

The Long and the Short of It

summarising event sequences with serial episodes

Jilles Vreeken & Nikolaj Tatti



Question at hand

How can we discover
the key patterns from an event sequence?



Event sequences

Alphabet Ω $\{ a, b, c, d, \dots \}$

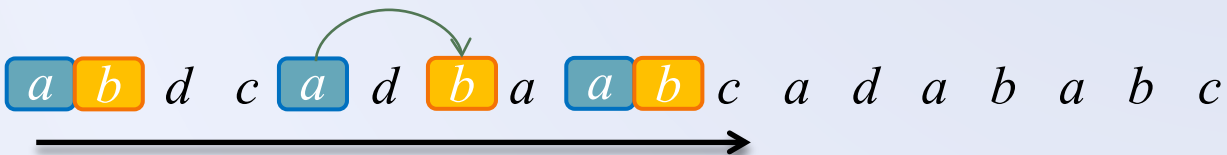
Data D

one, or multiple sequences

$\{ a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c \ ,$
 $a \ b \ d \ c \ a \ d \ b \ ,$
 $a \ b \ d \ c \ a \ d \ b \ a \ a \ , \dots \}$

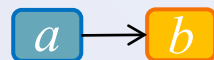
Event sequences

Alphabet Ω $\{ a, b, c, d, \dots \}$

Data D 
one, or
multiple
sequences
 $\{ a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c ,$
 $a \ b \ d \ c \ a \ d \ b ,$
 $a \ b \ d \ c \ a \ d \ b \ a \ a , \dots \}$

Patterns

serial
episodes



'subsequences
allowing gaps'

Summarising Event Sequences

The **ideal** outcome of pattern mining

- patterns that show the structure of the data
- preferably a small set, without redundancy or noise

Frequent pattern mining does **not** achieve this

- pattern explosion → overly many, overly redundant results

We pursue the ideal for serial episodes

- we want a group of patterns that summarise the data well
- we take a **pattern set** mining approach

Summarising Event Sequences

We want to find good summaries.

Three important questions

1. how do we **score** a summary?
2. how do we **cover** a sequence given a pattern set?
3. how do we **find** good pattern sets?

Scoring a Summary

We want models that generalise the data

and hence, we need a score that

- **rewards** models that identify real structure, and
- **punishes** redundancy and noise

No off-the-shelf score available for serial episodes

- e.g. no well-founded priors
- we can, however, make these goals concrete by MDL

MDL for Event Sequences

By MDL we define

the optimal set of serial episodes as the set that describes the data most succinctly

We use 2-part MDL

- we are interested in the patterns, after all

To use MDL, we need

- a lossless encoding for our models,
- a lossless encoding for the data given a model

(for itemsets, see Siebes et al, 2006, Vreeken et al 2011)

Models

pattern	code	gap	non-gap
<i>abc</i>	<i>p</i>	<i>?</i>	<i>!</i>
<i>da</i>	<i>q</i>	<i>?</i>	<i>!</i>
<i>a</i>	<i>a</i>	-	-
<i>b</i>	<i>b</i>	-	-
<i>c</i>	<i>c</i>	-	-
<i>d</i>	<i>d</i>	-	-

As models we use **code tables**

- dictionaries of patterns & codes
- always contains all singletons

We use optimal prefix codes

- easy to compute,
- behave predictably,
- good results,
- more details follow

Encoding Event Sequences

Data D : $a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c$



Encoding 1: using only singletons

C_p 

CT_1 : a 
 b 
 c 
 d 

The length of the code  for pattern X

$$L(\text{blue box with } X) = -\log(p(\text{blue box with } X)) = -\log\left(\frac{usg(X)}{\sum usg(Y)}\right)$$

The length of the code stream

$$L(C_p) = \sum_{X \in CT} usg(X) L(\text{blue box with } X)$$

Encoding Event Sequences

Data D : $a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c$



Encoding 2: using patterns

C_p $p \ d \ a \ q \ b \ p$

C_g $! \ ? \ ! \ ? \ ! \ ! \ !$

Alignment: $a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c$



$p \ ! \ ? \ !$ $q \ ? \ !$ $p \ ! \ !$

CT_2 :

a	a		
b	b		
c	c		
d	d		
abc	p	$?$	$!$
da	q	$?$	$!$

gaps ↓ non-gaps ↓

Encoding Event Sequences

Data D : $a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c$



Encoding 2: using patterns

C_p p d a q b p

C_g ! ? ! ? ! ! !

