Distributed Subgroup Mining

Michael Wurst and Martin Scholz

Artificial Intelligence Group, University of Dortmund, Germany
{wurst,scholz}@ls8.cs.uni-dortmund

Abstract. Subgroup discovery is a popular form of supervised rule learning, applicable to descriptive and predictive tasks. In this work we study two natural extensions of classical subgroup discovery to distributed settings. In the first variant the goal is to efficiently identify global subgroups, i.e. the rules an analysis would yield after collecting all the data at a single central database. In contrast, the second considered variant takes the locality of data explicitly into account. The aim is to find patterns that point out major differences between individual databases with respect to a specific property of interest (target attribute). We point out substantial differences between these novel learning problems and other kinds of distributed data mining tasks. These differences motivate new search and communication strategies, aiming at a minimization of computation time and communication costs. We present and empirically evaluate new algorithms for both considered variants.

1 Introduction

The aim of data mining is to find useful patterns in large data collections. Distributed computing plays an important role in this process for several reasons. First, data mining often requires huge amounts of resources in storage space and computation time. To make systems scalable, it is important to develop mechanisms that distribute the work load among several sites in a flexible way. Second, data is often inherently distributed to several databases, making a centralized processing of this data very inefficient and prone to security risks. Algorithms for several data mining tasks were proposed, for example for distributed association rule mining [1], classification, clustering and dimensionality reduction [2].

The goal of subgroup discovery [3] is to identify interesting rules. In this paper we study two distributed subgroup discovery tasks. The first one aims at the discovery of global subgroups from distributed databases using distributed algorithms. Its two objectives are to find the same rules as centralized learners [4], and to minimize communication costs. The second distributed task aims to detect relative local subgroups, which describe characteristics of individual databases concerning a target value. Such rules can, for instance, capture information how sales deviate at certain branches of a company.

In Sec. 2 existing work on subgroup discovery is presented and extended to distributed settings in Sec. 3. Novel algorithmic solutions for distributed subgroup discovery are proposed in Sec. 4 and evaluated in Sec. 5.

2 Subgroup discovery

Subgroup discovery aims at the identification of interesting and interpretable rules [3, 5]. Each subgroup describes a subset of the overall population in a database, which deviates from the overall behavior in terms of a given property of interest. Among the typical applications of subgroup discovery are the identification of homogeneous groups of clients in marketing, and the induction of interpretable rules in medical domains. Case studies illustrating benefits of subgroup discovery for decision support can be found in [6]. An example application in a medical domain with a focus on incorporating background knowledge and user-defined preferences is described in [7].

Formally, the learning task shares some of the characteristics of classifier induction. In particular it is also a supervised learning task. Instances of the population are referred to as *examples* $x \in \mathcal{X}$ in this paper, while the property of interest can be formalized as a target attribute $\mathcal{Y}$. Every subgroup is represented by a rule $A \rightarrow C$ with antecedent $A : \mathcal{X} \rightarrow \{true, false\}$ that is a conjunction of literals, and a conclusion $C : \mathcal{Y} \rightarrow \{true, false\}$.

In subgroup discovery the interestingness of rules is measured in terms of a user-given *utility function*, a parameter of the task itself. Hence, subgroup discovery can be considered to define a very broad rule discovery framework, covering classification as a specific case. However, the focus of subgroup discovery differs from inducing classifiers. Although the discovered sets of rules can well be used to make predictions if interpreted probabilistically [8], subgroup discovery is typically used for descriptive data analysis. Different selection metrics reflecting rule interestingness have been motivated in [5]. The main objective of utility functions is to trade-off between two quantities, which both indicate interestingness but tend to be diametric. These quantities are formalized in the following two definitions. To ease notation we denote the absolute number of true positives of a rule r with $p(r)$, and the number of its false positives with $n(r)$. The argument is omitted if clear from the context. P and N denote the number of positives and negatives in the complete dataset.

