Machine Learning for Programming

Martin Vechev
Department of Computer Science
ETH Zurich
Machine Learning for Programming

Probabilistically likely solutions to problems impossible to solve otherwise

Publications:
Predicting Program Properties from “Big Code”, ACM POPL’15
Programming with Big Code, SNAPL 2015
Code Completion with Statistical Language Models, ACM PLDI’14
Machine Translation for Programming Languages, ACM Onward’14
Statistical Feedback Generation for Programs, ETH TR
Fast and Precise Statistical Code Completion, ETH TR

Tools:
JSNICE (used worldwide)
   statistical de-obfuscation
SLANG
   statistical code synthesis
SAGE
   statistical feedback generation

More information: http://www.srl.inf.ethz.ch/
Scene Completion

Input
Scene Completion
[Scene Completion Using Millions of Photographs, ACM SIGGRAPH 2007]
Program Synthesis

```java
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
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);
Martin is talking at the ExCape summer school now.
Programming Language Translation

Phrase-based statistical translation of programming languages, ACM Onward 2014

```
C#  Java  Translate

Console.WriteLine("Hi");
...

System.out.println("Hi");
...
```
Image de-noisingification
function FZ(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;
}
JSNice.org
[Predicting program properties from Big Code, ACM POPL 2015]

193 countries
100,000 users
Top ranked tool

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. Great for #learning and reverse engineering.

Alvaro Sanchez @alvasavi · Jun 19
This is gold.
Statistical renaming, Type info

Alex Vanston @mvdot · Jun 30
I've been looking for this for years: JS NICE buff.ly/1pQ5qfr #javascript #unminify #deobfuscate #makeltReadable

Kamil Tomšík @cztomisik · Jun 6
tell me how this works!
de-minify #jquery #javascript incl. args, vars & #jsdoc impressive! jsnice.org
Machine Learning for Programming

- Applications
- Intermediate Representation
- Analyze Program (PL)
- Train Model (ML)
- Query Model (ML)
# 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</td>
<td>Sequences (sentences)</td>
<td>Translation Table</td>
<td>Trees</td>
<td>Graphical Models (CRFs)</td>
<td>Feature Vectors</td>
</tr>
<tr>
<td>Representation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></td>
<td>SVM</td>
<td>Structured SVM</td>
<td></td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>argmax $P(y \mid x)$</td>
<td>$y \in \Omega$</td>
<td>Greedy</td>
<td>MAP inference</td>
<td></td>
</tr>
</tbody>
</table>
## Machine Learning for Programming

### Applications
- Code completion
- Deobfuscation
- Program synthesis
- Feedback generation
- Translation

### Intermediate Representation
- Sequences (sentences)
- Translation Table
- Trees

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

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

### Query Model (ML)
- \[
  \text{argmax } \ P(y | x) \\
  y \in \Omega
  \]

More information and tutorials at:
Goal

function f(a) {
    var b = document.getElementById(a);
    return b;
}
function f(a) {
    var b = document.getElementById(a);
    return b;
}

unknown facts: a b

known facts: f document getElementById
function $f(a)$ {
    var $b = \text{document.getElementById}(a);$  
    return $b;$ 
}
Challenges

Facts to be predicted are dependent

Many candidate choices

Must quickly learn from huge codebases

Prediction should be fast (real-time)
Key Idea

Phrase the problem of predicting program facts as

Structured Prediction for Programs
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]

First connection between Programs and Conditional Random Fields
Conditional Random Field

[J. Lafferty, A. McCallum, F. Pereira, ICML 2001]

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

Undirected Probabilistic Graphical Model

Captures dependence between facts to be predicted

Represents a conditional distribution on known facts
Conditional Random Field

[J. Lafferty, A. McCallum, F. Pereira, ICML 2001]

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

Example: Let \( y = \text{first}, \text{last} \) and \( x = \text{town} \)
Conditional Random Field

[J. Lafferty, A. McCallum, F. Pereira, ICML 2001]

$$P(y \mid x) = \frac{1}{Z(x)} \exp(w^T f(y, x))$$

Example: Let $y =$ first, last and $x =$ town

<table>
<thead>
<tr>
<th>first</th>
<th>town</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>Boston</td>
<td>0.1</td>
</tr>
<tr>
<td>James</td>
<td>Boston</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>first</th>
<th>last</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>Smith</td>
<td>0.7</td>
</tr>
<tr>
<td>James</td>
<td>Chandra</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Conditional Random Field

[J. Lafferty, A. McCallum, F. Pereira, ICML 2001]

