Apriori

Find Frequent Item Sets and Association Rules with the Apriori Algorithm

Contents

Introduction

Frequent item set mining and association rule induction [Agrawal *et al.* 1993, 1994] are powerful methods for so-called *market basket analysis*, which aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, online shops etc. With the induction of frequent item sets and association rules one tries to find sets of products that are frequently bought together, so that from the presence of certain products in a shopping cart one can infer (with a high probability) that certain other products are present. Such information, especially if expressed in the form of rules, can often be used to increase the number of items sold, for instance, by appropriately arranging the products on the shelves of a supermarket or on the pages of a mail-order catalog (they may, for example, be placed adjacent to each other in order to invite even more customers to buy them together) or by directly suggesting items to a customer, which may be of interest for him/her.

An *association rule* is a rule like "If a customer buys wine and bread, he/she often buys cheese, too." It expresses an association between (sets of) *items*, which may be products of a supermarket or a mail-order company, special equipment options of a car, optional services offered by telecommunication companies etc. An association rule states that if we pick a customer at random and find out that he/she selected certain items (bought certain products, chose certain options etc.), we can be confident, quantified by a percentage, that he/she also selected certain other items (bought certain other products, chose certain other options etc.).

Of course, we do not want just any association rules, we want "good" rules, rules that are "expressive" and "reliable". The standard measures to assess association rules are the *support* and the *confidence* of a rule, both of which are computed from the *support* of certain item sets. These notions are discussed here in more detail. However, these standard criteria are often not sufficient to restrict the set of rules to the interesting ones. Therefore some additional rule evaluation measures are considered here.

The main problem of association rule induction is that there are so many possible rules. For example, for the product range of a supermarket, which may consist of several thousand different products, there are billions of possible association rules. It is obvious that such a vast amount of rules cannot be processed by inspecting each one in turn. Therefore efficient algorithms are needed that restrict the search space and check only a subset of all rules, but, if possible, without missing important rules. One such algorithm is the apriori algorithm, which was developed by [Agrawal *et al.* 1994] and which is implemented in a specific way in my apriori program. A brief description of some implementation aspects can be found in these papers:

- **Induction of Association Rules: Apriori Implementation**
  Christian Borgelt and Rudolf Kruse
  15th Conference on Computational Statistics (Compstat 2002, Berlin, Germany)
  Physica Verlag, Heidelberg, Germany 2002
  (6 pages) cstat_02.pdf (105 kb) cstat_02.ps.gz (91 kb)
- **Efficient Implementations of Apriori and Eclat**
  Christian Borgelt.
  Workshop of Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL, USA).
  (9 pages) fimi_03.pdf (304 kb) fimi_03.ps.gz (197 kb)

By the way: Earlier versions of my apriori program are incorporated in the well-known data mining tool Clementine (apriori version 1.8 in Clementine version 5.0, apriori version 2.7 in Clementine version 7.0), available from SPSS. Newer versions of Clementine still use my program, but I am not completely sure about the version number of the underlying apriori program.

Enjoy,
Christian Borgelt

back to the top

Basic Notions

This section introduces the basic notions needed to talk about frequent item sets and association rules. These notions are *item*, *transaction*, *support* and *confidence*.

Items and Transactions

On an abstract level, the input to frequent item set mining and association rule induction consists of a bag or multiset of *transactions* that are defined over a set of *items*, sometimes called the *item base*. These items may be products, special equipment items, service options etc. For an abstract treatment it suffices that items have an identity, that is, that it is possible to distinguish one item from another. The item base is the set of all considered items, for example, the set of all products that are sold by a given supermarket, mail-order company or online shop. Any subset of the item base is called an *item set*.

A transaction is simply an item set and it represents, for example, the set of products bought by a customer. Since two or more customers may, in principle, buy the exact same set of products, we cannot model the whole of all "shopping baskets" or "shopping carts" (bought, say, in a given week) as a *set* of transactions, since in a set each element is unique. There are several solutions to this problem: one may, for example, model the whole of all transactions as a bag or multiset (a generalization of a set, which allows for multiple occurrences of the same element) or as a vector (where elements at different positions may be the same, but are still distinguished by their position) or by extending each transaction with a unique *transaction identifier* (or *tid* for short). (Note that the position of a transaction in a vector representation is an implicit transaction identifier.) Still another possibility consists in using a standard *set* of (unique) transactions and assigning to each of them an occurrence counter. Here I will use the bag/multiset terminology, even though an occasional "set of transactions" may have slipped through. (This should always be read as "bag/multiset of transactions".)

Note that the item base (the set of all considered items) is often not given explicitly, but only implicitly as the union of all given transactions. This is also the case for my apriori program, which by default only takes a transaction file as input. However, it is also possible to specify the item base explicitly with an optional item appearances file (discussed in this section). This can be useful, for example, if one wants to restrict the analysis to a subset of all items.

back to the top

Support of an Item Set

Let S be an item set and T the bag/multiset of all transactions under consideration. Then the *absolute support* (or simply the *support*) of the item set S is the number of transactions in T that contain S. Likewise, the *relative support* of S is the fraction (or percentage) of the transactions in T which contain S.

More formally, let S be an item set and U the bag/multiset of all transactions in T that contain all items in S. Then

$\text{supp}_{\text{abs}}(S) = |U|$

is the absolute support of S and

$\text{supp}_{\text{rel}}(S) = (|U| / |T|) *100\%$

is the relative support of S. |U| and |T| are the number of elements in U and T, respectively.

In a supermarket setting, the item set S may be a set like S = { bread, wine, cheese } and T may be the bag/multiset of all "baskets" or "carts" of products bought by the customers of a supermarket - in a given week if you like. U is the bag/multiset of all transactions in T that contain all items in S (and may also some other items). For example, if a customer buys the set X = { milk, bread, apples, wine, sausages, cheese, onions, potatoes }, then S is obviously a subset of X, hence X is in U. If there are 318 customers, each giving rise to one transaction, and 242 of these customers bought such a set X or a similar one that contains S, then $\text{supp}_{\text{abs}}(S) = 242$ and $\text{supp}_{\text{rel}}(S) = 76.1\%$.

