Øving 2 Datavarehus

Alexander Fredheim

February 2020

1 Apriori Algorithm

1.a

Here we are going to use the apriori algorithm to generate frequent itemsets, which we are later going to use to find association rules.
We have minsupport = 0.5 which for our table means that every itemset must have a minimum support of 5 in our initial dataset.

	Itemset	**Supportcount**
C1	{A}	6
	{B}	8
	{C}	10
	{D}	4
	{E}	5
	{F}	3
	{G}	8
	{H}	7

Table 1: our initial 1-element set

	Itemset	**Supportcount**
L1	{A}	6
	{B}	8
	{C}	10
	{E}	5
	{G}	8
	{H}	7

Table 2: frequent 1-element sets with supportcount >= minsupport

C2

Itemset	Supportcount
{A, B}	5
{A, C}	6
{A, E}	1
{A, G}	6
{A, H}	5
{B, C}	8
{B, E}	4
{B, G}	7
{B, H}	5
{C, E}	5
{C, G}	8
{C, H}	7
{E, G}	3
{E, H}	3
{G, H}	5

Table 3: 2-element sets generated from frequent 1-element sets

L2

Itemset	Supportcount
{A, B}	5
{A, C}	6
{A, G}	6
{A, H}	5
{B, C}	8
{B, G}	7
{B, H}	5
{C, E}	5
{C, G}	8
{C, H}	7
{G, H}	5

Table 4: frequent 2-element sets with supportcount >= minsupport

	Itemset	Supportcount
C3	{A, B, C}	5
	{A, B, G}	5
	{A, B, H}	4
	{A, C, G}	6
	{A, C, H}	5
	{A, G, H}	5
	{B, C, G}	7
	{B, C, H}	5
	{B, G, H}	4
	{C, G, H}	5

Table 5: 3-element sets generated from frequent 2-element sets

Note that in table 5 we don't include {C, E, G} and {C, E, H} as they have subsets {E, G} and {E, H} which doesn't meet the minimum support requirement

	Itemset	Supportcount
L3	{A, B, C}	5
	{A, B, G}	5
	{A, C, G}	6
	{A, C, H}	5
	{A, G, H}	5
	{B, C, G}	7
	{B, C, H}	5
	{C, G, H}	5

Table 6: frequent 3-element sets with supportcount >= minsupport

	Itemset	Supportcount
C4	{A, B, C, G}	5
	{A, C, G, H}	5

Table 7: 4-element sets generated from frequent 3-element sets

Note that in table 7 we don't include {B, C, G, H} as it has subset {B, G, H} which doesn't meet the minimum support requirement

	Itemset	Supportcount
L4	{A, B, C, G}	5
	{A, C, G, H}	5

Table 8: frequent 4-element sets with supportcount >= minsupport

1.b

For this task we are going to generate some rules for our frequent 4-element sets {A, B, C, G} and {A, C, G, H}. We use minconf = 0.8.

Rule generation for element set {A, B, C, G}:

Rules	**Confidence**
{B, C, G} => {A}	0.71
{A, C, G} => {B}	0.83
{A, B, G} => {C}	1.00
{A, B, C} => {G}	1.00

Table 9: possible rules generated from {A, B, C, G} and their corresponding confidence

We see in table 9 that {B, C, G} => {A} does not meet our confidence threshold. We can therefore prune all possible rules containing A on the right hand side

Rules	**Confidence**
{A, G} => {B, C}	0.83
{A, C} => {B, G}	0.83
{A, B} => {C, G}	1.00

Table 10: possible rules further generated from the rules of {A, C, G}, {A, B, G} and {A, B, C} and their corresponding confidence. After pruning

Rules	**Confidence**
{A} => {B, C, G}	0.83

Table 11: possible rules further generated from the rules of {A, G}, {A, C} and {A, G} and its corresponding confidence. After pruning

Rule generation for element set {A, C, G, H}:

Rules	Confidence
{C, G, H} => {A}	1.00
{A, G, H} => {C}	1.00
{A, C, H} => {G}	1.00
{A, C, G} => {H}	0.83

Table 12: possible rules generated from {A, C, G, H} and their corresponding confidence

Rules	Confidence
{G, H} => {A, C}	1.00
{C, H} => {A, G}	0.71
{C, G} => {A, H}	0.63
{A, H} => {C, G}	1.00
{A, G} => {C, H}	0.83
{A, C} => {G, H}	0.83

Table 13: possible rules further generated from the rules of {G, H}, {C, H}, {C, G}, {A, H}, {A, G} and {A, C} and its corresponding confidence.

