
 CMU SCS

15-826: Multimedia Databases and Data Mining


Lecture #26: Data Mining - trees + assoc.
rules
C. Faloutsos

 CMU SCS

Must-read Material

- Agrawal, R., S. Ghosh, et al. (Aug. 23-27, 1992). An Interval Classifier for Database Mining Applications. VLDB Conf. Proc., Vancouver, BC, Canada.
- Han + Kamber, chapter 6.1-4 (1st Edition); or Chapter 5.1-4 (2nd Edition)
- Mehta, M., R. Agrawal, et al. (1996). SLIQ: A Fast Scalable Classifier for Data Mining. EDBT, Avignon, France.

15-826 Copyright: C. Faloutsos (2009) 2

 CMU SCS

Outline

Goal: 'Find similar / interesting things'

- Intro to DB
- Indexing - similarity search
-  • Data Mining

15-826 Copyright: C. Faloutsos (2009) 3

CMU SCS

Data Mining - Detailed outline

- data warehouses; data cubes; OLAP
- classifiers
- association rules

15-826 Copyright: C. Faloutsos (2009) 4

CMU SCS

Classifiers - outline

- Case study: 'Interval Classifier' ('IC')
- recent developments and variations

15-826 Copyright: C. Faloutsos (2009) 5

CMU SCS

Tree Classifiers

Database issues: how about huge (training) datasets?

Case study: Interval Classifier [Agrawal+92]
Goal: build a classifier (eg., for target mailing)
Differences from AI/ML:

- retrieval efficiency (could use DBMS indices!)
- generation efficiency (large training dataset)

15-826 Copyright: C. Faloutsos (2009) 6

CMU SCS

Tree Classifiers - 'IC'

Proposed method: use classification tree, but

- split a range (= num. attribute) into k sub-ranges, as opposed to just 2
- do 'dynamic pruning' (ie., don't expand a node that is fairly homogeneous)

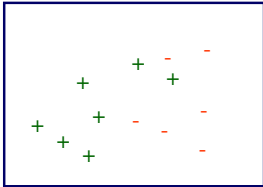
15-826 Copyright: C. Faloutsos (2009) 7

CMU SCS

Decision trees

- Pictorially, we have

num. attr#2
(eg., chol-level)



num. attr#1 (eg., 'age')

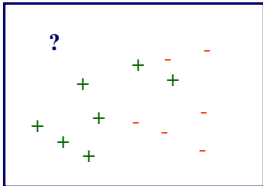
15-826 Copyright: C. Faloutsos (2009) 8

CMU SCS

Decision trees

- and we want to label '?'

num. attr#2
(eg., chol-level)



num. attr#1 (eg., 'age')

15-826 Copyright: C. Faloutsos (2009) 9

CMU SCS

Decision trees

- so we build a decision tree:

num. attr#2
(eg., chol-level)

40

?

+	+ - -
+ +	+ +
+ +	- - -
+ +	- - -

50
num. attr#1 (eg., 'age')

15-826 Copyright: C. Faloutsos (2009) 10

CMU SCS

Tree Classifiers - 'IC'

Sketch of algorithm

make-tree():

- partition set in groups by label
- obtain histograms for each group and each attribute
- Apply goodness function to pick winning attribute A'
- Partition the domain of A' into "strong" and "weak" intervals
- For each "strong" interval: assign it to majority label
- For each "weak" interval: make-tree()

15-826 Copyright: C. Faloutsos (2009) 11

CMU SCS

Tree Classifiers - 'IC'

- "strong" interval: = homogeneous (or close enough)
- k: depends on # of distinct values
- 'interval' = 'range' for a continuous attribute;
- 'interval' = 'value' for a categorical one
- histograms: equi-width

Classification accuracy: comparable to standard algorithms (ID3, C4)

15-826 Copyright: C. Faloutsos (2009) 12

CMU SCS

Tree Classifiers - 'IC'

Conclusions: compared to standard algorithms (ID3, C4):

- Faster, because of
 - *k*-way splitting and
 - dynamic pruning
- comparable classification accuracy

15-826 Copyright: C. Faloutsos (2009) 13

CMU SCS

Classifiers - outline

- Case study: 'Interval Classifier' ('IC')
- ➔ • newer developments and variations

15-826 Copyright: C. Faloutsos (2009) 14


CMU SCS

Classifiers - newer methods

- SLIQ [Mehta+96]
- SPRINT [Shafer+, vldb96]
- PUBLIC [Rastogi+Shim, vldb98]
- RainForest [Gehrke+, 2000]