The goal of frequent item set mining is to find all item sets (that is, all subsets of the item base) that occur in the given bag/multiset of transactions with at least a user-specified minimum support. My apriori program always uses a minimum support threshold to select item sets. The default value for the minimum support is 10%. This value can be changed with the option `-s`. In addition, a maximum support can be specified with the option `-S` (this may be useful for certain applications). The default value for the maximum support is 100% (that is, there is no restriction). Note that the arguments to these options, if positive, is interpreted as a percentage. If, however, the given argument is negative, it is interpreted as an absolute number (number of transactions) rather than a percentage.

Note that the default operation mode of my apriori program is to find such frequent item sets. In order to find association rules, the target type has to be changed (see here).

back to the top

Confidence of an Association Rule

When we search for association rules, we do not want just any association rules, but "good" association rules. To measure the quality of association rules, [Agrawal *et al.* 1994], the inventors of the apriori algorithm, introduced the *confidence* of a rule. The confidence of an association rule R = "A and B -> C" (with items A, B and C) is the support of the set of all items that appear in the rule (here: the support of S = { A, B, C }) divided by the support of the antecedent (also called "if-part" or "body") of the rule (here X = { A, B)). That is,

conf(R) = (supp({A, B, C}) / supp({A, B})) *100%.

(Note that it does not matter whether the confidence is computed from the absolute or the relative support of an item set, as long as the same support type is used in both the numerator and the denominator of the fraction.)

More intuitively, the confidence of a rule is the number of cases in which the rule is correct relative to the number of cases in which it is applicable. For example, let R = "wine and bread -> cheese". If a customer buys wine and bread (and maybe some other items), then the rule is applicable and it says that he/she can be expected to buy cheese. If he/she does not buy wine or does not buy bread or buys neither, then the rule is not applicable and thus (obviously) does not say anything about this customer.

If the rule is applicable, it says that the customer can be expected to buy cheese. But he/she may or may not buy cheese, that is, the rule may or may not be correct (for this customer). Naturally, we are interested in how good the prediction of the rule is, that is, how often its prediction that the customer buys cheese is correct. The rule confidence measures this: it states the percentage of cases in which the rule is correct. It states this percentage relative to the number of cases in which the antecedent holds, since these are the cases in which the rule makes a prediction that can be true or false. If the antecedent does not hold, then the rule does not make any prediction, so these cases are excluded.

Rules are reported as association rules if their confidence reaches or exceeds a given lower limit (minimum confidence, to be specified by a user). That is, we look for rules that have a high probability of being true: we look for "good" rules, which make correct (or very often correct) predictions. My apriori program always uses a minimum confidence to select association rules. The default value for the minimum confidence is 80%. This value can be changed with the option `-c`.

In addition to the rule confidence my apriori program lets you select from several other (additional) rule evaluation measures, which are explained below, but it will also use rule confidence. If you want to rely entirely on some other measure, you can do so by setting the minimal rule confidence to zero. (Attention: If you have a large number of items, setting the minimal rule confidence to zero can result in *very* high memory consumption.)

back to the top

Support of an Association Rule

The support of association rules may cause some confusion, because I use this term in a different way than [Agrawal *et al.* 1993, 1994] do. For them, the support of an association rule "A and B -> C" is the support of the set S = { A, B, C }. This may be fine if rule confidence is the only evaluation measure, but it causes problems if some other measure is used. For these other measures it is often much more appropriate to call the support of the antecedent of the association rule, that is, the support of X = { A, B } in the example above, the support of the association rule.

The difference can also be stated in the following way: for [Agrawal *et al.* 1993, 1994], the support of the rule is the (absolute or relative) number of cases in which the rule is correct (that is, in which the presence of the item C follows from the presence of the items A and B), whereas for me (and thus my apriori program) the support of a rule is the (absolute or relative) number of cases in which it is applicable (that is, in which the antecedent of the rule holds), although in some of these cases it may be false (because only the items A and B are present, but the item C is missing).

One reason for this choice, as already mentioned, is that the definition of [Agrawal *et al.* 1993, 1994] does not work well for evaluation measures other than rule confidence. This is explained in more detail below. Another reason is that I prefer the support of a rule to say something about the "statistical" support of a rule and its confidence, that is, from how many cases the confidence is computed in order to express how well founded the statement about the confidence is.

Maybe an example will make this clearer. Suppose you have a die which you suspect to be biased. To test this hypothesis, you throw the die, say, a thousand times. 307 times the 6 turns up. Hence you assume that the die is actually biased, since the relative frequency is about 30% although for an unbiased die it should be around 17%. Now, what is the "statistical" support of this statement, that is, on how many experiments does it rest? Obviously it rests on all 1000 experiments and not only on the 307 experiments in which the 6 turned up. This is so, simply because you had to do 1000 experiments to find out that the relative frequency is around 30%, and not only the 307 in which a 6 turned up (doing only these experiments is obviously impossible).

Or suppose you are doing an opinion poll to find out about the acceptance of a certain political party, maybe with the usual question "If an election were held next Sunday ...?" You ask 2000 persons, of which 857 say that they would vote for the party you are interested in. What is the support of the assertion that this party would get around 43% of all votes? It is the size of your sample, that is, all 2000 persons, and not only the 857 that answered in the positive. Again you had to ask all 2000 people to find out about the percentage of 43%. Of course, you could have asked fewer people, say, 100, of which, say, 43 said that they would vote for the party, but then your statement would be less reliable, because it is less "supported". The number of votes for the party could also be 40% or 50%, because of some random influences. Such deviations are much less likely, if you asked 2000 persons, since then the random influences can be expected to cancel out.

The rule support can be used to filter association rules by stating a lower bound for the support of a rule (minimum support). This is equivalent to saying that you are interested only in such rules that have a large enough statistical basis (since my apriori program uses the term "support" in my interpretation and not in the one used by [Agrawal *et al.* 1993, 1994]). The default value for this support limit is 10%. It can be changed with the option `-s`. Note that the argument, if positive, is interpreted as a percentage. If, however, the given argument is negative, it is interpreted as an absolute number (number of transactions) rather than a percentage.

