Data Mining[1]
CS102 - April 25 2017

Approximate terminology, though there's some overlap, and terms are used sloppily or interchangeably:

- **Data(base) operations -** Executing specific operations or queries over data
- **Data mining -** Looking for patterns in data
- **Machine learning -** Using data to make inferences or predictions

Early data mining success stories:

- Victoria's Secret
- Walmart
- "Beer and diapers"

We'll cover data mining on *market-basket data*, with patterns being *frequent itemsets* and *association rules*.

- Examples of other types of data: *graphs* (of the node-and-link variety), *streams*, *text* (known as "text mining")
- Examples of other types of patterns: looking for *similar items*, looking for *structural patterns in large networks*, looking for *clusters* and/or *anomalies*

Market-Basket Data

Originated with retail data, specifically grocery stores, where a *market basket* is a set of items purchased together. More generally, market-basket data is any data where there's a fixed (possibly very large) set of *items*, and a (usually large) number of *transactions* consisting of one or more of the items. Examples:

- Items: groceries, Transaction: grocery cart
- Items: online goods, Transaction: (virtual) shopping cart
- Items: college courses, Transaction: student transcript
- Items: students, Transaction: party
- Items: movies, Transaction: person
- Items: symptoms, Transaction: patient
- Items: menu items, Transaction: customer
- Items: words, Transaction: document

Frequent Itemsets

Sets of items that occur together frequently in transactions

1. How large is a "set"?
2. What does "frequently" mean?

Look for sets containing at least *min-set-size* items, may also constrain *max-set-size*

Support: # transactions containing set / total # transactions
Look for sets with support > *support-threshold*

[1] Notes adopted from Jennifer Widom's CS102 offering in 2016.

Example

T1: beer, eggs, milk
T2: beer, diapers, milk
T3: chips, eggs
T4: eggs, milk
T5: beer, chips, diapers, milk

min-set-size = 2, *support-threshold* = 0.3
Frequent itemsets?
Answer: *beer/milk, beer/diapers, diapers/milk, eggs/milk, beer/diapers/milk*

Computing Frequent Itemsets Using Python

File Shop.csv with tid,item pairs

```
import csv
transactions={} # dictionary from TID to list of items
items={} # dictionary from item to list of TIDs
with open('Shop.csv', 'rU') as csvfile:
  data = csv.reader(csvfile)
  for row in data:
    if row[0] not in transactions: transactions[row[0]]=[row[1]]
    else: transactions[row[0]].append(row[1])
    if row[1] not in items: items[row[1]]=[row[0]]
    else: items[row[1]].append(row[0])
numtransactions = len(transactions)

# compute all pairs of items, alphabetical
pairs = []
for i1 in items:
  for i2 in items:
    if i1<i2: pairs.append([i1,i2,0])

# append number of transactions containing each pair
for p in pairs:
  for t in transactions:
    if p[0] in transactions[t] and p[1] in transactions[t]: p[2] += 1

# compute frequent itemsets of two
frequent2 = []
for p in pairs:
  if float(p[2])/float(numtransactions) > 0.3: frequent2.append(p)
print 'FREQUENT ITEMSETS OF TWO:'
for f in frequent2: print '  ', f[0], f[1]

# compute all triples of items where first two are in frequent itemsets
```

```
# of two, alphabetical
triples = []
for f in frequent2:
  for i in items:
    if f[0] < i and f[1] < i: triples.append([f[0],f[1],i,0])

# append number of transactions containing each triple
for tr in triples:
  for t in transactions:
    if tr[0] in transactions[t] and tr[1] in transactions[t]
    and tr[2] in transactions[t]: tr[3] += 1

# compute frequent itemsets of three
frequent3 = []
for t in triples:
  if float(t[3])/float(numtransactions) > 0.3: frequent3.append(t)
print 'FREQUENT ITEMSETS OF THREE:'
for f in frequent3: print '  ', f[0], f[1], f[2]
```

Association Rules

Set1 → Set2: when Set1 occurs in a transaction, Set2 often occurs in the same transaction
Commonly limit to looking for rules where Set2 is a single item

1. How large is Set1?
2. What does “often” mean?

Look for sets Set1 containing at least *min-set-size* items, may also constrain *max-set-size*

Confidence: # transactions containing Set1 and Set2 / # transactions containing Set1
Look for sets with confidence > *confidence-threshold*

Still consider *Support:* # transactions containing Set1 / total # transactions
Look for sets with support > *support-threshold* (i.e., Set1 should be frequent itemset)

Example

Same transactions T1-T5 as above
min-set-size = 1, *max-set-size* = 1, *confidence-threshold* = 0.5, *support-threshold* = 0.5
Association rules?
Answer: *Beer → Diapers, Beer → Milk, Eggs → Milk, Milk → Beer*

Computing Association Rules Using Python

Homework!