CT_2 :

a	a	
b	b	
c	c	
d	d	
abc	p	?
da	q	?

$\downarrow$ gaps $\downarrow$ non-gaps

The length of a gap code ? for pattern X

$$L(\text{?}) = -\log(p(\text{?} \mid p))$$

and analogue for non-gap codes !

Encoding Event Sequences

By which, the encoded size of D given CT and C is

$$L(D \mid CT) = L(C_p \mid CT) + L(C_g \mid CT)$$

Next, we discuss how to compute $L(CT \mid C, D)$

Encoding a Code Table

$L(CT \mid C, D)$ consists of

X	X	$?$	$!$
$\vdots$			
Y	Y	$?$	$!$
a	a		
$\vdots$			
z	z		

Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in D

$$L_{\mathbb{N}}(|\Omega|) + L_{\mathbb{N}}(||D||) + \log \binom{||D|| - 1}{|\Omega| - 1}$$

X	X	?	!
$\vdots$			
Y	Y	?	!
$\vdots$			
a	a		
$\vdots$			
z	z		

Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in D

$$L_{\mathbb{N}}(|\Omega|) + L_{\mathbb{N}}(||D||) + \log \binom{||D|| - 1}{|\Omega| - 1}$$

X	X	$?$	$!$
$\vdots$			
Y	Y	$?$	$!$
$\vdots$			
a	a		
$\vdots$			
z	z		

Encoding a Code Table

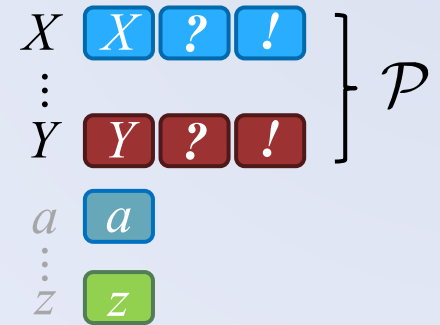
$L(CT \mid C, D)$ consists of

1) base singleton counts in D

$$L_{\mathbb{N}}(|\Omega|) + L_{\mathbb{N}}(||D||) + \log \binom{||D|| - 1}{|\Omega| - 1}$$

2) number of patterns, total, and per pattern usage

$$L_{\mathbb{N}}(|\mathcal{P}| + 1) + L_{\mathbb{N}}(usg(\mathcal{P}) + 1) + \log \binom{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1}$$



Encoding a Code Table

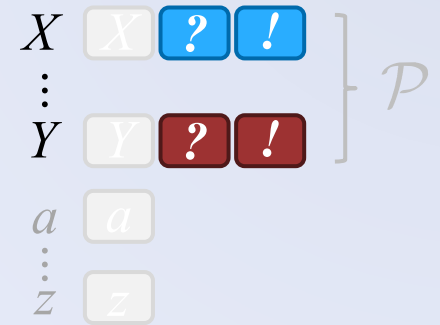
$L(CT \mid C, D)$ consists of

1) base singleton counts in D

$$L_{\mathbb{N}}(|\Omega|) + L_{\mathbb{N}}(||D||) + \log \binom{||D|| - 1}{|\Omega| - 1}$$

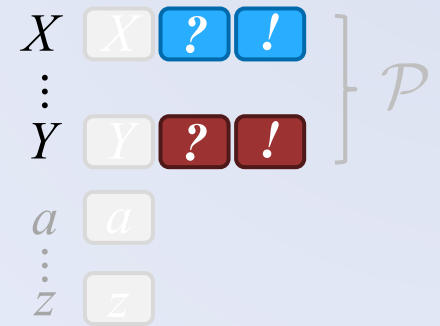
2) number of patterns, total, and per pattern usage

$$L_{\mathbb{N}}(|\mathcal{P}| + 1) + L_{\mathbb{N}}(usg(\mathcal{P}) + 1) + \log \binom{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1}$$



Encoding a Code Table

$L(CT \mid C, D)$ consists of



1) base singleton counts in D

$$L_{\mathbb{N}}(|\Omega|) + L_{\mathbb{N}}(||D||) + \log \binom{||D|| - 1}{|\Omega| - 1}$$