The minimum support is combined with the minimum confidence to filter association rules. That is, my apriori program generates only association rules, the confidence of which is greater than or equal to the minimum confidence *and* the support of which is greater than or equal to the minimum support.

Despite the above arguments in favor of my definition of the support of an association rule, a rule support compatibility mode is available (due to the overwhelming pervasiveness of the original definition). With the option `-o` the original rule support definition can be selected. In this case the support of an association rule is the support of the set with all items in the antecedent (also called "if-part" or "body") *and* the consequent (also called "then-part" or "head") of the association rule, that is, the support of an association rule as defined in [Agrawal *et al.* 1993, 1994].

back to the top

Target Types

The target type, which can be selected via the option `-t`, is either frequent item sets (default, option `-ts`), closed item sets (option `-tc`), maximal item sets (option `-tm`) or association rules (option `-tr`).

Note that association hyperedges, which were a separate target type in earlier versions of my apriori program, are now covered by the target type of frequent item sets. In order to find association hyperedges, choose rule confidence as the additional evaluation measure (option `-ec`) and averaging as the aggregation mode (option `-aa`, see this section for more explanations). (I am grateful to Bastien Duclaux for requesting the possibility to generate association hyperedges.)

Frequent Item Sets (default, option -ts)

Often one only wants to find frequent item sets. That is, one wants to find all item sets with a support exceeding a certain threshold, the so-called *minimum support*. For my apriori program this is the default operation mode (since version 4.36, earlier versions had association rules as the default mode). However, this mode can also be selected explicitly with the option `-ts`.

Closed Item Sets (option -tc)

A frequent item set is called *closed* if no superset has the same support. If the option `-tc` is given, the found frequent item sets are subsequently filtered and only the closed item sets are reported.

Maximal Item Sets (option -tm)

A frequent item set is called *maximal* if no superset is frequent, that is, has a support exceeding the minimum support. If the option `-tm` is given, the found frequent item sets are subsequently filtered and only the maximal item sets are reported.

Association Rules (option -tr)

My apriori program generates association rules if the the option `-tr` is given. Note, however, that it produces only association rules with a single item in the consequent (also called "then-part" or "head" of the rule). This restriction is due to the following considerations:

In the first place, association rule mining usually produces too many rules even if one confines oneself to rules with only one item in the consequent. So why should one make the situation worse by allowing more than one item in the consequent? (It merely blows up the size of the output.)

Secondly, I do not know any application in which rules with more than one item in the consequent are of any real use. The reason, in my opinion, is that such more complex rules add almost nothing to the insights about the data set. To understand this, consider the simpler rules that correspond to a rule with multiple items in the consequent, that is, rules having the same antecedent, but consequents with only single items from the consequent of the complex rule. All of these rules must necessarily be in the output, because neither their support (in my interpretation, see this section) nor their confidence can be less than that of the more complex rule. That is, if you have a rule C D <- A B, you will necessarily also have the rules C <- A B and D <- A B in the output. Of course, these latter two rules together do *not* say the same as the more complex rule. However, what do you gain from the additional information the more complex rule gives you? How can you use it? And is this little extra information worth having to cope with a much bigger rule set?

back to the top

Extended Rule Selection

If association rules are selected using a minimum confidence, the following problem arises: "Good" rules (rules that are often true) are not always "interesting" rules (rules that reveal something about the interdependence of the items). You certainly know the examples that are usually given to illustrate this fact. For instance, it is easy to discover in a medical database that the rule "pregnant -> female" is true with a confidence of 100%. Hence it is a perfect rule, it never fails, but, of course, this is not very surprising. Although the measures explained below cannot deal with this problem (which is semantical), they may be able to improve the results in a related case.

Let us look at the supermarket example again and let us assume that 60% of all customers buy some kind of bread. Consider the rule "cheese -> bread", which holds with a confidence of, say, 62%. Is this an important rule? Obviously not, since the fact that the customer buys cheese does not have a significant influence on him/her buying bread: The percentages are almost the same. But if you had set a confidence limit of 60%, you would get both rules "-> bread" (confidence 60%) and "cheese -> bread" (confidence 62%), although the first would suffice (the first, because it is the simpler of the two). The idea of all measures, which can be used in addition or instead of rule confidence, is to handle such situations and to suppress the second rule.

In addition, consider the following case: Assume that the confidence of the rule "cheese -> bread" is not 62% but 35%. With a confidence limit of 60% it would not be selected, but it may be very important to know about this rule! Together with cheese, bread is bought much less frequently than it is bought at all. Is cheese some kind of substitute for bread, so that one does not need any bread if one has cheese? Ok, maybe this is not a very good example. However, what can be seen is that a rule with low confidence can be very interesting, since it may capture an important influence. Furthermore, this is a way to express negation (although only in the consequent of a rule), since "cheese -> bread" with confidence 35% is obviously equivalent to "cheese -> no bread" with confidence 65%. This also makes clear why the support of the item set that contains all items in the antecedent ("if-part", "head") *and* the consequent ("then-part", "body") of the rule is not appropriate for this measure. An important rule may have confidence 0 and thus a support (in the interpretation of [Agrawal *et al.* 1993, 1994]) of 0. Hence it is not reasonable to set a lower bound for this kind of support.

I hope that the intention underlying all of these explanations is clear by now: Potentially interesting rules differ significantly in their confidence from the confidence of rules with the same consequent, but a simpler antecedent. Adding an item to the antecedent is informative only if it significantly changes the confidence of the rule. Otherwise the simpler rule suffices.

Unfortunately the measures other than rule confidence do not solve the rule selection problem in the very general form in which it was stated above. It is not that easy to deal with all rules that have a simpler antecedent, to keep track of which of these rules were selected (this obviously influences the selection of more complicated rules), to deal with the special type of Poincare paradox that can occur etc. Hence the measures always compare the confidence of a rule with the confidence of the rule with an empty antecedent, that is, with the relative frequency of the consequent.

