Flexibly Mining Better Subgroups

Hoang Vu Nguyen
Jilles Vreeken
How can we efficiently discover the globally optimal cut points for any subgroup discovery objective function?
Question of the day

How can we efficiently discover the globally optimal cut points for any subgroup discovery objective function?
How can we efficiently discover the locally optimal cut points for any subgroup discovery objective function?
Subgroup Discovery

Find conditions on attributes such that distribution of the targets on the conditioned data is different from that of the global data.

For example
- when $Temperature \leq 6$ there are fewer bikers than usual
- when $20 \leq Temperature \leq 25$ and $65 \leq Humidity \leq 75$ there are more bikers than usual
Example Subgroup

The number of gold atoms in a micro-cluster strongly determines its homo-lumo gap

\[ \text{Condition} = \{\text{"N odd"}\} \]

(together with Mario Boley, work in progress)
Binary Features

A **condition on an attribute** is essentially a **binary feature**
- subgroup discovery essentially relies on feature construction

For **nominal data**, extracting binary features is **easy**
- there are only $2^{|\text{dom}(A)|}$ features for each attribute $A$, after all

For **numeric or ordinal data**, this is much **harder**
- there are $2^n$ possible features for each attribute $A$
- standard approach is to simply use $k$ equi-width or height bins
### Eye of the beholder

There exist very many quality measures
- each with specific properties, for target-specific data types

<table>
<thead>
<tr>
<th>Measure</th>
<th>Univariate</th>
<th>Multivariate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nominal</td>
<td>Ordinal</td>
</tr>
<tr>
<td>WRAcc</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>z-score</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hellinger distance</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Quadratic divergence</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>
Discovering subgroups

Very complicated combinatorial problem
- **humonguous** search space
  - all possible conditions on all possible attributes
- **unstructured** search space
  - useful objective functions are not monotone/submodular

Standard approach
- naively binarise your data
- sample or search to discover top-$k$ best subgroups
Discovering subgroups

Very complicated combinatorial problem
- **humonguous** search space
  all possible conditions on all possible attributes
- **unstructured** search space
  useful objective functions are not monotone/submodular

Standard approach
- naively binarise your data
- sample or search to discover top-\(k\) best subgroups

What we fix in this paper
Quality measures are **highly specific** to problem settings

- can we define a **general** and **efficient** algorithm to find cut points?

### Quality measures

<table>
<thead>
<tr>
<th>Measure</th>
<th>Univariate</th>
<th>Multivariate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nominal</td>
<td>Ordinal</td>
</tr>
<tr>
<td>WRAcc</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>z-score</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hellinger distance</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Quadratic divergence</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>
For attribute $A$, discover the **binary features**, i.e. grid $g$, that gives **maximal average quality** for objective $\phi$

$$\arg\max_{g \in \mathcal{F}} \frac{1}{|g|} \sum_{i=1}^{|g|} \phi(b^i_g)$$

This leaves $|\mathcal{F}| = O(2^n)$ grids to evaluate...

- luckily, the search space is **structured**

(we also consider maximal **total** quality, but this leads to worse results)
Structure in space

Let $g$ be the optimal partitioning of attribute $A$ into $k$ bins.

We observe

$$\sum_{i=1}^{k} \phi(b^i_g) = \phi(b^k_g) + \sum_{i=1}^{k-1} \phi(b^i_g)$$

This means that $\{b^1_g, ..., b^{k-1}_g\}$ is the optimal partitioning of $A \leq l^k_g$ into $k - 1$ bins.

We can use dynamic programming!
FLEXI, the algorithm

**Algorithm 1 FLEXI**

1: **Initialisation**
2: Initialise $\beta \ll |D|$ micro-bins
3: **2-partitions**
4: Compute 2-partition scores
5: **3 to $\beta$ partitions**
6: Use dynamic programming to compute best scores for partitions into 3 to $\beta$ parts
7: **Return**
8: Identify and return best result
FLEXI, the algorithm