We see in table 13 that {C, H} => {A, G} and {C, G} => {A, H} does not meet our confidence threshold. We can therefore prune all possible rules containing subsets of either {A, G} or {A, H} on the right hand side of the rule.

Rules	Confidence
{A} => {C, G, H}	0.83

Table 14: the only rule derived from the rules in table 13, not containing supersets of {A, G} or {A, H}

we end up with the following rules for {A, B, C, G} and {A, C, G, H}:

Rules	**Confidence**
{A, C, G} => {B}	0.83
{A, B, G} => {C}	1.00
{A, B, C} => {G}	1.00
{A, G} => {B, C}	0.83
{A, C} => {B, G}	0.83
{A, B} => {C, G}	1.00
{A} => {B, C, G}	0.83
{C, G, H} => {A}	1.00
{A, G, H} => {C}	1.00
{A, C, H} => {G}	1.00
{A, C, G} => {H}	0.83
{G, H} => {A, C}	1.00
{A, H} => {C, G}	1.00
{A, G} => {C, H}	0.83
{A, C} => {G, H}	0.83
{A} => {C, G, H}	0.83

Table 15: all rules generated from {A, B, C, G} and {A, C, G, H}

2 FP-Growth

We start by scanning the items and sorting it by support (decreasing). Our minsupport is still 0.5 which means minimum support is still 5

Item	**Supportcount**
C	10
B	8
G	8
H	7
A	6
E	5

Table 16: our elements sorted by frequency. Elements not meeting support requirements removed

TID	**Initial Items**	**Ordered Items**
110	A,C,F,G,H	C,G,H,A
111	B,C,E,D,G	C,B,G,E
112	B,C,E,F,G	C,B,G,E
113	A,B,C,G	C,B,G,A
114	C,D,E,H	C,H,E
115	A,B,C,G,H	C,B,G,H,A
116	A,B,C,D,G,H	C,B,G,H,A
117	B,C,E,G	C,B,G,E
118	A,B,C,F,G,H	C,B,G,H,A
119	A,B,C,D,E,G,H	C,B,G,H,A,E

Table 17: Transactions with the prioritized ordering

we now create the FP-tree, transaction by transaction, following the order in "ordered items from table 17"

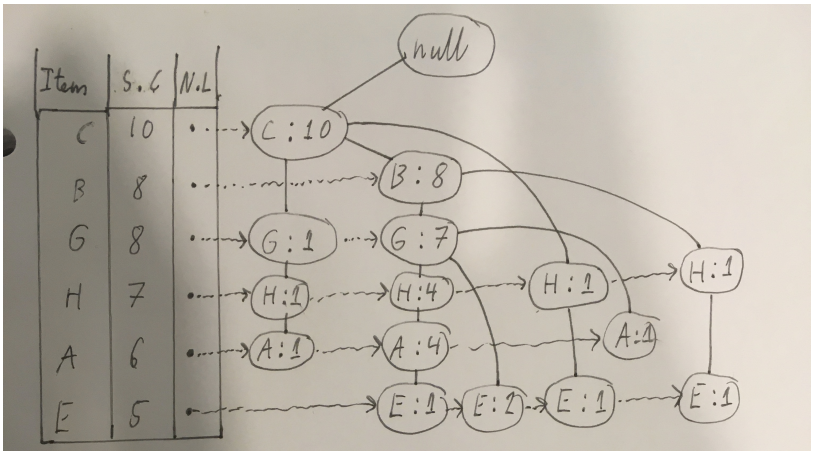

Figure 1: FP-tree

we now need to find the frequent itemsets by constructing a conditional FP-tree. To do this we start bottom up for the items in table 16 and look at all trees ending with the given item. We then use the divide and conquer approach where necessary to generate the needed trees. we get the following table

