Part 2

MDL in Action

Jilles Vreeken

Explicit Coding

Ad hoc sounds bad, but is it really?

- Bayesian learning for instance, is inherently subjective, plus
- biasing search is a time-honoured tradition in data analysis

Using an explicit encoding allows us
to steer towards the
type of structure we want to discover

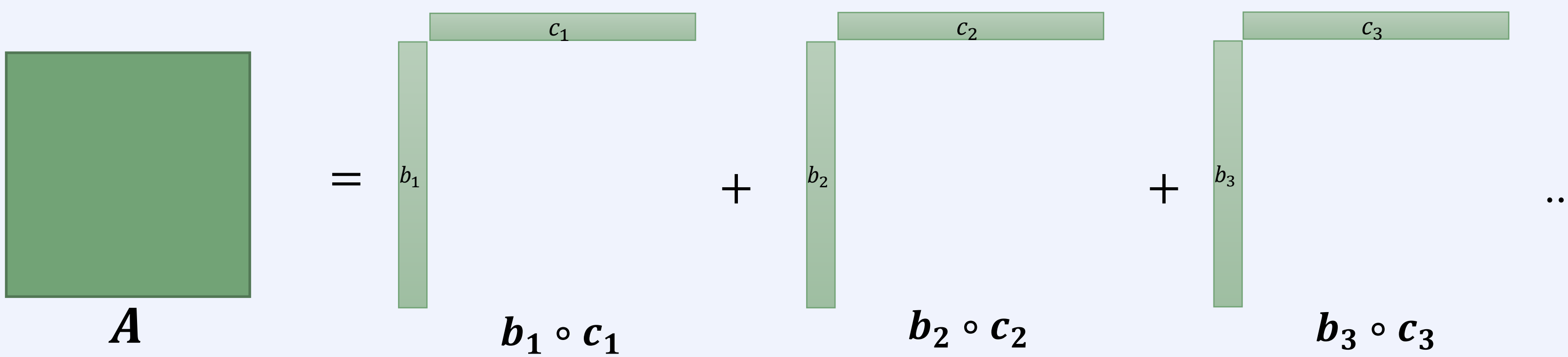
We so also mitigate one of the practical weak spots of AIT

- all data is a string, but wouldn’t it be nice if the structure you found would not depend on the order of the data?

Matrix Factorization

The rank of a matrix $\boldsymbol{A}$ is

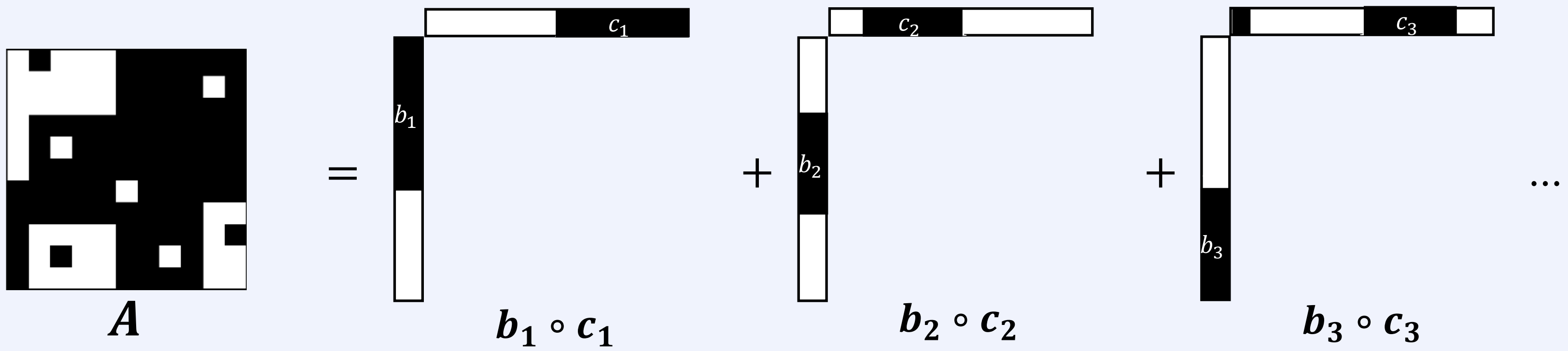
- number of rank-1 matrices that when summed form $\boldsymbol{A}$ (Schein rank)



Boolean Matrix Factorization

The rank of a Boolean matrix $\boldsymbol{A}$ is

- number of rank-1 matrices that when summed form $\boldsymbol{A}$ (**Schein rank**)

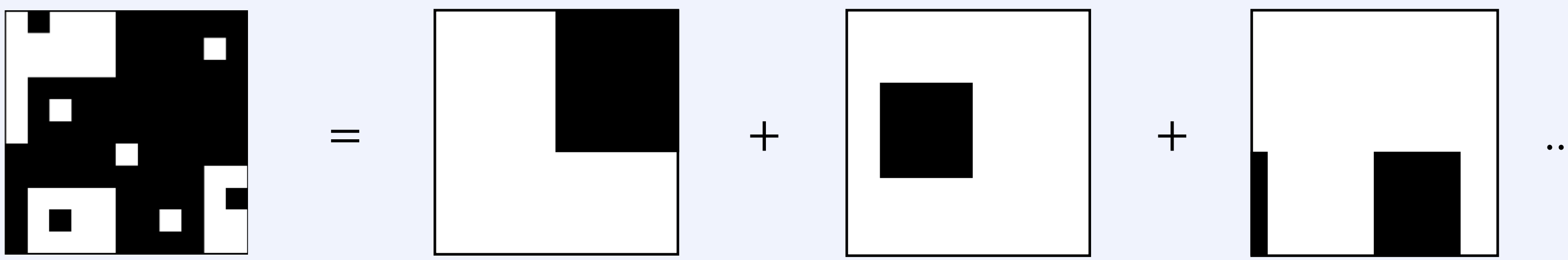


(Miettinen et al 2006, 2008)

Boolean Matrix Factorization

The rank of a Boolean matrix $\boldsymbol{A}$ is

- number of rank-1 matrices that when summed form $\boldsymbol{A}$ (Schein rank)
- noise quickly inflate the 'true' latent rank to $\min(n, m)$



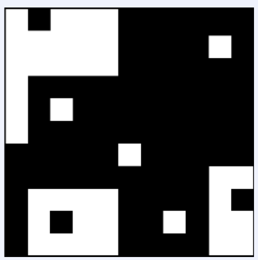
$\boldsymbol{A}$ = $\boldsymbol{b}_1 \circ \boldsymbol{c}_1$ + $\boldsymbol{b}_2 \circ \boldsymbol{c}_2$ + $\boldsymbol{b}_3 \circ \boldsymbol{c}_3$ …

(Miettinen et al 2006, 2008)

Boolean Matrix Factorization

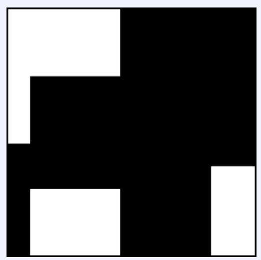
Noise quickly inflates the rank to $\min(n, m)$

- how can we determine the ‘true’ latent rank?



