Data Science Research Software

for Experiential Learning

Michael Hahsler

OIT/EMIS, SMU

February 5, 2020

1. Motivation

2. Example: Reinforcement Learning

3. Example: Clustering Association Rules

Section 1

Motivation

Teaching Portfolio

Data Science

- EMIS/DS 1300: A Practical Introduction to Data Science
- EMIS 2360: Engineering Economy
- EMIS 3309: Information Engineering
- EMIS 5/7361 Computer Simulation Techniques
- EMIS/CSE 5/7331: Data Mining
- EMIS/CSE 8331: Advanced Data Mining
- CSE 8091: Advanced Scientific Computing with R

Computer Science

- CSE 1341: Principles of Computer Science
- CSE 1342: Programming Concepts
- CSE 5/7337: Information Retrieval and Web Search
- CSE 5/7342: Concepts of Language Theory and Their Applications
- CSE 7343: Operating Systems and System Software

Data Science Research

SMU | Bobby B. Lyle School of Engineering

Goal: To apply, implement and improve state-of the art techniques for knowledge discovery from large scale noisy data.

Current Focus Areas:

- **Combinatorial optimization** for sequencing and ordering problems with applications for visualization and scheduling.
- **Data stream mining** with applications to hurricane intensity prediction, simulation data analytics for earth quake induced liquefaction , metagenomics and cybersecurity.
- Apply **reinforcement learning** and predictive modeling to develop optimal policies. Applications are type-2 diabetes decisions based on electronic health care records.

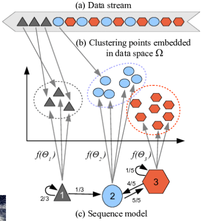
Collaborators: 8 SMU faculty, 15 students, 7 collaborators

Supported by

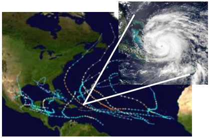
NIH | National Human Genome Research Institute | NSF | T-System | UT Southwestern Medical Center | Net-Centric I/UCRC

Simulation Data Analytics

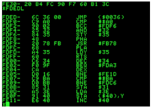
Massive-scale Sequence Modeling & Data Stream Mining



Meteorology



Cybersecurity



Metagenomics

Health Care Analytics

http://michael.hahsler.net

Research Software Development $+$ Teaching $=$ Experiential Learning?

Michael Hahsler

Home Teaching Research Publications Talks Software Lab CV

I am the lead developer and maintainer of several extension packages for the R software environment for statistical computing and graphics. R has been consistently voted one of the most important tools for data mining and analytics and being able to work with R is one of the highest paying analytics skills.

Development versions of our software are available on GitHub.

- Association Rule Mining
  - **arules**: a package for mining association rules and frequent itemsets. [intro]
    CRAN 1.6-4 - 5 months ago | rdocumentation 1.6-4 | GitHub last commit november 2019 | downloads 1.4M
  - **arulesViz**: A package for visualizing association rules based on package arules. [intro]
    CRAN 1.3-3 - 8 months ago | rdocumentation 1.3-3 | GitHub last commit september 2019 | downloads 717K
  - **arulesSequences**: Add-on package to handle and mine frequent sequences (lead developer: Christian Buchta).
    CRAN 0.2-22 - 9 months ago | rdocumentation 0.2-22 | downloads 91K
  - **arulesCBA**: Add-on package for classification based on association rules (lead developer: Ian Johnson).
    CRAN 1.1.6 - 22 days ago | rdocumentation 1.1.6 | GitHub last commit january | downloads 36K
  - **arulesNBMiner**: Add-on package to mine NB-frequent itemsets (see paper) and NB-precise rules.
    CRAN 0.1-5 - 5 years ago | rdocumentation 0.1-5 | GitHub last commit january 2018 | downloads 37K
- Bioinformatics
  - **rRDP**: Seamlessly interfaces the Ribosomal Database Project (RDP) classifier (version 2.9) which implements a Naive Bayesian Classifier (NBC) for biological sequences. [intro]
    in Bioc 5 years | rank 678 / 1823
  - **rMSA**: Interface for Popular Multiple Sequence Alignment Tools like ClustalW, MAFFT, MUSCLE and Kalign. [intro]
    GitHub last commit june 2017
  - **rBLAST**: Interfaces the Basic Local Alignment Search Tool (BLAST) to search genetic sequence databases with the Bioconductor infrastructure.
    GitHub last commit june 2019
  - **QuasiAlign**: Efficient sequence alignment using alignment-free methods. [Prototype at R-Forge]
- Combinatorial Optimization
  - **dbscan**: A fast reimplementation of several density-based algorithms of the DBSCAN family for spatial data. Includes the DBSCAN (density-based spatial clustering of applications with noise) and OPTICS (ordering points to identify the clustering structure) clustering algorithms and the LOF (local outlier factor) algorithm. The implementations use the k-d tree data structure (from library ANN) for faster k-nearest neighbor search. An R interface to fast kNN and fixed-radius NN search is also provided.
    CRAN 1.1-5 - 3 months ago | rdocumentation 1.1-5 | GitHub last commit october 2019 | downloads 374K
  - **seriation**: Infrastructure for seriation with an implementation of several seriation/sequencing techniques to reorder matrices, dissimilarity matrices, and dendrograms. [intro]
    CRAN 1.2-8 - 5 months ago | rdocumentation 1.2-8 | GitHub last commit august 2019 | downloads 956K
  - **TSP**: Basic infrastructure and some algorithms for the traveling salesperson problem (TSP). The package provides some simple algorithms and an interface to Concorde, currently the fastest TSP solver for the Traveling Salesperson Problem. [intro]
    CRAN 1.1-8 - 7 days ago | rdocumentation 1.1-8 | GitHub last commit january | downloads 911K

