Machine Learning for Programming

Martin Vechev
ETH Zurich and DeepCode.ai
Learning and probabilistic models based on Big Data have revolutionized entire fields

Natural Language Processing (e.g., machine translation)

Computer Vision (e.g., image captioning)

Medical Computing (e.g., disease prediction)

Can we bring this revolution to programmers?
Machine Learning for Programs

Task → Statistical Programming Tool → Solution

Probabilistic model

Number of repositories

15 million repositories

Billions of lines of code

High quality, tested, maintained programs

Last 5 years
Why now?

Advances in Programming Languages
[Automated Reasoning, Synthesis, Constraint Solving]

Advances in Machine Learning
[Deep Learning, Graphical Models, Language Models]

Confluence of streams

machine learning-based programming tools
new rules, new ideas, new opportunities

Data
[> 15 million public repositories]
Machine Learning for Programs

Probabilistically likely solutions to problems hard to solve otherwise

Joint work with:

- Veselin Raychev
- Andreas Krause
- Pavol Bielik
- Christine Zeller
- Svetoslav Karaivanov
- Pascal Roos
- Benjamin Bichsel
- Timon Gehr
- Petar Tsankov
- Mateo Panzacchi

Publications

- Program Synthesis for Character Level Language Modeling, *ICLR’17*
- Learning a Static Analyzer from Data, *CAV’17*
- Statistical Deobfuscation of Android Applications, *ACM CCS’16*
- Probabilistic Mode for Code with Decision Trees, *ACM OOPSLA’16*
- PHOG: Probabilistic Mode for Code, *ACM ICML’16*
- Learning Programs from Noisy Data, *ACM POPL’16*
- Predicting Program Properties from “Big Code”, *ACM POPL’15*
- Code Completion with Statistical Language Models, *ACM PLDI’14*
- Machine Translation for Programming Languages, *ACM Onward’14*

more: [http://plml.ethz.ch](http://plml.ethz.ch)
Dimensions:
Machine Learning for Programming
## Dimensions: Machine Learning for Programming

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Deobfuscation</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Representation</td>
<td>Sequences (sentences)</td>
<td>Translation Table</td>
<td>Trees</td>
<td>Graphical Models (CRFs)</td>
<td>Feature Vectors</td>
</tr>
<tr>
<td>Analyze Program (PL)</td>
<td>typestate analysis</td>
<td>scope analysis</td>
<td>control-flow analysis</td>
<td>alias analysis</td>
<td></td>
</tr>
<tr>
<td>Train Model (ML)</td>
<td>Neural Networks</td>
<td>N-gram/back-off</td>
<td>SVM</td>
<td>PCFG</td>
<td>Structured SVM</td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>$\text{argmax } P(y \mid x)$</td>
<td>$y \in \Omega$</td>
<td>Greedy</td>
<td>MAP inference</td>
<td></td>
</tr>
</tbody>
</table>
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

SLANG

camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);
r.setOutFormat(MediaRecorder.MPEG4);

Statistical language models
Recurrent neural networks

+ Typestate analysis
Alias analysis
Importance of Static (Semantic) Analysis

**precision vs. % of data used**

- Red: no alias analysis
- Green: with alias analysis

1% 10% 100%

**static analysis benefit = 10x more data**
Statistical Language Translation

[ACM Onward’14, Svetoslav Karaivanov, Veselin Raychev, M.V.]

Phrase-based Statistical Machine Translation

Prefix Parsing of Context-Free Grammars
function chunkData(str, step) {
    var colNames = [];
    var len = str.length;
    var i = 0;
    for (; i < len; i += step)
        if (i + step < len)
            colNames.push(str.substring(i, i + step));
        else
            colNames.push(str.substring(i, len));
    return colNames;
}
This Page Amsterdam @thispage_ams · Jul 16
Do you write ugly JavaScript code? Not to worry. JSNice will make it look like you are a superstar coder. Yai! - buff.ly/1HR4JL7

Ingvar Stepanyan @RReverser · Aug 6
JSNice.org became my must-have tool for code deobfuscation.

