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 $\mathbf{A}$ is

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

$$
\mathbf{A} = b_1 \circ c_1 + b_2 \circ c_2 + b_3 \circ c_3 + \ldots
$$
Boolean Matrix Factorization

The rank of a Boolean matrix $A$ is

- number of rank-1 matrices that when summed form $A$ (Schein rank)

$$A = b_1 \circ c_1 + b_2 \circ c_2 + b_3 \circ c_3 + \ldots$$

(Miettinen et al. 2006, 2008)
Boolean Matrix Factorization

The rank of a Boolean matrix $A$ is

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

$$A = b_1 \circ c_1 + b_2 \circ c_2 + b_3 \circ c_3 + \ldots$$

(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?

\[ A \approx B \circ C \]
Boolean Matrix Factorization

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

$$ A = B \odot C \oplus E $$

(Miettinen & Vreeken 2012, 2014)
Boolean Matrix Factorization

Encoding the structure

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

\[ A = B \circ C \oplus E \]

(Miettinen & Vreeken 2012, 2014)
Boolean Matrix Factorization

Encoding the structure

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

\[ A = B \odot C \oplus E \]
Boolean Matrix Factorization

Encoding the noise

\[ L(E) = \log nm + \log \left(\frac{nm}{|E|}\right) \]

\[ A = B \circ C \oplus E \]
Boolean Matrix Factorization

MDL for BMF

\[ L(D, H) = L(B) + L(C) + L(E) \]

\[ A = B \odot C \oplus E \]

(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

Event sequences

Alphabet $\Omega$  \: $\{ a, b, c, d, \ldots \}$

Data $D$

one, or multiple sequences

\[ a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c \ a \ d \ a \ b \ a \ b \ c \]

\[ \{ a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c ,
\quad a \ b \ d \ c \ a \ d \ b ,
\quad a \ b \ d \ c \ a \ d \ b \ a \ a , \ldots \} \]
Event sequences

Alphabet $\Omega$  
\{ $a$, $b$, $c$, $d$, ... $\}$

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

Patterns  
serial episodes
\{ $a$ $b$ $d$ $c$ $a$ $d$ $b$ $a$ $a$ $b$ $c$ $a$ $b$ $d$ $c$ $a$ $d$ $b$ $a$ $b$ $c$ $\}$

one, or multiple sequences

Patterns  
'subsequences allowing gaps'

Event sequences

Alphabet $\Omega$

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

Data $D$

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

Patterns

serial episodes

\begin{align*}
& a \rightarrow b \\
& d \rightarrow b \rightarrow c
\end{align*}

‘subsequences allowing gaps’

Models

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

<table>
<thead>
<tr>
<th>pattern</th>
<th>code</th>
<th>gap</th>
<th>non-gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>p</td>
<td>?</td>
<td>!</td>
</tr>
<tr>
<td>da</td>
<td>q</td>
<td>?</td>
<td>!</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Encoding Event Sequences

Data \( D \): \[ \begin{array}{ccccccccc}
  a & b & d & c & a & d & b & a & b & c
\end{array} \]

Encoding 1: using only singletons

\[ C_p \quad \begin{array}{ccccccccc}
  a & b & d & c & a & d & b & a & a & b & c
\end{array} \]

CT_1: \[ \begin{array}{cccc}
  a & a \\
  b & b \\
  c & c \\
  d & d
\end{array} \]

The length of the code \( X \) for pattern \( X \)

\[ L(X) = -\log(p(X)) = -\log\left(\frac{\text{usg}(X)}{\sum \text{usg}(Y)}\right) \]

The length of the code stream

\[ L(C_p) = \sum_{X \in CT} \text{usg}(X)L(X) \]
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: $a \ b \ d \ c \ a \ d \ b \ a \ a \ b \ c$

$CT_2$: $a \ b \ c \ d \ a \ b \ p \ ? \ ! \ q \ ? \ !$

$D$: $\begin{align*}
D_1: & \quad p \ ! \ ? \ ! \\
D_2: & \quad q \ ? \ ! \ p \ ! !
\end{align*}$
Encoding Event Sequences

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

Encoding 2: using patterns

$C_p: \begin{array}{ccccccc}
p & d & a & q & b & p \\
\end{array}$

$C_g: \begin{array}{cccccccc}
\end{array}$

$CT_2$: $a \ a \ b \ b \ c \ c \ d \ d \ p \ ? \ ? \ ! \ ! \ ! \ !$

The length of a gap code $?$ for pattern $X$

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

and analogue for non-gap codes $!$
Encoding Event Sequences

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

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

which leaves us to define $L(CT \mid C)$
Encoding a Code Table

\[ L(CT \mid C, D) \] consists of

\[ X, \quad Y, \quad a, \quad z \]
Encoding a Code Table

\[ L(CT \mid C, D) \] consists of

1) base singleton counts in \( D \)

\[ L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{||D|| - 1}{|\Omega| - 1} \right) \]
Encoding a Code Table