$\boldsymbol{A}$

$\approx$



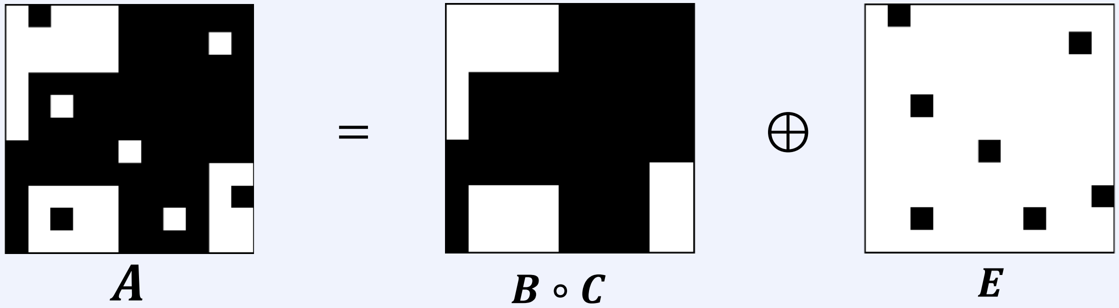
$\boldsymbol{B} \circ \boldsymbol{C}$

(Miettinen & Vreeken 2012, 2014)

Boolean Matrix Factorization

Separating structure and noise

- matrices B and C contain structure, matrix E contains noise

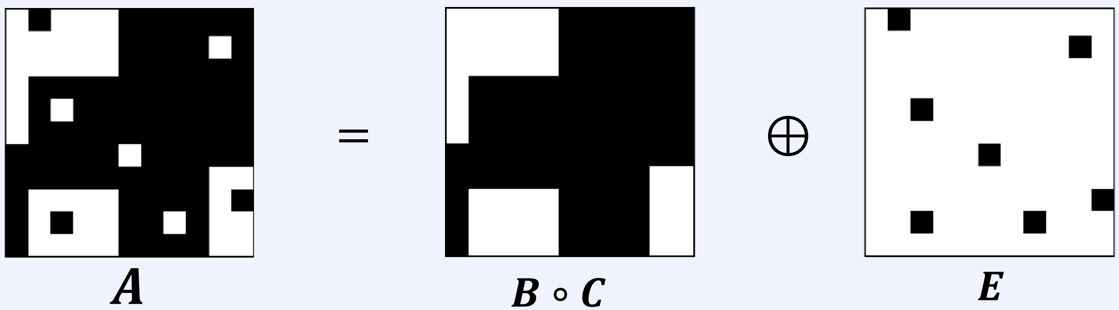


(Miettinen & Vreeken 2012, 2014)

Boolean Matrix Factorization

Encoding the structure

$$L(\boldsymbol{B}) = \log n + \sum_{b \in \boldsymbol{B}} \left[\log n + \log \binom{n}{|b|} \right]$$

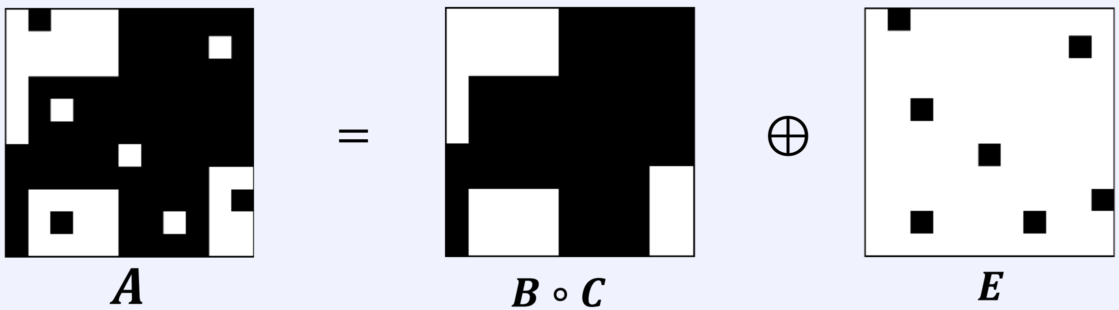


(Miettinen & Vreeken 2012, 2014)

Boolean Matrix Factorization

Encoding the structure

$$L(\boldsymbol{C}) = \log m + \sum_{c \in \boldsymbol{C}} \left[\log m + \log \binom{m}{|c|} \right]$$

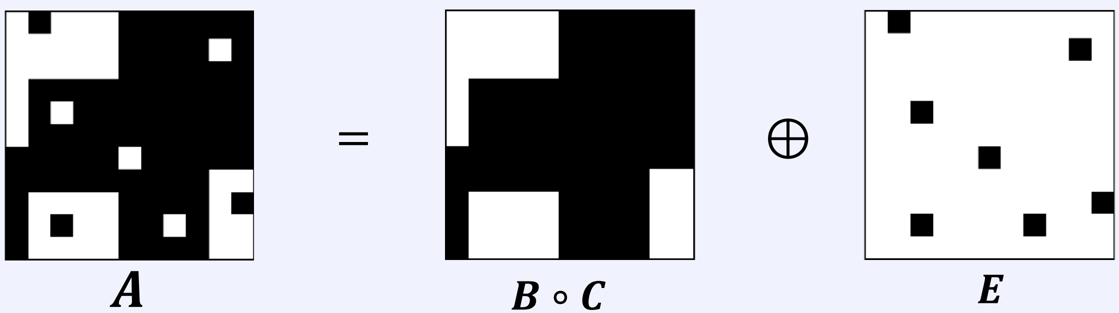


(Miettinen & Vreeken 2012, 2014)

Boolean Matrix Factorization

Encoding the noise

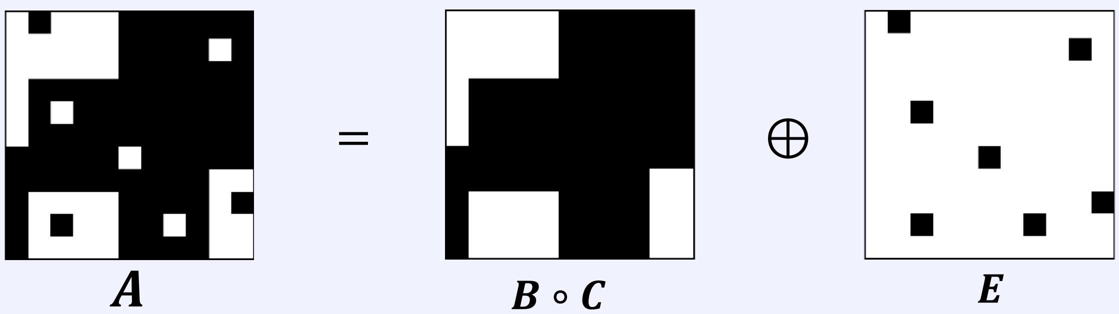
$$L(\boldsymbol{E}) = \log nm + \log \binom{nm}{|\boldsymbol{E}|}$$



Boolean Matrix Factorization

MDL for BMF

$$L(D, H) = L(\boldsymbol{B}) + L(\boldsymbol{C}) + L(\boldsymbol{E})$$



(Miettinen & Vreeken 2012, 2014)

Pattern Mining

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

MDL allows us to effectively pursue the ideal
- we want a group of patterns that summarise the data well
- we take a **pattern set** mining approach

(Tatti & Vreeken 2012, Bertens et al. 2016, Bhattacharyya & Vreeken 2017)
(for transaction data, Vreeken et al (2011), for graphs Koutra et al (2014))

Event sequences

Alphabet Ω $\{a, b, c, d, \ldots\}$

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

one, or multiple sequences

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

(Tatti & Vreeken 2012, Bertens et al. 2016, Bhattacharyya & Vreeken 2017)

Event sequences

Alphabet Ω $\{a, b, c, d, \ldots\}$

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

one, or multiple sequences

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

Patterns

serial episodes $a \rightarrow b$

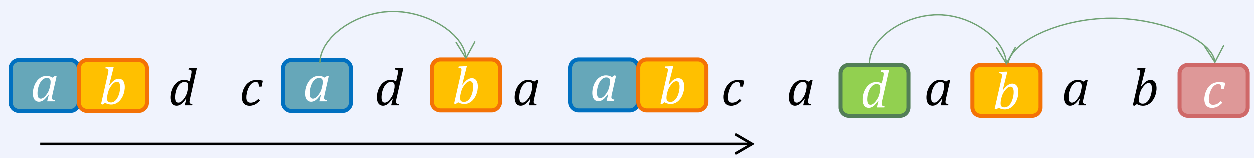
'subsequences allowing gaps'