Brevity @seekbrevity · Jul 28
JSNice is an amazing tool for de-minifying javascript files. JSNice.org, it’s great for learning and reverse engineering.

Alvaro Sanchez @alvasi · Jun 10
This is gold.
Statistical renaming, Type inference and Deobfuscation. jsnice.org

Alex Vanston @mvdot · Jun 7
I've been looking for this for years: JS NICE buff.ly/1pQ5qfr javascript 
#unminify #deobfuscate #makeItReadable

Kamil Tomšík @cztomsk · Jun 6
tell me how this works!
#de-minify #jquery #javascript incl. args, vars & jsl Inpressive! jsnice.org
JSNice: over a year of usage
May 2015 – May 2016

Total: 12GB of code
function chunkData(e, t)
    var n = [];
    var r = e.length;
    var i = 0;
    for (; i < r; i += t)
        if (i + t < r)
            n.push(e.substring(i, i + t));
        else
            n.push(e.substring(i, r));
    return n;

function chunkData(str, step)
    var colNames = [];
    var len = str.length;
    var i = 0;
    for (; i < len; i += step)
        if (i + step < len)
            colNames.push(str.substring(i, i + step));
        else
            colNames.push(str.substring(i, len));
    return colNames;

Key Challenges

Facts to be predicted are dependent

Prediction should be fast (real-time)
(hard, millions of possible choices)
Key Idea

Phrase the problem of predicting program facts as

Structured Prediction for Programs
function chunkData(e, t)
    var n = [];
    var r = e.length;
    var i = 0;
    for (; i < r; i += t)
        if (i + t < r)
            n.push(e.substring(i, i + t));
        else
            n.push(e.substring(i, r));
    return n;
function chunkData(e, t)
    var n = [];
    var r = e.length;
    var i = 0;
    for (; i < r; i += t)
        if (i + t < r)
            n.push(e.substring(i, i + t));
        else
            n.push(e.substring(i, r));
    return n;
MAP inference

function chunkData(e, t)
    var n = [];
    var r = e.length;
    var i = 0;
    for (; i < r; i += t)
        if (i + t < r)
            n.push(e.substring(i, i + t));
        else
            n.push(e.substring(i, r));
    return n;
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;
function chunkData(e, t)
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t)
    if (i + t < r)
      n.push(e.substring(i, i + t));
    else
      n.push(e.substring(i, r));
  return n;

function chunkData(str, step)
  var colNames = [];
  var len = str.length;
  var i = 0;
  for (; i < len; i += step)
    if (i + step < len)
      colNames.push(str.substring(i, i + step));
    else
      colNames.push(str.substring(i, len));
  return colNames;

argmax \mathbf{w}^T f(i, t, r, \text{length})
A glimpse of learning...
[N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007]

\[ P(y | x) = \frac{1}{Z(x)} \exp(\mathbf{w}^T \mathbf{f}(y, x)) \]

Given a data set: \( D = \{ \mathbf{x}^i, y^i \}_{j=1..n} \) learn weights \( \mathbf{w}^T \)

Optimization objective (max-margin training):

\[ \forall j \forall y \sum w_i f_i(x^{(j)}, y^{(j)}) \geq \sum w_i f_i(x^{(j)}, y) + \Delta(y, y^{(j)}) \]

for all samples

Given prediction is better than any other prediction by a margin

Avoids expensive computation of the partition function \( Z(x) \)
JSNice

[Predicting program properties from Big Code, ACM POPL 2015]
Statistical Program de-obfuscation

[ACM CCS’16, Benjamin Bichsel, Petar Tsankov, Veselin Raychev, M.V.]