**Algorithm 1 FLEXI**

1: Create initial disjoint bins \( \{c_1, \ldots, c_\beta\} \) of \( A \)
2: Create a double array \( qual[1 \ldots \beta][1 \ldots \beta] \)
3: Create an array \( b[1 \ldots \beta][1 \ldots \beta] \) to store bins
4: for \( i = 1 \rightarrow \beta \) do
5: \( b[1][i] = \bigcup_{k=1}^{i} c_k \) and \( qual[1][i] = \phi(b[1][i]) \)
6: end for
7: for \( \lambda = 2 \rightarrow \beta \) do
8: \hspace{1em} for \( i = \lambda \rightarrow \beta \) do
9: \hspace{2em} \( pos = \arg \max_{1 \leq j \leq i-1} \) \( qual[\lambda - 1][j] + \phi(\bigcup_{k=j+1}^{i} c_k) \)
10: \hspace{2em} \( qual[\lambda][i] = qual[\lambda - 1][pos] + \phi(\bigcup_{k=pos+1}^{i} c_k) \)
11: \hspace{2em} Copy all bins in \( b[\lambda - 1][pos] \) to \( b[\lambda][i] \)
12: \hspace{2em} Add \( \bigcup_{k=pos+1}^{i} c_k \) to \( b[\lambda][i] \)
13: \hspace{1em} end for
14: end for
15: \( \lambda^* = \arg \max_{1 \leq \lambda \leq \beta} \frac{1}{\lambda} qual[\lambda][\beta] \)
16: Return \( b[\lambda^*][\beta] \)

FLEXI can be used with **any** quality function \( \phi \)

To ensure **efficiency**, we need a smart way to compute \( \phi\left(\bigcup_{k=j}^{i} c_k\right) \)

For **five** measures we show how to do this
Instantiating FLEXI_w

Weighted Relative Accuracy

- standard quality measure for single binary target

\[
WRAcc(S) = \frac{s}{n} \left( \frac{s_+}{s} - \frac{n_+}{n} \right)
\]

Compares the ratios of positive samples \( \frac{s_+}{s} \) within subgroup \( S \) to that of the whole data, \( \frac{n_+}{n} \)

How can we efficiently pre-compute \( WRAcc \left( \bigcup_{k=j}^i c_k \right) \)?
Instantiating $FLEXI_w$

Pre-computing Weighted Relative Accuracies

1) $\textbf{for } i = 1 \rightarrow \beta \textbf{ do}$
   \begin{align*}
   &\quad \text{count}[i] = \text{number of positive labels in } D_{c_i} \\
   &\quad \text{compute } WRAcc(c_i) \text{ based on } \text{count}[i]
   \end{align*}
   $O(n)$

2) $\textbf{for } i = 2 \rightarrow \beta \textbf{ do}$
   \begin{align*}
   &\quad \theta = \text{count}[i] \\
   &\textbf{for } j = i - 1 \rightarrow 1 \textbf{ do}$
   \begin{align*}
   &\quad \theta = \theta + \text{count}[j] \\
   &\quad \text{set # of positive labels in } \bigcup_{k=j}^{i} c_k \text{ to } \theta \\
   &\quad \text{compute } WRAcc(\bigcup_{k=j}^{i} c_k)
   \end{align*}
   $O(\beta^2)$

Done!
Instantiating FLEXI

We show how to instantiate

- $\text{FLEXI}_w$ with WRAcc at $O(n + \beta^2)$
- $\text{FLEXI}_z$ with Z-score at $O(n + \beta^2)$
- $\text{FLEXI}_h$ with Hellinger distance at $O(n\beta^2d)$
- $\text{FLEXI}_k$ with Kullback Leibler at $O(n\beta^2d)$
- $\text{FLEXI}_q$ with quadratic divergence at $O(n^2d)$

As $\beta$ is typically small, between 5 to 40, the first four scale \textit{linear} in $n$

(see the paper for details on how to speed up $\text{FLEXI}_q$ using sampling)
Experiments show that FLEXI outperforms the state of the art in **quality**, **flexibility**, and **efficiency**.
Experiments show that \textsc{flexi} outperforms the state of the art in \textbf{quality}, \textbf{flexibility}, and \textbf{efficiency}.

<table>
<thead>
<tr>
<th>Data</th>
<th>FLEXI\textsubscript{w}</th>
<th>EF</th>
<th>EW</th>
<th>SD</th>
<th>UD</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>0.08 (100)</td>
<td>0.07 (88)</td>
<td>0.07 (88)</td>
<td>0.07 (88)</td>
<td>0.06 (75)</td>
<td>0.07 (88)</td>
</tr>
<tr>
<td>Cover</td>
<td>0.12 (100)</td>
<td>0.04 (33)</td>
<td>0.08 (66)</td>
<td>0.04 (33)</td>
<td>0.05 (42)</td>
<td>0.04 (33)</td>
</tr>
<tr>
<td>Bank</td>
<td>0.04 (100)</td>
<td>0.02 (50)</td>
<td>0.03 (75)</td>
<td>0.02 (50)</td>
<td>0.02 (50)</td>
<td>0.02 (50)</td>
</tr>
<tr>
<td>Network</td>
<td>0.18 (100)</td>
<td>0.10 (56)</td>
<td>0.12 (67)</td>
<td>0.14 (78)</td>
<td>0.12 (67)</td>
<td>0.14 (78)</td>
</tr>
<tr>
<td>Drive</td>
<td>0.11 (100)</td>
<td>0.03 (27)</td>
<td>0.08 (73)</td>
<td>0.05 (45)</td>
<td>0.06 (55)</td>
<td>0.05 (45)</td>
</tr>
<tr>
<td>Year</td>
<td>0.12 (100)</td>
<td>0.06 (50)</td>
<td>0.06 (50)</td>
<td>0.07 (58)</td>
<td>0.06 (50)</td>
<td>0.07 (58)</td>
</tr>
</tbody>
</table>