Definition 1. *For a given database $E \subseteq X \times Y$ the* support *of a rule $A \rightarrow C$ is denoted as $Sup(A \rightarrow C)$. It is defined as*

$$Sup(A \rightarrow C) := \frac{|\{A(x) \mid \langle x, y \rangle \in E\}|}{|E|} = p + n,$$

the fraction of examples $\langle x, y \rangle \in E$ for which the antecedent A evaluates to true.

The notion of rule support is well-known from frequent itemset mining [9].

Definition 2. *The* bias *of a rule $r : A \rightarrow C$ is defined as the difference between the conditional distribution of C given A and the default probability of C:*

$$Bias(r) := \frac{p}{p+n} - \frac{P}{P+N}.$$

The bias reflects the degree to which a subgroup differs from expectation, i.e. that the target attribute is distributed as in the overall dataset. A broad variety of utility functions have been suggested for rule discovery, most of which are monotone in the bias and support of rules. Please refer to [10] for an overview. The most popular utility function for subgroup discovery is the weighted relative accuracy [5, 11], which is exemplarily used in our algorithms proposed in Sec. 4.

Definition 3. *The* weighted relative accuracy *of a rule r is defined as*

$$WRAcc(r) := Sup(r) \cdot Bias(r) = \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right).$$

For selecting rules only the order induced by an evaluation metric is relevant. Since P and N are constants for any fixed dataset we may multiply the term above with $\frac{(P+N)^2}{N}$ and reach at a more convenient formulation:

$$\begin{aligned} WRAcc \cdot \frac{(P+N)^2}{N} &= (p+n) \cdot \frac{(P+N)}{N} \cdot \left(\frac{p}{p+n} - \frac{P}{P+N}\right) \\ = \frac{p \cdot (P+N) - (p+n) \cdot P}{N} &= p - \frac{P}{N} \cdot n = p - c \cdot n \end{aligned} \tag{1}$$

for a database-dependent constant $c \in \mathbb{R}^+$. This simple reformulation reflects the fact that ROC isometrics of the weighted relative accuracy are parallel lines with a slope of 1. Learners optimizing this evaluation metric handle class skews differently than e.g. predictive accuracy does. Several search strategies have been proposed to find rules optimizing this metric for different settings.

The most straight-forward subgroup discovery task is to identifying a set of k rules with highest utility scores, where k is a user-given parameter. The ILP system MIDOS [4] is the best-known algorithm for this task, optimizing the weighted relative accuracy metric. It is designed for multi-relational learning and searches the space of rules exhaustively, except for safe pruning. The use of a refinement operator allows to evaluate rules from general to specific, while making sure that no rule is evaluated twice. The pruning strategy exploits the operator's top-down search. The support of each rule r decreases monotonically with each refinement, so for p positive and n negative examples the upper bound

$$WRAcc(r) \leq Sup(r) \cdot \left(1 - \frac{P}{P+N}\right) \tag{2}$$

allows to prune all refinements of a rule with low support if it cannot improve over the k-th best rule found so far.

In order to speed up the subgroup discovery process adaptive sampling has been proposed [12]. The learning task needs to be reformulated to account for the inevitable risk of drawing a poor sample. Hence, the goal is to find k approximately best rules with high probability. For the most relevant utility functions probabilistic guarantees can be given to find good rules with high probability.

Several authors have addressed subgroup discovery in the presence of (or relative to) prior knowledge. A recently presented system exploits different kinds of background knowledge to select relevant features, to discretize variables in a meaningful way, and to exploit user-given preferences for guiding search heuristics [7]. Knowledge-based sampling [8] generically incorporates probabilistic prior knowledge and can be combined with any of the other approaches. It yields unexpected patterns and supports the induction of accurate classifier ensembles.

3 Discovering subgroups from distributed data

3.1 Global distributed subgroup mining

An extension to classical subgroup discovery that has not yet been investigated by the data mining community is the discovery of subgroups from distributed data. We start with a few definitions for evaluating rules on distributed data to ease the formulation of the formal learning problems studied in this work.

In the remainder of this paper the global data E is assumed to be distributed to nodes $\{1, \ldots, m\}$, each holding a local subset $E_i \subset E$ so that $E = \biguplus_{i=1}^{m} E_i$. The number of positives and negatives at site i are denoted as P_i and N_i.

