

Session 4

FP-Growth and Apriori parameter studies

FP-Growth finds the same frequent itemsets as Apriori without generating candidates. The session compares the two and then runs Apriori with the exact parameter ranges the manual specifies.

Objectives

× Do not copy. Read for understanding and the viva

- Complete questions 9 to 11 of the manual: fp-growth and apriori parameter studies
- Prepare the deliverable before the lab and finish it during the session
- Be ready to explain every step in the viva

Questions Covered

× Do not copy. Read for understanding and the viva

Question	Requirement	Status
Q9	Find the frequent patterns using FP-Growth algorithm on contactlenses.arff and...	Complete
Q10	Generate association rules using Apriori algorithm with Bank.arff relation	Complete
Q11	Generate association rule for the credit card promotion dataset using Apriori...	Complete

Preparation

× Do not copy. Read for understanding and the viva

- FP-Growth in WEKA needs binary or nominal attributes; use `NominalToBinary` on other datasets first.
- Translate the manual's wording into WEKA fields: upper bound 100 percent, lower bound 20 percent, delta 5 percent, minMetric 0.8, numRules 5 for question 10(a).
- Question 10(b) uses the lift metric: set `metricType` to Lift and `minMetric` to 1.5.

Question 9

Problem Statement

■ Write in lab record

Find the frequent patterns using FP-Growth algorithm on `contactlenses.arff` and `test.arff` datasets.

Solution

■ Write in lab record

Steps

1. Open file `contact-lenses.arff`. Filter, Choose, unsupervised, attribute, `NominalToBinary`; click the name and set `transformAllValues` True and `binaryAttributesNominal` True, OK, Apply. The 5 attributes become 12 binary ones named `age=young`, `age=pre-presbyopic`, ..., `contact-lenses=none`, each with values 0 and 1. (FP-Growth also accepts the raw nominal file, but then for a two-valued attribute only its second value, `positiveIndex` 2, is an item, so `tear-prod-rate=reduced` would never appear in a rule.)
2. Associate, Choose, `weka.associations.FPGrowth`. Options: `positiveIndex` 2, `numRulesToFind` 10, `metricType` Confidence, `minMetric` 0.9, `delta` 0.05, `lowerBoundMinSupport` 0.1, `upperBoundMinSupport` 1.0, `findAllRulesForSupportLevel` False. Start.
3. `test.arff` is not a WEKA file; it is the 10-transaction market basket below, one binary attribute per item. Open file `test.arff` (already binary with values 0 and 1, so no filter). Same FPGrowth, `lowerBoundMinSupport` 0.3, `minMetric` 0.7, Start.
4. Compare with Apriori: on the same working relation choose Apriori with the same support and confidence and Start. The rules are the same; only the algorithm and the output format differ.
5. Run `fpgrowth.py` on both files to see the header table, the FP-tree and the full list of frequent patterns, which WEKA's FPGrowth does not print.

Program

- Lab record: every tab is one file of the answer. Write all of them.

fpgrowth.py

fpgrowth.py

```

1  #!/usr/bin/env python3
2  """FP-Growth frequent-pattern miner with WEKA-style rule output.
3
4  Standard library only. Builds the FP-tree (items ordered by frequency),
5  prints the header table and the tree, mines every frequent itemset by
6  recursive conditional pattern bases (no candidate generation), then derives
7  rules the way weka.associations.FPGrowth prints them.
8
9  Item rule (same as WEKA): if every attribute has exactly two values, only th
10 second value counts as 'present' (positiveIndex 2). Otherwise each
11 attribute=value pair is an item.
12
13 Usage: python3 fpgrowth.py FILE.arff [-M minSupport] [-C minConfidence] [-N
14 """
15 import argparse
16 import itertools
17 import re
18 from collections import Counter
19 from decimal import ROUND_HALF_UP, Decimal
20
21
22 def read_arff(path):
23     relation, attrs, rows, in_data = "", [], [], False
24     with open(path) as f:
25         for line in f:
26             line = line.strip()
27             if not line or line.startswith("%"):
28                 continue
29             if in_data:
30                 rows.append([v.strip().strip("'\"") for v in line.split(",")
31 elif line.lower().startswith("@relation"):
32                 relation = line.split(None, 1)[1].strip("'\"")
33 elif line.lower().startswith("@attribute"):
34                 m = re.match(r"@attribute\s+(['^']*|\"[^\"]*\"|\s+)\s+\{(.*)",
35 attrs.append((m.group(1).strip("'\""), [v.strip() for v in m
36 elif line.lower().startswith("@data"):
37                 in_data = True
38     return relation, attrs, rows
39
40
41 class Node:
42     __slots__ = ("item", "count", "parent", "children")
43

