Clustering and Classification of UVM Test Failures Using Machine Learning Techniques

Andy Troung, Daniel Hellström, Harry Duque, Lars Viklund
Axis Communications AB, Lund, Sweden
Problem Statement

• Applying coverage-driven constrained random verification
  – Test suites often consists of a large number of test invocation
  – Same base tests run many times
    • Different seeds
    • Adjusted random distributions
  – One problem often results in many test invocations failing

• Analysis of test failures
  – Which failures have a common cause?
  – What is the type of root cause for each failure?

• Can this analysis be automated using machine learning?
Input Data

- Simulation log files from UVM test benches
- Semi-structured text
- Consists mostly of UVM report messages
- Dataset used for evaluation:
  - 12500 samples labeled with failure root cause
  - Originated from 29 UVM test benches
Features

• Convert input into numerical feature vectors
• Examples of information extracted:
  – UVM test name and UVM configuration settings from command line
  – UVM report messages
  – Simulator warnings and error messages
• Some information is abstracted
• Mostly the frequency of occurrence is used as the feature
• Original set consists of 616 features
Dimensionality Reduction

• Feature selection
  – Remove less relevant features
  – Manual or automatic

• Feature extraction
  – Merge existing features

• Applied manual feature selection
  – Based on domain knowledge
  – Reduced set down to 287 features
  – Baseline used for evaluation

• Evaluated algorithms for automatic selection and extraction
Classification

- Supervised learning problem
- Training using a dataset associating input with output
  - Input is simulation log file
  - Output is failure root cause
- Algorithm creates a model that predicts output for new input data
Taxonomy of Test Failure Root Causes

- Environment
  - Constraint Solver
    - Infrastructure
    - Disk Quota
    - Job Killed
  - IT
    - Connectivity
    - Contradiction
    - Missing
    - Inefficient
  - Systemverilog
    - Functional Coverage
    - Configuration Issue
    - Configuration Object

- Test Bench
  - Factory
  - Parameters

- Reference Model
  - UVC
    - Bug
    - Algsim
    - Data Conversion
  - SystemC
  - Python
  - OpenCL

- Design

Status:
- Ok
- Fail
Classification Algorithms

• Nine classification algorithms evaluated
  – Selected because they use different strategies
• Three dimensionality reduction algorithms evaluated
  – Two feature selection algorithms
  – One feature extraction algorithm
• Implementations from *scikit-learn*
• Initial evaluation using default hyperparameter settings
Classification Algorithm Evaluation

• Dataset divided into a training set and a test set
  – 10000 samples for the training set
  – 2500 samples for the test set

• Evaluation steps:
  – Initial evaluation using training set and $k$-fold cross validation
  – Optimization of the most promising algorithms
  – Final evaluation using test set
Confusion Matrix

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>True Positive</td>
<td>False Negative</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>False Positive</td>
<td>True Negative</td>
<td></td>
</tr>
</tbody>
</table>
Classification Metrics

• **Accuracy**
  – How often is the classifier correct?
  – \((TP + TN) / \text{Total}\)

• **Precision**
  – When it predicts yes, how often is it correct?
  – \(TP / (TP + FP)\)

• **Recall**
  – When it is actually yes, how often does it predict yes?
  – \(TP / (TP + FN)\)

• **\(F_1\)-score**
  – Harmonic mean of precision and recall
# Initial Classification Results

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F$_1$-score</th>
<th>Train (s)</th>
<th>Predict (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline feature set</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random forest</td>
<td>0.899</td>
<td>0.907</td>
<td>0.904</td>
<td>0.905</td>
<td>0.277</td>
<td>0.132</td>
</tr>
<tr>
<td>SVC poly</td>
<td>0.556</td>
<td>0.864</td>
<td>0.560</td>
<td>0.609</td>
<td>52.655</td>
<td>56.315</td>
</tr>
<tr>
<td>SVC rbf</td>
<td>0.806</td>
<td>0.845</td>
<td>0.800</td>
<td>0.813</td>
<td>17.311</td>
<td>36.422</td>
</tr>
<tr>
<td>LinearSVC</td>
<td>0.851</td>
<td>0.856</td>
<td>0.852</td>
<td>0.852</td>
<td>72.463</td>
<td>0.184</td>
</tr>
<tr>
<td>Decision tree</td>
<td>0.892</td>
<td>0.901</td>
<td>0.899</td>
<td>0.899</td>
<td>0.342</td>
<td>0.067</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>0.841</td>
<td>0.851</td>
<td>0.840</td>
<td>0.842</td>
<td>62.498</td>
<td>0.191</td>
</tr>
<tr>
<td>K-neighbors</td>
<td>0.883</td>
<td>0.890</td>
<td>0.887</td>
<td>0.888</td>
<td>0.522</td>
<td>56.618</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.643</td>
<td>0.763</td>
<td>0.652</td>
<td>0.607</td>
<td>0.180</td>
<td>0.933</td>
</tr>
</tbody>
</table>
Summary of Initial Classification Evaluation

• Three algorithms performed better than the others
  – Random forest
  – Decision tree
  – k-nearest neighbors
• Impact of dimensionality reduction
  – Slightly lower scores
  – Significantly reduced computation time
Optimization of Classification Algorithms