→ Research Software by Michael Hahsler

Section 2

Example: Reinforcement Learning

Reinforcement Learning

"Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment." (Kaelbling, Littman and Moore 1996)

It is related to optimal control (Bertsekas 1995) and adaptive control (Burghes and Graham 1980).

Problem

During each step of interaction:

1. the agent receives as input some indication of the current state of the environment s
2. the agent chooses an action a
3. the action changes the state of the environment
4. the value of this state transition is communicated to the agent through a signal r

The objective is typically to choose actions to maximize some notion of cumulative reward.

- The environment is often modeled as a Markov Decision Process (MDP).
- The considered MDPs are typically
  - only approximately known, and
  - too large to be solved with dynamic programming.

Markov Decision Process

A Markov decision process (MDP) is a discrete time stochastic control process (Bellman 1957).

Components:

- a set of environment and agent states, S, and a set of actions, A
- transition probabilities $T(s' \mid s, a) = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$
- a immediate reward function $R(s, s', a)$
- a policy $\pi : S \times A \to [0, 1]$

Objective: Find the policy that maximizes the expected cumulative discounted reward for horizon H.

$$\mathrm{E}[R] = \mathrm{E}\left[\sum_{t=0}^{H} \gamma^t R_t \mid s_0 = s\right]$$

Value iteration (Bellman back-up)

$V_0^*(s) = 0 \; \forall s \in S$, For $i = 0, 1, \ldots, H-1$ and all $s \in S$ do

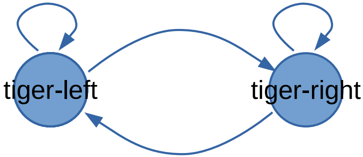
$$V_{i+1}^*(s) = \max_{a \in A} \gamma \sum_{s'} T(s' \mid s, a) \left[R(s, s', a) + V_i^*(s')\right]$$

Policy: Structural results show that it is sufficient to consider policies that are

- stationary (i.e., optimal action only depends on the last state), and
- deterministic (i.e., $\pi : A \times S \to \{0, 1\}$).

Example: Michael's Sleepy Tiger Problem

A sleepy tiger is put ***in front of a door*** *and treasure is put behind the other door. You can go to a door and open it or do nothing. Whenever you open a door, the tiger is put randomly in front of a door and treasure is put again behind the other door. You have 5 tires.*
What should you do?



Observable states

This is easy since you can see the state of the system.

Code - Michael's Sleepy Tiger Problem

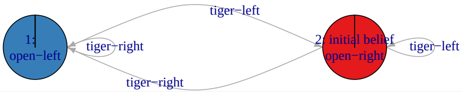
```r
library(pomdp)

Tiger_MDP <- MDP(
  name = "Michael's Sleepy Tiger Problem",
  states = c("tiger-left" , "tiger-right"),
  actions = c("open-left", "open-right", "do-nothing"),
  start = "tiger-left",

  transition_prob = list("open-left" =  "uniform", "open-right" = "uniform",
    "do-nothing" = "identity"),

  # the reward helper expects: action, start.state, end.state, observation, value
  reward = rbind(
    R_("do-nothing",                    v =    0),
    R_("open-left",  "tiger-left",  v = -100),
    R_("open-left",  "tiger-right", v =   10),
    R_("open-right", "tiger-left",  v =   10),
    R_("open-right", "tiger-right", v = -100)
  )
)
```

