Information Retrieval & Data Mining [COMP0084]

Introduction to machine learning & data mining – Part 1

Vasileios Lampos

Computer Science, UCL

@lampos

Preliminaries

- In this lecture:
  - Association rule mining (*data mining*) — Apriori algorithm
  - Introduction to machine learning — Part 1

- Useful additional reads
  - Chapters 2 and 4 of "*Web Data Mining*" by Bing Liu (2006)
  - Chapters 3 and 4 of "*The Elements of Statistical Learning*" by Hastie, Tibshirani, and Friedman (2008)
  - Chapter 5 of "*Speech and language processing*" by Jurafsky and Martin (2021)

- Some slides adapted from Bing Liu's course — cs.uic.edu/~liub/teach/cs583-fall-21/cs583.html

- Many slides were adapted from Prof. Emine Yilmaz's lectures in previous years

Data mining — Definition

- **Data mining** is the process of discovering (*mining*) useful patterns from or conducting inferences based on various types of *data* sources such as structured information repositories (e.g. databases), text, images, sound, video, and so on.
- **Multi-disciplinary**: machine learning (or AI more broadly), statistics, databases, information retrieval — *but the distinction between machine learning and data mining is becoming increasingly difficult, especially from an applications perspective.*
- Strong research community: Knowledge Discovery and Data Mining or **KDD** — kdd.org
- Why? Gaining knowledge from a database is not as simple issue database queries
- Applications include marketing, recommendations, scientific data analysis, and *any task involving large amounts of data*

- Today: a basic look into **Association rule mining** / **learning** — *perhaps the most important task proposed and studied by the data mining community*
- Introduced by Agrawal, Imielinski, and Swami in 1993 — dl.acm.org/doi/pdf/10.1145/170035.170072
- Applicable on categorical / discrete data (e.g. product categories, movies, songs)
- Initially used for market basket analysis to understand how products purchased by customers are related, e.g.

spaghetti → *basil* [**support** = 0.1%, **confidence** = 25%]

Association rule mining — Notation & definitions

market basket transactions

t_1 : {almonds, cashews, pistachios}
t_2 : {almonds, bananas}
$\ldots$
t_n : {cashews, oranges, pistachios}

- A set of all the m **items**, $I = \{i_1, i_2, \ldots, i_m\}$
  — e.g. "almonds" is an item
- A set of all the n **transactions**, $T = \{t_1, t_2, \ldots, t_n\}$
- A transaction t_i is a set of items, and hence $t_i \subseteq I$

Association rule mining — Notation & definitions

market basket transactions

t_1 : {almonds, cashews, pistachios}
t_2 : {almonds, bananas}
$\dots$
t_n : {cashews, oranges, pistachios}

- An **itemset** is a set of items
  – e.g. $X = \{\text{almonds, cashews}\}$
- A k-**itemset** is an itemset with k items
  – e.g. $X = \{\text{almonds, cashews, pistachios}\}$ is a 3-itemset
- A transaction t_i contains the set of items (**itemset**) $X \subseteq I$, if $X \subseteq t_i$
- An **association rule** between itemsets X, Y is an implication of the form:

$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \varnothing$$

Association rule mining — Support & confidence

- **Association rule** (a *pattern*): $X \rightarrow Y$
  - when X occurs, Y occurs with a certain *support* and *confidence*

- **support** $= \dfrac{(X \cup Y)\,.\,\text{count}}{n}$
  - probability that a transaction will contain both itemsets X and Y, $\Pr(X \cup Y)$
  - how many times X and Y appear together in all (n) transactions in T divided by n

Association rule mining — Support & confidence

- Association rule (a *pattern*): $X \rightarrow Y$
  – when X occurs, Y occurs with a certain *support* and *confidence*

- **support** $= \dfrac{(X \cup Y)\,.\,\text{count}}{n} \quad \sim \quad \Pr(X \cup Y)$

- **confidence** $= \dfrac{(X \cup Y)\,.\,\text{count}}{X\,.\,\text{count}}$
  – conditional probability that a transaction that contains X will also contain Y, $\Pr(Y \mid X)$
  – how many times a transaction that contains X also contains Y divided by the number of transactions that contain X

Association rule mining

- **Goal:** Find all association rules $(X \rightarrow Y)$ that satisfy a pre-specified (*by us!*) **minimum support** (also abbreviated as **`minsup`**) and **minimum confidence** (**`minconf`**)

- Key features
  - **Completeness**, i.e. find all rules
    Note that $X \rightarrow Y$ and $Y \rightarrow X$ are different rules. **Why?**
  - Mining with data on hard disk (*because it is not always feasible to load everything in memory*)