(Tatti & Vreeken 2012, Bertens et al. 2016, Bhattacharyya & Vreeken 2017)

Event sequences

Alphabet Ω $\{ a, b, c, d, \ldots \}$

Data D

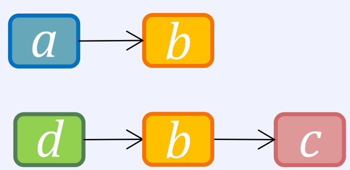


one, or multiple sequences

$\{ a \; b \; d \; c \; a \; d \; b \; a \; a \; b \; c,$
$a \; b \; d \; c \; a \; d \; b,$
$a \; b \; d \; c \; a \; d \; b \; a \; a, \ldots \}$

Patterns

serial episodes



'subsequences allowing gaps'

Models

pattern	code	gap	non-gap
abc	*p*	*?*	*!*
da	*q*	*?*	*!*
a	*a*	-	-
b	*b*	-	-
c	*c*	-	-
d	*d*	-	-

As models we use **code tables**

- dictionary 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 $\boxed{a}\boxed{b}\boxed{d}\boxed{c}\boxed{a}\boxed{d}\boxed{b}\boxed{a}\boxed{a}\boxed{b}\boxed{c}$

CT_1:

a	$\boxed{a}$
b	$\boxed{b}$
c	$\boxed{c}$
d	$\boxed{d}$

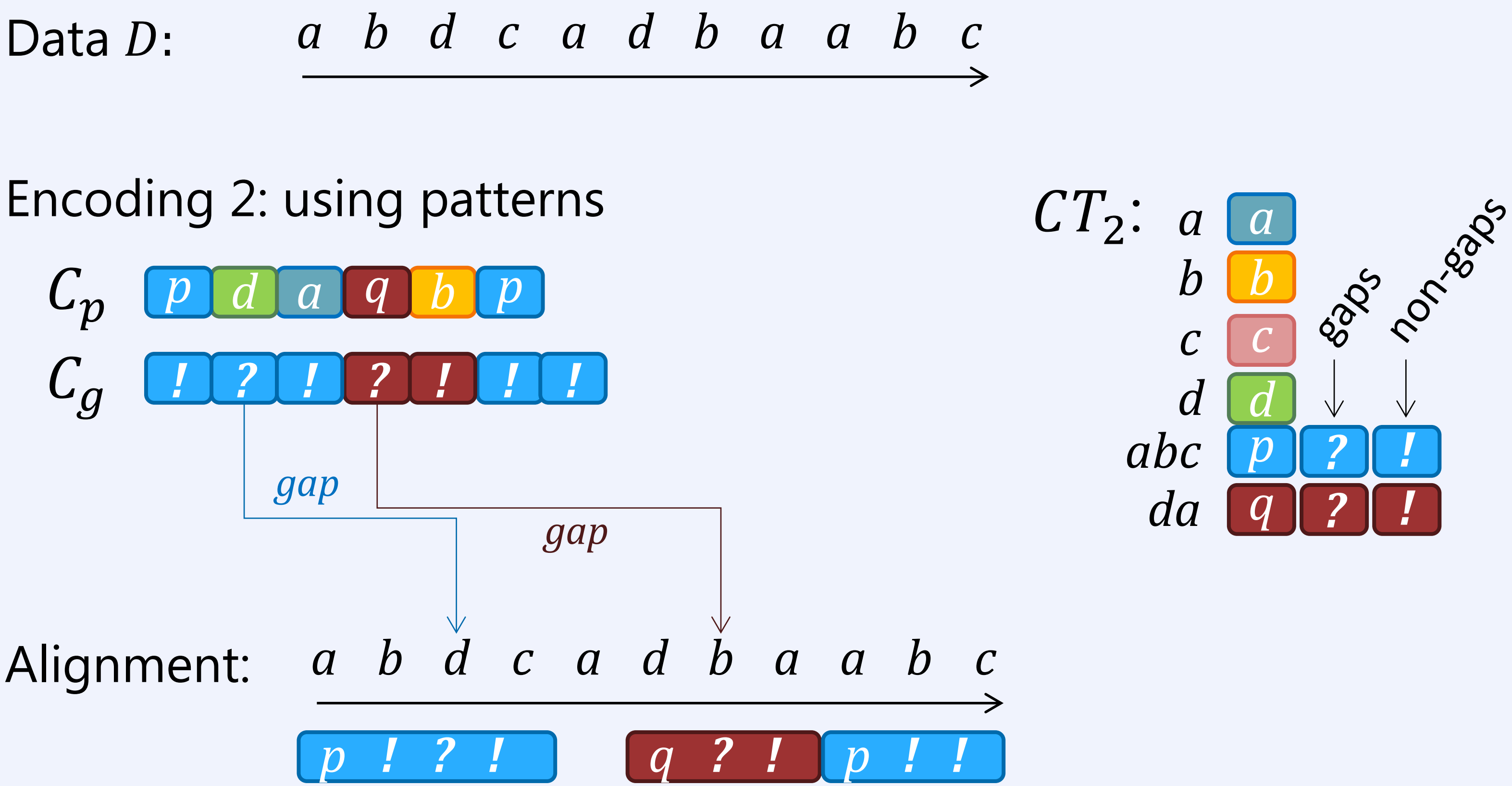
The length of the code $\boxed{X}$ for pattern X

$$L(\boxed{X}) = -\log\left(p(\boxed{X})\right) = -\log\left(\frac{usg(X)}{\sum usg(Y)}\right)$$

The length of the code stream

$$L(C_p) = \sum_{X \in CT} usg(X) L(\boxed{X})$$

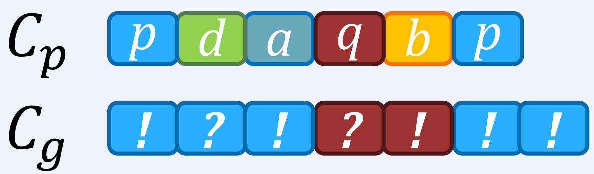
Encoding Event Sequences

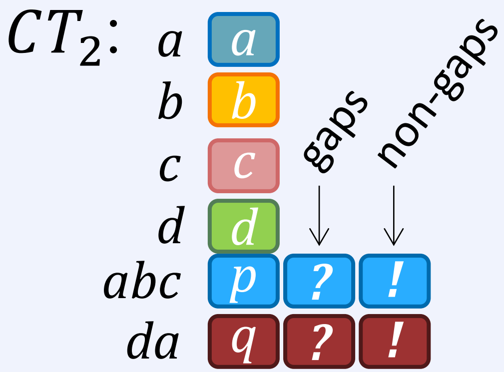


Encoding Event Sequences

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

Encoding 2: using patterns





The length of a gap code ? for pattern X

$$L(?) = -\log(p(? \mid p))$$

and analogue for non-gap codes !

