Nearest Neighbor Sampling for Cross Company Defect Predictors

Burak Turhan, Ayse Bener, (Bogazici University, Turkey)
Tim Menzies (WVU, USA)

DEFECTS'08
Seattle, USA
Is PROMISE useful?
http://promisedata.org

- Repository
  - Data
  - Papers

- Annual meetings
  - ICSE’09
  - Boetticher, Guhe, Menzies, Ostrand

- Journal special issues
  - IEEE software
  - Journal Emp. SE

- Arguably: good science
- Arguably: bad science
- But what generality?
Generality in SE

• Do results from project X in company A…
  – Apply to project Y in company B?
  – If yes, then can use imported data
    • And the PROMISE data becomes very useful
    • And no need for tedious local data collection
  – If no, then must use local data
    • And no generality in SE
    • PROMISE is a playground, useful for sharpening our pencils

• This talk
  – Cost/benefits of local vs imported data for defect prediction
    • While local is much better…
    • But, with a little row selection, imported data surprisingly useful
Estimating post-release faults

We have a big problem!

Need more domain understanding!

Lack of generality if using naïve measures when

Estimation of pre-release issue reports

Suprisingly high levels of repetabability

? Less environment change

Hypothesis: intra-development team properties easier to estimate that pre-post-release
Setting up

data,
features
learners,
performance measures
Data
(the usual suspects, plus 3)

- [http://promisedata.org](http://promisedata.org)
- Seven NASA data sets (ground and flight systems)
- Three new data sets from Turkish whitegoods
  - Held in reserve, tested later

<table>
<thead>
<tr>
<th>source</th>
<th>project</th>
<th>language</th>
<th>description</th>
<th>examples</th>
<th>features</th>
<th>loc</th>
<th>%defective</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>pc1</td>
<td>C++</td>
<td>Flight software for earth orbiting satellite</td>
<td>1,109</td>
<td>21</td>
<td>25,924</td>
<td>6.94</td>
</tr>
<tr>
<td>NASA</td>
<td>kc1</td>
<td>C++</td>
<td>Storage management for ground data</td>
<td>845</td>
<td>21</td>
<td>42,965</td>
<td>15.45</td>
</tr>
<tr>
<td>NASA</td>
<td>kc2</td>
<td>C++</td>
<td>Storage management for ground data</td>
<td>522</td>
<td>21</td>
<td>19,259</td>
<td>20.49</td>
</tr>
<tr>
<td>NASA</td>
<td>cm1</td>
<td>C++</td>
<td>Spacecraft instrument</td>
<td>498</td>
<td>21</td>
<td>14,763</td>
<td>9.83</td>
</tr>
<tr>
<td>NASA</td>
<td>kc3</td>
<td>JAVA</td>
<td>Storage management for ground data</td>
<td>458</td>
<td>39</td>
<td>7749</td>
<td>9.38</td>
</tr>
<tr>
<td>NASA</td>
<td>mw1</td>
<td>C++</td>
<td>A zero gravity experiment related to combustion</td>
<td>403</td>
<td>37</td>
<td>8341</td>
<td>7.69</td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar4</td>
<td>C</td>
<td>Embedded controller for white-goods</td>
<td>107</td>
<td>30</td>
<td>9196</td>
<td>18.69</td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar3</td>
<td>C</td>
<td>Embedded controller for white-goods</td>
<td>63</td>
<td>30</td>
<td>5624</td>
<td>12.70</td>
</tr>
<tr>
<td>NASA</td>
<td>mc2</td>
<td>C++</td>
<td>Video guidance system</td>
<td>61</td>
<td>39</td>
<td>6134</td>
<td>32.29</td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar5</td>
<td>C</td>
<td>Embedded controller for white-goods</td>
<td>36</td>
<td>30</td>
<td>2732</td>
<td>22.22</td>
</tr>
</tbody>
</table>

4,102
Features

• Q: Why just static code features? Why not:
  – Churn? [Nikora & Munson]
  – Details on development team? [Nagappan et al ICSE’08]
  – Requirements features? [Jian, Cukic, Menzies, ISSRE’07]

• A: beside the point.
  – This report is one study of local vs imported data.
  – Future work: repeat for other kinds of data
Learner

- Naïve Bayes (log filtering on the numerics)
- Why? Because nothing (yet) found demonstrably better for these data sets
Performance reporting

- N-way cross-val
- PD (a.k.a. recall), PF
- Balance: \( \text{balance} = \text{bal} = 1 - \frac{\sqrt{(0 - pf)^2 + (1 - pd)^2}}{\sqrt{2}} \)
- Not precision: unstable for small targets