Definition 4. *For any rule $r : A \rightarrow C$ the absolute number of covered positives and covered negatives in node i are denoted as*

$$p_i(r) := |\{A(x) \wedge C(y) \mid \langle x, y \rangle \in E_i\}| \quad \textit{and} \quad n_i(r) := |\{A(x) \wedge \overline{C(y)} \mid \langle x, y \rangle \in E_i\}|.$$

This allows to restate the support and bias of rules for individual databases E_i.

Definition 5. *The* local support *of rule r at a site i is defined as*

$$Sup_i(r) := \frac{p_i(r) + n_i(r)}{|E_i|},$$

while the local bias *is defined as*

$$Bias_i(r) := \frac{p_i(r)}{p_i(r) + n_i(r)} - \frac{P_i(r)}{P_i(r) + N_i(r)}$$

Global utility functions can be adapted in a straight-forward manner based on these local quantities. We confine ourselves to weighted relative accuracy.

Definition 6. *Local weighted relative accuracy of rule r at node i is defined as*

$$WRAcc_i(r) := Sup_i(r) \cdot Bias_i(r).$$

The first studied distributed subgroup discovery task is referred to as *global subgroup discovery.* It aims at the identification of the same k best subgroups in the global data E, but without shifting all the data to a single database.

Global subgroup discovery is an unexpectedly hard problem. If the distribution underlying different databases E_i may deviate from the global distributions, i.e. they cannot be considered to be uniform subsamples of E, then globally best rules may perform poor at *all* local sites [13]. More precisely, collecting all the locally best rules with respect to $WRAcc_i$ does not necessarily yield a set that contains one of the k globally best rules, neither exactly nor approximately in the sense of the approximately k best rules problem (see Sec. 2). As a consequence, algorithms addressing global subgroup discovery need to exchange either examples or models and counts if guarantees are required. A new algorithm tailored towards the specific characteristics of the task will be presented in Sec. 4.1.

3.2 Relative local subgroup discovery

The novel task of relative subgroup mining takes the locality of data explicitly into account. A rule is considered to be interesting, if it is well supported by local data, and if its local confidence deviates substantially from the corresponding confidence when evaluating the same rule globally.

Relative subgroups are relevant in several domains. E.g. in a marketing application the corresponding rules may identify spatial regions in which the buying behavior of customers differs from that observed in other parts of the country. An unsupervised approach with a related aim, mining *high contrast frequent itemsets*, has recently been presented [14]. Based on entropy, it identifies itemsets with counts that are inhomogeneously distributed to the different sites. In this paper we address supervised relative rule discovery, a learning task proposed in recent prior work [13]. It aims at the identification of rules maximizing the following evaluation metric:

Definition 7. *The* relative local utility *of a rule r at node i is defined as*

$$RLU_i(r) := Sup_i(r) \cdot (Bias_i(r) - Bias(r) + c_i)\,,\ \textit{with}\ \ c_i := \frac{P_i}{P_i + N_i} - \frac{P}{P+N}.$$

Different class skews P_i/N_i are of minor interest in this setting, so the term c_i is used to focus on deviations of globally and locally different conditional class distributions for subsets covered by considered rules. This turns the term in brackets into deviations of local and global confidences, as motivated above.
As for $WRAcc$, a more convenient version of the RLU metric can be derived:

$$\begin{aligned} RLU_i(r) &= Sup_i(r) \cdot \left(\frac{p_i(r)}{p_i(r) + n_i(r)} - \frac{p(r)}{p(r) + n(r)} \right) \\ &= |E_i|^{-1} \cdot \left(p_i(r) - \underbrace{p(r) \cdot \frac{p_i(r) + n_i(r)}{p(r) + n(r)}}_{=:\hat{p}_i(r)} \right) = \frac{p_i(r) - \hat{p}_i(r)}{|E_i|} \end{aligned}$$

The term $\hat{p}_i(r)$ can be interpreted as the estimated number of positives within the subset covered by rule r at site i. This estimate is based on the fraction of positives in the subset of the *global* data that are covered by the rule, i.e. on the global confidence. A factor-equivalent metric to RLU is $RLU_i^*(r) := p_i(r) - \hat{p}_i(r)$.

