A Toolbox for Software Mining

Rahul Premraj
Tom Zimmermann
Project #1
Quality of bug reports

What makes a good bug report?

How do they differ from bad ones?

Related:
Cool Paper Title
[Your names here, Big SE conference]
Project #2
Predicting lifetime of bug reports

How long will a bug exist?

What influences its time to fix?

Related:
Predicting Eclipse Bug Lifetimes
[Panjer, Mining Challenge 2007]

9 days

???

days
Project #3
Detecting duplicates of bug reports

Is a new bug report a duplicate of another?

Related:
Detection of Duplicate Defect Reports Using Natural Language Processing [Runeson et al., ICSE 2007]
Project #4
Triaging bug reports

How can we assign bugs to developers?

Related:
Who Should Fix This Bug?
[Anvik et al., ICSE 2006]
Project #5
Visualizing bug reports (networks)

How can we visualize individual bugs?

How can we visualize all bugs?

Related:
Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships [D’Ambros et al., CSMR 2006]
Project #6
Predicting defects with spam filters

Which files have bugs?
Which ones don’t?

Related:
Spam Filter Based Approach for Finding Fault-Prone Software Modules
[Mizuno et al., MSR 2007]
Pattern mining

Mining Version Histories to Guide Software Changes (Lecture 3)
Xelopes Library

http://www.prudsys.com/Produkte/Algorithmen/Xelopes/

Commercial?
The call relation is easy to obtain by disassembling executable C code.
The call relation is easy to obtain by disassembling executable C code.

Blocks are patterns!
The call relation is easy to obtain by disassembling executable C code.
The call relation is easy to obtain by disassembling executable C code.
The call relation is easy to obtain by disassembling executable C code.

{sync, lock, unlock} has support = 3 and size = 3
Many possible patterns
Concept analysis
Concept analysis

Concept Analysis computes all blocks (patterns)
Concept analysis

Concept Analysis computes all blocks (patterns)
Concept analysis

*Concept Analysis computes all blocks (patterns)*
Support vs. Size

Slide provided by C. Lindig
Support vs. Size

Support decreases monotonically from top to bottom.
Support vs. Size

Support decreases monotonically from top to bottom.

Support can be used as a cut-off criterion (support $\geq 3$).

Slide provided by C. Lindig
Support vs. Size

Support decreases monotonically from top to bottom.

Support can be used as a cut-off criterion (support ≥ 3).

Slide provided by C. Lindig
Support vs. Size

Support decreases monotonically from top to bottom.

Support can be used as a cut-off criterion (support \(\geq 3\)).

Size increases monotonically from top to bottom.
### Example patterns

<table>
<thead>
<tr>
<th>Subject</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruby CVS</td>
<td>va_end   va_start</td>
</tr>
<tr>
<td>Linux 2.6 (s)</td>
<td>mutex_lock mutex_unlock</td>
</tr>
<tr>
<td>Linux 2.6 (xs)</td>
<td>kmem_cache_alloc kmem_cache_free</td>
</tr>
<tr>
<td>Python SVN</td>
<td>PyErr_SetString PyExc_TypeError</td>
</tr>
<tr>
<td>Ruby CVS</td>
<td>rb_fix2int rb_num2int</td>
</tr>
</tbody>
</table>

Slide provided by C. Lindig
Decision trees
# Decision trees

## Play golf dataset

<table>
<thead>
<tr>
<th>Independent variables</th>
<th>Dep. var</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTLOOK</td>
<td>TEMPERATURE</td>
</tr>
<tr>
<td>sunny</td>
<td>85</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
</tr>
<tr>
<td>rain</td>
<td>75</td>
</tr>
<tr>
<td>sunny</td>
<td>75</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
</tr>
</tbody>
</table>
Decision trees

Dependent variable: PLAY

OUTLOOK?
- sunny
  - HUMIDITY?
    - $\leq 70$
      - Play 2
      - Don't Play 0
    - $> 70$
      - Play 0
      - Don't Play 3
- overcast
  - Play 4
  - Don't Play 0
- rain
  - WINDY?
    - TRUE
      - Play 0
      - Don't Play 2
    - FALSE
      - Play 3
      - Don't Play 0
Weka

http://www.cs.waikato.ac.nz/ml/weka/
WinMine Toolkit

http://research.microsoft.com/~dmax/winmine/tooldoc.htm
Classification with SVM

Dataset:
- binary=i.dll  depends_on=h.dll m.dll  failures=0
- binary=m.dll  depends_on=a.exe d.dll  failures=0
- binary=h.dll  depends_on=e.exe j.exe  failures=0
- binary=d.dll  depends_on=i.dll e.exe  failures=1
- binary=l.dll  depends_on=g.exe b.dll  failures=0
- binary=b.dll  depends_on=m.dll i.dl k.dlll  failures=0
- binary=g.exe  depends_on=m.dll  failures=0
The black box: SVMs
SVM: Maximal margins
SVMs: Predictions

[Diagram showing SVM hyperplane and data points categorized as 'Failure-free' and 'Failure-prone'.]
SVMs: Non-linear data
SVMs: Non-linear data

Kernel trick!
Precision and Recall

High precision = returned elements are relevant
High recall = relevant elements are returned
## Precision and Recall

<table>
<thead>
<tr>
<th>Actually has property</th>
<th>Predicted to have property</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