package a.b.c
class a extends SQLiteHelper {
    SQLiteDatabase b; public a(Context ctx) { b = getWritableDatabase(); }
    Cursor c(String str) { return b.rawQuery(str); }
}

class DBHelper extends SQLiteHelper {
    SQLiteDatabase db;
    public DBHelper(Context ctx) {
        db = getWritableDatabase();
    }
    Cursor execSQL(String str) { return db.rawQuery(str); }
}

package com.example.dbhelper

class DBHelper extends SQLiteHelper {
    SQLiteDatabase db;
    public DBHelper(Context ctx) {
        db = getWritableDatabase();
    }
    Cursor execSQL(String str) { return db.rawQuery(str); }
}

👍 Funny Reddit post/comment

[-] Tycon712 • 3 points 2 days ago
Can someone tell me what the point of using Proguard is if there are tools out there like this?

permalink embed pocket

[-] theheartbreakpug • 6 points 2 days ago
As far as I know, this is brand new. I asked the creator of ProGuard a week ago how hard it is to unobfuscate code after it’s run through proguard. He said it strips all the names out of the code so it’s essentially impossible. I'm super impressed by what they’ve done here.
Nice2Predict.org: scalable structured prediction framework

[Pavol Bielik, Veselin Raychev, Matteo Panzacchi, Nick Baumann, M.V.]

fully, open sourced, Apache license

used by various groups worldwide

Fast, Approximate MAP inference

Fast, Parallel, Structured SVM and Pseudo-Likelihood Training

Arbitrary factors and indicator functions
Fundamental Problem

Data → Learning → Model

- Widely Applicable
- Efficient Learning
- High Precision
- Explainable Predictions

Probabilistic Model
Training dataset $D$

```python
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

query:

```python
f.open("file" | "r");
```

```python
f.?
```
Training dataset $D$

```c
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

Query:

```c
f.open("file" | "r");

f. ?
```

3-gram model on tokens

Hindle et. al., ACM ICSE’12

\[
P(open \mid f. ) \sim 3/6 \\
P(read \mid f. ) \sim 2/6 \\
P(write \mid f. ) \sim 1/6
\]

Context $\gamma$
Training dataset $D$

```c
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

query:

```c
f.open("file" | "r");
```

3-gram model on tokens

Hindle et. al., ACM ICSE’12

\[
\begin{align*}
P(\text{open} | \text{f.}) & \sim \frac{3}{6} \\
P(\text{read} | \text{f.}) & \sim \frac{2}{6} \\
P(\text{write} | \text{f.}) & \sim \frac{1}{6}
\end{align*}
\]

context $\gamma$
Training dataset $D$

- `f.open("f2" | "r"); f.read();`
- `f.open("f2" | "w"); f.write("c");`
- `f.open("f1" | "r"); f.read();`

probabilistic model on APIs

Raychev et. al., ACM PLDI’14

- $P(\text{read} \mid \text{open}) \sim 2/3$
- $P(\text{write} \mid \text{open}) \sim 1/3$

query:

- `f.open("file" | "r"); f.?

3-gram model on tokens

Hindle et. al., ACM ICSE’12

- $P(\text{open} \mid f.) \sim 3/6$
- $P(\text{read} \mid f.) \sim 2/6$
- $P(\text{write} \mid f.) \sim 1/6$
Training dataset $D$

```python
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

**probabilistic model on APIs**

Raychev et. al., ACM PLDI’14

- $P(\text{read} \mid \text{open}) \sim 2/3$
- $P(\text{write} \mid \text{open}) \sim 1/3$

query:

```python
f.open("file" | "r");
```

**3-gram model on tokens**

Hindle et. al., ACM ICSE’12

- $P(\text{open} \mid f.) \sim 3/6$
- $P(\text{read} \mid f.) \sim 2/6$
- $P(\text{write} \mid f.) \sim 1/6$
Training dataset $D$

```java
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

### probabilistic model on APIs

*Raychev et. al., ACM PLDI’14*

- $P($read | open$) \sim 2/3$
- $P($write | open$) \sim 1/3$

**query:**