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

Example: Let \( y = \text{first, last} \) and \( x = \text{town} \)

<table>
<thead>
<tr>
<th>first</th>
<th>town</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>Boston</td>
<td>0.1</td>
</tr>
<tr>
<td>James</td>
<td>Boston</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[ P(\text{first, last} \mid \text{town}) = \frac{\exp (0.1 \cdot f_1 + 0.3 \cdot f_2 + 0.7 \cdot f_3 + 0.4 \cdot f_4)}{Z(\text{town})} \]
Structured SVM Training
[N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007]

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

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

[N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007]

\[ P(y \mid x) = \frac{1}{z(x)} \exp(w^T f(y, x)) \]

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

Optimization objective (max-margin training):

\[ \forall j \forall y \sum w_i f_i(x^{(j)}, y^{(j)}) \geq \sum w_i f_i(x^{(j)}, y) + \Delta(y, y^{(j)}) \]
Structured SVM Training
[N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007]

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

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

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
Structured SVM Training
[N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007]

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

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

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 \( Z(x) \)
Querying the model: MAP Inference

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

Given \( x \), we would like to predict \( y = y_1, y_2, \ldots, y_n \) that maximizes \( P(y | x) \)

This requires us to make a joint prediction, together for all \( y_1, y_2, \ldots, y_n \)
Querying the model: MAP Inference

\[ P(y \mid x) = \frac{1}{z(x)} \exp(w^T f(y, x)) \]

Given \( x \), we would like to predict \( y = y_1, y_2, \ldots, y_n \) that maximizes \( P(y \mid x) \).

This requires us to make a joint prediction, together for all \( y_1, y_2, \ldots, y_n \)

\[ y^{\text{best}} = \arg\max_{y \in \Omega_x} P(y \mid x) = \arg\max_{y \in \Omega_x} w^T f(y, x) \]
Querying the model: MAP Inference

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

Given \( x \), we would like to predict \( y = y_1, y_2, \ldots, y_n \) that maximizes \( P(y | x) \).

This requires us to make a joint prediction, together for all \( y_1, y_2, \ldots, y_n \).

\[ y^{\text{best}} = \arg\max_{y \in \Omega_x} P(y | x) = \arg\max_{y \in \Omega_x} w^T f(y, x) \]
Querying the model: Marginals

\[ y_{1}^{\text{best}} = \arg \max P(y_1 | x) \]  
\[ \ldots \ldots \ldots \]  
\[ y_{n}^{\text{best}} = \arg \max P(y_n | x) \]

\[ y^{\text{best}} = (y_{1}^{\text{best}}, \ldots, y_{n}^{\text{best}}) \]

\( \Sigma \Pi \) belief propagation algorithms answer marginal queries
Querying the model: Marginals

\[ y_1^{\text{best}} = \arg \max P(y_1 | x) \]

\[ y_{\text{best}} = (y_1^{\text{best}}, \ldots, y_n^{\text{best}}) \]

\[ \sum \text{propagation algorithms answer marginal queries} \]
Queries: MAP vs. Marginals

**MAP Inference:**  
\[(y_1, \ldots, y_n)^{\text{MAP}} = \arg\max_{y_1, \ldots, y_n} P(y_1, \ldots, y_n)\]

**VS.**

**Marginals:**  
\[(y_1, \ldots, y_n)^{\text{ML}} = (y_1^{\text{ML}}, \ldots, y_n^{\text{ML}})\]

\[y_1^{\text{ML}} = \arg\max_{y_1} P(y_1) \quad \ldots \quad y_n^{\text{ML}} = \arg\max_{y_n} P(y_n)\]
Queries: MAP vs. Marginals

**MAP Inference:**

\[
(y_1, \ldots, y_n)^{MAP} = \arg\max_{y_1, \ldots, y_n} P(y_1, \ldots, y_n)
\]

**VS.**

**Marginals:**

\[
(y_1, \ldots, y_n)^{ML} = (y_1^{ML}, \ldots, y_n^{ML})
\]

\[
y_1^{ML} = \arg\max_{y_1} P(y_1) \quad \ldots \quad y_n^{ML} = \arg\max_{y_n} P(y_n)
\]

Consider the following probability distribution:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.29</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.31</td>
</tr>
</tbody>
</table>
Queries: MAP vs. Marginals

MAP Inference: \( (y_1, \ldots, y_n)^{\text{MAP}} = \text{argmax}_{y_1, \ldots, y_n} P(y_1, \ldots, y_n) \)

VS.

Marginals: \( (y_1, \ldots, y_n)^{\text{ML}} = (y_1^{\text{ML}}, \ldots, y_n^{\text{ML}}) \)

\( y_1^{\text{ML}} = \text{argmax}_{y_1} P(y_1) \) \hspace{1cm} \( y_n^{\text{ML}} = \text{argmax}_{y_n} P(y_n) \)

Consider the following probability distribution:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>0.29</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0.15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>0.31</td>
</tr>
</tbody>
</table>

\( (1,1)^{\text{MAP}} \)  
\( \text{argmax}_{A,B} P(A, B) \)

\( (0,1)^{\text{ML}} \)  
\( \text{argmax}_{A} P(A) \)  
\( \text{argmax}_{B} P(B) \)
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;
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;
```

