Machine Learning for Programs

Martin Vechev

Pavol Bielik

Department of Computer Science

ETH Zurich
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)

A group of people shopping at an outdoor market. There are many vegetables at the fruit stand.
Learning and probabilistic models based on Big Data have revolutionized entire fields

<table>
<thead>
<tr>
<th>Natural Language Processing (e.g., machine translation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Vision (e.g., image captioning)</td>
</tr>
<tr>
<td>Medical Computing (e.g., disease prediction)</td>
</tr>
</tbody>
</table>

Can we bring this revolution to programmers?
Vision

New kinds of techniques and tools that learn from Big Code

15 million repositories
Billions of lines of code
High quality, tested, maintained programs

number of repositories

last 8 years
Probabilistic Learning from Big Code

Probabilistically likely solutions to problems impossible to solve otherwise

Publications
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

Publicly Available Tools
http://JSNice.org
statistical de-obfuscation

http://Nice2Predict.org
structured prediction framework

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

• Motivation
  • Applications

• Statistical Language Models
  • N-gram, Recurrent Networks, PCFGs
  • Application: code completion

____________________________

• Graphical Models
  • Markov Networks, Conditional Random Fields
  • Inference & Learning in Markov Networks
  • Application: predicting names and types

____________________________

• Learning Features for Programs
  • Combining Program Synthesis + Machine Learning
Statistical Programming Tools

Write new code [PLDI’14]:
Code Completion

Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
?


Statistical Programming Tools

Write new code [PLDI’14]:
Code Completion

Camera camera = Camera.open();
camera.SetDisplayOrientation(90);

Port code [ONWARD’14]:
Programming Language Translation
Write new code [PLDI’14]:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
? 
```

Port code [ONWARD’14]:
Programming Language Translation

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

Debug code:
Statistical Bug Detection