```

```
44     def __init__(self, item, parent):
45         self.item, self.count, self.parent, self.children = item, 0, parent,
46
47
48     def build(trans, min_count):
49         """trans: list of (frozenset of items, weight). Returns root, header, or
50         freq = Counter()
51         for items, w in trans:
52             for i in items:
53                 freq[i] += w
54         freq = {i: c for i, c in freq.items() if c ≥ min_count}
55         order = sorted(freq, key=lambda i: (-freq[i], i))
56         root, header = Node(None, None), {i: [] for i in order}
57         for items, w in trans:
58             node = root
59             for i in [x for x in order if x in items]:
60                 if i not in node.children:
61                     node.children[i] = Node(i, node)
62                     header[i].append(node.children[i])
63                 node = node.children[i]
64                 node.count += w
65         return root, header, order, freq
66
67
68     def mine(trans, min_count, suffix, out):
69         root, header, order, freq = build(trans, min_count)
70         for item in reversed(order): # least frequent item first
71             pattern = suffix | {item}
72             out[pattern] = freq[item]
73             base = [] # conditional pattern base of `item`
74             for node in header[item]:
75                 path, p = [], node.parent
76                 while p.item is not None:
77                     path.append(p.item)
78                     p = p.parent
79                 if path:
80                     base.append((frozenset(path), node.count))
81             if base:
82                 mine(base, min_count, pattern, out)
83         return root, header, order, freq
84
85
86     def show(node, depth=0):
87         for child in node.children.values():
```

```
88     print("  " * depth + f"{child.item}:{child.count}")
89     show(child, depth + 1)
90
91
92 def fmt(x):
93     """WEKA's Utils.doubleToString(x, 2): rounds half up, trailing zeros tri
94     s = str(Decimal(repr(x)).quantize(Decimal("0.01"), ROUND_HALF_UP)).rstri
95     return s or "0"
96
97
98 def main():
99     p = argparse.ArgumentParser()
100
101     0     p.add_argument("file")
102
103     1     p.add_argument("-M", type=float, default=0.2, help="minimum support (fra
104
105     2     p.add_argument("-C", type=float, default=0.9, help="minimum confidence")
106
107     3     p.add_argument("-N", type=int, default=10, help="rules to display")
108
109     4     a = p.parse_args()
110
111     5
112
113     6     relation, attrs, rows = read_arff(a.file)
114
115     7     binary = all(len(v) == 2 for _, v in attrs)
116
117     8     trans = []
118
119     9     for r in rows:
120
121         0         items = set()
122
123         1         for (name, values), v in zip(attrs, r):
124
125             2             if v == "?" or (binary and v != values[1]):
126
127                 3                 continue
128
129             4                 items.add(f"{name}={v}")
130
131         5         trans.append((frozenset(items), 1))
```

```
11
6     n = len(trans)
11
7     min_count = int(a.M * n + 0.5)
11
8     print(f"Relation: {relation}   Instances: {n}   Minimum support: {a.M} (
11
9
12
0     out = {}
12
1     root, header, order, freq = mine(trans, min_count, frozenset(), out)
12
2     print("\nHeader table (frequency-ordered items):")
12
3     for i in order:
12
4         print(f"   {i} {freq[i]}   ({len(header[i])} node(s))")
12
5     print("\nFP-tree (item:count, indented by depth):")
12
6     show(root)
12
7
12
8     by_size = Counter(len(k) for k in out)
12
9     print("\nFrequent itemsets by size:", dict(sorted(by_size.items())))
13
0     for k in sorted(by_size):
13
1         print(f"L({k}):")
13
2         for iset in sorted((s for s in out if len(s) == k), key=sorted):
13
3             print("   " + " ".join(sorted(iset)), out[iset])
13
4
13
5     rules = []
13
6     for iset, nxy in out.items():
13
7         if len(iset) < 2:
```

```

13
8         continue
13
9     for size in range(1, len(iset)):
14
10         for cons in itertools.combinations(sorted(iset), size):
14
11             cons = frozenset(cons)
14
12             prem = iset - cons
14
13             nx, ny = out[prem], out[cons]
14
14             conf = nxy / nx
14
15             if conf ≥ a.C - 1e-9:
14
16                 lift = conf / (ny / n)
14
17                 lev = nxy / n - (nx / n) * (ny / n)
14
18                 conv = (nx * (n - ny) / n) / (nx - nxy + 1)
14
19                 rules.append((prem, cons, nx, nxy, conf, lift, lev, conv)
15
20 rules.sort(key=lambda r: (-r[4], -r[3]))
15
21 print(f"\nFPGrowth found {len(rules)} rules (displaying top {min(a.N, le
15
22 for i, (prem, cons, nx, nxy, conf, lift, lev, conv) in enumerate(rules[:
15
23     print(f"{i:2d}. [{', '.join(sorted(prem))}]: {nx} ⇒ [{', '.join(so
15
24         f"    <conf:({fmt(conf)})> lift:({fmt(lift)}) lev:({fmt(lev)})
15
25
15
26
15
27 if __name__ == "__main__":
15
28     main()