```java
f.open("file" | "r");
```

### 3-gram model on tokens

*Hindle et. al., ACM ICSE’12*

- $P($open | f$) \sim 3/6$
- $P($read | f$) \sim 2/6$
- $P($write | f$) \sim 1/6$

**context** $\gamma$

What should the context be?
“...All problems in computer science can be solved by another level of indirection...”

-- David Wheeler
“...All problems in computer science can be solved by another level of indirection...”
-- David Wheeler

key idea: **synthesize a function** $f$: program $\rightarrow \gamma$
Creating probabilistic models: our method


1. **Pick** a structure of interest, e.g., ASTs:

2. **Define** a DSL for expressing functions:
   (can be Turing complete)

3. **Synthesize** $f_{best} \in \text{DSL}$ from Dataset $D$:

   $$f_{best} = \text{argmin} \ cost(D, f) \quad f \in \text{DSL}$$

4. Use $f_{best}$ to compute context and predict:

   $$f_{best} \left( \begin{array}{c} \ldots \end{array} \right) \rightarrow \gamma$$
Step 1: Pick Structure of Interest

Let it be abstract syntax trees (ASTs) of programs

`elem.notify({
    position: 'top',
    autoHide: false,
    delay: 100
});`
Step 2: Define a DSL over structure

**Syntax**

\[ T\text{Cond} ::= \varepsilon \mid \text{WriteOp} \ T\text{Cond} \mid \text{MoveOp} \ T\text{Cond} \]

\[ \text{MoveOp} ::= \text{Up, Left, Right, DownFirst, DownLast, NextDFS, PrevDFS, NextLeaf, PrevLeaf}, \text{PrevNodeType}, \text{PrevNodeValue}, \text{PrevNodeContext} \]

\[ \text{WriteOp} ::= \text{WriteValue, WriteType, WritePos} \]

**Semantics**

- Up
- Left
- WriteValue

\[ \gamma \leftarrow \gamma \cdot \square \]
Step 3: synthesize $f_{\text{best}}$

$$f_{\text{best}} = \arg\min_{f} \text{cost}(D, f)$$

$$f \in \text{DSL}$$
Step 3: synthesize $f_{best}$

$f_{best} = \arg\min f \in DSL \quad \text{cost}(D, f)$

DSL

$T\text{Cond} ::= \varepsilon \mid \text{WriteOp} T\text{Cond} \mid \text{MoveOp} T\text{Cond}$

$\text{MoveOp} ::= \text{Up, Left, Right, ...}$

$\text{WriteOp} ::= \text{WriteValue, WriteType, ...}$

$\text{cost}(D, f) = \text{entropy}(P)$

Generate candidate $f$ from $f_{best}$

$P(\text{element} \mid f)$ to scale: iterative synthesis on fraction of examples

$O(|D|)$

Dataset $D$ of millions ($\approx 10^8$)

Synthesizer

Build Probabilistic Model $P$
Step 4: use $f_{best}$ to predict

program

```
elem.notify{
  ...
  ...
  {
    position: 'top',
    hide: false,
    ?
  }
};
```

Context $\gamma$