Association rule mining — An example

- Transactions
- Let's set
  — **`minsup`** = 30%, and
  — **`minconf`** = 80%

t_1 : {almonds, cashews, pistachios}
t_2 : {almonds, bananas}
t_3 : {apples, bananas}
t_4 : {almonds, bananas, cashews}
t_5 : {almonds, bananas, cashews, oranges, pistachios}
t_6 : {cashews, oranges, pistachios}
t_7 : {cashews, oranges, pistachios}

- **Frequent itemset** examples:
  — {almonds, cashews} with support 3/7 (> **`minsup`**)
  — {cashews, pistachios} with support 4/7
  — {cashews, oranges, pistachios} with support 3/7
- **Association rule candidates** from the above frequent itemsets
  — almonds → cashews with confidence 3/4 (< **`minconf`**, *rejected*)
  — pistachios → cashews with confidence 4/4 (> **`minconf`**, *accepted*)
  — {cashews, oranges} → pistachios with confidence 3/3 (*accepted*)

Association rule mining — Algorithms

- Large number of different association rule mining algorithms
- Different strategies, data structures, computational efficiency, memory requirements
- But their output can only be the same:
  – Given a transaction data set T, minsup, and minconf, the set of association rules in T is uniquely determined.
- Let's briefly look at a foundational algorithm for association rule mining: **Apriori**

Association rule mining — Apriori

- **Apriori** is perhaps the most popular algorithm in data mining
- "Apriori" probably because it uses "prior" knowledge of frequent itemsets
- Proposed by Agrawal and Srikant in 1994 — vldb.org/conf/1994/P487.pdf
- Same two steps (*that we've seen previously*)
  - find all the itemsets with a minimum support (frequent itemsets)
  - then use the frequent itemsets to generate association rules

Apriori — Identify frequent itemsets

- The key idea of Apriori is the downward closure property (also known as the “Apriori property”):
  — Any subset of a frequent itemset is also a frequent itemset
  = Any subset of an itemset whose support is ≥ **`minsup`** has also support that is ≥ **`minsup`**

- If the itemset {a, b, c, d} with 4 items is frequent, then the $\left(2^4 - 2\right)$ non-empty sub-itemsets will also be frequent. These are: {a}, {b}, {c}, {d}, {a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d}, {a, b, c}, {a, b, d}, {a, c, d}, and {b, c, d}.

- **Contraposition:** if an itemset is not frequent, then any of its supersets cannot be frequent

Apriori — The gist of the algorithm

- Apriori is an iterative algorithm
  - given a minimum support
  - find all frequent 1-itemsets (denoted by `F[1]` in the source code)
  - use those to find all frequent 2-itemsets, and so on
    > `C[2]` is a list of frequent 2-itemset candidates based on `F[1]`
    > `F[2]` $\subseteq$ `C[2]` is a list with the frequent 2-itemsets
  - in each iteration k of the algorithm only consider itemsets that contain some frequent $(k-1)$-itemset

Apriori — An important detail (*item ordering*)

- Items should be sorted according to a sorting scheme — i.e. lexicographic order
- This order will be used throughout the algorithm as it helps to reduce redundant passes on the data, e.g. the frequent itemset {a, b, c, d} is identical to the frequent itemsets {c, d, a, b} or {b, a, d, c} — we only need to deal with {a, b, c, d} once.

Apriori — Pseudocode of the algorithm (*part 1*)

```
% T: all the transactions, MINSUP: frequent itemset minimum support
function apriori(T, MINSUP):
  % C[1] count of 1-itemsets, n transactions in T
  C[1], n ← initial-pass(T)
  % F[1] is the set of frequent 1-itemsets
  F[1] ← {f | f in C[1] AND f.count/n ≥ MINSUP}
  for k = 2; F[k-1] ≠ Ø; k++:
    % use the (k-1)—itemsets to generate k-itemset candidates, C[k]
    C[k] ← generate-candidates(F[k-1])
    for each transaction t in T:
      for each candidate c in C[k]:
        if c is in t:
          c.count++
    F[k] ← {c in C[k] | c.count/n ≥ MINSUP}

  return F
```

Apriori — Candidate itemset generation

- The `generate-candidates` function takes the $(k-1)$-frequent itemsets, denoted by `F[k-1]` in the source code, and returns a superset of k-frequent itemset candidates, denoted by `C[k]`