```

Output

```
python3 fpgrowth.py contact-lenses.arff -M 0.2 -C 0.9 (computed):
```

```
1 Relation: contact-lenses   Instances: 24   Minimum support: 0.2 (5 instances
2
3 Header table (frequency-ordered items):
4   contact-lenses=none 15   (1 node(s))
5   astigmatism=no 12   (2 node(s))
6   astigmatism=yes 12   (2 node(s))
7   spectacle-prescrip=hypermetrope 12   (4 node(s))
8   spectacle-prescrip=myope 12   (4 node(s))
9   tear-prod-rate=normal 12   (6 node(s))
10  tear-prod-rate=reduced 12   (4 node(s))
11  age=pre-presbyopic 8   (8 node(s))
12  age=presbyopic 8   (8 node(s))
13  age=young 8   (8 node(s))
14  contact-lenses=soft 5   (5 node(s))
15
16 FP-tree (item:count, indented by depth):
17 contact-lenses=none:15
18   astigmatism=no:7
19     spectacle-prescrip=myope:4
20     tear-prod-rate=reduced:3
21     age=young:1
22     age=pre-presbyopic:1
23     age=presbyopic:1
24     tear-prod-rate=normal:1
25     age=presbyopic:1
26     spectacle-prescrip=hypermetrope:3
27     tear-prod-rate=reduced:3
28     age=young:1
29     age=pre-presbyopic:1
30     age=presbyopic:1
31 astigmatism=yes:8
32   spectacle-prescrip=myope:3
33   tear-prod-rate=reduced:3
34   age=young:1
35   age=pre-presbyopic:1
36   age=presbyopic:1
37   spectacle-prescrip=hypermetrope:5
38   tear-prod-rate=reduced:3
39   age=young:1
40   age=pre-presbyopic:1
41   age=presbyopic:1
42   tear-prod-rate=normal:2
43   age=pre-presbyopic:1
```

```

44         age=presbyopic:1
45 astigmatism=no:5
46     spectacle-prescrip=myope:2
47         tear-prod-rate=normal:2
48         age=young:1
49         contact-lenses=soft:1
50         age=pre-presbyopic:1
51         contact-lenses=soft:1
52     spectacle-prescrip=hypermetrope:3
53         tear-prod-rate=normal:3
54         age=young:1
55         contact-lenses=soft:1
56         age=pre-presbyopic:1
57         contact-lenses=soft:1
58         age=presbyopic:1
59         contact-lenses=soft:1
60 astigmatism=yes:4
61     spectacle-prescrip=myope:3
62         tear-prod-rate=normal:3
63         age=young:1
64         age=pre-presbyopic:1
65         age=presbyopic:1
66     spectacle-prescrip=hypermetrope:1
67         tear-prod-rate=normal:1
68         age=young:1
69
70 Frequent itemsets by size: {1: 11, 2: 21, 3: 6}
71 (itemset listing omitted here; identical to the L(1), L(2), L(3) lists in Se
72
73 FPGrowth found 10 rules (displaying top 10)
74
75 1. [tear-prod-rate=reduced]: 12  $\Rightarrow$  [contact-lenses=none]: 12    <conf:(1)>
76 2. [spectacle-prescrip=myope, tear-prod-rate=reduced]: 6  $\Rightarrow$  [contact-lense
77 3. [spectacle-prescrip=hypermetrope, tear-prod-rate=reduced]: 6  $\Rightarrow$  [contac
78 4. [astigmatism=yes, tear-prod-rate=reduced]: 6  $\Rightarrow$  [contact-lenses=none]:
79 5. [astigmatism=no, tear-prod-rate=reduced]: 6  $\Rightarrow$  [contact-lenses=none]: 6
80 6. [contact-lenses=soft]: 5  $\Rightarrow$  [tear-prod-rate=normal]: 5    <conf:(1)> lif
81 7. [contact-lenses=soft, tear-prod-rate=normal]: 5  $\Rightarrow$  [astigmatism=no]: 5
82 8. [astigmatism=no, contact-lenses=soft]: 5  $\Rightarrow$  [tear-prod-rate=normal]: 5
83 9. [contact-lenses=soft]: 5  $\Rightarrow$  [astigmatism=no, tear-prod-rate=normal]: 5
84 10. [contact-lenses=soft]: 5  $\Rightarrow$  [astigmatism=no]: 5    <conf:(1)> lift:(2) 1