argmax \( w^T f(i, t, r, length) \)

Unknown facts: \( t, r, i, \ldots \)

Known facts: length, \ldots

i  t  w
i  step  0.5
j  step  0.4

i  r  w
i  len  0.6
i  length  0.3

\( r \)  length  \( w \)
length  length  0.5
len  length  0.3
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(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, length)
Structured Prediction for Programs
[Predicting program properties from Big Code, ACM POPL 2015]
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]
Structured Prediction for Programs
[Predicting program properties from Big Code, ACM POPL 2015]

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

var colNames = [];
var len = str.length;
var i = 0;
for (; i < len; i += step)
if (i + step < len)
  colNames.push(str.subs(i, i + step));
else
  colNames.push(str.subs(i, len));
return colNames;
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]

```javascript
var n = [];
var r = e.length;
var i = 0;
for (; i < r; i += t)
  if (i + t < r)
    n.push(e.subs(i, i + t));
  else
    n.push(e.subs(i, r));
return n;
```

```javascript
var colNames = [];
var len = str.length;
var i = 0;
for (; i < len; i += step)
  if (i + step < len)
    colNames.push(str.subs(i, i + step));
  else
    colNames.push(str.subs(i, len));
return colNames;
```
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]

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

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

150MB

Prediction Phase

Learning Phase

SSVM learning

max-margin training

Conditional Random Field

$P(y \mid x)$

program analysis

MAP inference

transform

program analysis

alias, call analysis

7M feature functions for names

70K feature functions for types
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]

**Prediction Phase**
- Program analysis
- MAP inference
- Transform
- Prediction

**Learning Phase**
- Program analysis
- SSVM learning
- Max-margin training
- Conditional Random Field $P(y | x)$
- 150MB

30 nodes, 400 edges

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

```
var colNames = [];
var len = str.length;
var i = 0;
for (; i < len; i += step)
  if (i + step < len)
    colNames.push(str.subs(i, i + step));
  else
    colNames.push(str.subs(i, len));
return colNames;
```
Structured Prediction for Programs

(Predicting program properties from Big Code, ACM POPL 2015)

Time: milliseconds

Prediction Phase

program analysis → MAP inference → transform

Learning Phase

program analysis → SSVM learning → max-margin training

30 nodes, 400 edges

150MB

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

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

var

150MB

Conditional Random Field

P(y | x)

max-margin training

7M feature functions for names
70K feature functions for types

alias, call analysis
Structured Prediction for Programs

[Predicting program properties from Big Code, ACM POPL 2015]
# 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>SVM</td>
<td>N-gram language model</td>
<td>Structured SVM</td>
<td></td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>argmax ( P(y \mid x) ) ( y \in \Omega )</td>
<td>Greedy</td>
<td>MAP inference</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next 10 years
Next 10 years

Knowledge

Methods

Applications

Frameworks

Potts model of programs?

Nice2Predict.org

SLANG

JS NICE

2014 2017 2020 2024

Time
Machine Learning for Programming

**TREND**

**IMPACT**

**DIMENSIONS**

**FUTURE**

More information: http://www.srl.inf.ethz.ch/