Code - Michael's Sleepy Tiger Problem

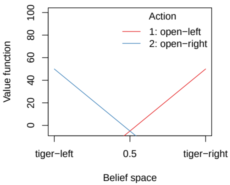
```
# do 5 epochs with no discounting
s <- solve_POMDP(Tiger_MDP, discount = 1, horizon = 5)
s
```

```
## Solved POMDP model: Michael's Sleepy Tiger Problem
##      solution method: grid
##      horizon: 5 (converged: FALSE)
##      total expected reward (for start probabilities): 50
```

```
# policy
plot(s, layout = igraph::layout.grid, edge.curved = TRUE, legend = FALSE)
```




```
plot_value_function(s, ylim = c(-5,100))
```



Useful Markov Decision Processes

Useful MDPs are more complicated:

- What if the tiger is not so sleepy and there is a chance that he catches you when you go to the door with the treasure?
- What if the tiger learns and is prepared the next time?
- What if you get more and more tired every try and thus get easier to catch?
- What if the tiger gets sleepy again if you decide to do nothing for a few tries?

Tony's Tiger Problem

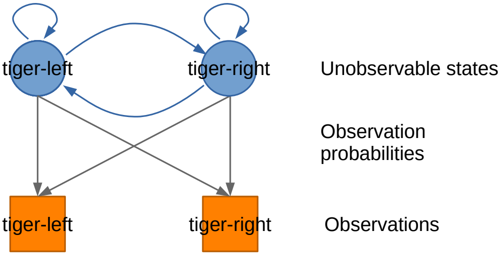
*A tiger is put with equal probability **behind** one of two doors, while treasure is put behind the other door. You can open a door or listen for tiger noises. However, **listening is not perfect.** When you open the door then the problem starts over. What do you do?*

(Cassandra, 1994)

$\rightarrow$ Why is this so much more difficult?

Partially Observable Markov Processes

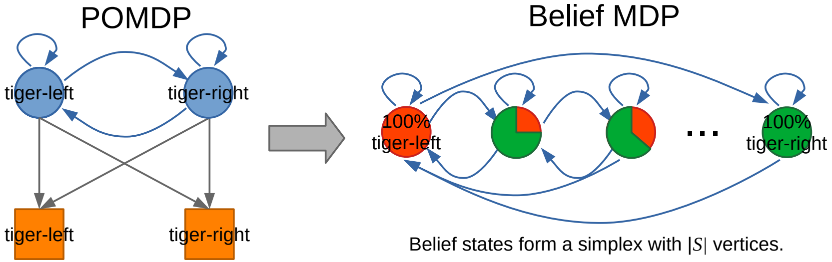
A generalization of a Markov decision process (MDP) where the agent cannot observe the state of the system directly, but receives a signal (an observation) at the end of every decision epoch (Åström 1965; Sondik 1971; Kaelbling, Littman and Cassandra 1998).



POMDPs add:

- A set of observations Ω
- Conditional observation probabilities $O(o \mid s', a) = Pr(o_{t+1} = o \mid s_{t+1} = s', a_t = a)$

Belief State MDP



Belief states form a simplex with $|S|$ vertices.

Transitions are reflected by a Belief Update

Let $b(s)$ be the probability that the system is in state s. Given an action a and an observation o we update the probability using:

$$b'(s') = \eta O(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s),$$

where η is a normalization factor.

Issue: The complexity of value iteration becomes $|S^2 \times A \times \Omega| \; H$ and Belief MDPs have an infinite state space.

Code - Tony's Tiger Problem

```
Tiger <- POMDP(
  name = "Tony's Tiger Problem",

  states = c("tiger-left", "tiger-right"),
  actions = c("open-left", "open-right", "listen"),
  observations = c("tiger-left", "tiger-right"),

  transition_prob = list("open-left" =  "uniform", "open-right" = "uniform",
    "listen" = "identity"),

  observation_prob = list("open-left" = "uniform", "open-right" = "uniform",
    "listen" = rbind(c(0.85, 0.15),
                     c(0.15, 0.85))),

  reward = rbind(
    R_("listen",                      v =   -1),
    R_("open-left",  "tiger-left",  v = -100),
    R_("open-left",  "tiger-right", v =   10),
    R_("open-right", "tiger-left",  v =   10),
    R_("open-right", "tiger-right", v = -100)
  )
)
```

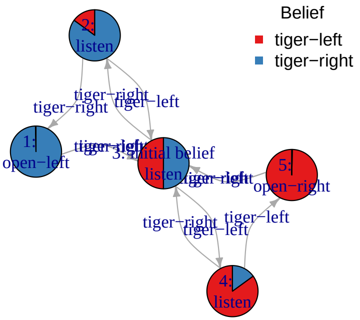
Code - Tony's Tiger Problem

```
# infinite horizon with discounting to get a converged policy
s <- solve_POMDP(Tiger, discount = .75, horizon = Inf)
s
```

