Defect Content Estimation for Inspections:
Regression and Machine Learning

Frank Padberg
Universität Karlsruhe
Germany
Our Task

reliably estimate

the number of defects in a software document

from the outcome of an inspection!
Estimation Methods

- capture–recapture methods (Eick et al. ICSE 1992)
- curve–fitting methods (Wohlin et al. ICSE 1998)
- studies show that estimates are far too unreliable (Briand et al. TSE 2000, Biffl et al. ICSE 2001)
- interval estimate method (Padberg ICSE 2002)
- outperforms other methods on benchmark dataset
Interval Estimate Method

• use empirical data from past inspections for estimating

• stochastic model relates inspection outcome (the $w_k$ and $d$) to the true number $N$ of defects

• use that relation to estimate $N$ for a new document from its inspection outcome
Regression Approach

- learn relationship between observable features of an inspection and true number of defects contained in the document
Regression Approach

- learn relationship between observable features of an inspection and true number of defects contained in the document
- view defect content estimation as a regression problem
Regression Approach

• learn relationship between observable features of an inspection and true number of defects contained in the document

• view defect content estimation as a regression problem

• again, need empirical database
Candidate Features

- derived from zero–one matrix
- use the $w_k$ and $d$ to get: TDD, AVE, MIN, MAX, STD
- example A1:
  \[(9, 7, 6, 13, 9, 6)\] and 23 yields

<table>
<thead>
<tr>
<th>TDD</th>
<th>AVE</th>
<th>MIN</th>
<th>MAX</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>8.3</td>
<td>6</td>
<td>13</td>
<td>2.4</td>
</tr>
</tbody>
</table>
Input Data for Linear Regression

- correlation analysis yields ranking
  
  \[ TDD > AVE > MIN > MAX > STD \]

- some datapoints:

<table>
<thead>
<tr>
<th>inspection</th>
<th>TDD</th>
<th>AVE</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>23</td>
<td>8.3</td>
<td>30</td>
</tr>
<tr>
<td>B1</td>
<td>20</td>
<td>6.0</td>
<td>28</td>
</tr>
<tr>
<td>C1</td>
<td>10</td>
<td>3.2</td>
<td>18</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>1.3</td>
<td>15</td>
</tr>
</tbody>
</table>
Regression Hyperplane

defects

all 16 inspections

© Frank Padberg 2002
Jackknife Validation

- leave out an inspection from the database
- compute the regression hyperplane using the remaining 15 inspections
- compute the regression estimate for the one inspection which was left out
- compare the estimate with the true value of the number of defects
Linear Regression Estimates

jackknife error of 11 percent
clearly outperforms capture–recapture

(11 percent versus 24)
Linear Regression versus Interval Estimates

similar performance on one half of the dataset

(7 percent each)
Non-Linear Regression: Neural Networks

\[
\text{logist}(x) = \frac{1}{1 + e^{-x}} \quad s_i = \text{logist} \left( \sum_j w_{ji} \cdot s_j \right)
\]
Neural Network Methodology

- determine a set of candidate features
- select an appropriate subset (feature selection)
- train different neural networks on the dataset
- select the best neural network (model selection)
Input Data for Non-Linear Regression

- non-linear feature selection yields ranking
  \[ TDD > STD > MAX > MIN > AVE \]

- STD instead of AVE

- some training patterns:

<table>
<thead>
<tr>
<th>inspection</th>
<th>TDD</th>
<th>STD</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>23</td>
<td>2.4</td>
<td>30</td>
</tr>
<tr>
<td>B1</td>
<td>20</td>
<td>1.7</td>
<td>28</td>
</tr>
<tr>
<td>C1</td>
<td>10</td>
<td>1.5</td>
<td>18</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>1.4</td>
<td>15</td>
</tr>
</tbody>
</table>
neural network with two hidden units in one layer
all 16 inspections
Neural Network Estimates

jackknife error of 6 percent
Neural Networks versus Capture–Recapture

clearly outperforms capture–recapture

(6 percent versus 24)
similar performance on one half of the dataset

(5 percent versus 7)
Neural Networks versus Linear Regression

outperforms linear regression
(6 percent versus 11, smaller variance)
Neural Network Advantages

• much flexibility when fitting to data
• detects non-linearity in the data
• gives guidelines which features to use
• works well even with small datasets
• automatically adapts to different document types and sizes
Neural Network Topology

- number of inputs
- number of hidden layers
- number of units in hidden layers
- connections between layers
Training a Neural Network

• fit regression function to training data

• non-linear optimization process (choose weights to minimize error on training data)

• might get caught in local minimum

• train networks with different initial weights
Model Selection

- good generalization (predictive power) is more important than a small training error

- can use cross-validation on additional dataset

- we use model evidence (Bayesian technique)

- model evidence works well if network is small
Empty Space Phenomenon

<table>
<thead>
<tr>
<th>features</th>
<th>patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>67</td>
</tr>
<tr>
<td>4</td>
<td>223</td>
</tr>
<tr>
<td>5</td>
<td>768</td>
</tr>
<tr>
<td>6</td>
<td>2790</td>
</tr>
</tbody>
</table>

maximum number of features that can be used depends on number of training patterns available
Overfitting

good fit to training patterns, but underlying smooth process poorly approximated
Technical Countermeasures

- Empty Space Phenomenon
  -> follow Silverman’s rule of thumb
  -> apply feature selection
  -> we use mutual information
Technical Countermeasures

- Empty Space Phenomenon
  - follow Silverman’s rule of thumb
  - apply feature selection
  - we use mutual information

- Overfitting
  - prefer small networks
  - prefer networks with small weights
  - use regularization during training
Mutual Information

\[ H(T) - H(T \mid X) = \]

\[ \int\int p(x, t) \cdot \log \frac{p(x, t)}{p(x)p(t)} \]

- measures stochastic dependence between target \( T \) and feature \( X \)
- detects non-linear dependencies
Regularization

• prefer networks with small weights $w_{ji}$

• minimize regularized error

$$\beta \cdot E_{\text{train}} + \alpha \cdot \sum w_{ji}^2$$

• $\alpha$ and $\beta$ are additional parameters
Iterative Training Procedure

alternate between optimizing the weights $w_{ji}$
and updating the parameters $\alpha, \beta$
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>mean error</th>
<th>max error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture–Recapture</td>
<td>24 %</td>
<td>67 %</td>
</tr>
<tr>
<td>Detection Profile</td>
<td>36 %</td>
<td>113 %</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>11 %</td>
<td>40 %</td>
</tr>
<tr>
<td>Interval Estimates</td>
<td>(7 %)</td>
<td>(14 %)</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>6 %</td>
<td>17 %</td>
</tr>
</tbody>
</table>

All three novel approaches are promising but need more empirical data for validation.
Regression Approach Summary

- uses empirical data from past inspections
- linear regression
- neural networks as non-linear regression
- outperforms existing methods

- see Ragg, Padberg, Schoknecht ICANN 2002
Let’s Try This, Too!