2) number of patterns, total, and per pattern usage

$$L_{\mathbb{N}}(|\mathcal{P}| + 1) + L_{\mathbb{N}}(usg(\mathcal{P}) + 1) + \log \binom{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1}$$

3) per pattern X : its length, elements, and number of gaps

$$L_{\mathbb{N}}(|X|) - \left[\sum_{x \in X} \log p(x \mid D) \right] + L_{\mathbb{N}}(gaps(X) + 1)$$

Encoding a Code Table

$L(CT \mid C, D)$ consists of

1) base singleton counts in D

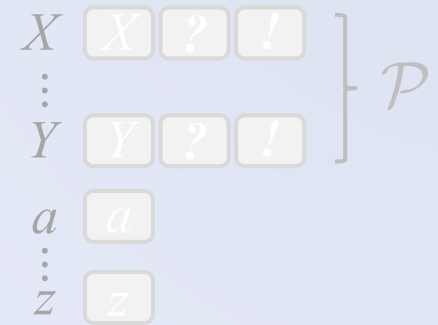
$$L_{\mathbb{N}}(|\Omega|) + L_{\mathbb{N}}(||D||) + \log \binom{||D|| - 1}{|\Omega| - 1}$$

2) number of patterns, total, and per pattern usage

$$L_{\mathbb{N}}(|\mathcal{P}| + 1) + L_{\mathbb{N}}(usg(\mathcal{P}) + 1) + \log \binom{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1}$$

3) per pattern X : its length, elements, and number of gaps

$$L_{\mathbb{N}}(|X|) - \left[\sum_{x \in X} \log p(x \mid D) \right] + L_{\mathbb{N}}(gaps(X) + 1)$$



Encoding Event Sequences

Now, our goal is to minimise

$$L(CT, D) = L(CT \mid C, D) + L(D \mid CT)$$

Summarising Event Sequences

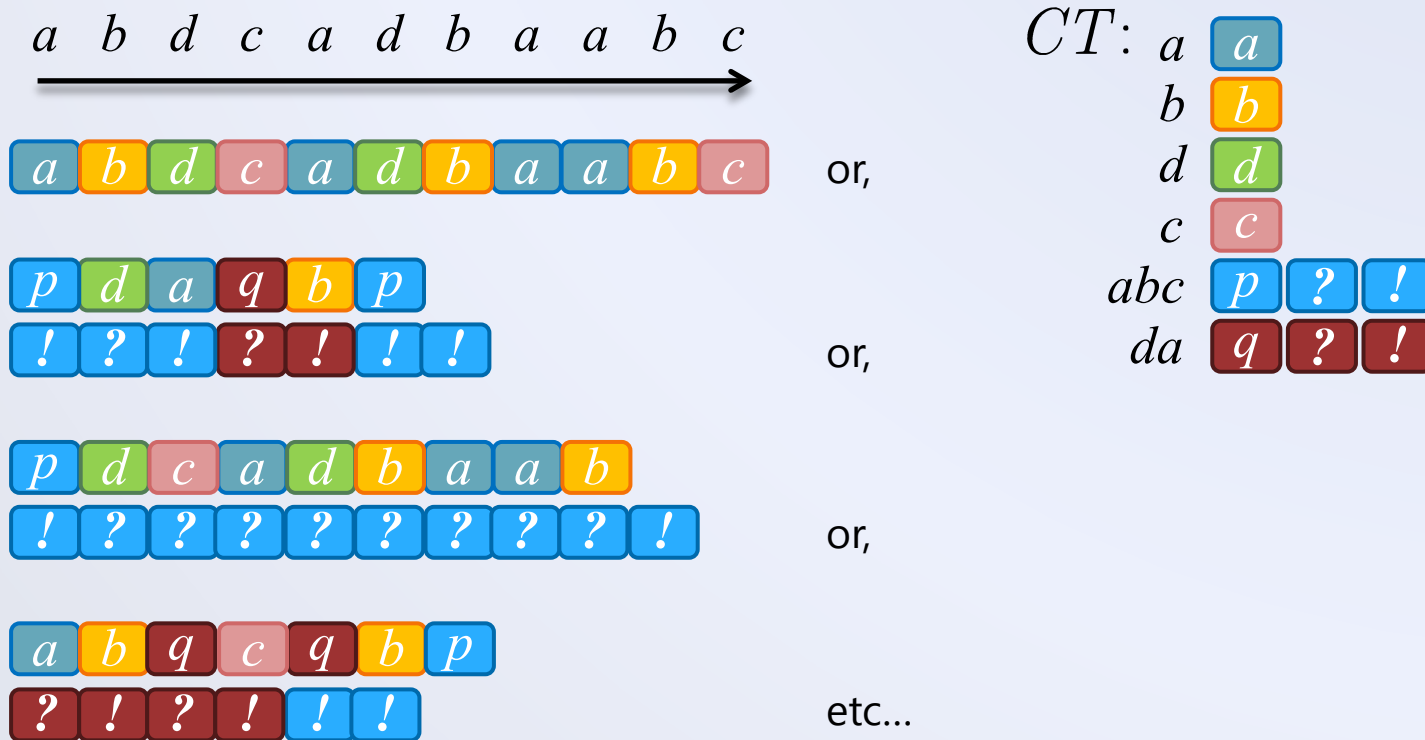
We want to find good summaries.