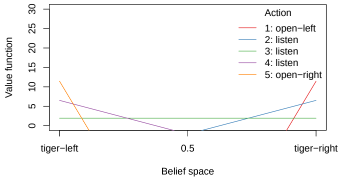
```
## Solved POMDP model: Tony's Tiger Problem
##      solution method: grid
##      horizon: Inf (converged: TRUE)
##      total expected reward (for start probabilities): 1.933439
```

Policy and Value Function

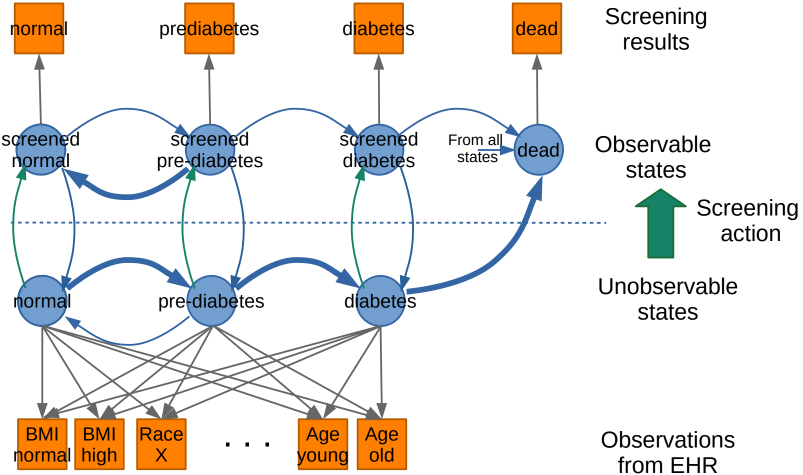
```
plot(s, edge.curved = TRUE)
```




```
plot_value_function(s, ylim = c(0,30))
```



A Real Example: Diabetes Screening



Note: This diagram is simplified (e.g., self-loops)

A Real Example: Diabetes Screening

- Actions: Screen, do nothing
- Estimate transition probabilities $T(s' \mid s, a)$
- Estimate reward structure $R(s', s, a, o)$
- **Observation space Ω and observation probabilities $O(o \mid s', a)$**

Observations:

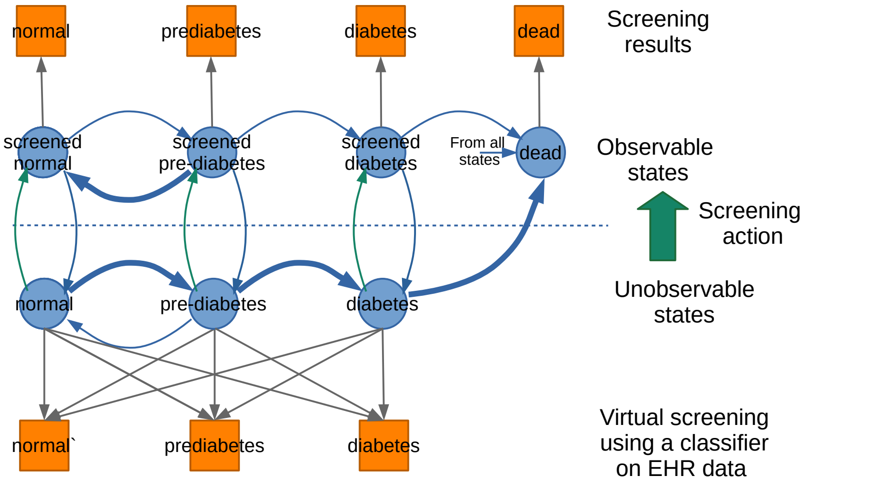
- Screening results are straight forward.
- Other observations can come from electronic health records (EHR)
  - 100s of signals (e.g., BMI, age, race, test results, medications)
  - missing values (e.g., a test was not performed or recorded)

Issues

- What signals from a set Z do we use?
- How do we construct a single observations? E.g., $\Omega = Z_1 \times Z_2 \times \cdots \times Z_{|Z|}$
- How do we estimate the observation probabilities?
- Can we solve a problem with complexity of the order of $|S^2 \times A \times \Omega|\ H$

Idea: Use a classifier to aggregate the signals into a small set of observations.