The task of discovering the best k relative local subgroups has been shown to be at least as hard as discovering global subgroups from distributed data [13].

4 Algorithms for distributed subgroup discovery

4.1 Distributed global subgroup discovery

In this section we propose an algorithm for distributed global subgroup mining based on count polling and distributed rule pruning based on optimistic estimates. A basic principle of the algorithm is that for each rule r all refinements of this rule r' are created and counted at exactly one node. We use a refinement operator as defined in [4]. The following definition assumes a fixed total order on the set of attributes.

Definition 8. *A refinement operator ρ is a function that maps each rule to the set of its direct successors. A rule $r' : A' \rightarrow C'$ is a direct extension of $r : A \rightarrow C$, if and only if $C = C'$, $A' = A \cup \{X_i = v\}$ for a variable X_i with the property that all attributes X_j in A have an index j which is strictly lower than i. The transitive relation $r' < r$ denotes, that r' is a refinement of r.*

Our pruning method exploits the following relationship. If for each node the counts for a rule r or a predecessor of r, denoted as r' are known, we can calculate a tight upper bound on the $WRAcc(r)$. If this highest possible score is worse than the currently k-best rule, then the algorithm can safely prune the rule r.

Lemma 1. *The (global) utility of a rule r is bounded by the following term*

$$WRAcc(r) \leq \frac{\sum_{i=1}^{m} p_i(r'_i)}{P+N} \cdot \left(1 - \frac{P}{P+N}\right) = \frac{N}{(P+N)^2} \sum_{i=1}^{m} p_i(r'_i),$$

where $r'_i = r$ or $r < r'_i$. For the most specific rules $p_i(r_i)$ is known for, this bound is tight.

Proof. The correctness of the lemma follows from eqn. (1), implying that $WRAcc$ is order-equivalent to $p - \frac{P}{N} \cdot n$. Hence, optimal refinements discard all negatives but no positives, which leads to the score used as an upper bound.

The difference to eqn. (2) is that the support is replaced by the fraction of true positives, a quantity which is strictly smaller unless r cannot further be improved by refinements, anyway. The pruning strategy exploits the fact that $WRAcc$ increases monotonically if refinements "discard" only negatives. It is maximized by refinements that discard all negatives and no positives. For this reason straightforward adaptations of eqn. (2) apply to the broad class of utility functions sharing this property of monotonicity, e.g. to the binomial test function. It is sufficient to substitute the tightest known counts during optimistic score computation in lemma 1 for each rule, and to optimistically assume that a subsequent refinement is able to discard only the covered negatives.

The lemma can be used to prune rules for which exact counts are available only from a subset of all nodes. If the upper bound for $WRAcc(r)$ is worse than the k-th best rule, r can be pruned without polling further counts. Lemma 1 also implies a second pruning condition. If a rule r' is pruned, then all refinements

$r < r'$ of this rule can be pruned as well, as their optimistic scores are known to be no better than the optimistic score of r'.

These pruning strategies are combined with count polling to derive an algorithm for distributed subgroup mining that scales linearly in the number of nodes. Each node i keeps three data structures. First, a list B_i containing the k currently best hypotheses. Second, a list of pruned hypotheses Z_i. These are rules for which it is known that no descendant can reach a score better than

$$kb_i := \min_{r \in B_i} WRAcc(r),$$

the k-th best score at node i. To this end an optimistic upper-bound is computed using lemma 1. Finally, each node keeps a list of all rules, for which it is polling counts. This list is denoted as Q_i.