Three important questions

1. how do we **score** a summary?
2. how do we **describe** a sequence given a pattern set?
3. how do we **find** good pattern sets?

How to Cover your String

There are many ways C to describe a sequence given a set of patterns. We are after the **optimum**.



How to Cover your String

There are many ways C to describe a sequence given a set of patterns. We are after a **good** one.

1. if we fix the **cover**, we can obtain the optimal code lengths
2. if we fix the **code lengths**, we can obtain the optimal cover by dynamic programming

We alternate these steps until **convergence**

Summarising Event Sequences

We want to find good summaries.

Three important questions

1. how do we **score** a summary?
2. how do we **describe** a sequence given a pattern set?
3. how do we **find** good pattern sets?

Mining Code Tables

There are very many possible pattern sets.

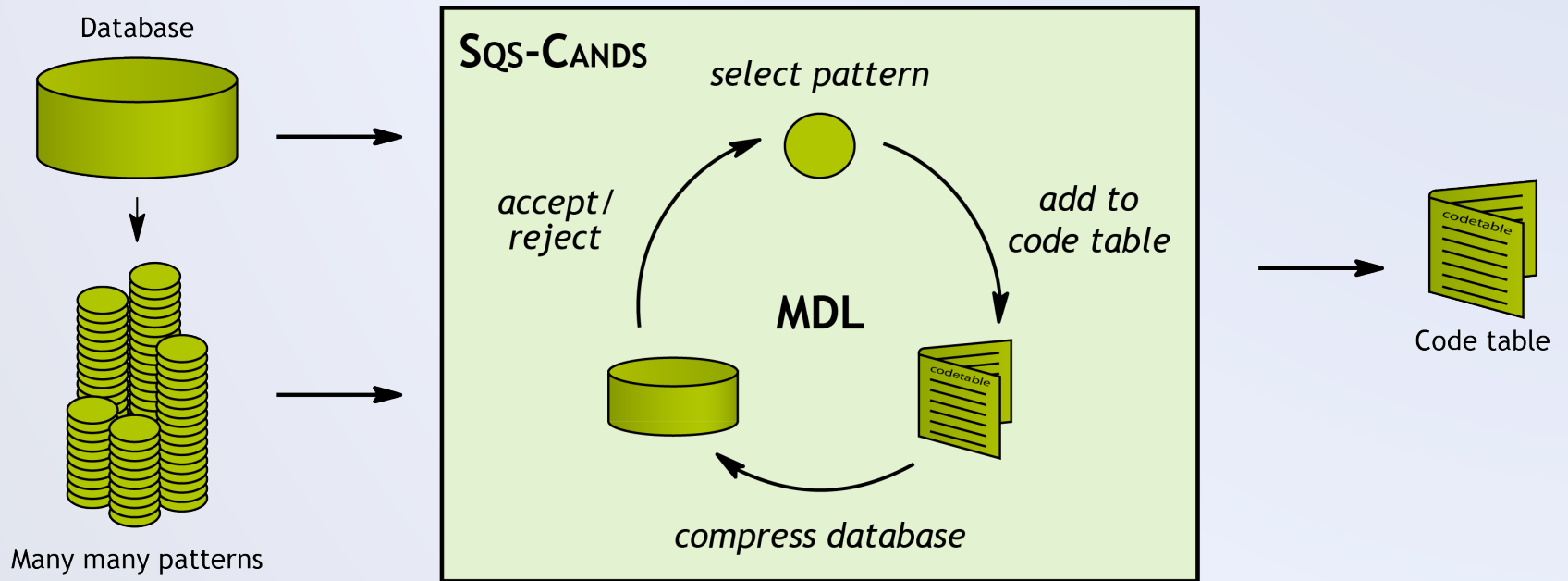
Again, we are after the **optimum**

However, again, the search space is huge, complex,
and does **not** exhibit trivial structure

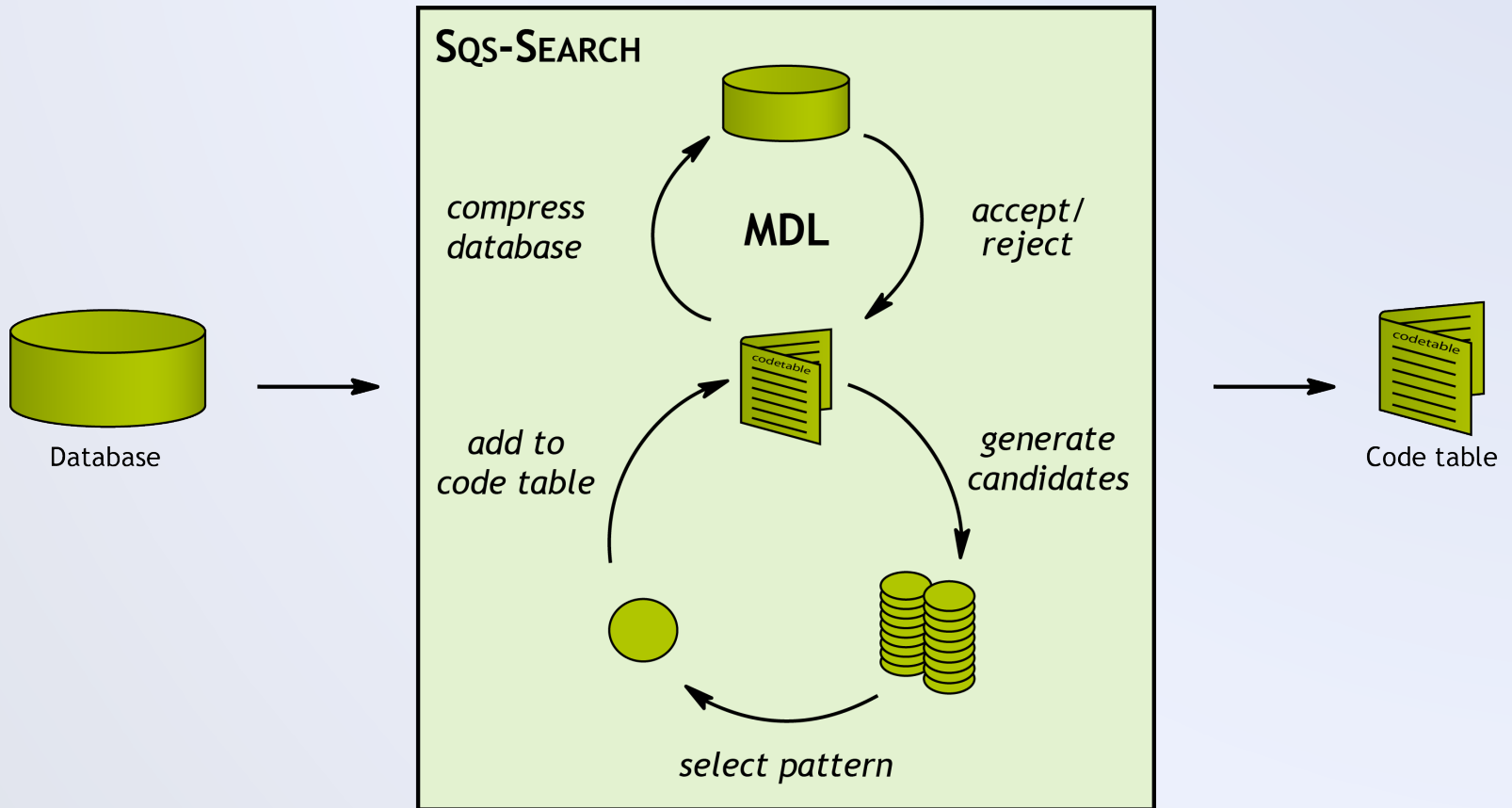
We propose two algorithms for mining code tables

