Who we are

Samuel Hopstock
samuel.hopstock@tngtech.com

Thomas Endres
thomas.endres@tngtech.com
Classical approaches on program evaluation

Static analysis
Classical approaches on program analysis

Dynamic analysis

https://devrant.com/rants/64755/unit-test-is-close-enough
Limitations

Complex Dependencies

Big Test Suite

https://thedailywtf.com/articles/The-Enterprise-Dependency
Our idea

Before
- Foo.java
- Bar.java

After
- Foo.java (no change)
- Bar.java (modified)
- Baz.java (new)

Test suite
Prediction:
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

Execution order:
- 1
- 4
- 7
- 8
- 0
- 2
- 3
- 5
- 6
- 9
Why machine learning?

It seems good for finding complex relations in data
Overview: challenges to solve

Original.java  ?  Modified.java

1 1 1 0 0 1 0 1 0 0 1

What model?

What predictions?
## Overview

### Data
- Vectors for machine learning
- Machine learning techniques

### Technical aspects

### Conclusion

### Related work
What data?
Data generation

Historical data (e.g. Git log)

natural learning data → best prediction performance
MonkeyRunner

Randomly change source code semantics

Similar: randomly rotating image data
original

```java
public class Physics {
    public int getEnergy(int m, int c) {
        return m - Math.pow(c, 8);
    }
}
```

modified

```java
public class Physics {
    public int getEnergy(int m, int c) {
        return m * Math.pow(c, 2);
    }
}
```
Overview

Data

Vectors for machine learning

Machine learning techniques

Technical aspects

Conclusion

Related work
Vectors?

Input data: source code

0 1 1 0 1 0 1

Output data: test results

0 0 0 1
Input vectors

List of changed classes

- ChangedClass
- AlsoChanged
- NotChanged
Input vectors based on AST
Input vectors based on AST

one-hot encoding

```
<table>
<thead>
<tr>
<th>Token Vectors</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token_0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Token_1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Token_2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Token_3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```
1 0 0 0
1 0 0 0
0 0 0 1
2 0 0 1
```
Input vectors based on AST

word2vec
Input vectors based on AST

code2vec
Input vectors based on AST
Token clustering with T-SNE

view in Tensorflow Projector:
https://goo.gl/VUP454
Token clustering with T-SNE
Difficulties

get vectors using CBOW

MonkeyRunner

package com.foo;
public class Foo {
    private int sum(int a, int b) {
        return a-b;
    }
    public static void main(String[] args) {
        public static void main(String[] args) {
            System.out.println(sum(1,2));
            System.out.println(sum(1,2));
        }
    }
}

package com.foo;
public class Foo {
    private int sum(int a, int b) {
        return a+b;
    }
    public static void main(String[] args) {
        System.out.println(sum(1,2));
    }
}

original

No change??

120.167
35.995
117.041
83.047
81.653
100.214
66.009

0.0
0.0
0.0
0.0
0.0
0.0
0.0
Difficulties
Why??

AST structure still the same!
Embed token parameters in vector?
More complex MonkeyRunner changes?
Output vectors

Test results:
- FAILED: Test A
- FAILED: Test B
- PASSED: Test C

Code quality/other metrics:
- Test coverage: 0.87
- Bugginess: 100
- WTF’s per minute: 0.91
Difficulties

Vector dimensions not fixed? (e.g. new classes/tests added)

Padding

desired size: 5

Auto-Encoder

hidden layer dimension: desired feature vector size
Overview

Data

Vectors for machine learning

Machine learning techniques

Technical aspects

Conclusion

Related work
Statistical classification
Support vector machines (SVM)

Feature vector | Classifier | Output
---|---|---
1 2 | SVM | 1
5 5 | SVM | 0

Diagram: Points classified into two classes, with linear decision boundary.
Statistical classification
Support vector machines (SVM)

<table>
<thead>
<tr>
<th>Feature vector</th>
<th>Classifier</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>SVM</td>
<td>1</td>
</tr>
<tr>
<td>5 5</td>
<td>SVM</td>
<td>0</td>
</tr>
</tbody>
</table>
Statistical classification

Support vector machines (SVM)

Feature vector  Classifier  Output

1 2  SVM 1

5 5  SVM 0

\[ z = x^2 + y^2 \]
Statistical classification

SVM with multilabel data

Feature vector

Output vector

svm_0

svm_1

svm_2

svm_3

0/1

0/1

0/1

0/1

0

1

1

0

0

0

0
Neural network

Simple example: fully connected, one hidden layer

Training data

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Cost: \( (2-0)^2 + (4-1)^2 \)

Goal: minimize cost
How: modify weights

Neural network

Input: Java files
Output: unit test results

90 -> 596

Expectation:

Java class ID

unit test ID

test not influenced by class
test influenced by class
Neural network

Input: Java files
Output: unit test results

Reality:
Java class ID
unit test ID

90 → 596

Test not influenced by class
Test influenced by class
Overfitting

Rule of thumb: ~10 samples per parameter
Our neural net: >120k weights → 1.2 million samples needed
Decision trees

Do you want a computer?