The algorithm is initialized by assigning all rules with an empty body to an arbitrary node. The computation then follows Fig. 1. A node that receives an assignment for a rule r generates all canonical refinements (direct successors) $\rho(r)$ and serves as their polling node. A rule $r : A \rightarrow C$ can be pruned (i) based on its optimistic score, or (ii) because it is *subsumed* by a previously pruned rule $r' : A' \rightarrow C'$, that is $C' = C$ and $A' \subset A$, so $\{A(x) \mid x \in \mathcal{X}\} \subset \{A'(x) \mid x \in \mathcal{X}\}$ and hence $p_i(r) < p_i(r')$ at all sites. For each refined rule r' the algorithm first obtains the local counts from the database and checks whether r' can be pruned. If the rule is pruned based on its optimistic score, the node additionally informs all other nodes about this step of pruning. In contrast, subsumption-based pruning of a rule r' does not require to broadcast r', since each node is known to have a rule subsuming r' in its list of pruned rules Z_i. If a rule is not pruned the node broadcasts a query for counts on r and adds r to the list of open hypotheses Q_i. The individual nodes then reply their local counts for r'. As more and more local counts arrive the bound on the global count gets tighter.

If all local counts for a rule r are available and r cannot be pruned, it is first checked, if the rule is better than kb_i. If this is the case, it is inserted into B_i as described above and broadcasted to all other nodes. Then the rule is assigned to a node that is responsible for generating and counting the canonical refinements of the rule. Besides the rule itself, the local counts for rule r are transmitted from all the nodes. This information is necessary to allow for pruning based on partially available counts, as described above. The node to which a rule is assigned is determined by the support of the rule. The rationale of this choice is, that such a node is the most likely to be able to prune the rule without querying other nodes for counts.

The algorithm has communication costs in $O(m|C|)$, where m is the number of nodes and C is the set of evaluated candidates. Hence, the algorithm scales linearly with the number of nodes and candidates. This can easily be seen from the fact that at most $O(m)$ messages are exchanged per candidate: a query for counts, its replies, and possibly a broadcast for a new best hypothesis *or* for pruning. These messages contain only rules and individual counts. Additionally, at most one delegation message for a rule is produced, containing a set of local counts. This message is of size $O(m)$.

```
// Update best rules
for $best_{ij}(r, WRAcc(r)) \in M_j$ do
  if $WRAcc(r) > kb_j$ then
    insert($B_j$,r);

// Update pruned rules
for $prune_{ij}(r) \in M_j$ do
  $Z_j = Z_j \cup \{r\}$;

// Obtain message counts
for $count_{ij}(r, n_i(r), p_i(r)) \in M_j$ do
  recalculate optscore($r$);
  if prunable(r) then
    $Z_j = Z_j \cup \{r\}$;
  else
    if counts-complete($r$) then
      if $WRAcc(r) > kb_j$ then
        best.insert($B_j$, $r$);
        bcast($best(r, WRAcc(r))$);
      $Q_j = Q_j \setminus \{r\}$;
      $m = argmax_i(n_i(r) + p_i(r))$;
      send($assign_{jm}(r, \{(p_1(r), ...)\})$);

  // Handle assignment to refine a rule
  for $assign_{ij}(r, \{(p_1(r), ...)\}) \in M_j$ do
    for $r' \in refinements(r)$ do
      recalculate optscore($r'$);
      if $not(prunable(r'))$ then
        bcast($query(r')$);
        $Q_j = Q_j \cup \{r'\}$;

  // Answer queries for local counts
  for $query_{ij}(r) \in M_j$ do
    send($count_{ji}(r, n_j(r), p_j(r))$);

prunable($r$):
  if $r \le r' : r' \in Z_j$ then
    return true;
  if $optscore(r) < kb_j$ then
    bcast($prune(r)$);
    return true;
  return false;
```

Fig. 1. Algorithm for distributed global subgroup mining at node j. M_j denotes the input message queue of node j. $best_{ij}$, $prune_{ij}$, $count_{ij}$, $query_{ij}$ and $assign_{ij}$ are messages, where i denotes the sender and j the receiver. The procedures above are executed as long as messages arrive.

4.2 Distributed relative subgroup discovery

Finding relative local subgroups differs from finding global subgroups in that each node finds an own, individual set of rules. The score of a rule is defined with respect to its local support and its relative bias. While the support of a rule r can easily be computed locally at each database, global counts for r are required for computing the bias. Global counts of rules are aggregated as described in the last section. There is one important difference however. Rules can only be pruned, if they are pruned at *every* node. We propose an algorithm that is based on count polling and optimistic pruning. The following tight optimistic pruning rule holds for the task of relative local subgroup mining.