Encoding Event Sequences

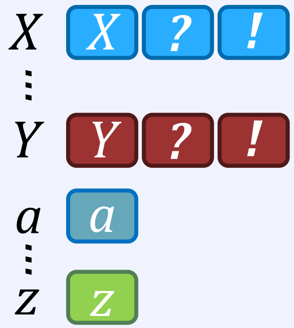
By which, the encoded size of D given CT and C is

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

which leaves us to define $L(CT \mid C)$

Encoding a Code Table

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

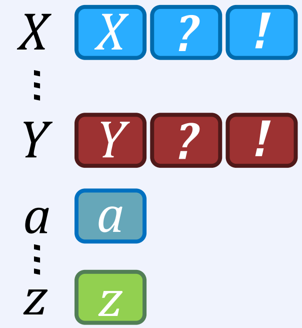


Encoding a Code Table

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

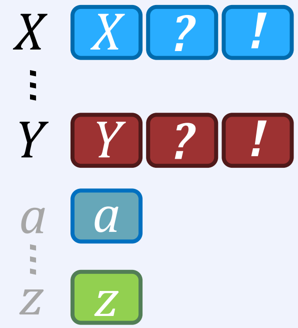
1) base singleton counts in D

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



(Rissanen 1983)

Encoding a Code Table



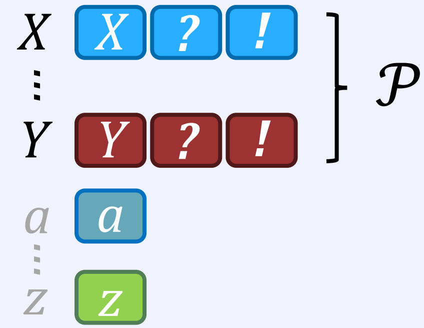
$L(CT \mid C, D)$ consists of

1) base singleton counts in D

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

(Rissanen 1983)

Encoding a Code Table



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

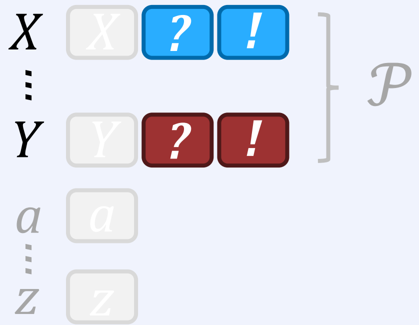
1) base singleton counts in D

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

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

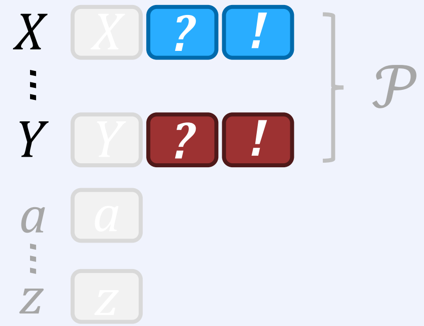
1) base singleton counts in D

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

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

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

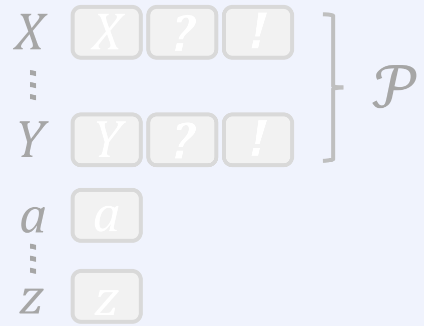
2) number of patterns, total, and per pattern usage

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

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

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

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

By which we have a lossless encoding.
In other words, an objective function.

By MDL, our goal is now to minimise

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

for how to do so, please see the papers

Tatti & Vreeken (2012) Bertens et al. (2016), Bhattacharyya & Vreeken (2017)
for transaction data, Vreeken et al (2011) Budhathoki & Vreeken (2015) for graphs Koutra et al (2014)

Experiments

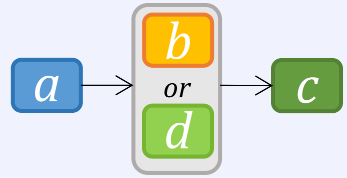
- **synthetic data** random ✓ no structure found
  HMM ✓ structure recovered
- **real data** text data for interpretation

	$\lvert\Omega\rvert$	$\lvert D\rvert$	SQS-CANDS $\# Cnds$	SQS-CANDS $\lvert\mathcal{P}\rvert$	SQS-SEARCH $\lvert\mathcal{P}\rvert$	ΔL
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://eda.mmci.uni-saarland/sqs)

Selected Results

Serial Episodes



Choice-episode

Ontological Episodes

PRES. ADDRESSES

unit[ed] state[s]
take oath
army navy
under circumst.
econ. public expenditur
exec. branch. governm.

JMLR

empirical, structural risk minimization

indep, principal component analysis

Mahalanobis, edit, Euclidean, pairwise distance

LOTR

he Verb Conj he
[he said that he]

Conf _ the Noun of
[and even the end of]

the Adj Noun and
[the young Hobbits and]

(Tatti & Vreeken 2012; Bhattacharyya & Vreeken 2017, Grosse & Vreeken 2017)

Clustering

The best clustering is the one that costs the least bits

- similar structure (patterns) within clusters
- different structure (patterns) between clusters

Partition your data such that

$$L(C) + \sum_{(D_i,H_i)\in C} L(D_i, H_i)$$

is minimal

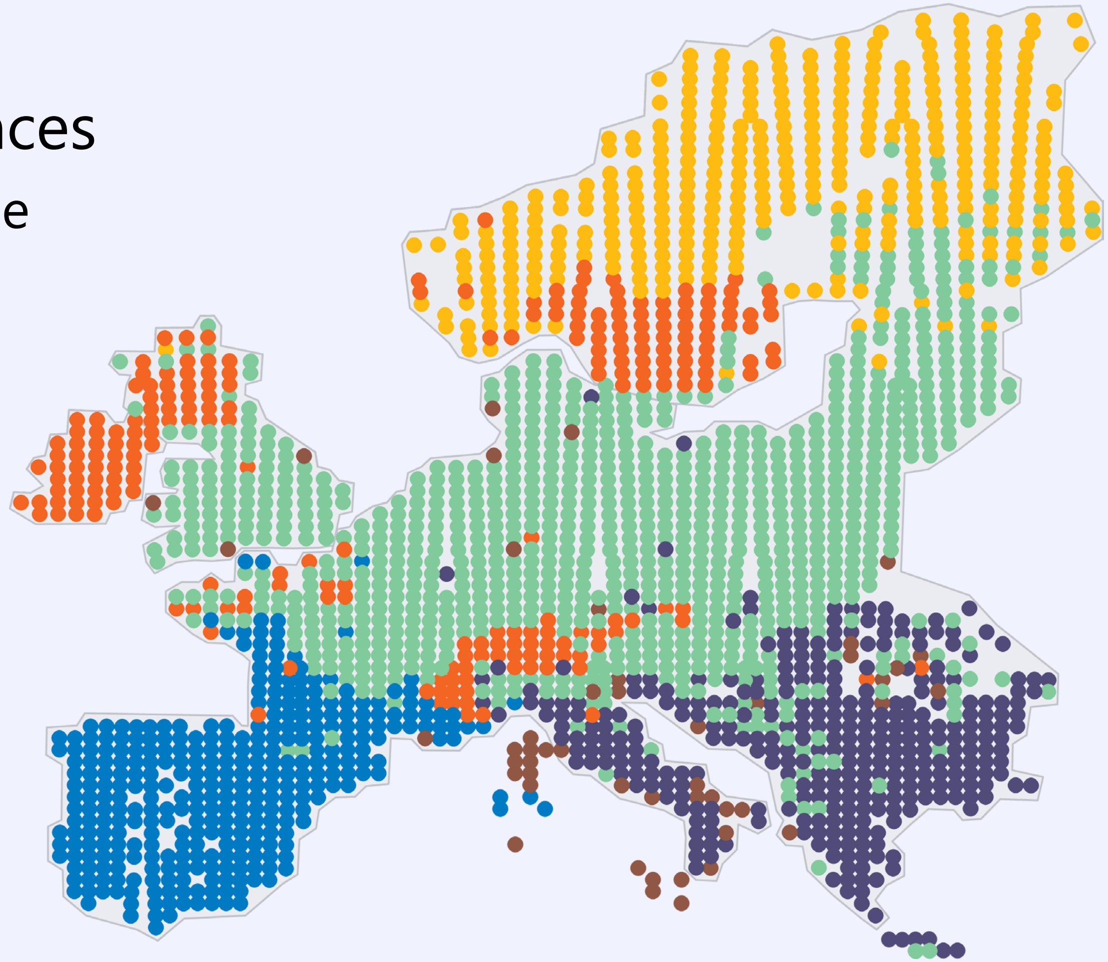
(similar to mixture modelling, but descriptive instead of predictive)

for itemsets, see Van Leeuwen et al (2009)