```

```
python3 fpgrowth.py test.arff -M 0.3 -C 0.7 (computed):
```

```
1 Relation: test   Instances: 10   Minimum support: 0.3 (3 instances)
2
3 Header table (frequency-ordered items):
4   bread=1 8   (1 node(s))
5   milk=1 8   (2 node(s))
6   butter=1 6   (2 node(s))
7   eggs=1 4   (3 node(s))
8   jam=1 3   (2 node(s))
9
10 FP-tree (item:count, indented by depth):
11 bread=1:8
12   milk=1:6
13     butter=1:4
14       eggs=1:1
15       jam=1:1
16     eggs=1:2
17   butter=1:2
18     jam=1:2
19 milk=1:2
20   eggs=1:1
21
22 Frequent itemsets by size: {1: 5, 2: 7, 3: 3}
23 L(1):
24   bread=1 8
25   butter=1 6
26   eggs=1 4
27   jam=1 3
28   milk=1 8
29 L(2):
30   bread=1 butter=1 6
31   bread=1 eggs=1 3
32   bread=1 jam=1 3
33   bread=1 milk=1 6
34   butter=1 jam=1 3
35   butter=1 milk=1 4
36   eggs=1 milk=1 4
37 L(3):
38   bread=1 butter=1 jam=1 3
39   bread=1 butter=1 milk=1 4
40   bread=1 eggs=1 milk=1 3
41
42 FPGrowth found 15 rules (displaying top 10)
43
```

```

44 1. [butter=1]: 6  $\implies$  [bread=1]: 6    <conf:(1)> lift:(1.25) lev:(0.12) conv:
45 2. [eggs=1]: 4  $\implies$  [milk=1]: 4    <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.
46 3. [butter=1, milk=1]: 4  $\implies$  [bread=1]: 4    <conf:(1)> lift:(1.25) lev:(0.0
47 4. [jam=1]: 3  $\implies$  [butter=1]: 3    <conf:(1)> lift:(1.67) lev:(0.12) conv:(1
48 5. [butter=1, jam=1]: 3  $\implies$  [bread=1]: 3    <conf:(1)> lift:(1.25) lev:(0.06
49 6. [bread=1, jam=1]: 3  $\implies$  [butter=1]: 3    <conf:(1)> lift:(1.67) lev:(0.12
50 7. [jam=1]: 3  $\implies$  [bread=1, butter=1]: 3    <conf:(1)> lift:(1.67) lev:(0.12
51 8. [jam=1]: 3  $\implies$  [bread=1]: 3    <conf:(1)> lift:(1.25) lev:(0.06) conv:(0.
52 9. [bread=1, eggs=1]: 3  $\implies$  [milk=1]: 3    <conf:(1)> lift:(1.25) lev:(0.06)
53 10. [bread=1]: 8  $\implies$  [butter=1]: 6    <conf:(0.75)> lift:(1.25) lev:(0.12) co