Classifier to Create Observations



Note: This diagram is simplified (e.g., missing self-loops)

Classifier to Create Observations

- Established algorithms and measures of predictive power (accuracy, kappa, AUC, etc.)
- Methods for handling missing values.
- Regularization and other feature selection methods.
- Observation matrix can be estimated from the confusion matrix. Normalize the reference columns (sometimes rows) to sum to 1.

```
##               Reference
## Prediction     Normal Prediabetes Diabetes
##   Normal          646         234       33
##   Prediabetes     530         507      202
##   Diabetes         59         109       93
##
## Overall Statistics
##                Accuracy : 0.516         95% CI : (0.496, 0.536)
##                   Kappa : 0.208
##
## Statistics by Class:
##                         Class: Normal Class: Prediabetes Class: Diabetes
## Sensitivity                     0.523              0.596          0.2835
## Specificity                     0.773              0.532          0.9194
## Balanced Accuracy               0.648              0.564          0.6015
```

Classifier to Create Observations

Reseach question: If we have several classifiers, which one is better for decision making?

Classifier 1:

Prediction/ Ref	Normal	Prediabetes	Diabetes
Normal	300	0	0
Prediabetes	0	300	0
Diabetes	0	0	300

Classifier 2:

Prediction/ Ref	Normal	Prediabetes	Diabetes
Normal	100	100	100
Prediabetes	100	100	100
Diabetes	100	100	100

Classifier to Create Observations

In general, choosing the best classifier depends on the structure of the decision problem (i.e., transition probabilities and reward structure).

Solution 1 - This is what we did so far

Solve the POMDP with each observation matrix (i.e., classifier) and pick the one that has the largest total expected reward.

Solution 2 - This is future research

Develop structural results that indicate, given properties of T and R, what properties O should have.

Simple Structural Result

Blackwell Dominance

Blackwell dominant observation matrices are better for any class of decision-maker.

$$O_2 \geq_B O_1$$

if and only if there exists a stochastic matrix R for which

$$O_1 = O_2 R$$

holds (Blackwell, 1951).

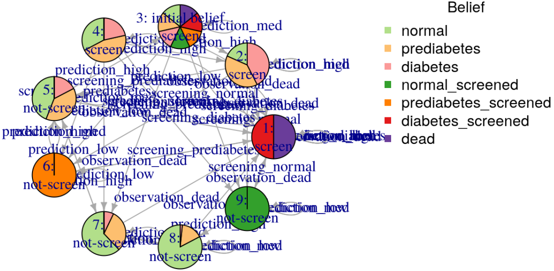
Application:

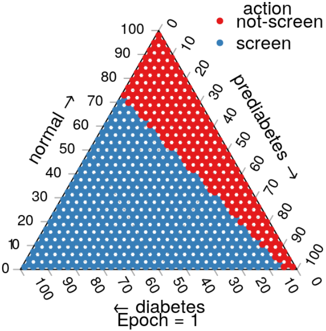
1. solve $R = O_2^{-1} O_1$.
2. check if R is stochastic.

Issues:

- If O_2 is not singular $\rightarrow$ use generalized (quasi) inverse.
- Only detects if O_2 is a noisy version of O_1, but classifiers produce different signals which cannot be compared using Blackwell dominance. $\rightarrow$ ϵ-dominance?

Example Policy and Belief Space





Results

Software

- New R package `pomdp` with infrastructure and solvers (pomdp-solve code contributed by Anthony Cassandra)
- See: https://github.com/farzad/pomdp

Experiential Learning Opportunity

- 1 student co-developer/co-author.
- Code development opportunities (e.g., visualization, solvers).
- Application opportunities.
- Theory development opportunities (structural results).
- Reinforcement learning focus for the next Advanced Data Mining course.

Research Output

- 4 Conference presentations
- 1 Journal publication under revision
- 2 Journal publications close to submission

Section 3

Example: Clustering Association Rules

Motivation

The aim of association analysis is to find *interesting* relationships between items (products, documents, people, genes, etc.) in transaction data.

Rule

A *rule* is defined as a probabilistic implication of the form

$$X \Rightarrow Y$$

where X and Y are itemsets.

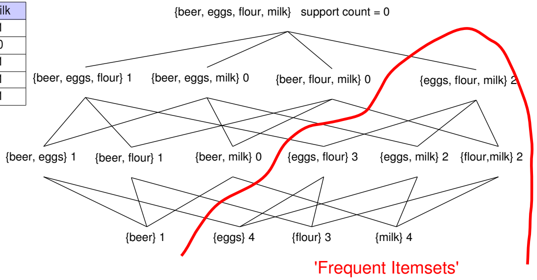
- **Support:** $\mathrm{supp}(X \Rightarrow Y) = \hat{P}(X, Y)$ is proportion of transactions which contain X and Y.
- **Confidence:** $\mathrm{conf}(X \Rightarrow Y) = \mathrm{supp}(X \cup Y)/\mathrm{supp}(X) = \hat{P}(Y \mid X)$
- **Association rule** $X \Rightarrow Y$ needs to satisfy:

$$\mathrm{supp}(X \cup Y) \geq \sigma \quad \text{and} \quad \mathrm{conf}(X \Rightarrow Y) \geq \delta$$