```
... for x in range(a):
print a[x]
```

likely error
Statistical Programming Tools

Write new code [PLDI’14]:
Code Completion

Camera camera = Camera.open();
camera.SetDisplayOrientation(90);

Port code [ONWARD’14]:
Programming Language Translation

Understand code/security [POPL’15]:
JavaScript Deobfuscation
Type Prediction

Debug code:
Statistical Bug Detection

For x in range(a):
print a[x]
JSNice.org

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

✅ Every country  ✅ 150,000+ users  ✅ Top ranked tool

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 @alvasavi · Jun 10
This is gold. Statistical renaming, Type inference and Deobfuscation.

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

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

CodeGeekz

20 Essential Tools for Coders

20 Best Freebies for Web Designers of Year 2014

JavaScript Diagram

right place. I have gathered 20 Ex)19 javascript development tasks. Following Dev Beauty, store your snippet and if you will find the list handy and access...

1. JS Nice

JSuICE

JavaScript速習教科書 — JSuICE.
Statistical Programming Tools

Write new code [PLDI’14]:
Code Completion

Port code [ONWARD’14]:
Programming Language Translation

Understand code/security [POPL’15]:
JavaScript Deobfuscation
Type Prediction

Debug code:
Statistical Bug Detection

All of these benefit from the probabilistic models for code.
Tutorial Outline

• Motivation
  • Applications

• **Statistical Language Models**
  • N-gram, Recurrent Networks, PCFGs
  • Application: code completion

• Graphical Models
  • Markov Networks, Conditional Random Fields
  • Inference & Learning in Markov Networks
  • Application: predicting names and types

• Learning Features for Programs
  • Combining Program Synthesis + Machine Learning
Machine Learning for Programs

- Applications
- Intermediate Representation
- Analyze Program (PL)
- Train Model (ML)
- Query Model (ML)
# Machine Learning for Programs

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze Program (PL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train Model (ML)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Machine Learning for Programs

<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 Trees</td>
<td>Graphical Models (CRFs)</td>
<td>Feature Vectors</td>
<td></td>
</tr>
</tbody>
</table>

- **Analyze Program (PL)**
- **Train Model (ML)**
- **Query Model (ML)
## Machine Learning for Programs

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

### Query Model (ML)
# Machine Learning for Programs

## Applications

<table>
<thead>
<tr>
<th>Code completion</th>
<th>Deobfuscation</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
</table>

## Intermediate Representation

<table>
<thead>
<tr>
<th>Sequences (sentences)</th>
<th>Translation Table</th>
<th>Trees</th>
<th>Graphical Models (CRFs)</th>
<th>Feature Vectors</th>
</tr>
</thead>
</table>

## Analyze Program (PL)

<table>
<thead>
<tr>
<th>typestate analysis</th>
<th>scope analysis</th>
<th>control-flow analysis</th>
<th>alias analysis</th>
</tr>
</thead>
</table>

## Train Model (ML)

<table>
<thead>
<tr>
<th>Neural Networks</th>
<th>SVM</th>
<th>Structured SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-gram language model</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Query Model (ML)
# Machine Learning for Programs

## Applications
- Code completion
- Deobfuscation
- Program synthesis
- Feedback generation
- 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
- SVM
- N-gram language model
- Structured SVM

## Query Model (ML)
- \[ \text{argmax} \ P(y | x) \]
- \( y \in \Omega \)
- Greedy MAP inference
# Machine Learning for Programs

![Diagram](image)

**Applications**
- Code completion
- Deobfuscation
- Program synthesis
- Feedback generation
- 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
- SVM
- Structured SVM
- N-gram language model

**Query Model (ML)**
- \[ \text{argmax} \ P(y \mid x) \]
- \[ y \in \Omega \]
- Greedy
- MAP inference

**Text**
- Machine Learning for Programs
- Code completion
- Deobfuscation
- Program synthesis
- Feedback generation
- Translation
- Sequences (sentences)
- Translation Table
- Trees
- Graphical Models (CRFs)
- Feature Vectors
- Typestate analysis
- Scope analysis
- Control-flow analysis
- Alias analysis
- Neural Networks
- SVM
- Structured SVM
- N-gram language model
- \[ \text{argmax} \ P(y \mid x) \]
- \[ y \in \Omega \]
- Greedy
- MAP inference
Motivation: Working with APIs
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

MediaRecorder rec = new MediaRecorder();

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");

...
Camera camera = Camera.open();
camera.setDisplayOrientation(90);

camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");

...
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");
...

Handles multiple objects
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();

MediaRecorder rec = new MediaRecorder();
rec.setCamera(camera);
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");
...

Handles multiple objects

Infers multiple statements
Statistical Code Synthesis: Capabilities

[Code Completion with Statistical Language Models, V. Raychev, M. Vechev, E. Yahav ,PLDI’14]

```java
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();

MediaRecorder rec = new MediaRecorder();
rec.setCamera(camera);
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");
...```

- Handles multiple objects
- Infers sequences not in training data
- Infers multiple statements
Key Insight

Regularities in code are similar to regularities in natural language
Regularities in code are similar to regularities in natural language

We want to learn that

```java
MediaRecorder rec = new MediaRecorder();
```

is before

```java
rec.setCamera(camera);
```
Key Insight

Regularities in code are similar to regularities in natural language

We want to learn that

```
    MediaRecorder rec = new MediaRecorder();
```

is before

```
    rec.setCamera(camera);
```

like in natural languages

```
    Hello
```

is before

```
    World!
```
she = new X();
me = new Y();
me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);
she = new X();
me = new Y();

me.sleep();
if (random()) {
    me.eat();
}

she.enter();
me.talk(she);
From Programs to Sentences

```
she = new X();
me = new Y();

typestate analysis

me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);

Alias analysis
```
From Programs to Sentences

declare new objects

\begin{verbatim}
she = new X();
me = new Y();
me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);
\end{verbatim}

Typestate analysis
Alias analysis
for abstract object \texttt{me}:
she = new X();
me = new Y();

me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);

Typestate analysis
Alias analysis

for abstract object me:
    Y_{init} sleep talk
From Programs to Sentences

she = new X();
me = new Y();

me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);

Typestate analysis
Alias analysis

for abstract object me:
    Y
      init  sleep  talk
      init  sleep  eat  talk
she = new X();
me = new Y();

me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);