Clustering

Mammals occurrences

- 2221 areas in Europe
- 50x50km each
- 123 mammals
- no location info



k=6, MDL 'optimal'

for itemsets, see Van Leeuwen et al (2009)

Classification

Split your data **per class**

- induce model per class

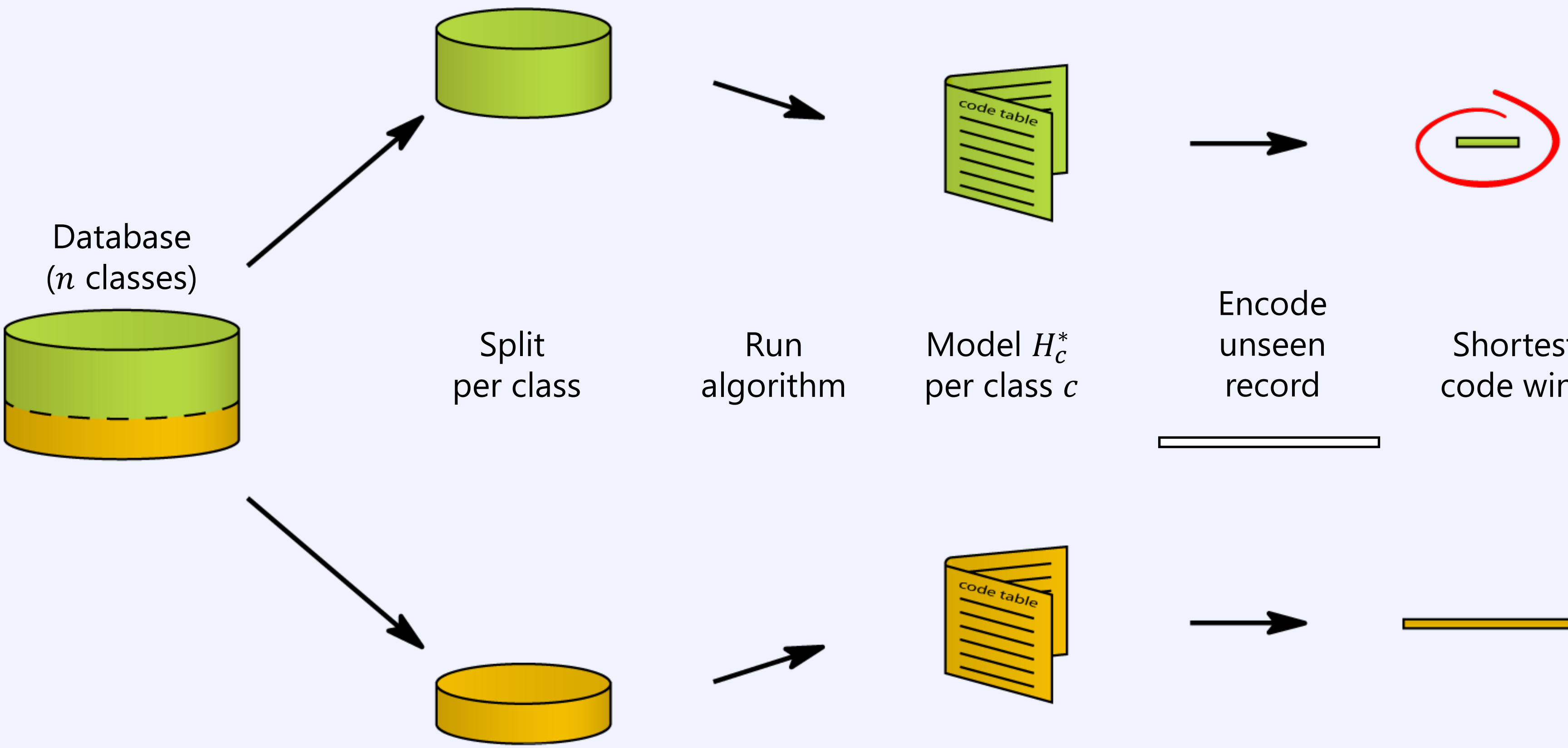
Then, for **unseen** instances

- **assign** class label of model that encodes it **shortest**

$$L(x \mid H_1) < L(x \mid H_2) \rightarrow P(x \mid H_1) > P(x \mid H_2)$$

(for itemsets, see ECML PKDD'06)

Classification by MDL



$$L(x \mid H_1) < L(x \mid H_2) \rightarrow P(x \mid H_1) > P(x \mid H_2)$$

Van Leeuwen et al (2006)

Outlier Detection

One-Class Classification (aka anomaly detection)

- lots of data for normal situation – insufficient data for target

Compression models the norm

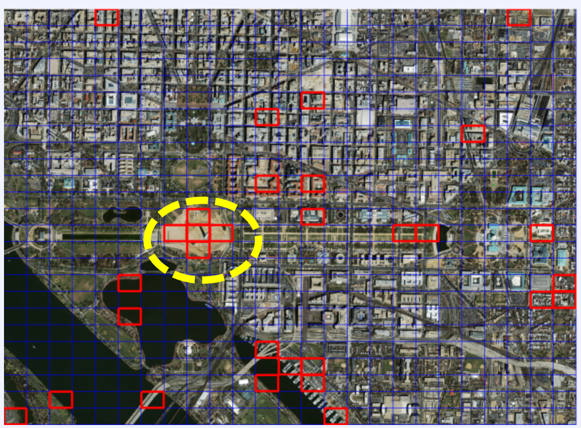
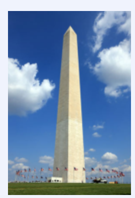
- anomalies will have high description length $L(t \mid H^*_{norm})$

Very nice properties

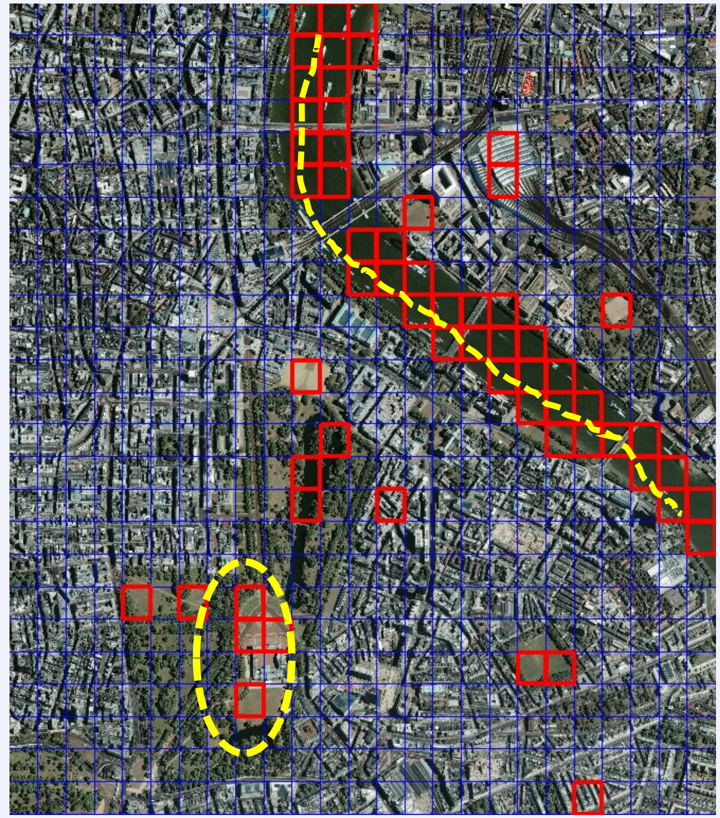
- performance high accuracy
- versatile no distance measure needed
- characterisation *'this part of t is incompressible'*