Minimum Support

Idea: Set a user-defined threshold for support since more frequent itemsets are typically more important. E.g., frequently purchased products generally generate more revenue.
Problem: For k items (products) we have $2^k - k - 1$ possible relationships between items. Example: $k = 100$ leads to more than 10^{30} possible associations.
Apriori property (Agrawal & Srikant 1994): The support of an itemset cannot increase by adding an item. Example: $\sigma = .4$ (support count ≥ 2)

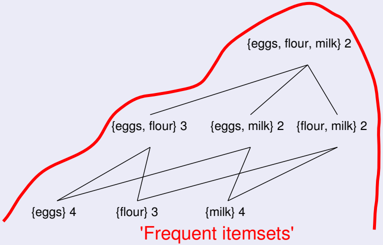
Transaction ID	beer	eggs	flour	milk
1	0	1	1	1
2	1	1	1	0
3	0	1	0	1
4	0	1	1	1
5	0	0	0	1



→ Basis for efficient algorithms (e.g., Apriori, Eclat).

Minimum Confidence

From the set of *frequent itemsets* all rules which satisfy the threshold for confidence $\mathrm{conf}(X \rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)} \geq \gamma$ are generated.



			Confidence
{eggs}	$\rightarrow$	{flour}	$3/4 = 0.75$
{flour}	$\rightarrow$	{eggs}	$3/3 = 1$
{eggs}	$\rightarrow$	{milk}	$2/4 = 0.5$
{milk}	$\rightarrow$	{eggs}	$2/4 = 0.5$
{flour}	$\rightarrow$	{milk}	$2/3 = 0.67$
{milk}	$\rightarrow$	{flour}	$2/4 = 0.5$
{eggs, flour}	$\rightarrow$	{milk}	$2/3 = 0.67$
{eggs, milk}	$\rightarrow$	{flour}	$2/2 = 1$
{flour, milk}	$\rightarrow$	{eggs}	$2/2 = 1$
{eggs}	$\rightarrow$	{flour, milk}	$2/4 = 0.5$
{flour}	$\rightarrow$	{eggs, milk}	$2/3 = 0.67$
{milk}	$\rightarrow$	{eggs, flour}	$2/4 = 0.5$

At $\gamma = 0.7$ the following set of rules is generated:

			Support	Confidence
{eggs}	$\rightarrow$	{flour}	$3/5 = 0.6$	$3/4 = 0.75$
{flour}	$\rightarrow$	{eggs}	$3/5 = 0.6$	$3/3 = 1$
{eggs, milk}	$\rightarrow$	{flour}	$2/5 = 0.4$	$2/2 = 1$
{flour, milk}	$\rightarrow$	{eggs}	$2/5 = 0.4$	$2/2 = 1$

Code: Mining Association Rules

```
library(arules)
data(Groceries)
rules <- apriori(Groceries, parameter = list(sup = 0.0005, conf = .5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   5e-04      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [164 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.05s].
## writing ... [42278 rule(s)] done [0.01s].
## creating S4 object  ... done [0.01s].
```

```
inspect(rules[1:2])
```

```
##     lhs                  rhs                 support confidence lift count
## [1] {salad dressing}  => {other vegetables} 0.00051 0.62       3.2  5
## [2] {rubbing alcohol} => {butter}           0.00051 0.50       9.0  5
```

Issue: Typically we find many rules and it is hard to make them actionable.

Grouping Association rules by Clustering

Group a set of m association rules

$$R = \{R_1, R_2, \ldots, R_m\}$$

into k subsets

$$S = \{S_1, S_2, \ldots, S_k\}$$

called clusters.

Such that rules in the same cluster are more similar to each other than to rules in different clusters.

Clustering Binary Vectors

A set of m association rules R can be represented as a set of n-dimensional vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$, where n is the total number of different items in the database.

k-Means Problem

Find a cluster assignment $S = \{S_1, S_2, \ldots, S_k\}$ which minimizes

$$WSS = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2,$$

where $\boldsymbol{\mu}_i$ is the cluster centroid (i.e., the mean of the points in S_i).

Advantage:
Fast and efficient heuristics.