Typestate analysis
Alias analysis

for abstract object me:
    Y\text{init} \text{ sleep talk}
    Y\text{init} \text{ sleep eat talk}

for abstract object she:
From Programs to Sentences

```java
she = new X();
me = new Y();

me.sleep();
if (random()) {
    me.eat();
}
she.enter();
me.talk(she);
```

Typestate analysis

Alias analysis

for abstract object `me`:

```
Y_{init} sleep talk
Y_{init} sleep eat talk
```

for abstract object `she`:

```
X_{init} enter talk_{param1}
```
From Programs to Sentences
Learn Regularities

Learn regularities in obtained sentences

Regularities in sentences $\Leftrightarrow$ regularities in API usage

If we see many sequences like:

$$X_{init} \text{ enter } \text{talk}_{\text{param1}}$$

then we should learn that $\text{talk}_{\text{param1}}$ is often after $\text{enter}$
Statistical Language Models

Given a sentence \( s = w_1 \; w_2 \; w_3 \; \ldots \; w_n \)

estimate \( P(\; w_1 \; w_2 \; w_3 \; \ldots \; w_n \;) \)

Queries:

\( P(\text{“Hello World!”}) \gg P(\text{“World! Hello”}) \)

\[ \arg \max_y P(\text{“Hello”} \; y) \]

\( y \)
Statistical Language Models

Given a sentence \( s = w_1 \, w_2 \, w_3 \ldots \, w_n \)

estimate \( P(\, w_1 \, w_2 \, w_3 \ldots \, w_n \, ) \)

Decomposed to conditional probabilities

\[ P(\, w_1 \, w_2 \, w_3 \ldots \, w_n \, ) = \prod_{i=1..n} P(\, w_i \mid w_1 \ldots w_{i-1} \, ) \]
Statistical Language Models

Given a sentence $s = w_1 \ w_2 \ w_3 \ldots \ w_n$

estimate $P(w_1 \ w_2 \ w_3 \ldots \ w_n)$

Decomposed to conditional probabilities

$$P(w_1 \ w_2 \ w_3 \ldots \ w_n) = \prod_{i=1..n} P(w_i \mid w_1 \ldots w_{i-1})$$

$P(\text{The quick brown fox jumped}) =$

$P(\text{The}) \ P(\text{quick} \mid \text{The}) \ P(\text{brown} \mid \text{The quick})$

$P(\text{fox} \mid \text{The quick brown}) \ P(\text{jumped} \mid \text{The quick brown fox})$
N-gram language model

Conditional probability only on previous \( n-1 \) words

\[
P( w_i \mid w_1 \ldots w_{i-1} ) \approx P( w_i \mid w_{i-n+1} \ldots w_{i-1} )
\]
N-gram language model

Conditional probability only on previous $n-1$ words

$$P( w_i \mid w_1 \ldots w_{i-1} ) \approx P( w_i \mid w_{i-n+1} \ldots w_{i-1} )$$

$n-1$ words
N-gram language model

Conditional probability only on previous n-1 words

\[ P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i \mid w_{i-n+1} \ldots w_{i-1}) \approx \frac{\#(w_{i-n+1} \ldots w_{i-1} w_i)}{\#(w_{i-n+1} \ldots w_{i-1})} \]
N-gram language model

Conditional probability only on previous \(n-1\) words

\[
P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i \mid w_{i-n+1} \ldots w_{i-1}) \approx \frac{\#(w_{i-n+1} \ldots w_{i-1} w_i)}{\#(w_{i-n+1} \ldots w_{i-1})}
\]

Training is achieved by counting n-grams. E.g., with 3-gram language model, we get:

\[
P(\text{jumped} \mid \text{The quick brown fox}) \approx P(\text{jumped} \mid \text{brown fox}) \approx \frac{\#(\text{brown fox jumped})}{\#(\text{brown fox})}
\]