Item	**Conditional Pattern Base**	**Cond-itional FP-tree**	**Frequent Patterns Generated**
E	{{C,B,H:1}, {C,B,G:2}, {C,H:1}, {C,B,G,H,A:1}}	<C:5>	{C,E:5}
A	{{C,G,H:1}, {C,B,G:1}, {C,B,G,H:4}}	<C:6,G:6,H:1>, <C:6,G:6,B:5,H:4>	{C,A:6}, {G,A:6}, {H,A:5} {B,A:5}, {C,G,A:6}, {C,B,A:5}, {G,B,A:5}, {C,H,A:5}, {G,H,A:5}, {C,G,H,A:5}, {C,G,B,A:5}
H	{{C,G:1}, {C,B:1}, {C:1}, {C,B,G:4}}	<C:7,G:1>, <C:7,B:5,G:4>	{C,H:7}, {B,H:5}, {G,H:5} {C,B,H:5}, {C,G,H:5}
G	{{C:1}, {C,B:7}}	<C:8,B:7>	{C,G:8}, {B,G:7}, {C,B,G:7}
B	{{C:8}}	<C:8>	{C,B:8}

Table 18: Table showing the generated frequent patterns

3 KNIME

The node setup from KNIME:

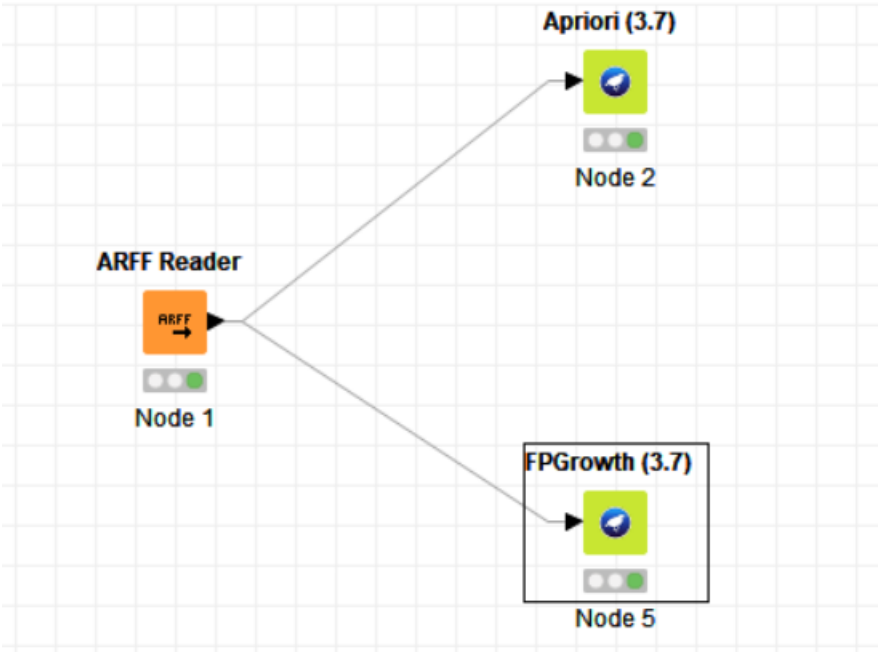

Figure 2: Nodes in KNIME

Apriori rules:

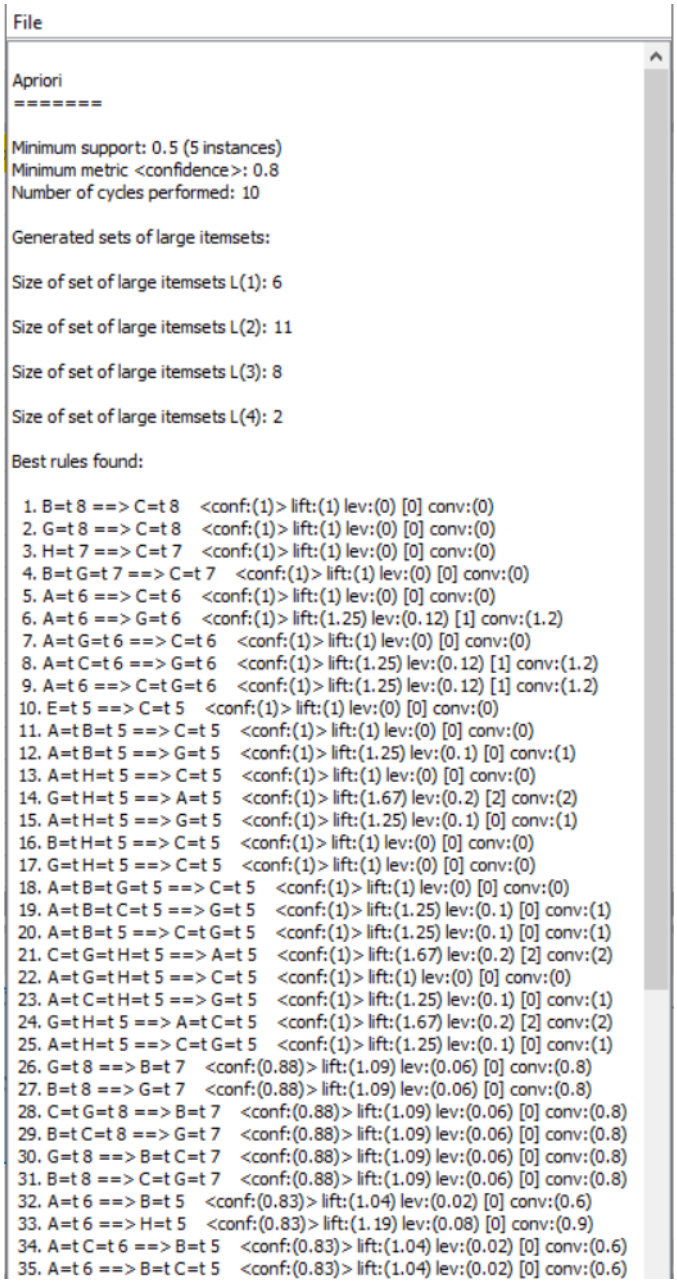

Figure 3: the 35 "best" rules generated by the apriori node in KNIME

FP rules:

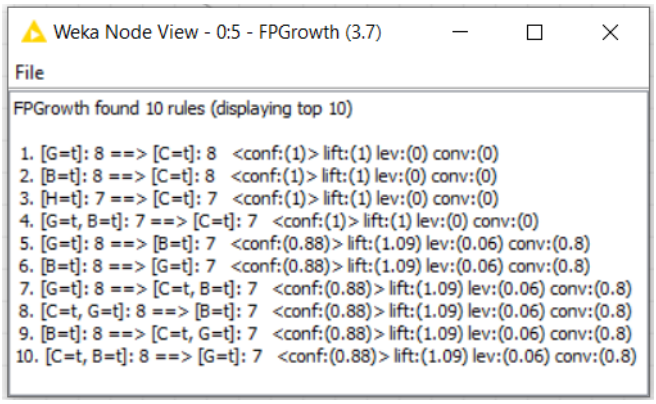

Figure 4: 10 rules generated by the FP-Growth node in KNIME

4 Compact Representation of Frequent Itemsets

The algorithm can be described by to steps:

1. generate frequent element sets by taking the union of all subsets of the closed frequent element sets
2. evaluate the support of any non-closed frequent k-element set by using the principle of k+1 frequent supersets. The support of any non frequent k-elementset is the maximum support of the frequent supersets.

This gives us the following table:

Frequent Element set	**Support**
{a}	11
{b}	10
{c}	6
{d}	13
{e}	8
{a,b}	7
{a,c}	6
{a,d}	11
{a,e}	7
{b,d}	7
{b,e}	8
{c,d}	6
{c,e}	5
{d,e}	6
{a,b,e}	7
{a,c,d}	6
{a,c,e}	5
{a,d,e}	5
{b,d,e}	4
{c,d,e}	5
{a,c,d,e}	5

Table 19: frequent item sets and their support