```

What WEKA prints for contact-lenses after NominalToBinary (item names carry the binary value, otherwise the same ten rules as above):

```

1 Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.0
2 Relation:    contact-lenses-weka.filters.unsupervised.attribute.NominalToBi
3 Instances:   24    Attributes:   12
4
5 FPGrowth found 10 rules (displaying top 10)
6
7 1. [tear-prod-rate=reduced=1]: 12  $\implies$  [contact-lenses=none=1]: 12    <conf:(
8 2. [spectacle-prescrip=myope=1, tear-prod-rate=reduced=1]: 6  $\implies$  [contact-l
9 ...
10 10. [contact-lenses=soft=1]: 5  $\implies$  [astigmatism=no=1, tear-prod-rate=normal=

```

FP-Growth against Apriori on contact-lenses at support 0.2:

	Apriori (Session 3)	FP-Growth
Frequent itemsets	L1 11, L2 21, L3 6	11, 21, 6 (same sets)
Rules at confidence 0.9	10	10, same rules
Passes over the data	one per level: 3, plus one per support step in WEKA's loop	2: one to count items, one to build the tree
Candidates counted	210 possible pairs joined, pruned to the ones with large subsets	none; patterns grow from conditional trees
Memory	candidate lists	the tree: 52 nodes for 24 rows here

Explanation

FP-Growth compresses the transactions into a prefix tree. Items are sorted by frequency (`contact-lenses=none` 15 first, `contact-lenses=soft` 5 last; ties broken alphabetically here, WEKA's tie order may differ) so that frequent items share tree prefixes: all 15 `none` rows hang under one node. The header table links every node of an item. Mining starts from the least frequent item: its conditional pattern base is the set of prefix paths above its nodes, a small conditional tree is built from them, and the recursion continues. No candidate is ever generated, so the 210 candidate pairs Apriori has to test on this file are never formed. Both algorithms find the identical frequent itemsets because the definition of frequent does not depend on the search; that is why the rule lists agree line for line. On `test.arff` the tree shows the pattern immediately: `bread` and `milk` head almost every path, `jam` only appears under `butter`, which gives the rule `jam  $\Rightarrow$  butter` with lift 1.67.

Question 10

Problem Statement

■ Write in lab record

Generate association rules using Apriori algorithm with Bank.arff relation

1. Set minimum support range as 20% to 100%, incremental decrease factor as 5% and confidence factor as 80% and generate 5 rules.
2. Set minimum support as 10%, delta 5%, minimum lift as 150% and generate 4 rules.

Solution

■ Write in lab record

Use `bank-nominal.arff` from Session 2 (bank-data with `id` removed, `age` and `income` in 3 bins, `children` nominal). Apriori refuses the raw file because `age`, `income` and `children` are numeric.

Steps

1. Open file `bank-nominal.arff`; confirm 600 instances, 11 attributes, all Nominal.
2. Associate, Choose Apriori, click the name and set, for part (a):

Manual wording	WEKA parameter	Value
minimum support range 20% to 100%	<code>lowerBoundMinSupport</code> and <code>upperBoundMinSupport</code>	0.2 and 1.0
incremental decrease factor 5%	<code>delta</code>	0.05
confidence factor 80%	<code>metricType</code> Confidence, <code>minMetric</code>	0.8
generate 5 rules	<code>numRules</code>	5

3. OK, Start. The Scheme line must read `weka.associations.Apriori -N 5 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.2 -S -1.0 -c -1`.

4. For part (b) set:

Manual wording	WEKA parameter	Value
minimum support 10%	<code>lowerBoundMinSupport</code>	0.1
delta 5%	<code>delta</code>	0.05
minimum lift 150%	<code>metricType</code> Lift, <code>minMetric</code>	1.5
generate 4 rules	<code>numRules</code>	4

5. OK, Start. Scheme line: `weka.associations.Apriori -N 4 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1`.

6. Tick `outputItemSets` and Start once more if you need the large itemset counts for the record.

Output

Expected shape (WEKA was not run here; bank-data is 600 rows and the counts depend on your bins, so copy the numbers from your own screen):

```

1  == Run information ==
2
3  Scheme:      weka.associations.Apriori -N 5 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0
4  Relation:    bank-nominal
5  Instances:   600
6  Attributes:  11
7  == Associator model (full training set) ==
8
9  Minimum support: 0.2 (120 instances)      ← or the first level at which 5 r
10 Minimum metric <confidence>: 0.8
11 Number of cycles performed: 16
12
13 Best rules found:
14
15  1. children=0 save_act=YES 190 ==> current_act=YES 158    <conf:(0.83)> lif
16  2. ...