\(#(\text{n-gram})\) - number of occurrences of n-gram in training data

Time complexity for each word encountered in training is constant, so training is usually fast.
Tri-gram language model

\[ P(w_1 \cdot w_2 \cdot w_3 \cdot \ldots \cdot w_n) \approx P(w_1) \times P(w_2 | w_1) \times P(w_3 | w_1 \cdot w_2) \times \ldots \times P(w_n | w_{n-2} \cdot w_{n-1}) \]

<table>
<thead>
<tr>
<th>3-grams</th>
<th># of occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>brown fox jumped</td>
<td>125</td>
</tr>
<tr>
<td>brown fox walked</td>
<td>45</td>
</tr>
<tr>
<td>brown fox snapped</td>
<td>30</td>
</tr>
</tbody>
</table>

\[ P(\text{jumped} | \text{brown fox}) \approx \frac{125}{200} \approx 0.625 \]
Tri-gram language model

\[ P (w_1 \cdot w_2 \cdot w_3 \cdot \ldots \cdot w_n) \approx P (w_1) \times P (w_2 | w_1) \times P (w_3 | w_1 \cdot w_2) \times \ldots \times P (w_n | w_{n-2} \cdot w_{n-1}) \]

<table>
<thead>
<tr>
<th>3-grams</th>
<th># of occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>brown fox jumped</td>
<td>125</td>
</tr>
<tr>
<td>brown fox walked</td>
<td>45</td>
</tr>
<tr>
<td>brown fox snapped</td>
<td>30</td>
</tr>
</tbody>
</table>

\[ P (\text{jumped} \mid \text{brown fox}) \approx \frac{125}{200} \approx 0.625 \]

\[ P (\text{brown fox jumped}) \approx \]

\[ \frac{200}{600} \times \frac{200}{200} \times \frac{125}{200} \approx 0.208 \]
Key Problem: Sparsity of Data

What if this number is 0?

\[
P(\text{jumped} \mid \text{red fox}) \approx \frac{\#(\text{red fox jumped})}{\#(\text{red fox})}
\]

The problem of sparsity gets worse as the size of the n-gram becomes larger.

We need to handle n-grams with 0 or few occurrences in the training data. Techniques that can do that are: smoothing, discounting
Solution: Smoothing Techniques

• Smoothing techniques give non-zero probability to n-grams not in the training data.

• Essentially, they try to estimate how likely it is that the n-gram is missing due to the limited size of the training data.

• They work by taking the probability mass of the existing n-grams and redistributing that mass over n-grams that occur zero times.

• Typically, probability of such n-grams is estimated by looking at the probabilities of n-1 grams, n-2 grams, unigrams, etc.
Smoothing: Intuitively

Distribution of probability mass before smoothing

Training Data

Smoothing

Distribution of probability mass after smoothing

Training Data

All other sentences get non-zero probability

All other sentences get 0 probability
Smoothing: Intuitively

Distribution of probability mass before smoothing

Training Data

All other sentences get 0 probability

Distribution of probability mass after smoothing

Training Data

All other sentences get non-zero probability
Smoothing Techniques

Backoff techniques:

\[
P_{\text{smooth}}(w_i | w_{i-N+1}) = \begin{cases} 
    P_{\text{ML}}(w_i | w_{i-n+1} \ldots w_{i-1}) \quad & \text{if } w_{i-n+1} \ldots w_i \text{ in training data} \\
    \gamma(w_{i-N+1}) \times P_{\text{smooth}}(w_i | w_{i-n+2} \ldots w_{i-1}) \quad & \text{otherwise}
\end{cases}
\]

Example smoothing techniques are: Witten-Bell Interpolated (WBI), Witten-Bell Backoff (WBB), Natural Discounting (ND), Stupid-Backoff (SB), …
Smoothing Techniques

Backoff techniques:

\[ P_{\text{smooth}}(w_i | w_{i-N+1}) = \begin{cases} 
P_{\text{ML}}(w_i | w_{i-n+1} \ldots w_{i-1}) & \text{if } w_{i-n+1} \ldots w_i \text{ in training data} \\
\gamma(w_{i-N+1}) \times P_{\text{smooth}}(w_i | w_{i-n+2} \ldots w_{i-1}) & \text{otherwise}
\end{cases} \]