- SQS-CANDS filters ordered lists of pre-mined candidates
- SQS-SEARCH mines good code tables directly from data

SQS-CANDIDATES



SQS-SEARCH



Experiments

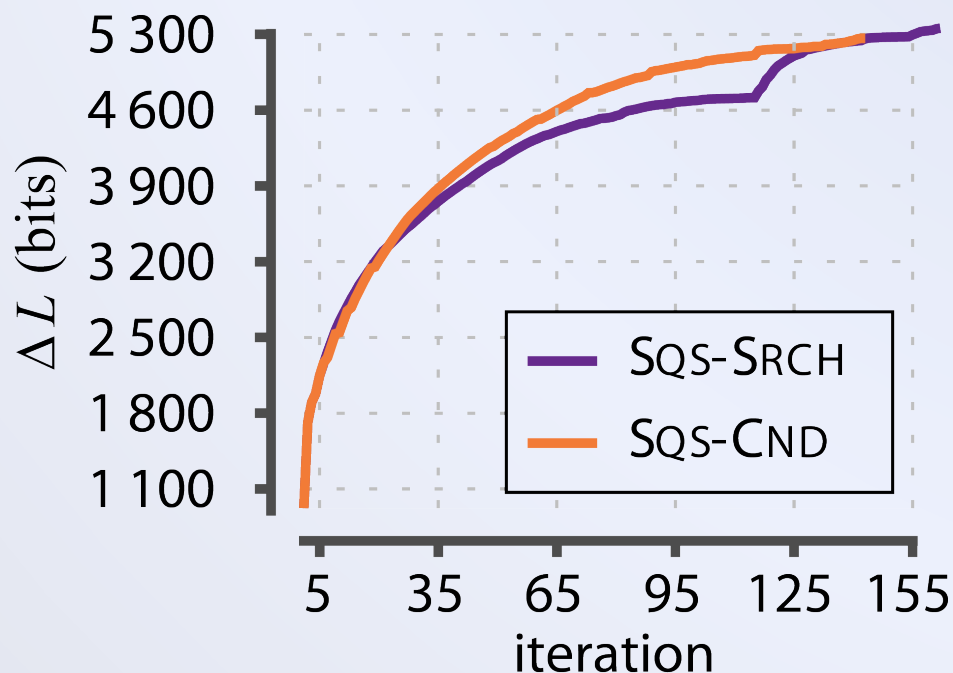
- synthetic data random HMM ✓ no structure found
 HMM ✓ structure recovered
- real data text data for interpretation

	$ \Omega $	$ D $	SQS-CANDS		SQS-SEARCH	ΔL
			$\# Cnds$	$ \mathcal{P} $	$ \mathcal{P} $	
Addresses	5 295	56	15 506	138	155	5k
JMLR	3 846	788	40 879	563	580	30k
Moby Dick	10 277	1	22 559	215	231	10k

(implementation available at <http://adrem.ua.ac.be/sqs>)

Optimising our Score

Both strategies show good convergence
SQS-SEARCH dips due to batch-wise search



Selected Results

JMLR

support vector machine
machine learning
state [of the] art
data set
Bayesian network

PRES. ADDRESSES

unit[ed] state[s]
public econ. expenditur
take oath
equal right
exercis power

Conclusions

Mining informative **sets of patterns**

- is an important aspect of exploratory data mining

SQS approximates the ideal for serial episodes

- SQS-CANDS filters a pre-mined candidate list
- SQS-SEARCH mines good code tables directly from data

Future work includes

- richer data and pattern types
- applying SQS in real-world settings

Thank you!

Mining informative **sets of patterns**

- is an important aspect of exploratory data mining

SQS approximates the ideal for serial episodes

- SQS-CANDS filters a pre-mined candidate list
- SQS-SEARCH mines good code tables directly from data

Future work includes

- richer data and pattern types
- applying SQS in real-world settings