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

1) base singleton counts in \( D \)

\[
L_N(|\Omega|) + L_N(\|D\|) + \log \left( \frac{\|D\| - 1}{|\Omega| - 1} \right)
\]
Encoding a Code Table

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

1) base singleton counts in $D$

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

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

$$L_\mathbb{N}(|\mathcal{P}| + 1) + L_\mathbb{N}(\text{usg}(\mathcal{P}) + 1) + \log \left( \frac{\text{usg}(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)$$
Encoding a Code Table

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

1) base singleton counts in $D$

$$L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{||D|| - 1}{|\Omega| - 1} \right)$$

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

$$L_N(|\mathcal{P}| + 1) + L_N(usg(\mathcal{P}) + 1) + \log \left( \frac{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)$$
Encoding a Code Table

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

1) base singleton counts in $D$

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

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

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

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 $D$

$$L_N(|\Omega|) + L_N(||D||) + \log \left( \frac{|D| - 1}{|\Omega| - 1} \right)$$

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

$$L_N(|\mathcal{P}| + 1) + L_N(usg(\mathcal{P}) + 1) + \log \left( \frac{usg(\mathcal{P}) - 1}{|\mathcal{P}| - 1} \right)$$

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

$$L_N(|X|) - \left[ \sum_{x \in X} \log p(x \mid D) \right] + L_N(gaps(X) + 1)$$
Encoding Event Sequences

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

By MDL, our goal is now to minimise

\[
L(CT, D) = L(CT | C) + L(D | CT)
\]

for how to do so, please see the papers
Experiments

- **synthetic data**
  - random
  - HMM
  - no structure found

- **real data**
  - text data
  - structure recovered for interpretation

<table>
<thead>
<tr>
<th></th>
<th>$\Omega$</th>
<th>$D$</th>
<th># Cnds</th>
<th>$P$</th>
<th>$\Delta L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addresses</td>
<td>5295</td>
<td>56</td>
<td>15506</td>
<td>138</td>
<td>155</td>
</tr>
<tr>
<td>JMLR</td>
<td>3846</td>
<td>788</td>
<td>40879</td>
<td>563</td>
<td>580</td>
</tr>
<tr>
<td>Moby Dick</td>
<td>10277</td>
<td>1</td>
<td>22559</td>
<td>215</td>
<td>231</td>
</tr>
</tbody>
</table>

(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]
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

\[ L(x | H_1) < L(x | H_2) \rightarrow P(x | H_1) > P(x | H_2) \]

(for itemsets, see ECML PKDD’06)
Classification by MDL

\[ L(x \mid H_1) < L(x \mid H_2) \Rightarrow P(x \mid H_1) > P(x \mid H_2) \]
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 $L(t \mid H^*_{\text{norm}})$

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
We can find the **causal skeleton** using **conditional independence** tests, but only **few edge directions**
We can find the **causal skeleton** using **conditional independence** tests, but only **few edge directions**.
If $X \rightarrow Y$, we have, up to an additive constant,

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

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

\[ L(M) + L(D|M) \]

\[ a_1 x + a_0 \quad \text{errors} \]

\[ a_{10} x^{10} + a_9 x^9 + \ldots + a_0 \{ \} \]
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?

\[ C = |L(X) + L(Y \mid X) - L(Y) + L(X \mid Y)| \]

- the higher the more certain

\[ L(X \rightarrow Y) \quad L(Y \rightarrow X) \]
Confidence and Significance

How certain are we?

\[ C = \left| \frac{L(X) + L(Y | X)}{L(X) + L(Y)} - \frac{L(Y) + L(X | Y)}{L(X) + L(Y)} \right| \]

- 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 \( L_0 = \frac{|L(X \rightarrow Y) - L(Y \rightarrow X)|}{2} \)

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

\[ P(L_0(D) - L(D) \geq k) \leq 2^{-k} \]
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 $L(y | H(x)) = -\sum_{y_i \in y} \log p(y_i | x_i)$
- else we can simply encode the residual error,
  - e.g. if $y$ is binary, we have $e = y \oplus \hat{y}$, and $L(y | H(x)) = \log n + \log \binom{n}{|e|}$
  - e.g. if $y$ 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
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 $H_3$
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 \( L(D \mid H^*) \), and this constant only depends on \( \mathcal{H} \)

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