Empirical Software Engineering

CSE 8340 — Fall 2002

Prof. Jeff Tian, tian@engr.smu.edu
CSE, SMU, Dallas, TX 75275
(214) 768-2861; Fax: (214) 768-3085
www.engr.smu.edu/~tian/class/8340.02f

Module IIc: Other Risk Id Techniques

- PCA/DA and Application in Telecom
- NN and Applications
- OSR and Application in NASA/SEL
PCA & DA Study: Overview

- **PCA**: principal component analysis
  - idea of linear transformation
  - produce uncorrelated i.v.
  - reduce dimensionality

- **DA**: discriminant analysis
  - discriminant function
  - combine with other techniques

- Application: BNR/Nortel Telecom

PCA & DA Study: Context/Design

● Goal/Motivation:
  ▶ risk id as a classification problem
  ▶ DA, TBM, NN, Pattern Recognition
  ▶ technique used: DA in connection with PCA and logistic analysis

● Experimental context:
  ▶ system profile: Table 1.
  ▶ module classification:
    – new, changed, unchanged
    – effectively use logistic regression

● Experimental design:
  ▶ observational (post-mortem analysis)
  ▶ training: 2/3; testing 1/3; randomly
PCA & DA Study: Data

- d.v.: faults
  - between coding and operational phases
  - fault distribution: Table 2
  - risk: fault-prone or faults \( \geq 5 \)

- i.v.: design metrics
  - early quality prediction
  - modules consists of files
    - mean = 12, median = 4
  - design metrics defined on call graph and control flow graph
  - 9 specific metrics: Table 3
PCA & DA Study: Analysis

- Problem: classification into fault-prune and no-fault-prune via discriminant analysis

- Analysis procedure/techniques:
  - encode categorical variable
  - standardization
  - PCA to make model stable
  - model selection
  - discriminant analysis

- Encode categorical variable (covariates)
  - encoding: change (see earlier)
  - logistic regression idea

- Standardization: transform data to make mean = 0, and std.dev. = 1.
PCA & DA Study: Analysis

- **PCA: why?**
  - correlated i.v.’s leads to unstable models
  - extreme case:
    - linearly dependent $\Rightarrow$ singularity
  - linear transformation (PCA) $\Rightarrow$
    - uncorrelated PCs (or domain metrics)

- **PCA: how?**
  - covariance matrix: $\Sigma$
  - solve $|\Sigma - \Lambda| = 0$ to obtain eigenvalues
    - $\lambda_j$ along the diagonal for the diagonal matrix $\Lambda$
  - $\lambda_j$’s in decreasing value
  - decomposition: $\Sigma = P^T \Lambda P$
  - $P$: matrix of eigenvectors
    - (transformation used)
PCA & DA Study: Analysis

• Obtaining PCA results:
  ▶ transformation: $D = ZT$, where
    – $Z$ is the standardized metrics
    – $T$ is the transformation matrix
  ▶ $\Lambda, P, T$ calculated by various statistical packages/tools

• PCA result interpretation:
  ▶ eigenvalues $\approx$ explained variance
  ▶ first few domain metrics (PCs) explain most of the variance
    (typically 3 to 5)
PCA & DA Study: Analysis

● Using PCA results:
  ▶ uncorrelated PCs
      ⇒ good/stable linear models
  ▶ only a few PCs are necessary
  ▶ establish significance level

● Models selection:
  ▶ choose domain metrics
  ▶ also choose covariates
  ▶ judge by the discriminant analysis model
  ▶ algorithm in Khoshgoftaar paper
PCA & DA Study: Analysis


- DA: how?
  - define discriminant function
  - classify into $G_1$ and $G_2$
    - $G_1$: not fault-prune
    - $G_2$: fault-prune
  - definitions: (4) through (8)
  - other/similar definitions possible

- DA evaluation:
  - fit: misclassification rate
  - prediction: from training to test sets
  - misclassification types:
    - I: $G_1$ to $G_2$ false alarm
    - II: $G_2$ to $G_1$ missed fault-prune
      prefer II to be lower than I
PCA & DA Study: Result

- PCA results: Table 4
  - only 3 PCs (domain metrics) needed
  - powerful tool to interpret data

- DA/PC model selection
  - $model_1$: PCs only, select all 3 PCs
  - $model_2$: all 3 PCs and 2 covariates

- DA results:
  - Tables 5, 6, 7, 8
  - DA fit: no significant differences
  - DA prediction: important differences
  - reuse an important factor
PCA & DA Study: Conclusions

• Positive results (Authors):
  ▶ DA for risk identification
    (identifying fault-prune modules)
  ▶ design metrics useful indicators
  ▶ reuse information valuable
  ▶ predictive quality more important

• Other observations (Tian):
  ▶ much needed before DA
  ▶ data treatment/transformation effect
  ▶ much statistics knowledge
  ▶ complexity and interpretation
  ▶ type I vs II misclassification
NN Study: Overview

- NN ideas and algorithms:
  - single neuron: computation unit
  - connection: layered network
  - algorithm: backward propagation