Quartile charts: 0% \(\cdots\) median \(\cdots\) 100%

\[\{4, 7, 15, 20, 31, 40, 52, 64, 70, 81, 90\}\]
Experiments

results,
implications
Experiment #1: local vs imported

- Repeat 20 times
- For NASA data
  - Seven test sets from 10% of each source
- Treatment 1 (using imported)
  - Train on the 6 other data sets
- Treatment 2 (using local)
  - Train on the remaining 90% of the local data
**Experiment #1: results**

<table>
<thead>
<tr>
<th>treatment</th>
<th>min</th>
<th>Q1</th>
<th>median</th>
<th>Q3</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>pd CC</td>
<td>50</td>
<td>83</td>
<td>97</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>WC</td>
<td>17</td>
<td>63</td>
<td>75</td>
<td>82</td>
<td>100</td>
</tr>
<tr>
<td>pf CC</td>
<td>14</td>
<td>53</td>
<td>64</td>
<td>91</td>
<td>100</td>
</tr>
<tr>
<td>WC</td>
<td>0</td>
<td>24</td>
<td>29</td>
<td>36</td>
<td>73</td>
</tr>
</tbody>
</table>

- **WC** = within-company local data
  - Cost: Lower PDs
  - Benefit: PFs much less

- **CC** = cross-company imported data
  - Benefit: massive increase in PD (highest ever seen)
  - Cost: large increase in PF.
  - Too many imported irrelevancies? (go to exp #2)
Experiment #2: local vs (imported+NN)

- Repeat 20 times

- Seven test sets from 10% of each source
- Treatment 1 (using imported)
  - Train on the 6 other data sets
- Treatment 2 (using local)
  - Train on the remaining 90% of the local data

- Treatment 3 (using imported+NN)
  - Initialize train set with 6 other data sets,
  - Prune the train set to just the 10 nearest neighbors (Euclidean) of the test set (discarding repeats)
Experiment #2: PD results

- **At best:** median PD of (imported+NN) falls halfway (ish) between imported and local
- **At worst:** PD of (imported+NN) worse
- But, always, variance in imported+NN very small

CC = imported; NN = imported+NearNeigh; WC = local
Experiment #2: PF results

- **At best**: median PF of (imported+NN) falls half-way (ish) between imported and local
- **At worst**: PF of (imported+NN) worse than local (but much less than imported data)
- Again, imported+NN variance very small

<table>
<thead>
<tr>
<th>rank</th>
<th>quartiles 0 25 50 75 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>1  WC 16 29 33 38 49</td>
</tr>
<tr>
<td></td>
<td>2  NN 40 43 44 45 46</td>
</tr>
<tr>
<td></td>
<td>3  CC 90 91 91 91 93</td>
</tr>
<tr>
<td>MW1</td>
<td>1  WC 8 21 26 29 47</td>
</tr>
<tr>
<td></td>
<td>2  NN 30 32 33 33 36</td>
</tr>
<tr>
<td></td>
<td>3  CC 67 68 68 69 70</td>
</tr>
<tr>
<td>PC1</td>
<td>1  WC 16 24 28 31 40</td>
</tr>
<tr>
<td></td>
<td>2  NN 45 48 48 49 53</td>
</tr>
<tr>
<td></td>
<td>3  CC 94 94 94 94 94</td>
</tr>
<tr>
<td>KC1</td>
<td>1  NN 22 23 24 25 27</td>
</tr>
<tr>
<td></td>
<td>2  WC 26 32 35 37 43</td>
</tr>
<tr>
<td></td>
<td>3  CC 59 60 60 60 60</td>
</tr>
<tr>
<td>KC2</td>
<td>1  NN 24 25 25 25 27</td>
</tr>
<tr>
<td></td>
<td>1  WC 10 21 26 31 40</td>
</tr>
<tr>
<td></td>
<td>2  CC 67 67 67 67 67</td>
</tr>
<tr>
<td>KC3</td>
<td>1  NN 17 18 18 19 20</td>
</tr>
<tr>
<td></td>
<td>2  WC 7 17 21 26 31</td>
</tr>
<tr>
<td></td>
<td>3  CC 26 27 27 27 27</td>
</tr>
<tr>
<td>MC2</td>
<td>1  NN 29 30 31 32 35</td>
</tr>
<tr>
<td></td>
<td>2  WC 0 27 36 45 73</td>
</tr>
<tr>
<td></td>
<td>3  CC 71 71 71 71 71</td>
</tr>
</tbody>
</table>