```

Part (b) prints `Minimum metric <lift>: 1.5` and rules whose lift column is in angle brackets, for example `income='(43758.136667-inf)' 120 ==> save_act=YES 110 conf:(0.92) <lift:(1.33)> ...` would be rejected at 1.5 while a rule between `mortgage=YES` and `pep=YES` with lift above 1.5 would be kept.

Rules that this dataset is known to produce at high confidence involve `current_act=YES` (76 percent of customers) as a consequent, and at high lift involve `income` bins with `save_act` and `children=0` with `pep`.

Explanation

Part (a) is a confidence search bounded by support: WEKA starts at support 0.95, lowers it by 0.05 a cycle, and stops at the first level where at least 5 rules have confidence 0.8 or better, or at 0.2. Because `current_act=YES` holds for three quarters of the rows, almost any premise predicts it with confidence above 0.8, so the top confidence rules are trivial. Part (b) fixes that by ranking on lift: $\text{lift} = \frac{\text{confidence}}{\text{support}(Y)}$ divides by how common the consequent is, so a rule predicting `current_act=YES` with confidence 0.83 gets lift 1.1 and fails the 1.5 threshold, while a rule that raises the chance of a rare consequent by half or more passes. The lower support bound of 0.1 (60 rows) lets rarer but stronger patterns through.

Question 11

Problem Statement

■ Write in lab record

Generate association rule for the credit card promotion dataset using Apriori algorithm with the support range 40% to 100%, confidence as 10%, incremental decrease as 5% and generate 6 rules.

Solution

■ Write in lab record

The credit card promotion database is the 15-row table from Roiger and Geatz: `income-range`, `magazine-promotion`, `watch-promotion`, `life-insurance-promotion`, `credit-card-insurance`, `sex` (all nominal) and `age` (numeric). It is small enough to compute for real.

Steps

1. Type the file below as `credit-card-promotion.arff` and Open file it.
2. Remove `age` (attribute 7): tick it and click Remove, because Apriori cannot use a numeric attribute. (Discretize would also work but the question does not ask for age rules.)

3. Associate, Choose Apriori, set `lowerBoundMinSupport` 0.4, `upperBoundMinSupport` 1.0, `delta` 0.05, `metricType` Confidence, `minMetric` 0.1, `numRules` 6, `outputItemSets` True. OK, Start.
4. Check the Scheme line: `weka.associations.Apriori -N 6 -T 0 -C 0.1 -D 0.05 -U 1.0 -M 0.4 -S -1.0 -c -1`.
5. Reproduce with `python3 apriori.py credit-card-promotion.arff -R 7 -M 0.4 -C 0.1 -N 6 -I`.

Program

credit-card-promotion.arff

```
1 % credit-card-promotion.arff - the 15-row credit card promotion database
2 % (Roiger and Geatz, Data Mining: A Tutorial-Based Primer, Table 2.3)
3 @relation credit-card-promotion
4
5 @attribute income-range {20-30K, 30-40K, 40-50K, 50-60K}
6 @attribute magazine-promotion {yes, no}
7 @attribute watch-promotion {yes, no}
8 @attribute life-insurance-promotion {yes, no}
9 @attribute credit-card-insurance {yes, no}
10 @attribute sex {male, female}
11 @attribute age numeric
12
13 @data
14 40-50K,yes,no,no,no,male,45
15 30-40K,yes,yes,yes,no,female,40
16 40-50K,no,no,no,no,male,42
17 30-40K,yes,yes,yes,yes,male,43
18 50-60K,yes,no,yes,no,female,38
19 20-30K,no,no,no,no,female,55
20 30-40K,yes,no,yes,yes,male,35
21 20-30K,no,yes,no,no,male,27
22 30-40K,yes,no,no,no,male,43
23 30-40K,yes,yes,yes,no,female,41
24 40-50K,no,yes,yes,no,female,43
25 20-30K,no,yes,yes,no,male,29
26 50-60K,yes,yes,yes,no,female,39
27 40-50K,no,yes,no,no,male,55
28 20-30K,no,no,yes,yes,female,19
```