Goal: how to make build decision trees, when the training set does not fit in memory

15-826 Copyright: C. Faloutsos (2009) 15

 CMU SCS


Classifiers - newer methods

Goal: how to make build decision trees, when the training set does not fit in memory

SLIQ: use vertical partitioning (att-value, record-id) for each attribute; keep the (label, record-id) list in main memory

SPRINT: like SLIQ, but attach 'label' on each attribute list: (attr-value, label, record-id)

15-826 Copyright: C. Faloutsos (2009) 16


 CMU SCS

Classifiers - conclusions

Recent methods: try to improve scalability/speed with

- 'dynamic' pruning
- elaborate file structures / data placement
- parallelism

15-826 Copyright: C. Faloutsos (2009) 17

 CMU SCS

Data Mining - Detailed outline

- data warehouses; data cubes; OLAP
- classifiers
- ➔ association rules

15-826 Copyright: C. Faloutsos (2009) 18

CMU SCS

Association rules - outline

➔ Main idea [Agrawal+SIGMOD93]

- performance improvements
- Variations / Applications
- Follow-up concepts

15-826 Copyright: C. Faloutsos (2009) 19

CMU SCS

Association rules - idea

[Agrawal+SIGMOD93]

- Consider 'market basket' case:
 - (milk, bread)
 - (milk)
 - (milk, chocolate)
 - (milk, bread)
- Find 'interesting things', eg., rules of the form:
 - milk, bread -> chocolate | 90%

15-826 Copyright: C. Faloutsos (2009) 20

CMU SCS

Association rules - idea

In general, for a given rule
 $I_j, I_k, \dots, I_m \rightarrow I_x | c$
 'c' = 'confidence' (how often people buy I_x , given that they have bought I_j, \dots, I_m)
 's' = support: how often people buy I_j, \dots, I_m, I_x

15-826 Copyright: C. Faloutsos (2009) 21

CMU SCS

Association rules - idea

Problem definition:

- given
 - a set of 'market baskets' (=binary matrix, of N rows/baskets and M columns/products)
 - min-support 's' and
 - min-confidence 'c'
- find
 - all the rules with higher support and confidence

15-826 Copyright: C. Faloutsos (2009) 22

CMU SCS

Association rules - idea

Closely related concept: "large itemset"

$I_j, I_k, \dots, I_m, I_x$
 is a 'large itemset', if it appears more than 'min-support' times

Observation: once we have a 'large itemset', we can find out the qualifying rules easily (how?)

Thus, let's focus on how to find 'large itemsets'

15-826 Copyright: C. Faloutsos (2009) 23

CMU SCS

Association rules - idea

Naive solution: scan database once; keep 2**|I| counters

Drawback?

Improvement?

15-826 Copyright: C. Faloutsos (2009) 24

CMU SCS

Association rules - idea

Naive solution: scan database once; keep $2^{*|I|}$ counters
 Drawback? 2^{*1000} is prohibitive...
 Improvement? scan the db $|I|$ times, looking for 1-, 2-, etc itemsets

Eg., for $|I|=3$ items only (A, B, C), we have

15-826 Copyright: C. Faloutsos (2009) 25

CMU SCS

Association rules - idea

Ⓐ	Ⓑ	Ⓒ	
100	200	2	first pass
			min-sup:10

15-826 Copyright: C. Faloutsos (2009) 26

CMU SCS

Association rules - idea

	Ⓐ,Ⓑ	Ⓐ,Ⓒ	Ⓑ,Ⓒ	
Ⓐ	Ⓑ	Ⓒ		first pass
100	200	2		min-sup:10

15-826 Copyright: C. Faloutsos (2009) 27

CMU SCS

Association rules - idea

Anti-monotonicity property:
if an itemset fails to be 'large', so will every superset of it (hence all supersets can be pruned)

Sketch of the (famous!) 'a-priori' algorithm
Let $L(i-1)$ be the set of large itemsets with $i-1$ elements
Let $C(i)$ be the set of candidate itemsets (of size i)

15-826 Copyright: C. Faloutsos (2009) 28

CMU SCS

Association rules - idea

Compute $L(1)$, by scanning the database.
repeat, for $i=2,3,\dots$,
 'join' $L(i-1)$ with itself, to generate $C(i)$
 two itemset can be joined, if they agree on their first $i-2$ elements
 prune the itemsets of $C(i)$ (how?)
 scan the db, finding the counts of the $C(i)$ itemsets - set this to be $L(i)$
 unless $L(i)$ is empty, repeat the loop
(see example 6.1 in [Han+Kamber])