- no
- yes

Do you want a working computer?

- yes
- no

https://www.pinterest.de/pin/80642649554451367
Decision trees

Training

<table>
<thead>
<tr>
<th>Color</th>
<th>Diameter</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>3</td>
<td>Apple</td>
</tr>
<tr>
<td>Yellow</td>
<td>3</td>
<td>Apple</td>
</tr>
<tr>
<td>Red</td>
<td>1</td>
<td>Grape</td>
</tr>
<tr>
<td>Red</td>
<td>1</td>
<td>Grape</td>
</tr>
<tr>
<td>Yellow</td>
<td>3</td>
<td>Lemon</td>
</tr>
</tbody>
</table>

```
Diameter >= 3?

false
  R 1  Grape
  R 1  Grape

true
  Color == Yellow?
  Y 3  Apple
  Y 3  Lemon

false
  G 3  Apple
  Apple 100%

true
  Apple 50%
  Lemon 50%
  Y 3  Apple
  Y 3  Lemon
```
Overview

Data

Vectors for machine learning

Machine learning techniques

Technical aspects

Conclusion

Related work
Frameworks

since 2007
Classification, regression and clustering algorithms
Language: Python with NumPy
Frameworks

since 2015
Calculations on tensors (more generalized form of vectors)
Popular for neural networks
Languages: Python, (Java, Go, C)
Simple neural network: Tensorflow

```python
# source: https://gist.github.com/vinhkhuc/e53a70f9e5c3f55852b0

def init_weights(shape):
    weights = tf.random_normal(shape, stddev=0.1)
    return tf.Variable(weights)

def forwardprop(X, w_1, w_2):
    h = tf.nn.sigmoid(tf.matmul(X, w_1))  # The \sigma function
    yhat = tf.matmul(h, w_2)  # The \varphi function
    return yhat

def get_data():
    dataset = load_dataset()
    data = dataset['data']
    target = dataset['target']
    N, M = data.shape
    all_X = np.ones((N, M + 1))
    all_X[:, 1:] = data
    num_labels = len(np.unique(target))
    all_Y = np.eye(num_labels)[target]  # One liner trick!
    return train_test_split(all_X, all_Y, test_size=0.33, random_state=RANDOM_SEED)

def main():
    train_X, test_X, train_y, test_y = get_data()
    x_size = train_X.shape[1]  # Number of input nodes: 90 features
    h_size = 180  # Number of hidden nodes
    y_size = train_y.shape[1]  # Number of outcomes (596 unit tests)
    X = tf.placeholder("float", shape=[None, x_size])
    y = tf.placeholder("float", shape=[None, y_size])
    w_1 = init_weights((x_size, h_size))
    w_2 = init_weights((h_size, y_size))
    yhat = forwardprop(X, w_1, w_2)
    predict = tf.argmax(yhat, axis=1)
    cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y, logits=yhat))
    updates = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
    sess = tf.Session()
    init = tf.global_variables_initializer()
    sess.run(init)
    for epoch in range(100):
        for i in range(len(train_X)):
            sess.run(updates, feed_dict={X: train_X[i: i + 1], y: train_y[i: i + 1]})
            train_accuracy = np.mean(np.argmax(train_y, axis=1) ==
                                      sess.run(predict, feed_dict={X: train_X, y: train_y}))
            test_accuracy = np.mean(np.argmax(test_y, axis=1) ==
                                     sess.run(predict, feed_dict={X: test_X, y: test_y}))
            print("Epoch = %d, train accuracy = %.2f%%, test accuracy = %.2f%%" % (epoch + 1, 100. * train_accuracy, 100. * test_accuracy))
    sess.close()
```

hidden_layer = Dense(180, input_shape=(90,))
output_layer = Dense(596)
model = Sequential([hidden_layer, output_layer])
model.compile(loss='mean_squared_error', optimizer='sgd')
model.fit(X_train, Y_train)
Specialized hardware

GPU learning

Intel Xeon + MKL

Movidius NCS
(e.g. for Raspberry Pi)

Clusters!
<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
</tr>
<tr>
<td>Vectors for machine learning</td>
</tr>
<tr>
<td>Machine learning techniques</td>
</tr>
<tr>
<td>Technical aspects</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
<tr>
<td>Related work</td>
</tr>
</tbody>
</table>
Recap: challenges

![Diagram showing the process from Original.java to Modified.java through VCS log and MonkeyRunner, followed by AST vectors, model, predictions, test results, and code quality considerations.]

What model?

1 1 0 0 1 0 1 0 0 1

1 0 0 1 1

Test results
Code quality...

Clustering GitHub repos

http://vmarkovtsev.github.io/techtalks-2017-moscow
VarNaming/VarMisuse

Source code as graph

VarNaming/VarMisuse
(Source Code as Graph)

Test failure prediction

Microsoft Dynamics AX 2012

Thank You!

Samuel Hopstock, samuel.hopstock@tngtech.com
Thomas Endres, thomas.endres@tngtech.com