```
{ }
  {hide}
  {hide}
  {hide, 3}
  {hide, 3}
  {hide, 3}
  {hide, 3}
```

\[
\{ \text{Previous Property, Parameter Position, API name} \} \]
Deep3: Experimental Results
[Probabilistic Model of JavaScript]

Dataset $D$: 150,000 files  
Training Time: $\sim100$ hours  
$f_{best} \sim 50,000$ instr.

<table>
<thead>
<tr>
<th>Probabilistic Model</th>
<th>Accuracy [APIs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last two tokens, Hindle et. al. [ICSE’12]</td>
<td>22.2%</td>
</tr>
<tr>
<td>Last two APIs, Raychev et. al. [PLDI’14]</td>
<td>30.4%</td>
</tr>
<tr>
<td><strong>Deep3</strong></td>
<td><strong>66.6%</strong></td>
</tr>
</tbody>
</table>

Details in: “Probabilistic Model for Code with Decision Trees”, ACM OOPSLA’16
**Applying the Concept to Natural Language**

**Dataset D:**
Hutter Prize Wikipedia Dataset

**Training Time:** \(~ 8 \text{ hours}\)

**f_{\text{best}} \sim 9,000 \text{ instr}**

Interpretable model, browse here:
http://www.srl.inf.ethz.ch/charmmodel.html

<table>
<thead>
<tr>
<th>Probabilistic Model</th>
<th>Bits-per-Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-gram (best)</td>
<td>1.94</td>
</tr>
<tr>
<td>Stacked LSTM (Graves 2013)</td>
<td>1.67</td>
</tr>
<tr>
<td>Char-based DSL synthesis</td>
<td>1.62</td>
</tr>
<tr>
<td>MRNN (Sutskever 2011)</td>
<td>1.60</td>
</tr>
<tr>
<td>MI-LSTM (Wu et al. 2016)</td>
<td>1.44</td>
</tr>
<tr>
<td>HM-LSTM* (Chung et al. 2016)</td>
<td>1.40</td>
</tr>
</tbody>
</table>
function isBig(v) {
    return v < this.length
}
[12, 5].filter(isBig);

VarPtsTo("global", h)
checkIfInsideMethodCall
checkMethodCallName
checkReceiverType
checkNumberOfArguments ...

VarPtsTo(this, h)
More Resources

Learning from Large Codebases,
ACM Doctoral Dissertation, Honorable Mention Award

• http://plml.ethz.ch

• Dagstuhl Seminar on Big Code Analytics, Nov 2015

• Data sets, tools, challenges: http://learningfrombigcode.org

Veselin Raychev
**Summary**

**OPPORTUNITY**

Advances in Programming Languages
- Automated Reasoning, Synthesis, Constraint Solving

Advances in Machine Learning
- Deep Learning, Graphical Models, Language Models

**RICH PROBLEM SPACE**

- Applications
  - Code completion
  - Deobfuscation
  - Program synthesis
  - Translation

- Intermediate Representation
  - Sequences (sentences)
  - Translation Table
  - Trees
  - Graphical Models (CRFs)
  - Feature Vectors

- Analyze Program (PL)
  - Typestate analysis
  - Scope analysis
  - Control-flow analysis
  - Alias analysis

- Train Model (ML)
  - Neural Networks
  - PCFGs
  - N-gram language model
  - SVM
  - Structured SVM

- Query Model (ML)
  - \( \text{argmax} \) \( f \in \mathcal{D}, f \in \mathcal{F} \)
  - MAP inference

**NEW PROBABILISTIC MODELS**

1. **Pick** a structure of interest, e.g., trees:

   \[ \text{TCond} ::= \# \mid \text{WriteOp TCond} \mid \text{MoveOp TCond} \]

   \[ \text{MoveOp} ::= \text{Up}, \text{Left}, \text{Right}, \text{DownFirst}, \text{DownLast}, \text{NextLeaf}, \text{Previous}, \text{PrevNodeType}, \text{PrevNodeValue} \]

   \[ \text{WriteOp} ::= \text{WriteValue}, \text{WriteType}, \text{WritePos} \]

2. **Define** a DSL for expressing functions:
   (can be Turing complete)

3. **Synthesize** \( f_{\text{best}} \in \mathcal{D} \) from Dataset \( D \):

   \[ f_{\text{best}} = \text{argmin}_{f \in \mathcal{D}} \text{cost}(D, f) \]

4. **Use** \( f_{\text{best}} \) on new structures:

   \[ f_{\text{best}} \left( \begin{array}{c} \text{structure} \end{array} \right) \rightarrow \gamma \]

**PRACTICAL IMPACT**

more: http://plml.ethz.ch