gains calculated so far and set it to τ. If $u(I) < \tau$, no superset of I gives a gain greater than or equal to τ, and hence we can safely prune all supersets of I at once. On the other hand, if $u(I) \geq \tau$, I is promising in the sense that there might exist a superset J of I such that $gain(J) \geq \tau$.

Definition 2. Suppose that τ is given and fixed. An itemset is a k-itemset if it contains exactly k items. An itemset I is promising if $u(I) \geq \tau$. Let P_k denote the set of promising k-itemsets. ∎

Thus we will search $P_1 \cup P_2 \cup \ldots$ for the optimal itemset. Next, to accelerate the generation of P_k, we introduce a candidate set for P_k.

Definition 3. An itemset I is *potentially promising* if every proper subset of I is promising. Let Q_k denote the set of all potentially promising k-itemsets. ∎

The following theorem guarantees that Q_k is be a candidate set for P_k.

Theorem 3. $Q_k \supseteq P_k$. ∎

$\tau := 0$;
$Q_1 := \{I \mid I$ is a 1-itemset.$\}$; $k := 1$;
repeat begin
    If $k > 1$, generate Q_k from P_{k-1};
    For each $I \in Q_k$, scan all the records to compute $u(I)$ and $gain(I)$;
    $\tau := \max(\tau,\ \max\{gain(I) \mid I \in Q_k\})$;
    $P_k := \{I \in Q_k \mid u(I) \geq \tau\}$; $X := P_k$; $k++$;
end until $X = \phi$;
Return τ with its corresponding itemset;

Fig. 1. AprioriGain for Computing the Optimal Itemset

The benefit of Q_k is that Q_k can be obtained from P_{k-1} without scanning all records that may reside in the secondary disk. To this end, we use the idea of the `apriori-gen` function of the Apriori algorithm [2]; that is, we select two members in P_{k-1}, say I_1 and I_2, such that I_1 and I_2 share $(k-2)$ items in common, and then check to see whether each $(k-1)$-itemset included in $I_1 \cup I_2$ belongs to P_{k-1}, which can be determined efficiently by organizing P_{k-1} as a hash tree structure. We repeat this process to create Q_k. Figure 1 presents the overall algorithm, which we call AprioriGain (Apriori for optimizing Gain).

3.4 Optimal Subsets of Categorical Values

We here suppose that x_i itself denotes a single categorical value. Let $\{c_1, \ldots, c_M\}$ be the domain of the categorical attribute, where M is at most the number of records N but is typically much smaller than N. Typical hypotheses h_t would be of the form:

$$h_t(x_i) = 1 \quad \text{iff} \quad x_i = c_j.$$

Computing the optimal choice of c_j that maximizes the accuracy gain is inexpensive. In practice, the number of categorical values M could be fairly large; for instance, consider the number of countries in the world. In such cases, the number of records satisfying $x_i = c_j$ could be relatively small, thereby raising the error of the hypothesis h_t. One way to overcome this problem is to use a subset S of $\{c_1, c_2, \ldots, c_M\}$ instead of a single value and to employ hypotheses of the form:

$$h_t(x_i) = 1 \quad \text{iff} \quad x_i \in S.$$

Our goal is then to find S that maximizes the sum of gains $\sum_{x_i \in S} g_i^t$. Although the number of possible subsets of $\{c_1, c_2, \ldots, c_M\}$ is 2^M, we are able to compute the optimal subset S in $O(M)$-time. First, without loss of generality, we assume that

$$\sum_{\{i|x_i=c_1\}} g_i^t \geq \sum_{\{i|x_i=c_2\}} g_i^t \geq \cdots \geq \sum_{\{i|x_i=c_M\}} g_i^t.$$

Otherwise, we rename the indexes so that the above property is guaranteed. It is easy to see the following property.

Theorem 4. Let $S = \{c_j \mid \sum_{\{i|x_i=c_j\}} g_i^t \geq 0\}$. S maximizes $\sum_{x_i \in S} g_i^t$. ∎

Thus we only need to find the maximum index k such that

$$\sum_{\{i|x_i=c_k\}} g_i^t \geq 0,$$

returning $S = \{c_j \mid j = 1, \ldots, k\}$ as the answer. Consequently, the optimal subset can be computed in $O(M)$-time (or, $O(N)$-time since $M \leq N$).

4 Discussion

To improve the prediction accuracy of AdaBoost, we have presented efficient algorithms for several classes of simple but expressive hypotheses. In the literature, boosting algorithms have been developed in the machine learning community, while optimization algorithms for association rules and optimized ranges/regions have been proposed and studied in the database and data mining communities. This paper sheds light on a strong connection between AdaBoost and optimization algorithms for data mining.

Acknowledgement Special thanks go to Jun Sese and Hisashi Hayashi. Jun Sese implemented AdaBoost empowered with the method of computing optimal ranges. Hisashi Hayashi also developed AdaBoost with AprioriGain. Both of them applied their implementations to real datasets. We also thank Carlos Domingo, Naoki Katoh, and Osamu Watanabe for motivating me to pursue this work. Comments from anonymous referees were useful to improve the readability.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.
3. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(2):105–139, 1999.
4. J. Bentley. Programming pearls. *Communications of the ACM*, 27(27):865–871, Sept. 1984.
5. S. Brin, R. Rastogi, and K. Shim. Mining optimized gain rules for numeric attributes. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 15-18 August 1999, San Diego, CA USA*, pages 135–144. ACM Press, 1999.
6. C. Domingo and O. Watanabe. A modification of adaboost: A preliminary report. *Research Reports, Dept. of Math. and Comp. Sciences, Tokyo Institute of Technology*, (C-133), July 1999.
7. Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
8. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.
9. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997.
10. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized accociation rules: Scheme, algorithms, and visualization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 13–23. ACM Press, 1996.
11. S. Khanna, S. Muthukrishnan, and M. Paterson. On approximating rectangle tiling and packing. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, Jan. 1998.
12. S. Morishita. On classification and regression. In *Proceedings of Discovery Science, First International Conference, DS'98 — Lecture Notes in Artificial Intelligence*, volume 1532, pages 40–57, Dec. 1998.
13. S. Morishita and J. Sese. Traversing itemset lattices with statistical metric pruning. In *Proc. of ACM SIGACT-SIGMOD-SIGART Symp. on Database Systems (PODS)*, pages 226–236, May 2000.
14. R. E. Schapire. The strength of weak learnability (extended abstract). In *FOCS*, pages 28–33, 1989.
15. H. Tamaki and T. Tokuyama. Algorithms for the maxium subarray problem based on matrix multiplication. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 446–452, Jan. 1998.
16. K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing optimized rectilinear regions for association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 96–103, Aug. 1997.