I call the confidence of an association rule with an empty antecedent the *prior confidence* (of any rule with the same consequent), since it is the confidence that the item in the consequent of the rule will be present in a transaction prior to any information about other items that may be present. (Note that the prior confidence of a rule is obviously equal to the (relative) support of its consequent item.) The confidence of an association rule with non-empty antecedent (and the same consequent) I call the *posterior confidence* (or simple the *confidence*) of the rule, since it is the confidence that the item in the consequent of the rule will be present after it becomes known that the items in the antecedent of the rule are present.

All measures, which can be computed with my apriori program and can be used for filtering in addition to the rule confidence, are basically computed from these two values: the prior confidence and the posterior confidence. Only those association rules are reported for which the value of the chosen additional evaluation measure meets or exceeds a certain limit. The measures are chosen with the option `-e`, the limit is passed to the program via the option `-d`. The default value for the limit is 10%. Note that the option `-d` always interprets its argument as a percentage. As a consequence, the desired limit value may sometimes have to be multiplied by 100 in order to obtained the needed argument value.

Note that all additional rule evaluation measures are combined with the limits for rule confidence and rule support. That is, my apriori program reports only those rules, the confidence of which is greater than or equal to the minimum confidence, the support of which is greater than or equal to the minimum support, *and* for which the additional evaluation value (if selected) is greater than or equal to the limit for this measure. The default is to use no additional evaluation measure (but this may also be specified explicitly with the option `-ex`), that is, to rely only on rule confidence and rule support. Of course you can remove the restriction that the rule support and the rule confidence must exceed certain limits by simply setting either (or both) of these limits to zero. In this case rules are selected using only the limits for the additional evaluation measure. (Attention: If you have a large number of items, setting the minimum rule support or the minimum rule confidence to zero can result in *very* high memory consumption.)

back to the top

Absolute Confidence Difference to Prior (option `-ed`)

The simplest way to compare the posterior and the prior confidence of an association rule (since the former should differ considerably from the latter to make the rule interesting) is to compute the absolute value of their difference. That is, if "-> bread" has a confidence of 60% and "cheese -> bread" has a confidence of 62%, then the value of this measure is 2% (for the second rule). The parameter given with the option `-d` to the program states a lower bound for this difference (in percentage points). It follows that this measure selects rules, the (posterior) confidence of which differs more than a given threshold from the corresponding prior confidence.

For example, with the option `-d20` (and, of course, the option `-ed` to select this measure) only rules with a confidence less than 40% or greater than 80% would be selected for the item "bread" in the consequent. As a consequence, the selected rules are those, for which the antecedent considerably changes the confidence. (Note that, of course, for other items, with a different prior confidence, the upper and lower bounds are different. For example, they are 10% and 50% for a rule with a prior confidence of 30%.)

back to the top

Lift Value (option `-el`)

The so-called lift value is simply the quotient of the posterior and the prior confidence of an association rule. That is, if "-> bread" has a confidence of 60% and "cheese -> bread" has a confidence of 72%, then the lift value (of the second rule) is 72/60 = 1.2. Obviously, if the posterior confidence equals the prior confidence, the value of this measure is 1. If the posterior confidence is greater than the prior confidence, the lift value exceeds 1 (the presence of the antecedent items raises the confidence), and if the posterior confidence is less than the prior confidence, the lift value is less than 1 (the presence of the antecedent items lowers the confidence).

The value that can be passed to the program with the option `-d` is a lower limit for this measure: only rules for which this measure meets or exceeds the given value are reported. As a consequence, the selected rules are those that raise the confidence by at least a given minimum factor. Note, however, that the option `-d` always interprets its arguments as a percentage. Therefore, in order to filter rules with a lift value of at least 1.2, the option `-d120` must be provided (that is, the argument must be the limit for the lift value times 100%).

Absolute Difference of Lift Value to 1 (option `-ea`)

With the lift value (see preceding section) it is not possible to filter association rules for which the lift value *differs* more than a given amount from 1, because the limit passed to the program with the option `-d` is a *lower* limit. Hence only rules with a lift value that *exceeds* 1 by a given minimum amount (and thus, for which the presence of the antecedent items raises the confidence by at least a given factor) can be properly filtered.

In order to be able to filter rules for which the lift value *differs* by more than a given amount from 1, the absolute difference of the lift value to 1 can be used (option `-ea`). For example, if the prior confidence of an association rule (the support of its consequent) is 60% and the option `-d20` is given, rules with a confidence less than or equal to (1 -20%) *60% = 0.8 *60% = 48% or a confidence greater than or equal to (1 +20%) *60% = 1.2 *60% = 72% are selected. Similarly, if the prior confidence is 30%, the numbers are 0.8 *30% = 24% and 1.2 *30% = 36%.

As these examples show, the main difference between this measure and the absolute difference of the posterior and prior confidences is that the deviation that is considered to be significant depends on the prior confidence (12% deviation for 60% prior confidence, but only 6% deviation for 30% prior confidence). The idea is that for a high prior confidence the deviation of the posterior confidence must also be high, and if it is low, the deviation needs to be only low.

(I am grateful to Roland Jonscher, S-Rating GmbH, Berlin, Germany, who pointed out this measure to me.)

back to the top

Difference of Lift Quotient to 1 (option `-er`)

Instead of simply forming the absolute difference of the lift value of an association rule, one may also compute the difference of either the lift value or its reciprocal, whichever is smaller, to one. Since either the lift value or its reciprocal must be less than or at most equal to one (and necessarily non-negative), this yields a value between 0 and 1. This measure can be selected with the option `-er`, a lower limit can, as usual, be specified with the option `-d`.

For example, if the prior confidence of an association rule (the support of its consequent) is 60% and the option `-d20` is given, association rules with a confidence less than or equal to (1 -20%) *60% = 0.8 *60% = 48% or a confidence greater than or equal to 60% /(1 -20%) = 60% /0.8 = 75% are selected. On the other hand, if the prior confidence is only 30%, rules with a (posterior) confidence of no more than 0.8 *30% = 24% or at least 30% /0.8 = 37.5% are selected (with `-d20`).

