Entire Regularization Paths for Graph Data

Max Planck Institute for Biological Cybernetics

Koji Tsuda
Graph Regression

Training

- $(?, -0.2)$
- $(?, 0.7)$
- $(?, -0.5)$

Test

- $(?, ?)$
Substructure Representation

- 0/1 vector of pattern indicators
- Huge dimensionality!
- Need feature selection

(patterns)

(0, ..., 0, 1, 0, ..., 0, 1, 0, ...)

(A A)

(A A)

(B A)
Overview

Entire regularization paths
- LARS-LASSO (Efron et al., 2004), L1SVM
- Forward selection of features
- Trace the solution trajectory of L1-regularized learning

Path following algorithm for graph data
- Feature search -> pattern search
- Branch-and-bound algorithm
- DFS code tree, New Bound
Path Following Algorithms

- **LASSO regression**
  \[ \beta(\lambda) = \arg\min_{\beta} L(y, X\beta) + \lambda \|\beta\|_1. \]

- Follow the complete trajectory of \( \beta(\lambda) \)
  - \( \lambda \): Infinity to Zero

- **Active feature set** \( \mathcal{A} \)
  - Features corresponding to nonzero weights
Piecewise Linear Path

At a turning point,

- A new feature included into the active set,
  or
- An existing feature excluded from the active set
Practical Merit of Path Following

- Cross validation by grid search
  - Has to solve QP many times
  - Especially time-consuming for graph data

- Path following does not include QP
- Determine the CV-optimal regularization parameter in the finest precision
Pseudo code of path following

- Set initial point $\beta$ and direction $\gamma$
- Do
  - $d_1 =$ Step size if next event is inclusion
  - $d_2 =$ Step size if next event is exclusion
  - $d = \min(d_1,d_2)$
  - $\beta = \beta + d\gamma$
  - Update the active feature set
  - Set the next direction $\gamma$
- Until all features are included
Feature space of patterns

- Graph training data \( G = \{G_i\}_{i=1}^{n} \)
- Set of all subgraphs (patterns) \( T \)
- Each graph is represented as

\[
\mathbf{x}_i = (x_{it})_{t \in T}, \quad x_{it} = I(t \subseteq G_i)
\]
Main Search problem

\[ d_t = \min \left\{ \frac{\rho_0 - \sum w_i x_{it}}{\eta_0 - \sum v_i x_{it}}, \frac{\rho_0 + \sum w_i x_{it}}{\eta_0 + \sum v_i x_{it}} \right\}. \]

\( w_i, v_i, \rho_0, \eta_0 \) : constants computed from active set

\[ \text{Find pattern } \ t \in \mathcal{T} \text{ that minimizes } d_t \]
Tree-shaped Search Space

- Each node has a pattern
- Generate nodes from the root:
  - Add an edge at each step
Tree Pruning

If it is guaranteed that the optimal pattern is not in the downstream, the search tree can be pruned.
**Theorem (Pruning condition)**

- Traversed up to pattern $t$
- $d_t^*$: Minimum value so far
- No better pattern in the downstream, if

$$b_w + d_t^* b_v < |\rho_0| - d_t^* |\eta_0|.$$  

where

$$b_w = \max \left\{ \sum_{w_i < 0} |w_i| x_{it}, \sum_{w_i > 0} |w_i| x_{it} \right\}.$$
Reusing the search space

- Main search is solved repeatedly with different parameters
- More efficient to reuse the search space in next iterations
  - Node generation is expensive due to the minimum DFS code check
- Whole tree of patterns is kept in memory and progressively extended
Experiments

Naïve Method

- Enumerate all patterns whose edge size is smaller than $\text{maxpat}$
- Then, LAR-LASSO is applied

CPDB dataset

- 683 training graphs (chemical compounds)
- Classification dataset (mutagenetic or not)
- Converted to regression problem ($y=1,-1$)
How to measure the computational cost of our method

- Data divided into 90% train and 10% validation
- Record
  - Number of nodes in tree
  - Computation time

at the point of minimum validation error

![Diagram](image)
## Computational Cost

<table>
<thead>
<tr>
<th>maxpat</th>
<th>Naive</th>
<th>Progressive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#nodes</td>
<td>time</td>
</tr>
<tr>
<td>5</td>
<td>2142</td>
<td>0.51</td>
</tr>
<tr>
<td>6</td>
<td>5717</td>
<td>1.39</td>
</tr>
<tr>
<td>7</td>
<td>14309</td>
<td>3.79</td>
</tr>
<tr>
<td>8</td>
<td>33862</td>
<td>9.93</td>
</tr>
<tr>
<td>9</td>
<td>75814</td>
<td>25.64</td>
</tr>
<tr>
<td>10</td>
<td>161858</td>
<td>65.60</td>
</tr>
<tr>
<td>11</td>
<td>332553</td>
<td>164.20</td>
</tr>
<tr>
<td>12</td>
<td>665202</td>
<td>397.95</td>
</tr>
<tr>
<td>13</td>
<td>1302273</td>
<td>931.61</td>
</tr>
<tr>
<td>∞</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Solution Path

![Solution Path Graph]

- Iteration axis
- Value range from 0 to 0.15
- Multi-colored lines representing different iterations
- Graph title: Solution Path
Events
Conclusion

- Path following implemented for graph data
- Pattern search by the DFS code tree
- Hinge loss: To do
  - Search criterion more complicated
- Easily combined with itemset mining, tree mining, sequence mining
gboost MATLAB toolbox

- Graph classification by LPBoost + DFS Code Tree
- Includes an implementation of gspan
- www.kyb.mpg.de/people/nowozin/gboost
- Path following code will be available soon