Output

Computed with `apriori.py` (same format as WEKA's Associate output; the Run information lists 15 instances and the 6 remaining attributes):

```
1 Minimum support: 0.4 (6 instances)
2 Minimum metric <confidence>: 0.1
3 Number of cycles performed: 12
4
5 Generated sets of large itemsets:
6
7 Size of set of large itemsets L(1): 9
8
9 Large Itemsets L(1):
10 magazine-promotion=yes 8
11 magazine-promotion=no 7
12 watch-promotion=yes 8
13 watch-promotion=no 7
14 life-insurance-promotion=yes 9
15 life-insurance-promotion=no 6
16 credit-card-insurance=no 12
17 sex=male 8
18 sex=female 7
19
20 Size of set of large itemsets L(2): 10
21
22 Large Itemsets L(2):
23 magazine-promotion=yes life-insurance-promotion=yes 6
24 magazine-promotion=yes credit-card-insurance=no 6
25 magazine-promotion=no credit-card-insurance=no 6
26 watch-promotion=yes life-insurance-promotion=yes 6
27 watch-promotion=yes credit-card-insurance=no 7
28 life-insurance-promotion=yes credit-card-insurance=no 6
29 life-insurance-promotion=yes sex=female 6
30 life-insurance-promotion=no credit-card-insurance=no 6
31 credit-card-insurance=no sex=male 6
32 credit-card-insurance=no sex=female 6
33
34 Best rules found:
35
36 1. life-insurance-promotion=no 6  $\implies$  credit-card-insurance=no 6 <conf:(1
37 2. watch-promotion=yes 8  $\implies$  credit-card-insurance=no 7 <conf:(0.88)> li
38 3. magazine-promotion=no 7  $\implies$  credit-card-insurance=no 6 <conf:(0.86)>
39 4. sex=female 7  $\implies$  life-insurance-promotion=yes 6 <conf:(0.86)> lift:(1
40 5. sex=female 7  $\implies$  credit-card-insurance=no 6 <conf:(0.86)> lift:(1.07)
41 6. magazine-promotion=yes 8  $\implies$  life-insurance-promotion=yes 6 <conf:(0.
```

Explanation

With `minMetric` 0.1 the confidence filter is almost switched off, so the search is governed by support alone: WEKA lowers support from 0.95 until 6 rules exist, which happens at 0.4 (6 of 15 rows) after 12 cycles. At that level 9 single items and 10 pairs are large and no triple is, so every rule has one item on each side. The top rule, `life-insurance-promotion=no  $\implies$  credit-card-insurance=no`, has confidence $\frac{6}{6} = 1$ but lift only $\frac{1}{12/15} = 1.25$, because 12 of 15 customers have no credit card insurance anyway. The more useful rule is `sex=female  $\implies$  life-insurance-promotion=yes` (6 of 7, confidence 0.86, lift 1.43): women in this table took the life insurance promotion far more often than the base rate of 9 in 15. The exercise shows why a low confidence threshold is harmless when support is high, but the rules must be read with lift.

Viva Questions

✗ Do not copy. Read for understanding and the viva

Q: What does FP-Growth avoid that Apriori must do? **A:** Candidate generation and repeated scans; it builds a compressed tree in two scans and mines it recursively.

Q: What is a conditional pattern base? **A:** The set of prefix paths (with their counts) that lead to the nodes of one item in the FP-tree.

Q: Why sort items by frequency before inserting a transaction? **A:** So that common items sit near the root and many transactions share the same prefix, which keeps the tree small.

Q: In WEKA's FPGrowth what does `positiveIndex` 2 mean? **A:** For a binary attribute, only its second value is treated as the item being present.

Q: Why did part (a) of the Bank question give trivial rules? **A:** Confidence rewards common consequents; `current_act=YES` is true for most rows so nearly any premise predicts it.

Q: How is "minimum lift 150%" entered in WEKA? **A:** `metricType` Lift and `minMetric` 1.5.

Q: In the credit card run, why did the support stop at 0.4 and not lower? **A:**

`LowerBoundMinSupport` was 0.4 and six rules already passed there, so the loop ended.