Note that the main difference to the absolute difference of the lift value to one (see the preceding section) is that positive and negative deviations are treated differently. With the absolute difference of the lift value to one, positive and negative deviations (posterior confidence exceeding and falling short of the prior confidence, respectively) are treated the same. The deviation must be by the same amount (in percentage points) in order to select the rule. With this measure (difference of lift quotient to 1), however, a negative deviation by a certain number of percentage points can lead to the selection of the rule, while the same positive deviation (in percentage points) does not lead to a selection of the rule. For the example above, with a prior confidence of 60%, the positive deviation limit is 15%, the negative deviation limit only 12%. Likewise, for a prior confidence of 30%, the positive deviation limit is 7.5%, while the negative deviation limit is only 6%.

back to the top

Normalized chi^2 Measure (option `-en`)

The chi^2-measure is well known from statistics. It is often used to measure the difference between a conjectured independent distribution of two discrete variables and the actual joint distribution in order to determine how strongly two variables depend on each other (or whether they are independent). The two (binary) variables considered here are the consequent and the antecedent of the rule, namely whether (all) the items contained in them are present or not (variable X: all antecedent items are present (X=1) versus at least one absent (X=0), and variable Y: consequent item present (Y=1) versus consequent item absent (Y=0)).

The chi^2-measure (as it is usually defined in statistics) contains the number of cases it is computed from as a factor. Even though this is, of course, highly relevant in statistics, this is not very convenient if one wants to filter association rules that can have different support. Hence in my apriori program this factor is removed by simply dividing the measure by the number of transactions (the absolute support of the item set comprising the items in the antecedent of the association rule, cf. the explanations in this section). With this normalization, the chi^2 measure can assume values between 0 (no dependence) and 1 (very strong - or actually perfect - dependence). The value that can be passed with the `-d` option is a lower bound for the strength of the dependence of the consequent on the antecedent in percent (0 - no dependence, 100 - perfect dependence). Only those association rules are selected, in which the consequent depends on the antecedent with this or a higher degree of dependence.

back to the top

p-Value Computed from chi^2 Measure (option `-ep`)

Provided that the two considered variables are independent, the chi^2-measure has a chi^2-distribution, which, for two binary variables, has one degree of freedom. Hence it is possible to compute the probability that the chi$^{\wedge 2}$-value of an association rule under consideration can be observed by chance under the assumption that the antecedent and the consequent of the rule are independent.

However, since with this measure small p-values indicate interesting rules, my apriori program uses rather 1-p as the evaluation measure, so that the general interpretation of the value given as an argument to the option `-d` as a lower limit can be kept. Note also that the option `-d` interprets its argument as a percentage, so that `-d99` means that only association rules with a p-value smaller than 100% - 99% = 1% = 1 - 0.99 = 0.01 are reported.

back to the top

Information Difference to Prior (option `-ei`)

The information difference to the prior is simply the *information gain* criterion (also called *mutual information* that is also used, for example, in decision tree learners like C4.5 to select the split attributes. Its basic idea is as follows: Without any further information about other items in the set, we have a certain probability (or, to be exact, a relative frequency) distribution for, say "bread" and "no bread". Let us assume it is 60% : 40% (prior confidence of the item "bread", just as above). This distribution has a certain entropy

$H = - P(\text{bread}) \log_2 P(\text{bread}) - P(\text{no bread}) \log_2 P(\text{no bread})$,

where P(bread) is equivalent to the (relative) support of "bread", which in turn is equivalent to the prior confidence of "bread". The entropy of a probability distribution is, intuitively, a lower bound on the number of yes-no-questions you have to ask in order to determine the actual value. This cannot be understood very well with only two possible values, but it can be made to work for this case too. I skip the details here, they are not that important for this application of information gain.

After we get the information that the items in the antecedent of the association rule are present (say, cheese), we have a different probability distribution, say 35% : 65%. I.e., P(bread|cheese) = 0.35 and P(no bread|cheese) = 0.65. If we also know the support of the item "cheese" (let it be P(cheese) = 0.4 and P(no cheese) = 0.6), then we can also compute the probabilities P(bread|no cheese) = 0.77 and P(no bread|no cheese) = 0.23. Hence we have two posterior probability distributions: one in case cheese is also present, and one in case cheese is not present. The question now is: How much information do we receive by observing whether the antecedent of the rule holds or not? Information is measured as a reduction of entropy. Hence the entropies of the two conditional probability distributions (one for "cheese" and one for "no cheese") are computed and summed weighted with the probability of their occurrence (that is, the relative frequency of "cheese" and "no cheese", respectively). This gives the (expected value of the) posterior or conditional entropy. The difference of this value to the prior entropy (see above) is the gain in information from the antecedent of the rule or, as I called it here, the information difference to the prior.

The value that can be given via the `-d` option is a lower bound for the information gain, measured in hundreds of a bit (percent of one bit). (Note that, since all items can only be present or absent, the information gain can be at most one bit.)

back to the top

p-Value Computed from G-Statistic (option `-eg`)

The so-called G-statistic is a less well-known alternative to the chi^2-measure that is based on information gain (or mutual information). Formally, it is proportional to the product of the information gain and the number of cases the information gain is computed from (here: the (absolute) support of the antecedent of the association rule). Under independence, the G-statistic also has a chi^2-distribution and thus it can be used in a similar fashion as the chi^2-measure to compute a p-value. This measure (p-value computed from G-statistic) can be selected with the option `-eg`.

back to the top

Selection Behavior of the Measures

In the directory `apriori/doc` you can find a Gnuplot script named `arem.gp` (`arem` stands for "additional rule evaluation measures") which visualizes the behavior of some of the additional rule evaluation measures. This script draws eight 3d graphs, one for the absolute confidence difference, one for the difference of the lift quotient to one, three for the information difference to the prior confidence and three for the normalized chi^2 measure. All graphs show the value of the additional rule evaluation measure over a plane defined by the prior and the posterior confidence of a rule. The latter two measures need three graphs, since they depend also on the antecedent support of a rule as a third parameter. Setting a minimal value for an additional rule evaluation measure is like flooding the corresponding graph landscape up to a certain level (given as a percentage, since all considered measures assume values between 0 and 1). Only those association rules are reported that "sit on dry land".