CC= imported; NN=imported+NearNeigh; WC=local
• Can’t recommend imported+NN over local
  – Unless you are concerned with stability

• Imported+NN patches the problems with imported
  – Lowers the bad PFs
  – But can also lower PD

• But, if you have no local data,
  – You can get by with imported+NN

• Recommend a two phase approach
  – Start with imported+NN
  – Meanwhile, initiate a local data collection program

• Question: how long will you suffer with imported+NN?
  – How much local data do you need to collect?
  – Go to experiment #3
Experiment #3: Incremental learning

- Repeat 20 times
- Seven test sets from 10% of each source
- Treatment (using local)
  - Train on the 6 other data sets, in buckets of size 100, 200, 300, etc

Mann-Whitney:
- $Kc1, pc1$: no improvement after $|train| = 200$
- Rest: no improvement after $|train| = 100$
Generality

- The above patterns seen in NASA aerospace applications (pc1,kc1,kc3,cm1,kc3,mw1,mc2)
  - Repeat in Turkish whitegoods software (ar3,ar4,ar5)
  - Note: very different development methodologies

<table>
<thead>
<tr>
<th>source</th>
<th>project</th>
<th>language</th>
<th>description</th>
<th>(# modules)</th>
<th>examples</th>
<th>features</th>
<th>loc</th>
<th>%defective</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>pc1</td>
<td>C++</td>
<td>Flight software for earth orbiting satellite</td>
<td>1,109</td>
<td>21</td>
<td>25,924</td>
<td>6.94</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>kc1</td>
<td>C++</td>
<td>Storage management for ground data</td>
<td>845</td>
<td>21</td>
<td>42,965</td>
<td>15.45</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>kc2</td>
<td>C++</td>
<td>Storage management for ground data</td>
<td>522</td>
<td>21</td>
<td>19,259</td>
<td>20.49</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>cm1</td>
<td>C++</td>
<td>Spacecraft instrument</td>
<td>498</td>
<td>21</td>
<td>14,763</td>
<td>9.83</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>kc3</td>
<td>JAVA</td>
<td>Storage management for ground data</td>
<td>458</td>
<td>39</td>
<td>7749</td>
<td>9.38</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>mw1</td>
<td>C++</td>
<td>A zero gravity experiment related to combustion</td>
<td>403</td>
<td>37</td>
<td>8341</td>
<td>7.69</td>
<td></td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar4</td>
<td>C</td>
<td>Embedded controller for white-goods</td>
<td>107</td>
<td>30</td>
<td>9196</td>
<td>18.69</td>
<td></td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar3</td>
<td>C</td>
<td>Embedded controller for white-goods</td>
<td>63</td>
<td>30</td>
<td>5624</td>
<td>12.70</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>mc2</td>
<td>C++</td>
<td>Video guidance system</td>
<td>61</td>
<td>39</td>
<td>6134</td>
<td>32.29</td>
<td></td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar5</td>
<td>C</td>
<td>Embedded controller for white-goods</td>
<td>36</td>
<td>30</td>
<td>2732</td>
<td>22.22</td>
<td></td>
</tr>
</tbody>
</table>

4,102
What have we learned?

summary,
conclusions
Summary

• Experiment #1
  – cautioned against using unfiltered imported data

• Experiment #2:
  – tested a filter based on nearest neighbor
  – adequate stop-gap till local data available

• Experiment #3:
  – Stop-gap can be very short (less than 200 modules)
Conclusions: generality in SE

- Do results from project X in company A…
  - Apply to project Y in company B?
  - If yes, then can use imported data:
    - And the PROMISE data sets are more than just grad student playgrounds
  - If no, then must use local data:
    - no generality in SE

- At least for defect prediction from static code attributes
  - Local data yields best median performance (pd,pf) but worse variance
  - Imported data covers more cases,
    - but includes irrelevancies
  - Irrelevant sections can be pruned (NN= nearest neighbor)
    - produce predictors close (but not as good) as local data
  - You can use imported data (with NN) as a stop gap
    - And that stop gap need not be very lengthy

- As for other kinds of data….
  - Effort estimation: jury is out (Kitchenham ‘08)
  - #include futureWork.h
Questions, comments?
Implications of ceiling effects

- Maybe, the era of throwing learners at static code attributes is over
  - R.I.P.
- In the future, it may be better to improve the data
  - rather than improve the learners
- E.g.
  - filtering irrelevances (this work)
  - add details on development team [Nagappan et al ICSE’08]
  - add requirements features [Jian, Cukic, Menzies]