- NN applications:
  - use in command and control communication system (CCCS)
  - use in medical imaging system (MIS)
  - comparison baseline: multiple regression applied to the above two systems

- 1995 Paper by Khoshgoftaar, Pandya, and Lanning, ASE
NN Study: Context/Design

- Experimental context:
  - CCCS: large military software, in Ada (not much information given)
  - MIS: commercial software
    - 4500 routines, 400 KLOC
    - in Pascal, Fortran, assembler, and PL/M
    - 5 development, 3 year deployment

- Experimental design:
  - observational (post-mortem analysis)
  - training: 2/3; testing 1/3; randomly
NN Study: Data

- CCCS data:
  - d.v.: faults from system integration and test, and first year of deployment
  - i.v.: 8 selected out of 14
    (mostly D/C complexity and size)
  - 282 data points (2/3 vs 1/3)

- MIS data:
  - d.v.: changes due to faults discovered during system testing and maintenance.
  - i.v.: 11 similar metrics
  - 339 modules (less than 1/10) used
    (again, 2/3 training; 1/3 testing)
NN Study: Analysis

- NN ideas and organization:
  - neuron: basic computation unit
  - NN: multiple layers
  - each layer: multiple neurons
  - input from previous layer
  - output to next layer

- Computation at a neuron: 2 stages
  - weighted sum of input: \( h = \sum_{i=1}^{n} x_i \)
    (may include constant)
  - then activation function \( y = g(h) \)
    - threshold, piecewise-linear,
    - Gaussian, sigmoid (below), etc.
    - in Khoshgoftaar: \( y = \frac{1}{1 + e^{-h/T}} \)
  - illustration in Tian or Khoshgoftaar
NN Study: Analysis

- Overall computation:
  - layers of neurons
  - input layer: raw data feed
  - other layers: computation at \( n \) neurons
  - objective: minimize prediction error at the output layer

- algorithm: backward propagation
  - Fig. 4 in Tian SQP 2000
    (actually algorithm ideas, not exact)
  - trace through steps
  - error: deviance (sum of error sqr)
  - specific adjustments: Khoshgoftaar p.147
    (learning and momentum rates: \( \eta, \alpha \))
**NN Study: Result**

- Model performance measure:
  - $ARE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{e_i}{y_i} \right|$  
  - $y_i$ transformed by adding 1 to raw data
  - discussion in Khoshgoftaar p.143
  - other measure: rse (rel.std.err)

- NN model produced:
  - input data scaled to [0, 1]
  - initial weights: random in [–1, 1]
  - learning/momentum: $\eta = 1.5$, $\alpha = 0.7$
  - 1 hidden layer best, from 1, 2, 3 tried
  - 16 neurons for CCCS and 18 for MIS
NN Study: Summary

- NN result summary:
  - Table 1
  - comparison to regression model
    (RM details given in Tables 2 & 3)
  - NN is clearly superior

- Other comments (Tian):
  - good example of NN technique
  - not really focus on risk identification
    (estimating # faults)
  - but in other work by authors
  - Khoshgoftaar/Szabo (ref. in Tian/SQP):
    PCA & NN for risk id.
OSR Study: Overview

- OSR: optimal set reduction
  - pattern matching ideas
  - OSR technique: Briand et al., 1992 (OSR for effort estimation)
  - with application: 1993 Paper by Briand/Basili/Hetmanski, TSE 19(11)

- OSR applications:
  - NASA/SEL: fault risk
  - baseline for comparison:
    - classification trees
    - logistic analysis
    - (with or without PCA for LA)
  - demonstrate superiority
OSR Study: Context/Design

- Experimental context:
  - NASA/SEL software in Ada
  - 146 components and 260 KLOC

- Experimental design:
  - high risk: procedure/function with error
    - particularly, reading/writing to var/struct
  - otherwise, low risk
  - equal number of low and high risk components to build unbiased model
  - observational (post-mortem analysis)
OSR Study: Data

• Metrics data derived from hypotheses about the software design process

• Data categories (via hypotheses):
  ▶ context coupling
  ▶ parameterization
  ▶ visibility control
  ▶ reuse
  ▶ component size
  ▶ structural complexity

• Obtaining the data
  ▶ ASAP static-analysis program
  ▶ UNIX utilities and SAS program
  ▶ 15 metrics listed in Appendix A
OSR Study: Analysis

- OSR ideas and organization:
  - pattern extraction
  - algorithm sketch: Fig.9 in Tian/SQP
  - detailed algorithm: Appendix B
  - other detailed throughout paper
  - organization/modeling results:
    - no longer a tree, see example
    - general subsets, may overlap

- Similarities to TBM
  - pattern ~ partition
  - entropy ~ deviance
  - reduction of the above
OSR Study: Result

- Model performance measure: accuracy
  - completeness:
    % high risk modules identified
  - correctness:
    % high risk modules correctly identified
  - performance comparison: Table 1

- OSR conclusions:
  - accuracy: better than alternatives
    - LA tedious & less interpretable
    - TBM simplistic
  - interpretable structure (similar to TBM)

- Comments: fair comparison?