The first graph shows the behavior of the absolute confidence difference. For the diagonal, that is, the line where the prior and the posterior confidence are identical, its value is zero (as expected). The more the two confidences differ, the higher the value of this measure gets, but in a linear way.

The second graph shows the behavior of the lift quotient to one. Again its value is zero for the diagonal (as the value of all measures is) and becomes greater the more the prior and the posterior confidence differ. But it is much steeper for a small prior confidence than for a large one and it is non-linear.

The third to fifth graph show the information difference to the prior confidence for an antecedent support (which is identical to the rule support in my interpretation, see above) of 0.2 (20%), 0.3 (30%) and 0.4 (40%). The regions at the margins, where the measure is zero, correspond to impossible combinations of prior and posterior confidence and antecedent support. As you can see, the valley gets narrower with increasing antecedent support. That is, with the same minimal value for this measure, rules with low antecedent support need a higher confidence difference to be selected than rules with a high antecedent support. This nicely models the statistical significance of confidence changes: if you only have a few cases to support your rule, even a large deviation from the prior confidence can be explained by random fluctuations, since only a few transactions need to be different to produce a considerable change. However, if the antecedent support is large, even a small deviation (in percent) has to be considered as significant (non-random), since it takes a lot of changes to transactions to produce even a small change in the percentage. This dependence on the antecedent support of the rule and that the valley is not pointed at the diagonal (which means that even a low minimal value can exclude a lot of rules) is the main difference between the information gain and the normalized chi^2-measure on the one hand and the absolute confidence difference and difference of the lift quotient to one on the other.

The sixth to eighth graph show the normalized chi^2-measure for an antecedent support of 0.2 (20%), 0.3 (30%), and 0.4 (40%). The valleys are very similar to those for the information difference to the prior confidence, they only have slightly steeper flanks, especially for low antecedent support. So in practice there is no big difference between the information difference and the normalized chi^2-measure.

back to the top

Item Appearances

My apriori program provides a simple way to restrict the rules to generate w.r.t. the items that shall appear in them. It accepts a third optional input file, in which item appearance indicators can be given. For each item it can be stated whether it may appear in the antecedent (body) of a rule, in the head (consequent), or in both. A description of the format of this additional input file, including examples, can be found here. If no item appearances file is given, the rule selection is not restricted (any item can appear in antecedents and consequents of rules).

(I am grateful to the people at Integral Solutions Ltd., who developed the well-known data mining tool Clementine, but are now part of SPSS, for requesting the possibility to restrict item appearances.)

back to the top

Extended Item Set Selection

Since versions 4.20 (binary logarithm of support quotient) and 4.36 (other measures) there are also extended selection possibilities for frequent item sets. (These were added due to a cooperation with Sonja Gruen, RIKEN Brain Science Institute, Tokyo, Japan.)

Binary Logarithm of Support Quotient

A simple evaluation measure for item sets is the quotient of the actual support (as computed from the given transactions) and the expected support of an item set. The expected support of an item set can be computed from the support values of the individual items by assuming that the occurrences of all items are independent. Under independence we expect an item set to occur with a relative frequency that is the product of the relative occurrence frequencies of the individual items contained in it. Since this product quickly becomes very small (it decreases basically exponentially with the size of the item set), and thus the quotient of the actual and the expected frequency can become very large, it is advisable not to use the described support quotient directly, but its binary logarithm, so that its values stay in a manageable range. Computing the logarithm also has the advantage that the measure has the value zero for an item set that occurs exactly as often as expected under an assumption of item independence. The binary logarithm of the quotient of the actual and the expected support can be selected with the option `-eb`.

As for the additional rule evaluation measures, a minimum value for this measure can be set with the option `-d`. In this case only frequent item sets for which this measure exceeds the given threshold are reported. Note, however, that the option `-d` generally interprets its argument as a percentage. That is, if you only want item sets reported that occur at least twice as often as expected under independence (and thus desire the binary logarithm of the quotient of the actual and the expected support to be at least 1), you have to specify `-d100`.

back to the top

Additional Rule Evaluation Measures

Apart from the measure described in the preceding section (which is specific to item sets), all evaluation measures for association rules, as they were described in this section (including rule confidence, option `-ec`), can be used to filter item sets. The idea is to form all possible rules with one item in the consequent and all other items of the given item set in the antecedent. Each of these rules is then evaluated with the chosen measure and the results are aggregated. The aggregated value is the evaluation of the item set, and item sets can now be filtered by requiring a minimum value for this aggregation with the option `-d`.

There are four different aggregation modes for the evaluations of the rules that can be formed from an item set, which can be selected via the option `-a`:

`-ax` no aggregation (use first value)
`-am` minimum of individual measure values)
`-an` maximum of individual measure values)
`-aa` average of individual measure values)

Here "no aggregation (use first value)" means that only one association rule is formed. This rule has that item in the consequent that comes last in the order of the items. The evaluation of this rule is the evaluation of the item set. For all three other options all possible rules are formed.

The order of the items is determined with the option `-q`. By default the items are sorted ascendingly w.r.t. the sum of the sizes of the transactions they are contained in. (Note that this choice is based on efficiency issues: it usually leads to the fastest search. However, for use with this aggregation mode, you may want to choose a different order.) Other sorting options are ascendingly w.r.t. the item frequency or descendingly w.r.t. item frequency or the sum of the sizes of the transactions the items are contained in. With the option `-q0` the order in which the items are first encountered when the transactions are read is used. A specific, user-defined order may also be chosen, namely by using the option `-q0` and using the optional item appearances file (see this section), since the item appearances file is read before the transactions file. Therefore, in this case, the order of items in the appearances file determines the order of the items. If you want to use this option, make sure to specify "both" as the appearance indicator in order not to rule out the generation of any item set.

back to the top

Pruning with Additional Measures

By default only the minimum support is used to prune the search for frequent item sets. That is, if it is discovered that an item set does not meet the user-specified minimum support, no supersets of this item set are considered. (Note that this is a safe pruning rule, because no superset of an infrequent item set can be frequent.) With the option `-p` the additional item set evaluation measure can also be used for pruning the search additionally.