- Two steps
  - **Join**: generate all possible candidate k-itemsets `C[k]` based on `F[k-1]`
  - **Prune**: remove those candidates in `C[k]` that cannot be frequent, i.e. if a candidate itemset has a subset of items that is not already identified as a frequent itemset it should be removed

Apriori — Pseudocode of the algorithm (*part 2*)

```
% using frequent (k-1)-itemsets generate frequent k-itemset candidates
function generate-candidates(F[k-1]):
  C[k] ← ∅
  for every f1, f2 in F[k-1] where:
    a = f1 - f2 AND                       % set difference
    b = f2 - f1 AND                       % set difference
    (a AND b) are both of size 1 AND      % f1 and f2 differ by 1 element
    a < b do:                             % lexicographic comparison
      c ← {f1,b}                          % frequent k-itemset candidate
      C[k] ← {C[k],c}
      for each (k-1)-subset s of c do:
        if s not in F[k-1]:
          delete c from C[k]              % pruning non-frequent candidates
  return C[k]
```

Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

Let’s use Apriori to identify all frequent itemsets with minimum support of 30%

Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
C[1]:{almonds:4/7, apples:1/7, bananas:4/7, cashews:5/7, oranges:3/7,
      pistachios:4/7}

F[1]:{almonds, bananas, cashews, oranges, pistachios}

C[2]:{ {almonds, bananas}:3/7,      {almonds, cashews}:3/7,
       {almonds, oranges}:1/7,      {almonds, pistachios}:2/7,
       {bananas, cashews}:2/7,      {bananas, oranges}:1/7,
       {bananas, pistachios}:1/7,   {cashews, oranges}:3/7,
       {cashews, pistachios}:4/7,   {oranges, pistachios}:3/7 }
```

Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
C[2]:{ {almonds, bananas}:3/7,      {almonds, cashews}:3/7,
       {almonds, oranges}:1/7,      {almonds, pistachios}:2/7,
       {bananas, cashews}:2/7,      {bananas, oranges}:1/7,
       {bananas, pistachios}:1/7,   {cashews, oranges}:3/7,
       {cashews, pistachios}:4/7,   {oranges, pistachios}:3/7 }

F[2]:{ {almonds, bananas}, {almonds, cashews}, {cashews, oranges},
       {cashews, pistachios}, {oranges, pistachios} }
```

Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

F[2]:{ {almonds, bananas}, {almonds, cashews}, {cashews, oranges},
{cashews, pistachios}, {oranges, pistachios} }

C[3]:{ ~~{almonds, bananas, cashews}:2/7,~~
~~{cashews, oranges, pistachios}:3/7~~ } *** Incorrect ***

C[3]:{ {cashews, oranges, pistachios}:3/7 }

entry {almonds, bananas, cashews} will be pruned because {bananas, cashews} is not in F[2]

F[3]:{ {cashews, oranges, pistachios} }

Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

Apriori identified the following *frequent* itemsets with a minimum support of 30%:

```
F[1]:{almonds:4/7, bananas:4/7, cashews:5/7, oranges:3/7, pistachios:4/7}

F[2]:{ {almonds, bananas}:3/7,     {almonds, cashews}:3/7,
       {cashews, oranges}:3/7,     {cashews, pistachios}:4/7,
       {oranges, pistachios}:3/7 }

F[3]:{ {cashews, oranges, pistachios}:3/7 }
```

Apriori — Generating association rules from frequent itemsets

- Frequent itemsets do not directly provide association rules
- For each frequent itemset F
  For each non-empty subset A of F (*no repetitions*)
  - $B = F - A$
  - $A \rightarrow B$ is an association rule if confidence $(A \rightarrow B) \geq$ **`minconf`**

  $$\text{support}\,(A \rightarrow B) = \text{support}\,(A \cup B) = \text{support}\,(F)$$

  $$\text{confidence}\,(A \rightarrow B) = \frac{\text{support}\,(A \cup B)}{\text{support}\,(A)}$$

Apriori — Generating association rules (*example*)

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

`minsup` = 30%, **`minconf`** = 80%, let's use `F[3]:{ {cashews, oranges, pistachios}:3/7 }`

```
A = {{cashews, oranges}, {cashews, pistachios}, {oranges, pistachios},
     {cashews}, {oranges}, {pistachios}}
```

<u>A → B</u>		
{cashews, oranges} → pistachios		confidence = 1
{cashews, pistachios} → oranges		confidence = 0.75
{oranges, pistachios} → cashews		confidence = 1
cashews → {oranges, pistachios}		confidence = 0.6
oranges → {cashews, pistachios}		confidence = 1
pistachios → {cashews, oranges}		confidence = 0.75