Code: Clustering

```
cl <- kmeans(as(items(rules), "matrix"), centers = 20)
inspect(head(rules[cl$centers==1], n= 10))
```

```
##      lhs                           rhs                 support confidence lift count
## [1]  {chocolate,
##       female sanitary products} => {yogurt}            0.00051       0.56  4.0     5
## [2]  {tropical fruit,
##       female sanitary products} => {citrus fruit}      0.00061       0.75  9.1     6
## [3]  {jam,
##       soda}                     => {whole milk}        0.00081       0.62  2.4     8
## [4]  {dental care,
##       napkins}                  => {rolls/buns}        0.00051       0.62  3.4     5
## [5]  {tropical fruit,
##       instant coffee}           => {newspapers}        0.00061       0.50  6.3     6
## [6]  {whipped/sour cream,
##       instant coffee}           => {soda}              0.00071       0.54  3.1     7
## [7]  {whipped/sour cream,
##       instant coffee}           => {yogurt}            0.00081       0.62  4.4     8
## [8]  {sausage,
##       instant coffee}           => {soda}              0.00061       0.67  3.8     6
## [9]  {instant coffee,
##       bottled water}            => {other vegetables} 0.00061       0.60  3.1     6
## [10] {tropical fruit,
##       instant coffee}           => {soda}              0.00081       0.67  3.8     8
```

Suggested Solutions and Remaining Issues

Issues:

- Implies Euclidean distance, but matching 1s (same items in the rule) are much more important than matching 0s.
- Sparse, high-dimensional binary vectors.

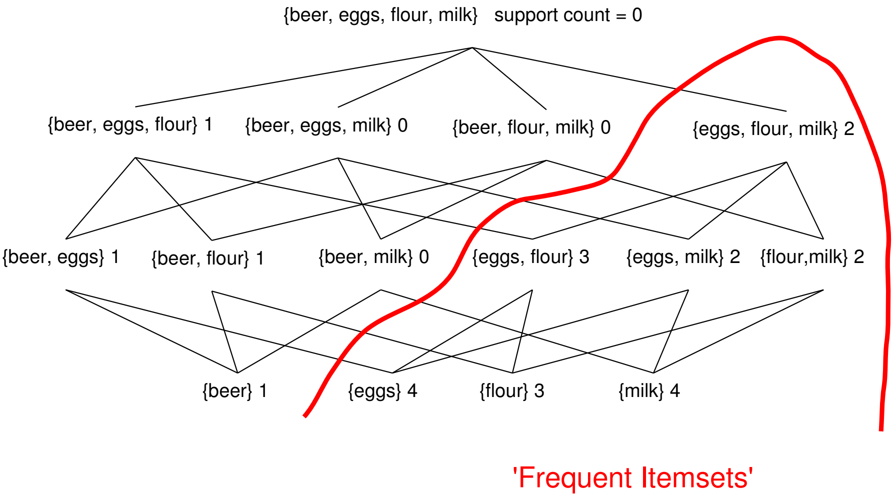
Proposed Solutions: Address issues with dimensionality and sparseness.

- Jaccard Index between binary vectors.
- Common covered transactions (Toivonen 1995 and Guha 1999).
- Use item hierarchies (Tuzhilin 2001).

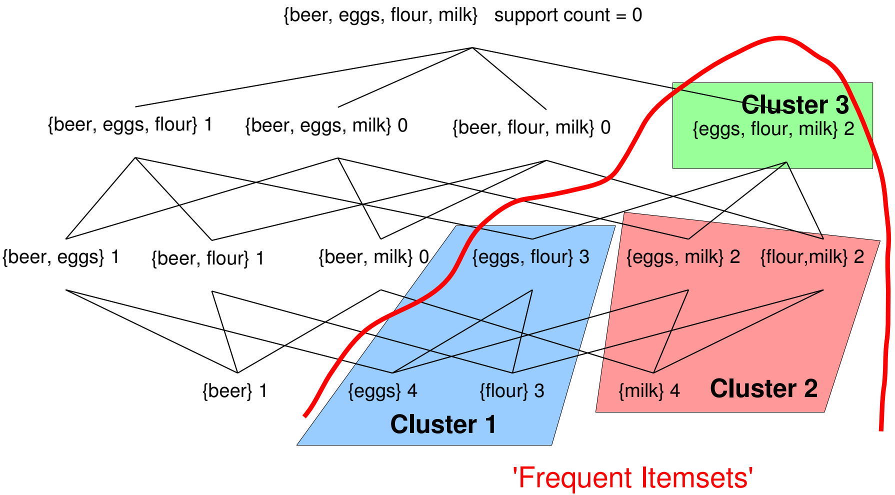
The following issues remain (Hahsler 2016):

- **Substitutes:** Grouping rules with substitutes, e.g., bread and beagles, is important.
- **Direction of association:** Most approaches do not differentiate between LHS and RHS.
- **Computational Complexity:** Distance matrix for a set of m rules requires $O(m^2)$ time and space.
- **Frequent itemset structure:** Clustering association rules will just rediscover subset structure of the frequent itemset lattice.