Lemma 2. *For relative local subgroup discovery, rules r with $p_i(r)$ positives, $n_i(r)$ negatives, and $\hat{p}_i(r)$ estimated positives covered by rule r at site i,*

$$RLU_i(r') \le \frac{p_i(r) - \max(0, \hat{p}_i(r) - n_i(r))}{|E_i|}$$

is a tight upper-bound for the local utilities of all rules $r' < r$.

Proof. Considering the factor-equivalent metric RLU^* it is easily seen that an optimal refinement of rule r reduces $\hat{p}_i(r)$ by covering less examples that are "predicted" positive, while not reducing $p_i(r)$. If the $n_i(r)$ negative examples covered by r are predicted positive by $\hat{p}_i(r)$, and if a refinement $r' < r$ exists that covers only the $p_i(r)$ positive examples, then we reach at a utility of

$$RLU_i^*(r') = p_i(r) - \max(0, \hat{p}_i(r) - n_i(r)).$$

This cannot be improved any further by refinements, since r' covers only positives, and further refinement reduces $p_i(r)$ at least as much as $\hat{p}_i(r) - n_i(r)$. Since $RLU^* = RLU \cdot |E_i|$ this proves the lemma.

Our algorithm for relative subgroup mining works as follows. Again, each node has a list of best rules, pruned rules, and open rules. Additionally, nodes keep a rule cache, that is used to store the global counts of rules for which a node serves as the polling node. The mapping of rules to responsible nodes is realized by a hash function.

Each node starts with an empty set of rule candidates. It then generates first-level rules that are evaluated locally. If a rule r can be pruned based on lemma 2 it is discarded. Otherwise, the node requests global counts $n(r)$ and $p(r)$ for r from a polling node that is determined by calculating a hash value for the rule. The node that receives this request checks whether it finds the rule in its cache. If so, it directly returns the corresponding global counts. Otherwise, the node first queries all other nodes for their corresponding local counts. After aggregating all local counts $n_i(r)$ and $p_i(r)$ the polling node stores and returns the global counts. Given the global counts and the local counts for a rule r, the exact utility score of r can be computed. If r is better than the k-best rule it is inserted into B_i as described in the last section. If r, and thus each of its refinements, receive an optimistic score that is worse than the lowest score in B_i, then r is pruned. Neither best rules nor pruned rules are broadcasted, as they are not relevant to other nodes.

While the pruning strategies for relative local subgroup mining are weaker than for distributed global subgroup mining, the approach still scales linearly with the number of nodes. Thus, relative local subgroup mining is in $O(|C|m)$, where $|C|$ are the candidates considered by at least one node. Relative local subgroup mining for all nodes is usually more expensive than global subgroup mining, because rules may only be pruned, if they would be pruned at all nodes.

5 Experiments

We performed experiments to analyze the properties of the proposed algorithms. As both algorithms are guaranteed to find the best rules, evaluation is only concerned with communication costs. These costs are evaluated on three datasets taken from the UCI library, mushroom, adult, and german. For adult and german numerical attributes were discretized using minimal entropy partitioning.

First of all the substantial difference between the tasks of subgroup and association rule mining is illustrated exemplarily. Association rule and frequent itemset mining rely on a user-provided support threshold and are usually applied