CompreX on Images


Catholic church, Vatican

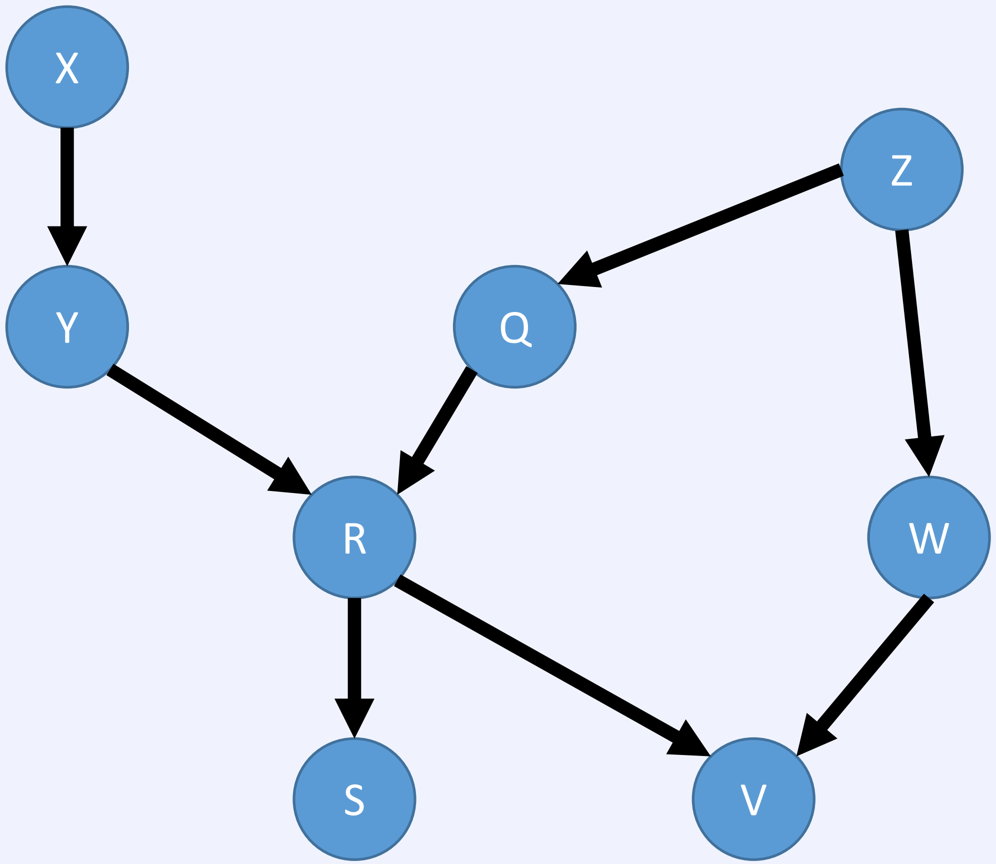



Washington Memorial, D.C.



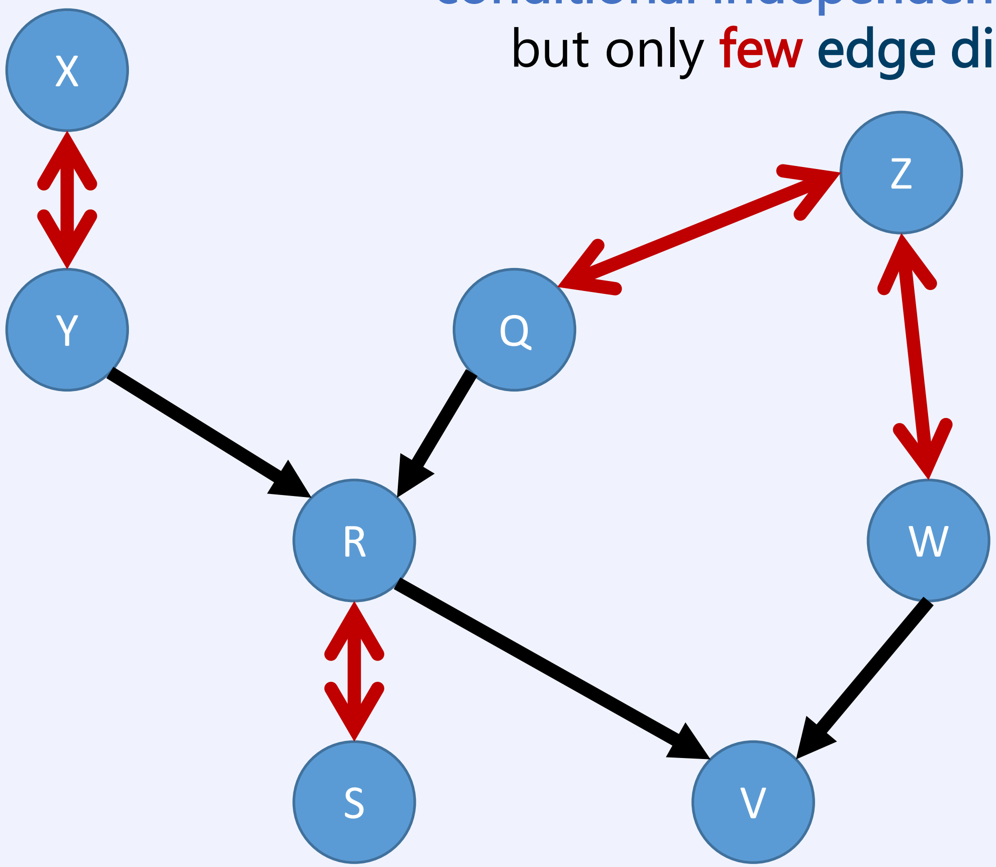
Thames river, Buckingham palace, plain fields, London

Causal Discovery



Causal Discovery

We can find the **causal skeleton** using **conditional independence** tests,
but only **few edge directions**

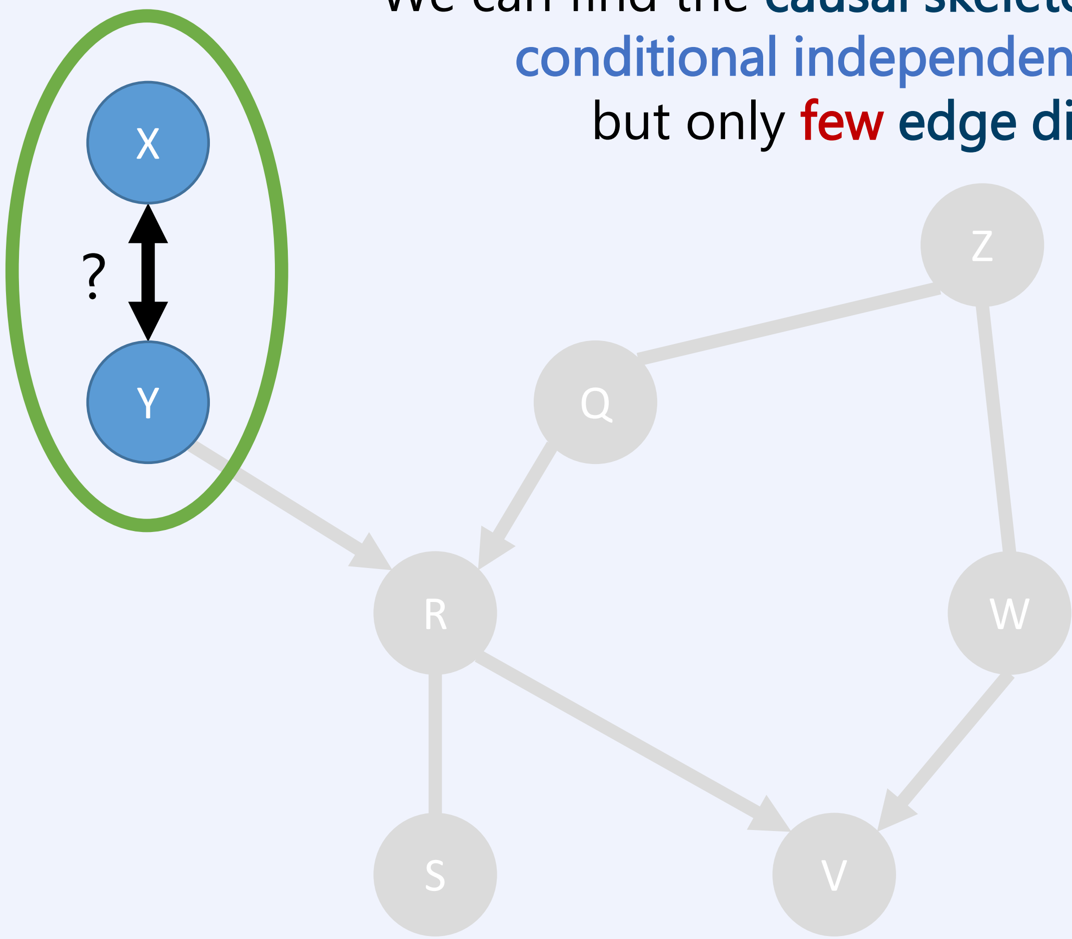


Spirtes et al (2000)