Average quality for top-50 subgroups (WRAcc)
Experiments show that FLEXI outperforms the state of the art in quality, flexibility, and efficiency.

<table>
<thead>
<tr>
<th>Data</th>
<th>FLEXI (_k)</th>
<th>SUM</th>
<th>EF</th>
<th>EW</th>
<th>SD</th>
<th>IPD</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>100</td>
<td>38</td>
<td>37</td>
<td>31</td>
<td>n/a</td>
<td>4</td>
<td>n/a</td>
</tr>
<tr>
<td>Cover</td>
<td>100</td>
<td>43</td>
<td>64</td>
<td>75</td>
<td>n/a</td>
<td>45</td>
<td>n/a</td>
</tr>
<tr>
<td>Bank</td>
<td>100</td>
<td>46</td>
<td>62</td>
<td>33</td>
<td>n/a</td>
<td>6</td>
<td>n/a</td>
</tr>
<tr>
<td>Network</td>
<td>100</td>
<td>55</td>
<td>68</td>
<td>55</td>
<td>n/a</td>
<td>21</td>
<td>n/a</td>
</tr>
<tr>
<td>Drive</td>
<td>100</td>
<td>42</td>
<td>64</td>
<td>85</td>
<td>89</td>
<td>42</td>
<td>62</td>
</tr>
<tr>
<td>Year</td>
<td>100</td>
<td>43</td>
<td>45</td>
<td>42</td>
<td>40</td>
<td>42</td>
<td>74</td>
</tr>
</tbody>
</table>

Average quality for top-50 subgroups
(Kullback-Leibler divergence)
Experiments show that FLEXI outperforms the state of the art in *quality*, *flexibility*, and *efficiency*.

<table>
<thead>
<tr>
<th>Data</th>
<th>FLEXI(_q)</th>
<th>SUM</th>
<th>EF</th>
<th>EW</th>
<th>IPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>100</td>
<td>18</td>
<td>7</td>
<td>8</td>
<td>23</td>
</tr>
<tr>
<td>Cover</td>
<td>100</td>
<td>60</td>
<td>41</td>
<td>39</td>
<td>53</td>
</tr>
<tr>
<td>Bank</td>
<td>100</td>
<td>31</td>
<td>47</td>
<td>59</td>
<td>66</td>
</tr>
<tr>
<td>Network</td>
<td>100</td>
<td>48</td>
<td>69</td>
<td>64</td>
<td>56</td>
</tr>
<tr>
<td>Drive</td>
<td>100</td>
<td>62</td>
<td>41</td>
<td>59</td>
<td>66</td>
</tr>
<tr>
<td>Year</td>
<td>100</td>
<td>26</td>
<td>27</td>
<td>21</td>
<td>55</td>
</tr>
</tbody>
</table>

Average quality for top-50 subgroups
(Quadratic divergence)
Experiments

Experiments show that FLEXI outperforms the state of the art in **quality**, **flexibility**, and **efficiency**.

Relative runtime to mine top-50 subgroups
Conclusions

We studied how to efficiently discover high quality binary features for subgroup discovery.

In short, **FLEXI**
- discovers binary features with maximal average quality
- highly flexible, operates with any objective function
- efficient due to dynamic programming
- complexity depends on $\phi$, yet often linear in size of the data

Future work
- feature construction to allow sampling high quality subgroups

(source code available at: eda.mmci.uni-saarland.de/flexi)
Thank you!

We studied how to efficiently discover high quality binary features for subgroup discovery.

In short, **Flexi**
- discovers binary features with maximal average quality
- highly flexible, operates with any objective function
- efficient due to dynamic programming
- complexity depends on $\phi$, yet often linear in size of the data

Future work
- feature construction to allow sampling high quality subgroups

(source code available at: eda.mmci.uni-saarland.de/flexi)