Interpolation techniques:

\[ P_{\text{smooth}}(w_i | w_{i-N+1}) = \lambda \times P_{\text{ML}}(w_i | w_{i-N+1}) + (1 - \lambda) \times P_{\text{smooth}}(w_i | w_{i-N+2}) \]

Example smoothing techniques are: Witten-Bell Interpolated (WBI), Witten-Bell Backoff (WBB), Natural Discounting (ND), Stupid-Backoff (SB), …
Summary: N-gram Language Model

Conditional probability only on previous $n-1$ words

$$P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1}) \approx \frac{\#(w_{i-n+1} \ldots w_{i-1} w_i)}{\#(w_{i-n+1} \ldots w_{i-1})}$$

+ Easy to implement
+ Fast to train (simply counting)

- Data sparsity issues
- Can be tricky to capture long distance relationships

Existing library implementing language models is SRILM:
http://www.speech.sri.com/projects/srilm/
Learn Regularities

Training phase

Program Analysis → Train Language Model

Language model

sentences

N-gram
Recurrent Neural Networks (RNNs)

probability of “The”
probability of “brown”

output layer
softmax classifier

hidden layer
(state i)

input word

“The”
Recurrent Neural Networks (RNNs)

![Diagram of RNNs showing input word, hidden layer (state i), output layer softmax classifier, probability of "The" and probability of "brown". The input word is "The" and "quick" with corresponding probabilities indicated in the diagram.]
Recurrent Neural Networks (RNNs)

output layer
softmax classifier

probability of “The”

probability of “brown”

hidden layer
(state i)

input word

“The”

“quick”
Recurrent Neural Networks (RNNs)

Matrices $U$, $V$ and $W$ learned through back propagation with time and some unfolding.

Matrices $U$, $V$ and $W$ learned through back propagation with time and some unfolding.
Recurrent Neural Networks (RNNs)

Matrices $U$, $V$ and $W$ learned through back propagation with time and some unfolding.

Matrix $U$ learns a low-dimensional continuous vector representation of input words also referred to as “word embedding”.
Word Embeddings

Matrices $U$, $V$ and $W$ learned through back propagation with time and some unfolding.

Matrix $U$ learns a low-dimensional continuous vector representation of input words also referred to as “word embedding”.

[Linguistic Regularities in Continuous Space Word Representations, Mikolov et.al., 2013 (NAACL-HLT-2013)]
Summary: RNNs

+ RNNs can learn dependencies beyond the prior several words (LSTM networks)
+ Active research area
+ Learns continuous representation of input vocabulary

- Slow to train
- Difficult to train
- Not interpretable

RNNLM-40 Download:
http://rnnlm.org/
https://www.tensorflow.org/
Learn Regularities

Training phase

Program Analysis

Sentences

Train Language Model

Language model

N-gram + RNN

Github

Atlassian Bitbucket
The SLANG System

Training phase

sentences

Program Analysis  Train Language Model  N-gram + RNN

Language model

Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();

MediaRecorder rec = new MediaRecorder();
rec.setCamera(camera);
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VIDEO_SOURCE.DEFAULT);
rec.setOutputFormat(MediaRecorder.OUTPUT_FORMAT.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");
...

incomplete program

completed program

Program Analysis

Training phase

Language model

sentences

Program Analysis  Train Language Model  N-gram + RNN

Language model
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    ? {smsMgr, list} // (Hole H1)
} else {
    ? {smsMgr, message} // (Hole H2)
}
Statistical Code Synthesis

Abstract object `smsMgr`:
- `getDefault()`
- `divideMessage(H1)`
- `getDefault(H2)`

```java
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    // {smsMgr, list} // (Hole H1)
} else {
    // {smsMgr, message} // (Hole H2)
}
```
Statistical Code Synthesis

smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    ? {smsMgr, list} // (Hole H1)
} else {
    ? {smsMgr, message} // (Hole H2)
}

Abstract object smsMgr:
  getDefault_result divideMessage H1
  getDefault_result H2
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    ? {smsMgr, list} // (Hole H1)
} else {
    ? {smsMgr, message} // (Hole H2)
}
Statistical Code Synthesis

```java
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    ? {smsMgr, list} // (Hole H1)
} else {
    ? {smsMgr, message} // (Hole H2)
}
```