Apriori — Generating association rules (*example*)

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

minsup = 30%, **minconf** = 80%, let's use `F[3]:{ {cashews, oranges, pistachios}:3/7 }`

```
A = {{cashews, oranges}, {cashews, pistachios}, {oranges, pistachios},
     {cashews}, {oranges}, {pistachios}}
```

A → B		
{cashews, oranges} → pistachios		**confidence = 1**
{cashews, pistachios} → oranges		confidence = 0.75
{oranges, pistachios} → cashews		**confidence = 1**
cashews → {oranges, pistachios}		confidence = 0.6
oranges → {cashews, pistachios}		**confidence = 1**
pistachios → {cashews, oranges}		confidence = 0.75

Apriori — Generating association rules from frequent itemsets

- To obtain an association rule $A \rightarrow B$, we need to compute the quantities: support $(A \cup B)$ and support (A)
- This information has already been recorded during itemset generation. No need to access the raw transaction data any longer.
- Not as time consuming a frequent itemset generation, although there are efficient algorithms to generate association rules as well

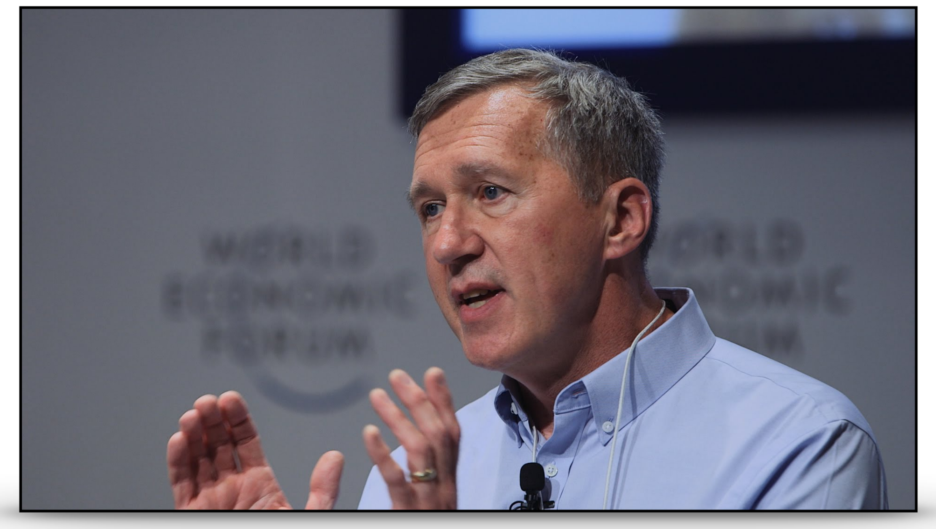
The (*very*) basics of machine learning

— definition
— supervised learning (*regression*, *classification*)
— unsupervised learning

Machine learning

- Arthur Samuel (IBM, 1959): "*Machine learning is the field of study that gives the computer the ability to learn* (**a task**) *without being explicitly programmed.*"
  — credited for coining the term
  — although we are still explicitly programming them to learn!

- Tom Mitchell (CMU, 1998): "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."
  — more formal definition
  — learning from experience (observations, data)



Notational conventions for this lecture

$x \in \mathbb{R}$ denotes a real-valued scalar

$\mathbf{x} \in \mathbb{R}^n$ denotes a real-value vector with n elements

$\mathbf{X} \in \mathbb{R}^{n \times m}$ denotes a real-valued matrix with n rows and m columns

$\mathbf{y} \in \mathbb{R}^m$ denotes m instances of a real valued response (or target) variable

$\hat{\mathbf{y}} \in \mathbb{R}^m$ denotes m inferences of a real valued response variable

$\|\mathbf{x}\|_k = \left(\sum_{i=1}^{n} |x_i|^k \right)^{\frac{1}{k}}$ denotes the L_k-norm of $\mathbf{x} \in \mathbb{R}^n$

Learning from experience

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  y_i denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample

$< \mathbf{x}_1, y_1 >$
$< \mathbf{x}_2, y_2 >$
$\dots$
$< \mathbf{x}_n, y_n >$

→ training → model function $f(\cdot)$

$< \mathbf{x}_{n+1}, \hat{y}_{n+1} = f\left(\mathbf{x}_{n+1}\right) >$
$\dots$
$< \mathbf{x}_{n+m}, \hat{y}_{n+m} = f\left(\mathbf{x}_{n+m}\right) >$

→ testing → loss / error function $\mathscr{L}\left(\hat{\mathbf{y}}, \mathbf{y}\right)$