Pruning with an additional item set evaluation measure comes in two flavors: forward and backward. Backward pruning (which is switched on with the option `-p-1`), means that all frequent item sets (item sets that reach or exceed the user-specified minimum support) are evaluated with the additional measure, but those item sets, the evaluation of which is less than the user-specified limit (option `-d`), are discarded.

Note that providing or not providing the option `-p-1` makes no difference for finding *frequent item sets*, since for this target type simply selecting an additional evaluation measure and a limit for its value already have the effect of discarding item sets for which the additional evaluation is too low. However, backward pruning can make a huge difference for finding *closed* or *maximal item sets*. The reason is that the option `-p` changes the order in which item sets are filtered by the additional evaluation measure and are filtered for closed (or maximal) item sets: by default (or if explicitly requested with the option `-p0`), the item sets are first filtered for closed (or maximal) item sets. Then only the closed (or maximal) item sets are evaluated with the additional measure and thus further reduced to those closed (or maximal) item sets that reach or exceed the user-specified evaluation limit. With the option `-p-1` (or any other negative number as the argument - all negative numbers have the same effect) all frequent item sets are first evaluated with the additional measure and those that do not reach or exceed the user-specified limit are discarded. Only afterwards the closed (or maximal) item sets of this reduced set of item sets are determined.

Forward pruning, on the other hand, means that in addition to a check whether the additional evaluation of an item set reaches or exceeds the user-specified limit, it is tested whether all of its subsets (with at least a certain size) reach or exceed the limit. Only if this is the case, the item set is reported. (Technically, this is achieved by not considering any superset of item sets that failed to reach or exceed the user-specified limit for the additional measure - hence the name "forward pruning".)

Since some additional measures (like, for example, the chi^2-measure) cannot reasonably be used to evaluate the empty set or single element sets, but also because this restriction can be useful for certain applications, the option `-p` allows for a parameter that states the minimum size of the subsets that are to be considered. For example, `-p2` means that only subsets with at least two elements are required to reach or exceed the user-specified limit; single element sets and the empty set need not do so. Similarly, `-p4` means that all subsets up to a size of 3 are ignored in the check; only subsets with at least 4 elements must reach or exceed the limit. Note that the option `-p1` usually produces no output, because all single element sets and the empty set are formally assigned an additional evaluation of 0 for most measures. Hence 2 is usually the smallest useful parameter for this option.

Note also that, for example, `-p4` does *not* mean that all reported item sets must have at least four elements. It rather means that there is no additional requirement for item sets up to four elements. If you want to require a minimum size for reported item sets, use the option `-m`.

back to the top

Difference of Support Quotient to 1

As with the preceding measure the quotient of actual and expected support of an item set is computed and compared to 1 (a value of 1 signifies independence of the items). A minimum value for this measure can be set with the option `-d`. In this case only frequent item sets for which this measure exceeds the given threshold are kept.

back to the top

Transaction Prefix Tree

The counting process can be sped up by organizing the transactions into a prefix tree. That is, the items in each transaction are sorted and then transactions with the same prefix are grouped together and are counted, as one may say, in parallel. This way of organizing the transactions was added in version 4.03 and is the default behavior now. If you prefer that the transactions are treated individually (i.e., the transactions are stored in a simple list and only one transaction is counted at a time), use the option `-h`.

back to the top

Program Invocation and Options

My apriori program is invoked as follows:

```
apriori [options] infile outfile [appfile]
```

The normal arguments are:

infile	file to read transactions from
outfile	file to write association rules / hyperedges to
appfile	file stating item appearances (optional)

The possible options are:

`-t#`	target type (default: association rules)
	(s: frequent item sets, c: closed item sets, m: maximal item sets, r: association rules)
`-m#`	minimum number of items per set/rule (default: 1)
`-n#`	maximum number of items per set/rule (default: no limit)
`-s#`	minimum support of a set/rule (default: 10%)
`-S#`	maximum support of a set/rule (default: 100%)
	(positive: percentage, negative: absolute number)
`-c#`	minimum confidence of a rule (default: 80%)
`-o`	use original definition of the support of a rule (body & head)
`-e#`	additional evaluation measure (default: none)
`-a#`	aggregation mode for evaluation measure (default: none)
`-z`	zero evaluation below expected support (default: evaluate all)
`-d#`	minimal value of add. evaluation measure (default: 10%)
`-p#`	(min. size for) pruning with evaluation (default: no pruning)
	(< 0: backward, > 0: forward)
`-l#`	sort item sets in output by their size (default: no sorting)
	(< 0: descending, > 0: ascending)
`-g`	write output in scanable form (quote certain characters)
`-h#`	record header for output (default: "")
`-k#`	item separator for output (default: " ")
`-i#`	implication sign for association rules (default: " <- ")
`-v#`	output format for set/rule information (default: " (%1S)")
`-q#`	sort items w.r.t. their frequency (default: 1)
	(1: ascending, -1: descending, 0: do not sort,
	(2: ascending, -2: descending w.r.t. transaction size sum)
`-u#`	filter unused items from transactions (default: 0.5)
	(0: do not filter items w.r.t. usage in item sets,
	<0: fraction of removed items for filtering,
	>0: take execution times ratio into account)
`-j`	use quicksort to sort the transactions (default: heapsort)
`-x`	do not prune search with perfect extensions
`-y`	a-posteriori pruning of infrequent item sets
`-H`	do not organize transactions as a prefix tree
`-b#`	blank characters (default: " ")
`-f#`	field separators (default: " \t\r")
`-r#`	record separators (default: "\n")
`-C#`	comment characters (default: "#")
`-!`	print a list of additional rule evaluation measures

(`#` always means a number, a letter, or a string that specifies the parameter of the option.)