### Precision

\[
\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}
\]

### Recall

\[
\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}
\]

### Accuracy

\[
\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{FN} + \text{TN})}
\]
Random splits
Random splits

Random 2/3 of binaries

Training
(build a model)
Random splits

Random 2/3 of binaries

Training
(build a model)

Testing
(assess the model)

Remaining 1/3 of binaries
Random splits

Random 2/3 of binaries

Training
(build a model)

1000×

Testing
(assess the model)

Remaining 1/3 of binaries
Precision
Precision

2 out of 3 binaries predicted as failure-prone are observed as failure-prone.
Recall
Recall

3 out of 4 binaries observed as failure-prone are predicted as failure-prone
Visualization

http://www.inf.unisi.ch/phd/dambros/tools/
yEd - Java Graph Editor

Regression Analysis
k-Nearest Neighbour

What describes the two?
k-Nearest Neighbour

What describes the two?

taste  colour
size   smell
shape  texture
k-Nearest Neighbour

So what is this one?
k-Nearest Neighbour

So what is this one?
k-Nearest Neighbour

So what is this one?

- taste
- colour
- size
- smell
- shape
- texture
k-Nearest Neighbour

So what is this one?

- taste
- colour
- size
- smell
- shape
- texture

- taste
- colour
- size
- smell
- shape
- texture
The car mechanic can infer that if the ignition doesn’t turn on, there might be a problem with the spark plug.

Court case A is similar to my case B. Can I reuse it’s solution to win mine?

Code Reuse ;-)
k-Nearest Neighbour
k-Nearest Neighbour
k-Nearest Neighbour
k-Nearest Neighbour

Measures of Similarity: colour, shape, size, texture, taste, smell

Input: apple

Output: similarity to other fruits
k-Nearest Neighbour

Measure Similarity

taste

colour

size

shape

smell

texture

Classification
**k-Nearest Neighbour**

What about *prediction* problems?

Recall the **Desharnais** data set from the Exercises...
What about prediction problems?

Recall the Desharnais data set from the Exercises...
k-Nearest Neighbour

What about prediction problems?

Recall the Desharnais data set from the Exercises...
k-Nearest Neighbour

What about prediction problems?

Recall the Desharnais data set from the Exercises...
k-Nearest Neighbour

What about prediction problems?

Recall the Desharnais data set from the Exercises...

![Diagram with connections to projects and their similarity scores]

- Project A: 0.64
- Project B: 0.32
- Project C: 0.78
- Project D: 0.82
- Project E: 0.53
**k-Nearest Neighbour**

What about **prediction** problems?

Recall the **Desharnais** data set from the Exercises...

---

**Project A**

0.64

**Project B**

0.32

**Project C**

0.78

**Project D**

0.82

**Project E**

0.53

---

**k-NN (k=3)**

- Project D
- Project C
- Project A
**k-Nearest Neighbour**

What about **prediction** problems?

Recall the **Desharnais** data set from the Exercises...

![Diagram showing project recommendations based on k-NN (k=3) algorithm]

- Project A: 0.64
- Project B: 0.32
- Project C: 0.78
- Project D: 0.82
- Project E: 0.53

New Project

**k-NN (k=3)**

- Project D
- Project C
- Project A

**Average the solution!**
k-Nearest Neighbour
Evaluation of Results
Evaluation of Results

Actual Effort
Evaluation of Results

Actual Effort

Predicted Effort
Evaluation of Results

Actual Effort

Predicted Effort

Overestimation
Evaluation of Results

Actual Effort

Predicted Effort

Over-estimation

Actual Effort
Evaluation of Results

Actual Effort

Predicted Effort

Actual Effort

Predicted Effort

Over-estimation
Evaluation of Results

Actual Effort       Predicted Effort

Actual Effort       Predicted Effort

Over-estimation

Under-estimation
Evaluation of Results

- Actual Effort
- Predicted Effort

Over-estimation
Under-estimation

Average of Absolute Residuals
Evaluation of Results

Actual Effort
Predicted Effort

Over-estimation
Under-estimation

Average of Absolute Residuals

\[ \frac{+ \ldots +}{n} \]

\( n = \text{number of predictions made} \)
Evaluation of Results

Pred(x)
% of predictions that lie within x% of Actual Effort
Evaluation of Results

Pred(x)
% of predictions that lie within x% of Actual Effort
Evaluation of Results

Pred(x)
% of predictions that lie within x% of Actual Effort
Evaluation of Results

Pred(x) % of predictions that lie within x% of Actual Effort

- **Bug 1**
- **Bug 2**
- **Bug 3**
- **Bug 4**
- **Bug 5**
- **Bug 6**
Evaluation of Results

Pred(x)
% of predictions that lie within x% of Actual Effort

4 of 6 predictions made lie within x% of Actual Effort
Pred(x) = 66.67%
Clustering
Clustering
Clustering
Clustering
Clustering
Bayesian Filtering
Bayesian Filtering
Bayesian Filtering
Bayesian Filtering
Bayesian Filtering

Congratulations! You won the lottery...

$23 million in your custody

Nigeria

SPAM

DEARBORN Brand
Boneless HAM
Water Added
Bayesian Filtering

Congratulations! You won the lottery...

$23 million in your custody

Congratulations! You won the competition...

Nigeria

SPAM

DEANPORN

Boneless HAM