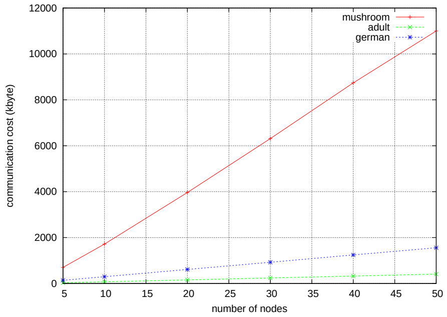

Fig. 2. Communication costs for distributed global subgroup mining

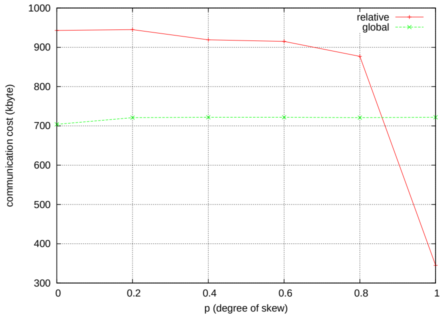

Fig. 3. Data skew / communication costs for global and relative subgroup mining

to find huge amounts of rules. Subgroup discovery finds only the k best rules with respect to a user-specified utility function, not requiring a threshold. Even if the best rule utility was known to a frequent itemset mining algorithm in advance, it would be more costly to generate all itemsets based on a corresponding support threshold in a distributed setting than to run distributed subgroup discovery; state-of-the-art algorithms for distributed frequent itemset mining evaluate at least all frequent itemsets at all nodes. E.g. the german dataset contains more than 50.000 frequent itemsets using the support-based pruning threshold of the MIDOS algorithm (see eqn. (2)) in combination with the (usually unknown) utility of the best subgroup. In contrast, the global subgroup discovery algorithm evaluates less than 3.000 *candidates*.

Still, the communication costs for our algorithm grow no more than linearly in the number of nodes. We validated this property in a first experiment, measuring costs by accounting 4 bytes for each rule transmitted over the network and 2 bytes for each count. To be able to measure the impact of data skews in the distribution of data to individual nodes we used the following procedure. First, the data was clustered using an EM algorithm. The number of clusters was chosen as the number of nodes. We use a parameter p_{skew} denoting the probability that an example is assigned to a node according to the corresponding cluster. Otherwise it is assigned randomly at equal probability. For $p_{skew} = 1$ each node receives all data points in its corresponding cluster. For $p_{skew} = 0$ all examples are distributed randomly. This allows to adjust the data skew between both extremes. The results for the datasets using $p_{skew} = 0$ and finding one global rule ($k = 1$) for rules of constrained length as in MIDOS (we searched for best rules containing up to 3 literals) are shown in Fig. 2. For all three datasets the curves confirm our theoretical findings concerning the scalability of our method. Please note, that in this experiment each database contains about the same amount of data, which is the worst case for our method.

The second experiment compares the communication costs for distributed global and relative subgroup mining for varying degrees of skew. The results of mining the most interesting rule of length up to 3 literals for the mushroom data set is shown in Fig. 3 for a network of $m = 5$ nodes. We see that distributed global subgroup mining shows a low sensitivity regarding the data skew. For relative subgroup mining the situation is different. Given a low skew, the costs for finding relative subgroups increases. The reason is that relative subgroups can only be found if the data distribution among nodes deviates. For low skews only rules with very low scores can be identified, which however forces all nodes to search a very large search space as pruning cannot be applied. Reaching at a certain level of skew the distributions deviate sufficiently to identify corresponding logical rules, leading to a sharp decrease of costs in Fig. 3 for relative subgroup mining.

6 Discussion and conclusion

Discovering distributed global and relative local subgroups are two novel knowledge discovery tasks. Since subgroup discovery is a supervised learning task it could be approached with state of the art distributed classification algorithms, e.g. distributed boosting [15] in order to find probabilistic rule ensembles as in [8]. Distributed boosting and similar algorithms are however not complete, thus do not guarantee to find optimal rules. As noted in [15] the quality of rules that can be discovered depends on the distribution of examples over the individual databases. Results presented in [13] support this observation. For this reason we focused on complete algorithms for distributed rule mining.