Marx & Vreeken, Conditional Independence Testing by Stochastic Complexity (2019)

Causal Inference

We can find the **causal skeleton** using **conditional independence** tests, but only **few edge directions**



Spirtes et al (2000)

Marx & Vreeken, Conditional Independence Testing by Stochastic Complexity (2019)

Algorithmic Markov Condition

If $X \rightarrow Y$, we have,

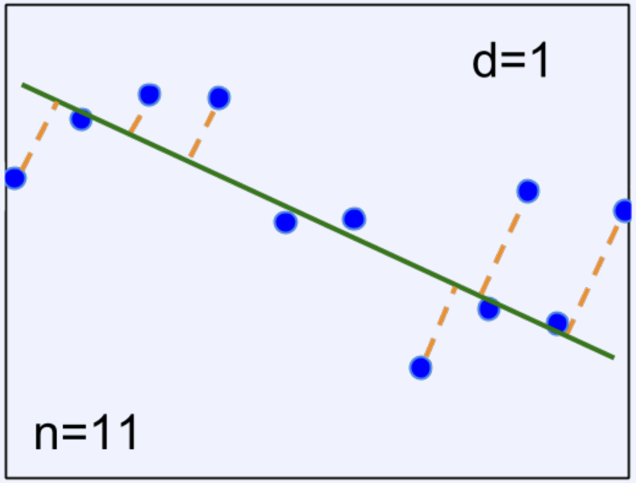
up to an additive constant,

$$K\big(P(X)\big) + K\big(P(Y|X)\big) \leq K\big(P(Y)\big) + K\big(P(X|Y)\big)$$

That is, we can do causal inference by identifying the factorization of the joint with the lowest Kolmogorov complexity

(Janzing & Schölkopf, IEEE TIT 2012)

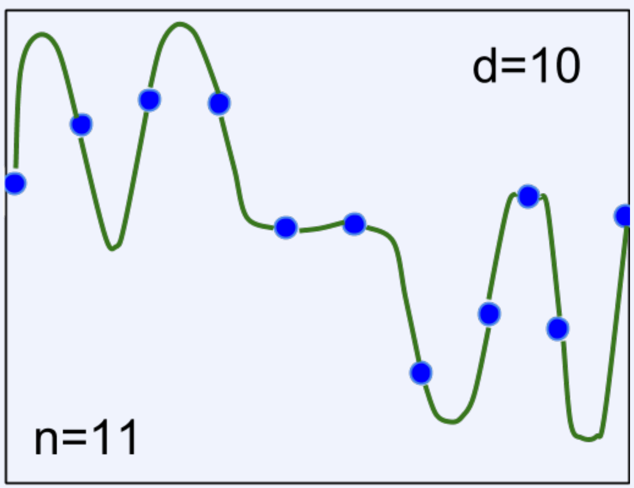
MDL and Regression



$$L(M) + L(D|M)$$

$a_1\, x + a_0$ errors

VS.



$$a_{10}\, x^{10} + a_9\, x^9 + \dots + a_0 \;\{\;\}$$

(Grünwald 2007)

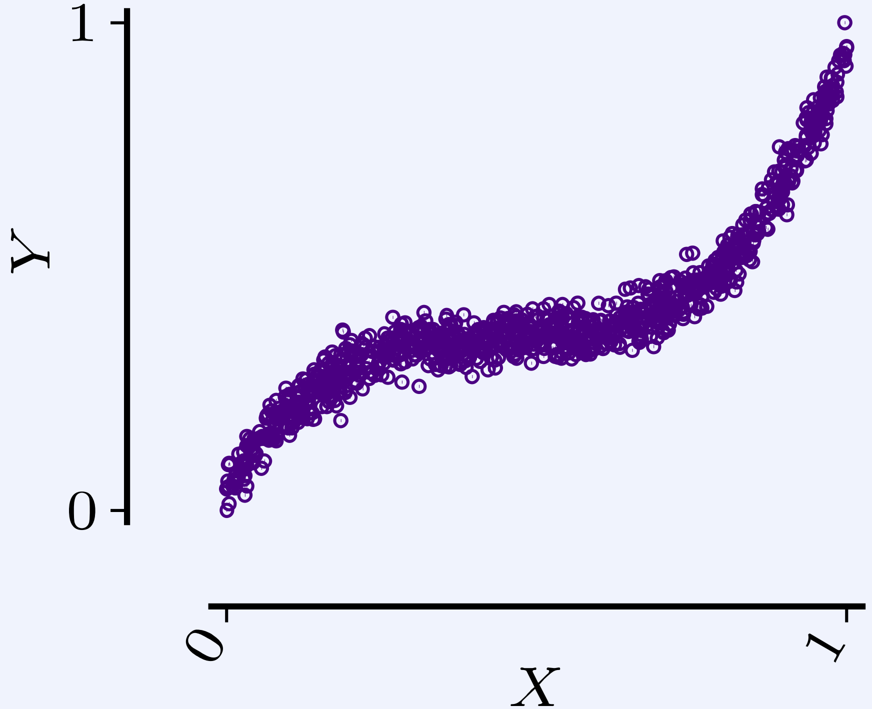
Modelling the Data

We model Y as

$$Y = f(X) + \mathcal{N}$$

As f we consider linear, quadratic, cubic, exponential, and reciprocal functions, and model the noise using a 0-mean Gaussian. We choose the f that minimizes

$$L(Y \mid X) = L(f) + L(\mathcal{N})$$



Confidence and Significance

How certain are we?

$$\mathbb{C} = |\underbrace{L(X) + L(Y \mid X)}_{L(X \to Y)} - \underbrace{L(Y) + L(X \mid Y)}_{L(Y \to X)}|$$

- the higher the more certain

Confidence and Significance

How certain are we?

$$\mathbb{C} = \left| \frac{L(X) + L(Y \mid X)}{L(X) + L(Y)} - \frac{L(Y) + L(X \mid Y)}{L(X) + L(Y)} \right|$$

- the higher the more certain
- robust w.r.t. sample size

Is a given inference significant?