• Tune hyperparameters
• Small hyperparameter space
  – Decision tree and $k$-nearest neighbor
  – Exhaustive search
• Large hyperparameter space
  – Random forest
  – Random search
## Classification Results After Optimization

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F$_1$-score</th>
<th>Train (s)</th>
<th>Predict (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest</td>
<td>0.907</td>
<td>0.915</td>
<td>0.913</td>
<td>0.913</td>
<td>8.410</td>
<td>2.074</td>
</tr>
<tr>
<td></td>
<td>+0.9%</td>
<td>+0.9%</td>
<td>+1.0%</td>
<td>+0.9%</td>
<td>+2936.1%</td>
<td>+1471.2%</td>
</tr>
<tr>
<td>Decision tree</td>
<td>0.896</td>
<td>0.904</td>
<td>0.902</td>
<td>0.902</td>
<td>0.385</td>
<td>0.113</td>
</tr>
<tr>
<td></td>
<td>+0.4%</td>
<td>+0.3%</td>
<td>+0.3%</td>
<td>+0.3%</td>
<td>+12.6%</td>
<td>+68.7%</td>
</tr>
<tr>
<td>K-neighbors</td>
<td>0.885</td>
<td>0.891</td>
<td>0.891</td>
<td>0.891</td>
<td>0.101</td>
<td>0.994</td>
</tr>
<tr>
<td></td>
<td>+0.3%</td>
<td>+0.1%</td>
<td>+0.7%</td>
<td>+0.5%</td>
<td>-9.0%</td>
<td>-41.0%</td>
</tr>
</tbody>
</table>
## Final Classification Results

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;-score</th>
<th>Train (s)</th>
<th>Predict (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random forest</td>
<td>0.907</td>
<td>0.916</td>
<td>0.912</td>
<td>0.913</td>
<td>5.344</td>
<td>0.182</td>
</tr>
<tr>
<td>Decision tree</td>
<td>0.895</td>
<td>0.902</td>
<td>0.900</td>
<td>0.900</td>
<td>0.111</td>
<td>0.002</td>
</tr>
<tr>
<td>K-neighbors</td>
<td>0.880</td>
<td>0.888</td>
<td>0.886</td>
<td>0.886</td>
<td>0.059</td>
<td>0.113</td>
</tr>
</tbody>
</table>
Summary of Classification Evaluation

- Random forest performs best
- Algorithms generalizes well to new data
- Learning and classification run-times are reasonable
- Most common misclassification is design bug vs. reference model bug
Significance of Data

\[ F_1 \text{-score} \]

Number of samples

© Accellera Systems Initiative
Prediction Confidence

Misclassification rate vs Prediction confidence
Clustering

• Unsupervised learning problem
• Divide the dataset into clusters (groups)
• Clusters will contain samples that are similar
  – In this case same failure root cause
• Different types of algorithms:
  – Prototype-based
  – Density-based
  – Graph-based
Clustering Algorithms

- Three different clustering algorithms evaluated
  - One of each of the three main types
- One dimensionality reduction algorithms evaluated
- Two different visualization algorithms evaluated
- Implementations from *scikit-learn*
- Hyperparameter settings were optimized
Clustering Algorithm Evaluation

- Metrics used for classification are not applicable
- No risk of overfitting since there is no training
  - No need for a separate test set
  - No need for cross-validation
- Measure similarities between clusters while ignoring permutations
- Adjusted Rand Index (ARI)
  - Based on pair-counting
  - Range [-1, 1]
- Adjusted Mutual Information (AMI)
  - Based on information theory
  - Range [0, 1]
## Clustering Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AMI</th>
<th>ARI</th>
<th>Computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline feature set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-means</td>
<td>0.505</td>
<td>0.480</td>
<td>0.079</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>0.568</td>
<td>0.530</td>
<td>0.086</td>
</tr>
<tr>
<td>Agglomerative Clustering</td>
<td>0.540</td>
<td>0.515</td>
<td>0.036</td>
</tr>
<tr>
<td><strong>Dimensionality reduction using PCA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-means</td>
<td>0.543</td>
<td>0.513</td>
<td>0.041</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>0.593</td>
<td>0.545</td>
<td>0.007</td>
</tr>
<tr>
<td>Agglomerative Clustering</td>
<td>0.543</td>
<td>0.519</td>
<td>0.006</td>
</tr>
</tbody>
</table>
Visualization of Clustering

- **t-SNE Ground Truth**
- **MDS Ground Truth**
- **t-SNE DBSCAN**
- **MDS DBSCAN**
Tool Implementation

• Prototype tool based on the most suitable algorithms
• Implemented in Python using *scikit-learn*
• Pre-processing of log file using regular expressions
• Interactive visualization of clustering results
Future Work

• Investigate dependency on test bench coding style
• Investigate dependency on report verbosity level
• Use other data in addition to log files
• Improve performance of clustering algorithms
  – More careful feature engineering
  – Tuning of hyperparameters
Conclusions

- Machine learning can effectively be applied to classify UVM test failures
  - Random forest yielded an accuracy of 0.907 and an $F_1$-score of 0.913
- Machine learning for clustering UVM test failures was less convincing
  - DBSCAN yielded an AMI score of 0.593 and an ARI score of 0.545
- Clustering algorithms can provide a good overview when used in combination with visualization algorithms
- The investigated algorithms show promise as tools to reduce the time invested in analyzing failures in large test suites
Questions