15-826 Copyright: C. Faloutsos (2009) 29

CMU SCS

Association rules - outline

- Main idea [Agrawal+SIGMOD93]
- ➡ performance improvements
- Variations / Applications
- Follow-up concepts

15-826 Copyright: C. Faloutsos (2009) 30

CMU SCS

Association rules - improvements

- Use the independence assumption, to second-guess large itemsets a few steps ahead
- eliminate 'market baskets', that don't contain any more large itemsets
- Partitioning (eg., for parallelism): find 'local large itemsets', and merge.
- Sampling
- report only 'maximal large itemsets' (dfn?)
- FP-tree (seems to be the fastest)

15-826 Copyright: C. Faloutsos (2009) 31

CMU SCS

Association rules - improvements

- FP-tree: no candidate itemset generation - only two passes over dataset
- Main idea: build a TRIE in main memory

Specifically:

- first pass, to find counts of each item - sort items in decreasing count order
- second pass: build the TRIE, and update its counts

(eg., let A,B, C, D be the items in frequency order:)

15-826 Copyright: C. Faloutsos (2009) 32

CMU SCS

Association rules - improvements

- eg., let A,B, C, D be the items in frequency order:)

32 records
10 of them have A
4 have AB
2 have AC
1 has C

15-826 Copyright: C. Faloutsos (2009) 33

CMU SCS

Association rules - improvements

- Traversing the TRIE, we can find the large itemsets (details: in [Han+Kamber, §6.2.4])
- Result: much faster than 'a-priori' (order of magnitude)

15-826 Copyright: C. Faloutsos (2009) 34

CMU SCS

Association rules - outline

- Main idea [Agrawal+SIGMOD93]
- performance improvements
- ➔ Variations / Applications
- Follow-up concepts

15-826 Copyright: C. Faloutsos (2009) 35

CMU SCS

Association rules - variations

1) Multi-level rules: given concept hierarchy

- 'bread', 'milk', 'butter' -> foods;
- 'aspirin', 'tylenol' -> pharmacy

look for rules across any level of the hierarchy, eg
 'aspirin' -> foods

(similarly, rules across dimensions, like 'product',
 'time', 'branch':
 'bread', '12noon', 'PGH-branch' -> 'milk')

15-826 Copyright: C. Faloutsos (2009) 36

CMU SCS

Association rules - variations

2) Sequential patterns:
 'car', 'now' -> 'tires', '2 months later'
 Also: given a stream of (time-stamped) events:
 A A B A C A B A C

find rules like
 B, A -> C
 [Manilla+KDD97]

15-826 Copyright: C. Faloutsos (2009) 37

CMU SCS

Association rules - variations

3) Spatial rules, eg:
 'house close to lake' -> 'expensive'

15-826 Copyright: C. Faloutsos (2009) 38

CMU SCS

Association rules - variations

4) Quantitative rules, eg:
 'age between 20 and 30', 'chol. level <150' ->
 'weight > 150lb'
 Ie., given **numerical** attributes, how to find rules?

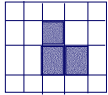
15-826 Copyright: C. Faloutsos (2009) 39

CMU SCS

Association rules - variations

4) Quantitative rules
Solution:

- bucketize the (numerical) attributes
- find (binary) rules
- stitch appropriate buckets together:

salary  age

15-826 Copyright: C. Faloutsos (2009) 40

CMU SCS

Association rules - outline

- Main idea [Agrawal+SIGMOD93]
- performance improvements
- Variations / Applications
- ➔ Follow-up concepts

15-826 Copyright: C. Faloutsos (2009) 41

CMU SCS

Association rules - follow-up concepts

Associations rules vs. correlation.
Motivation: if
 milk, bread
is a 'large itemset', does this means that there is a positive correlation between 'milk' and 'bread' sales?

15-826 Copyright: C. Faloutsos (2009) 42

CMU SCS

Association rules - follow-up concepts

What to do, then?

15-826 Copyright: C. Faloutsos (2009) 43

CMU SCS

Association rules - follow-up concepts

What to do, then?

A: report only pairs of items that are indeed correlated - ie, they pass the Chi-square test

The idea can be extended to 3-, 4- etc itemsets (but becomes more expensive to check)

See [Han+Kamber, §6.5], or [Brin+,SIGMOD97]

15-826 Copyright: C. Faloutsos (2009) 44

CMU SCS

Association rules - Conclusions

Association rules: a new tool to find patterns

- easy to understand its output
- fine-tuned algorithms exist

15-826 Copyright: C. Faloutsos (2009) 45