Frequent Itemset Structure



Frequent Itemset Structure



Grouping Rules Using Lift

Brin et al (1997) introduced the *lift* of an association rules as

$$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)\text{supp}(Y)} = \frac{\hat{P}(X,Y)}{\hat{P}(X)\hat{P}(Y)}$$

- Deviation from independence of LHS and RHS.
- 1 indicates independence.
- Larger lift values ($\gg 1$) indicate strong association.

Idea (Hahsler 2016)

Rules with a LHS that have strong associations with the same set of RHS (i.e., have a high lift value) are similar and thus should be grouped together.

Example

If $\{butter, cheese\} \rightarrow \{bread\}$ and $\{margarine, cheese\} \rightarrow \{bread\}$ have a similarly high lift, then the LHS should be grouped.
Note: butter and margarine are substitutes!

Clustering Rules using Lift

Let

$$R = \{\langle X_1, Y_1, \theta_1\rangle, \ldots, \langle X_i, Y_i, \theta_i\rangle, \ldots, \langle X_n, Y_n, \theta_n\rangle\},$$

- where X_i is the LHS,
- Y_i is the RHS, and
- θ_i is the lift value for the i-th rule.

Process

1. Find A, the set of unique LHS and C, the unique RHS.
2. Create a $A \times C$ matrix $M = (m_{ac})$.
3. Populate with $m_{ac} = \theta_i$ where X_i has index a in A and Y_i has index c in C.
4. Impute missing values (we use a neutral lift value of 1).
5. Cluster rules by grouping columns and/or rows in M.

Clustering Rules using Lift

We define now the distance between the LHS of two rules, X_i and X_j, as the Euclidean distance

$$d_{\text{Lift}}(X_i, X_j) = ||\mathbf{m_i} - \mathbf{m_j}||,$$

where $\mathbf{m_i}$ and $\mathbf{m_j}$ are the column vectors representing all rules with X_i and X_j, respectively.

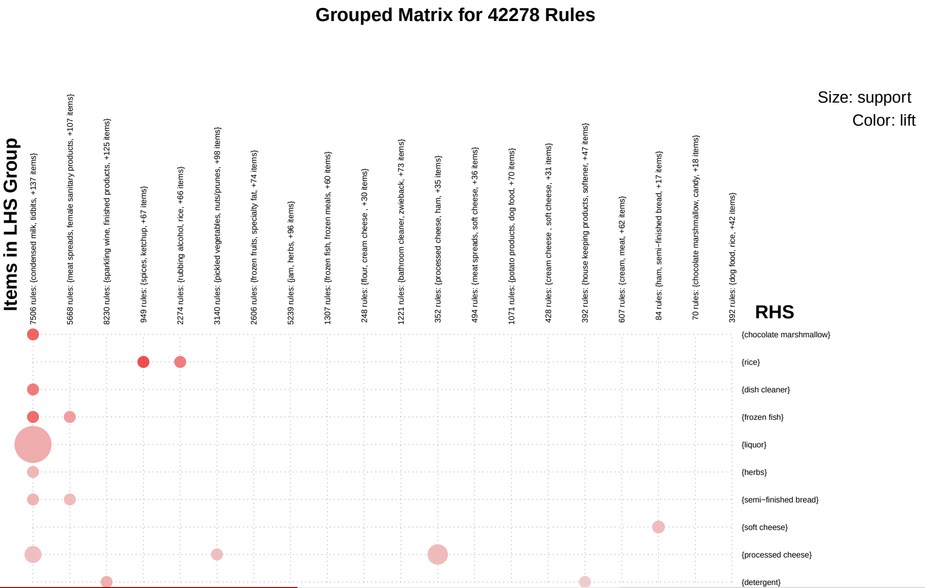
We can use now hierarchical clustering, k-medoids or k-means. For efficiency reasons we use a k-means heuristic to minimize the WSS

$$\text{argmin}_S \sum_{i=1}^{k} \sum_{\mathbf{m_j} \in \mathbf{S_i}} ||\mathbf{m_j} - \boldsymbol{\mu}_\mathbf{i}||^\mathbf{2},$$

Note: Most tools create rules with a single item in the RHS $\rightarrow$ no need for grouping.

Example

```r
library("arulesViz")
plot(rules, method="grouped", control = list(gp_labels = gpar(cex = .5)), interactive = TRUE)
```



Results

Software

- A family of arules 4 packages on CRAN
- 30k+ downloads per month
- 10+ packages by other researchers integrate with arules
- See: https://github.com/mhahsler/arules

Experiential Learning Opportunity

- 4 student co-developers/co-authors
- Use in data mining courses/machine learning courses worldwide.
- Current development of associative classifiers using Keras/TensorFlow.

Research Output

- 9 Journal publications
- 6 Conference proceedings
- 3 Book chapters

Thank you!