- our null hypothesis L_0 is that ***X*** and ***Y*** are only **correlated**, we have $L_0 = \frac{|L(X \to Y) - L(Y \to X)|}{2}$

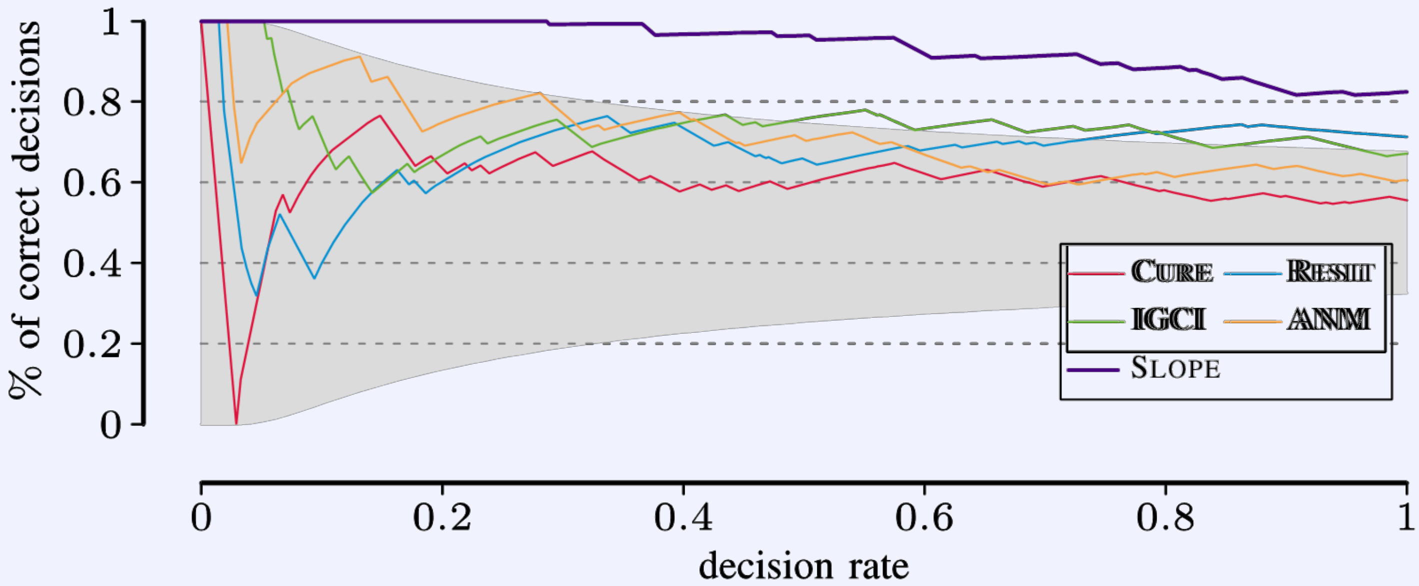
- we can use the no-hypercompression inequality to test significance

$$\mathrm{P}(L_0(D) - L(D) \geq k) \leq 2^{-k}$$

Grünwald (2007), Marx & Vreeken (2017)

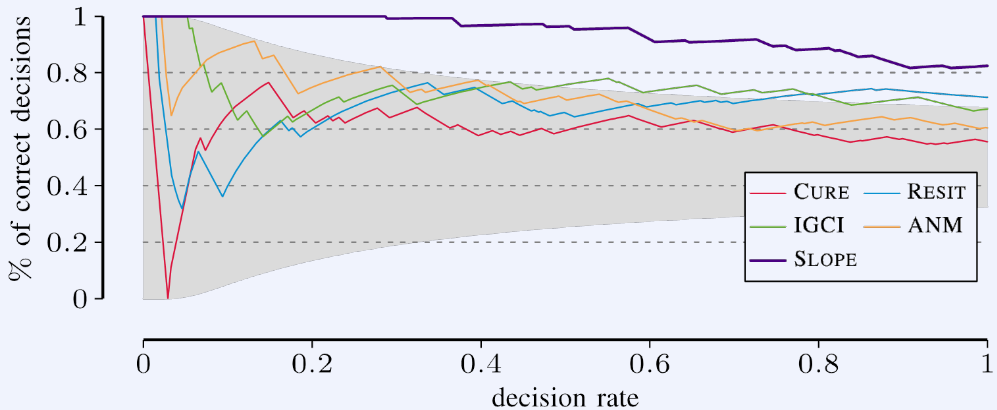
Performance on Benchmark Data

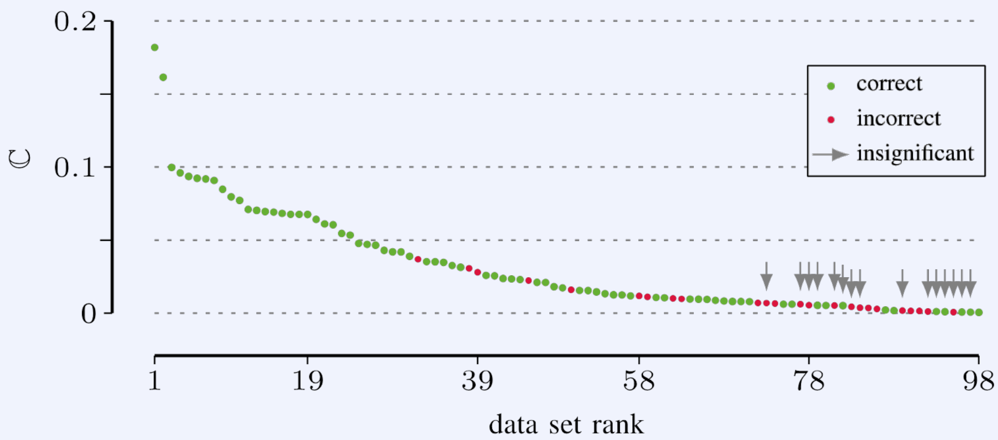
(Tübingen **97 univariate numeric cause-effect pairs, weighted)**



Performance on Benchmark Data

(Tübingen **97 univariate numeric cause-effect pairs, weighted**)





Inferences of state of the art algorithms **ordered** by **confidence** values.

SLOPE is 85% accurate with $\alpha = 0.001$

Deep Learning

Model selection in deep learning is hard

- way too many ‘free’ parameters for standard regularizers,
- no meaningful prior over networks, and
- uniform prior will lead to overfitting

How about an MDL approach?

- what is the description length of a neural network?

MDL for Neural Networks

Suppose neural network $H \in \mathcal{H}$ predicts target y given x

$$\hat{y} = H(x)$$

How do we encode data given the model?

- if $H(x)$ is probabilistic, we have $L(\,\boldsymbol{y} \mid H(\boldsymbol{x})\,) = -\sum_{y_i \in \boldsymbol{y}} \log p(y_i | x_i)$
- else we can simply encode the residual error,
  - e.g. if $\boldsymbol{y}$ is binary, we have $\boldsymbol{e} = \boldsymbol{y} \oplus \hat{\boldsymbol{y}}$, and $L(\,\boldsymbol{y} \mid H(\boldsymbol{x})\,) = \log n + \log \binom{n}{|\boldsymbol{e}|}$
  - e.g. if $\boldsymbol{y}$ is continuous, we can encode using a zero-mean Gaussian

MDL for Neural Networks

Suppose neural network $H \in \mathcal{H}$ predicts target y given x

$$\hat{y} = H(x)$$

How do we encode the model?

- we could encode all of the parameters, but that's highly ad hoc
- instead, we can use the notion of **prequential coding**

Dawid (1984) Barron et al (1998) Grünwald (2007) Blier & Ollivier (2018)

Prequential Coding

Simple, elegant idea:

"Update your model after every message"

That is, we re-train our network after 'every' new label

- we initialize topology $H \in \mathcal{H}$ with fixed weights
- we transmit the first k labels using H_0
- we now train H on this first batch of k labelled points, we obtain H_1
- we transmit the second k labels using H_1
- we now train H on the first two batches, and obtain H_3

Dawid (1984) Barron et al (1998) Grünwald (2007) Blier & Ollivier (2018)

Prequential Coding

Simple, elegant idea:

"Update your model after every message"

$$L(D \mid \mathcal{H}) = \sum_{D_i} L(D_i \mid H_{i-1})$$

Best of all, this is not a crude, but a refined MDL code!

- depends fully on how H behaves on the data
- no arbitrary choices on how to encode H
- within a constant of $L(D|H^*)$, and this constant only depends on $\mathcal{H}$

Dawid (1984) Barron et al (1998) Grünwald (2007)

Schedule

8:00am	Opening
8:10am	Introduction to MDL
8:50am	MDL in Action
9:30am	——— *break* ———
10:00am	Stochastic Complexity
11:00am	MDL in Dynamic Settings