Common Mistakes

✗ Do not copy. Read for understanding and the viva

- Running FPGrowth on a nominal file without `NominalToBinary` and then reporting that the first value of every two-valued attribute never appears in a rule.
- Entering 80 instead of 0.8 for `minMetric`, or 20 instead of 0.2 for the support bound; WEKA wants fractions.
- Leaving `metricType` on Confidence for the lift question and reading the `lift:` column by eye instead of letting WEKA rank by it.
- Forgetting to remove or discretise `age` in the credit card file, so Start stays disabled.

Formula Sheet

✗ Do not copy. Read for understanding and the viva

Association rules

For a rule $X \Rightarrow Y$ over N transactions:

$$\text{support}(X \Rightarrow Y) = \frac{|X \cup Y|}{N}, \quad \text{confidence}(X \Rightarrow Y) = \frac{|X \cup Y|}{|X|}, \quad \text{lift}(X \Rightarrow Y) = \frac{\text{confidence}(X \Rightarrow Y)}{\text{support}(Y)}$$

WEKA's Apriori starts at the upper bound of minimum support and lowers it by the delta each pass until the requested number of rules is found or the lower bound is reached.

Entropy, information gain and Gini

For a set S with class proportions $p_1, \dots, p_c$:

$$H(S) = - \sum_{i=1}^c p_i \log_2 p_i, \quad \text{Gain}(S, A) = H(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} H(S_v), \quad \text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2$$

ID3 splits on the attribute with the highest gain; J48 (C4.5) uses the gain ratio

$\text{Gain}(S, A) / \text{SplitInfo}(S, A)$ where $\text{SplitInfo}(S, A) = - \sum_v \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$.

Classifier evaluation

From the confusion matrix with true positives TP , false positives FP , false negatives FN , true negatives TN :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Kappa compares observed agreement p_o (accuracy) with the agreement expected by chance p_e :

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \quad p_e = \sum_i \frac{(\text{row}_i \text{ total}) (\text{column}_i \text{ total})}{N^2}$$

The ROC curve plots true positive rate $TP / (TP + FN)$ against false positive rate $FP / (FP + TN)$; the area under it (AUC) is 0.5 for guessing and 1.0 for a perfect classifier.

Naive Bayes and k-nearest neighbour

$$P(C | x_1, \dots, x_n) \propto P(C) \prod_{i=1}^n P(x_i | C)$$

k-NN assigns the majority class among the k nearest training records under Euclidean distance

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

after normalising each attribute to $[0, 1]$ with $x' = (x - x_{\min}) / (x_{\max} - x_{\min})$.

Linear regression

$$\hat{y} = \beta_0 + \beta_1 x, \quad \beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

WEKA reports the correlation coefficient, mean absolute error and root mean squared error $\sqrt{\frac{1}{N} \sum (y_i - \hat{y}_i)^2}$.

Clustering

k-means minimises the within-cluster sum of squared errors over clusters $C_1, \dots, C_k$ with centroids μ_j :

$$SSE = \sum_{j=1}^k \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mu_j\|^2, \quad \mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

Hierarchical (agglomerative) clustering merges the two closest clusters each step; linkage defines closeness: single $\min d(a, b)$, complete $\max d(a, b)$, average $\frac{1}{|A||B|} \sum d(a, b)$.

DBSCAN calls a point a core point when at least minPts points lie within radius ϵ ; clusters grow from core points, and points reachable from none are noise.

Session Summary

■ Write in lab record

- Question 9: `fpgrowth.py` written; FP-tree, header table and frequent patterns produced for `contact-lenses.arff` (11, 21, 6 itemsets, same 10 rules as Apriori) and for the hand-written `test.arff`; WEKA FPGrowth steps with NominalToBinary recorded
- Question 10: `Bank.arff` parameter table mapped to `lowerBoundMinSupport`, `upperBoundMinSupport`, `delta`, `metricType`, `minMetric`, `numRules` for both runs; Scheme lines and expected output shape recorded
- Question 11: `credit-card-promotion.arff` typed (15 rows), age removed, Apriori at support 0.4 to 1.0, delta 0.05, confidence 0.1, 6 rules computed and interpreted with lift

[✎ Edit this page](#)

< Previous
Session 3

Next >
Session 5