A note on the option `-j`: Constructing the prefix tree for the transactions requires sorting the transactions. Since version 4.17 heapsort is the default sorting method for the transactions, because it turned out that in conjunction with the item sorting (and especially for artificial datasets like `T10I4D100K`) quicksort can lead to very bad processing times (almost worst case behavior, that is, O(n^2) run time for the sorting). However, sometimes this is not a problem and then quicksort is slightly faster, which can be activated with the option -j.

back to the top

Input Format

Format of the Transactions File

A text file structured by field and record separators and blanks. Record separators, not surprisingly, separate records, usually lines, field separators fields (or columns), usually words. Blanks are used to fill fields (columns), e.g. to align them. In the transactions file each record must contain one transaction, i.e. a list of item identifiers, which are separated by field separators. An empty record is interpreted as an empty transaction.

Examples can be found in the directory `apriori/ex` in the source package. Refer to the file `apriori/ex/readme`, which explains how to process the different example files in the directory `apriori/ex` in the source package.

back to the top

Format of the Item Appearances File

A text file structured by field and record separators and blanks. (Note: For this file the same field and record separators and blanks are used as for the transactions file.)

The first record, which must have one field, contains the default appearance to be used with all items not mentioned in the appearances file. Other records state the appearance of specific items. The first field states the item, the second the appearance indicator. If no appearance indicator is given, the item will be ignored (i.e. may appear neither in the antecedent (body) nor in the consequent (head) of a rule). Empty records are ignored.

The following appearance indicators are recognized:

- item may appear only in rule antecedents (bodies):
  `i in b body a ante antecedent`
- item may appear only in rule consequents (heads):
  `o out h head c cons consequent`
- item may appear in rule antecedents (bodies) or in rule consequents (heads):
  `io inout bh b&h ac a&c both`
- item may appear neither in rule antecedents (bodies) nor in rule consequents (heads):
  `n neither none ign ignore -`

Example 1: Generate only rules with item "x" in the consequent.

```
in
x out
```

Example 2: Item "x" may appear only in a rule consequent (head), item "y" only in a rule antecedent (body); appearance of all other items is not restricted.

```
both
x head
y body
```

Providing no item appearances file is equivalent to an item appearances file containing only an indicator like "both", which does not restrict the appearance of any items.

back to the top

Output Format

The output format for item sets and association rules is fairly flexible. Especially the output of the additional information about item sets and association rules (support, confidence etc.) can be formatted freely.

Output Format for Frequent Item Sets

Each line of the output file contains one item set in the format

```
A B C [...] <add. info>
```

where A, B and C are item identifiers, and `<add. info>` is determined by the additional information output format (option `-v`). The item separator can be changed with the option `-k`.

The output format for the additional rule information can be any string with the following special format symbols (similar to the style of the special format symbols of the `printf` function in the programming language C):

%%	a percent sign
%a	absolute item set support

%s	relative item set support as a fraction
%S	relative item set support as a percentage
%e	additional evaluation measure
%E	additional evaluation measure as a percentage

All format specifiers can be extended by an integer number between the percent sign and the defining character, which specifies the number of decimal digits to be printed (expect those that refer to absolute support values, which are necessarily integer numbers). If no such number is given, no decimals are reported (the number is rounded to an integer).

The default additional information output format is for item sets "`  (%1S)`". That is, the relative support of the item set is printed with one decimal digit, enclosed in parentheses. Two spaces separate the additional information from the last item in the set.

back to the top

Output Format for Association Rules

Each line of the output file describes one association rule in the format

```
C <- A B [...] <add. info>
```

where A, B and C are item identifiers, and `<add. info>` is determined by the additional information output format (option `-v`). The implication sign can be changed with the option `-i`. The item separator can be changed with the option `-k`.

The output format for the additional rule information can be any string with the following special format symbols (similar to the style of the special format symbols of the `printf` function in the programming language C):

%%	a percent sign
%a	absolute item set support
%s	relative item set support as a fraction
%S	relative item set support as a percentage
%b	absolute body set support
%x	relative body set support as a fraction
%X	relative body set support as a percentage
%h	absolute head item support
%y	relative head item support as a fraction
%Y	relative head item support as a percentage
%c	rule confidence as a fraction
%C	rule confidence as a percentage
%l	lift value of a rule (confidence/prior)
%L	lift value of a rule as a percentage
%e	additional evaluation measure
%E	additional evaluation measure as a percentage

All format specifiers can be extended by an integer number between the percent sign and the defining character, which specifies the number of decimal digits to be printed (expect those that refer to absolute support values, which are necessarily integer numbers). If no such number is given, no decimals are reported (the number is rounded to an integer).

The default additional information output format for rules is "`  (%1X, %1C)`". That is, the relative support of the rule antecedent and the confidence of the rule are printed with one decimal digit. These two values are separated by a comma and enclosed in parentheses. Two spaces separate the additional information from the last item in the rule.

back to the top

Compilation Options

The program can be compiled with two additional compilation options (see `makefile`), namely `-DBENCH` and `-DALIGN8`.

Compiling the program with `-DBENCH` produces a version that prints some benchmark information on termination, in particular about the memory used during the item set tree construction (number of nodes, counters, necessary counters, child pointers, necessary child pointers). Collecting the memory usage information slightly, but negligibly increases the execution time.

Compiling the program with `-DALIGN8` produces a version for 64 bit machines (architecture model: pointers are 64 bits, integers are 32 bits wide) that require pointers to be aligned to addresses that are divisible by 8. (Note that this is not the case for standard Intel or AMD processors, which can read 64 bit pointers from any address that is divisible by 4. For such systems it is not necessary to use the flag `-DALIGN8`.) The needed adaptations slightly, but negligibly increase memory consumption. (I am grateful to Anthony Casaletto, SPSS Inc., for helping me a lot to identify these alignment problems, by compiling and testing the program on a 64 bit machine with alignment requirements, since I do not have access to one.)

back to the top

Copying

back to the top

Download

Download page with most recent version.

back to the top

Contact

E-mail:	christian.borgelt@softcomputing.es christian@borgelt.net
Snail mail: Old	Christian Borgelt Intelligent Data Analysis and Graphical Models Research Unit European Center for Soft Computing Edificio Cientifico-Tecnológico, 3ª Planta c/ Gonzalo Gutiérrez Quirós s/n 33600 Mieres (Asturias) Spain
Phone:	+34 985 456545
Fax:	+34 985 456699

back to the top