Existing complete algorithms for distributed rule mining are mostly concerned with finding association rules [1]. A straightforward extension of the Apriori algorithm is Count Distribution (CD) [16]. At each round, every database generates all $k + 1$ candidates from the globally large k-itemsets and broadcasts all counts to all other nodes. This procedure causes communication costs of $\Omega(|C|m^2)$, where $|C|$ is the number of candidates and m is the number of nodes. One way to improve the CD algorithm is to use a designated node for each candidate that is responsible for polling and redistributing all counts of the candidate itemset. This method is applied in the FDM algorithm [17]. It reduces the communication complexity of the algorithm to $\Theta(|C|m)$. Two additional pruning techniques are applied in FDM. Local pruning is based on the observation that for an item to be frequent it must be frequent at least at one node. Only for such items counts need to be exchanged. Second, nodes use an optimistic estimate for the support of an itemset based on partial counts received from other nodes. If this estimate is smaller than the minimal support, the candidate can be pruned. The idea of a polling site, as introduced by FDM helps to avoid costly broadcasts and is very general.

The real power of the above approaches lies in their local pruning strategies, however, which do not apply to distributed global subgroup mining as shown in [13]; globally optimal rules can simultaneously be inferior at each individual node, while pruning strategies applied to distributed frequent itemset mining rely on the fact that globally frequent itemsets *must* be frequent at least at one node. This reflects that subgroup utility functions are lacking the monotonicity of rule support, a prerequisite for efficient itemset mining. This substantial difference remains even for more sophisticated pruning strategies as proposed in [14, 18].

Association rule based approaches are not applicable to relative subgroup mining either, because the relative score of each rule does not only depend on its local support, but also on the (independent) local and global rule confidences.

Hence, we presented two new algorithms for distributed subgroup discovery that guarantee to deliver optimal rules at communication costs linear in the number of nodes and rule candidates, an essential property for scalable distributed algorithms. The complexity was shown theoretically and confirmed empirically.

References

1. Zaki, M.J.: Parallel and Distributed Association Mining: A Survey. IEEE Concurrency **7** (1999)
2. Park, B.H., Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications. In Ye, N., ed.: Data Mining Handbook. IEA (2002)
3. Klösgen, W.: Subgroup discovery. In: Handbook of Data Mining and Knowledge Discovery. Oxford University Press (2002)
4. Wrobel, S.: An Algorithm for Multi–relational Discovery of Subgroups. In: Principles of Data Mining and Knowledge Discovery: First European Symposium (1997)
5. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In Advances in Knowledge Discovery and Data Mining. AAAI Press (1996)
6. Lavrac, N., Cestnik, B., Gamberger, D., Flach, P.: Decision support through subgroup discovery: three case studies and the lessons learned. MLJ **57** (2004)
7. Atzmüller, M., Puppe, F., Buscher, H.P.: Exploiting background knowledge for knowledge-intensive subgroup discovery. In: Proc. of IJCAI (2005)
8. Scholz, M.: Sampling-Based Sequential Subgroup Mining. In: Proc. of KDD (2005)
9. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large data bases. In: Proc. of VLDB (1994)
10. Fürnkranz, J., Flach, P.: ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms. MLJ **58** (2005)
11. Nada Lavrac, N., Flach, P., Zupan, B.: Rule Evaluation Measures: A Unifying View. In: 9th International Workshop on Inductive Logic Programming (1999)
12. Scheffer, T., Wrobel, S.: Finding the Most Interesting Patterns in a Database Quickly by Using Sequential Sampling. JMLR **3** (2002)
13. Scholz, M.: On the Tractability of Rule Discovery from Distributed Data. In: Proc. of ICDM (2005)
14. Otey, M.E., Parthasarathy, S., Wang, C., Veloso, A., Meira, W.: Parallel and Distributed Methods for Incremental Frequent Itemset Mining. IEEE Transactions on Systems, Man, and Cybernetics, Part B **34** (2004) 2439–2450
15. Lazarevic, A., Obradovic, Z.: Boosting algorithms for parallel and distributed learning. Distributed and Parallel Databases Journal **11** (2002)
16. Agrawal, R., Shafer, J.C.: Parallel mining of association rules. IEEE Transactions On Knowledge And Data Engineering **8** (1996)
17. Cheung, D., Han, J., Ng, V., Fu, A., Fu, Y.: A Fast Distributed Algorithm for Mining Association Rules. In: International Conference on Parallel and Distributed Information Systems. (1996)
18. Schuster, A., Wolff, R.: Communication-efficient distributed mining of association rules. In: Proc. of SIGMOD (2001)