Abstract object `smsMgr`:
- `getDefault_result`
- `divideMessage H1`
- `getDefault_result H2`

Abstract object `list`:
- `divideMessage_result H1`
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    ? {smsMgr, list} // (Hole H1)
} else {
    ? {smsMgr, message} // (Hole H2)
}
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    ? {smsMgr, list} // (Hole H1)
} else {
    ? {smsMgr, message} // (Hole H2)
}
Abstract object $\text{smsMgr}$:

$$\text{getDefault}_{\text{result}} \ \text{divideMessage} \ H1$$

$$\text{arg max } P(\text{getDefault}_{\text{result}} \ \text{divideMessage} \ H1)$$
Abstract object `smsMgr`:

| getDefault\_result | divideMessage | sendMultipartTextMessage | 0.0033 |

\[
\arg\max_{\text{H1}} P(\text{getDefault}\_\text{result} \text{ divideMessage} \text{ H1})
\]
Statistical Code Synthesis

Abstract object `smsMgr`:

<table>
<thead>
<tr>
<th>Method</th>
<th>Completion probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDefaultr</code> <code>result</code></td>
<td>divideMessage <code>sendMultpartTextMessage</code></td>
</tr>
<tr>
<td><code>getDefaultr</code> <code>result</code></td>
<td>divideMessage <code>sendTextMessage</code></td>
</tr>
</tbody>
</table>
## Statistical Code Synthesis

Abstract object `smsMgr`:

<table>
<thead>
<tr>
<th>Operation Combination</th>
<th>Completion Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDefault_result</code> divideMessage sendMultipartTextMessage</td>
<td>0.0033</td>
</tr>
<tr>
<td><code>getDefault_result</code> divideMessage sendTextMessage</td>
<td>0.0016</td>
</tr>
<tr>
<td><code>getDefault_result</code> sendTextMessage</td>
<td>0.0073</td>
</tr>
<tr>
<td><code>getDefault_result</code> sendMultipartTextMessage</td>
<td>0.0010</td>
</tr>
<tr>
<td><code>divideMessage_result</code> sendMultipartTextMessage, param3</td>
<td>0.0821</td>
</tr>
<tr>
<td><code>length</code> length</td>
<td>0.0132</td>
</tr>
<tr>
<td><code>length</code> split</td>
<td>0.0080</td>
</tr>
<tr>
<td><code>length</code> sendTextMessage, param3</td>
<td>0.0017</td>
</tr>
</tbody>
</table>
## Statistical Code Synthesis

Abstract object `smsMgr`:

<table>
<thead>
<tr>
<th>Method</th>
<th>Completion probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDefault</code></td>
<td></td>
</tr>
<tr>
<td><code>divideMessage</code></td>
<td></td>
</tr>
<tr>
<td><code>sendMultipartTextMessage</code></td>
<td>0.0033</td>
</tr>
<tr>
<td><code>length</code></td>
<td></td>
</tr>
<tr>
<td><code>split</code></td>
<td></td>
</tr>
<tr>
<td><code>sendTextMessage</code></td>
<td></td>
</tr>
</tbody>
</table>

- `getDefault` result `divideMessage` `sendMultipartTextMessage` 0.0033
- `getDefault` result `divideMessage` `sendTextMessage` 0.0016
- `getDefault` result `sendTextMessage` 0.0073
- `getDefault` result `sendMultipartTextMessage` 0.0010
- `divideMessage` result `sendMultipartTextMessage` param3 0.0821
- `length` `length` 0.0132
- `length` `split` 0.0080
- `length` `sendTextMessage` param3 0.0017
# Statistical Code Synthesis

Abstract object `smsMgr`:

<table>
<thead>
<tr>
<th>Method Combination</th>
<th>Completion Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDefault</code> <code>result</code> <code>divideMessage</code> <code>sendMultipartTextMessage</code></td>
<td>0.0033</td>
</tr>
<tr>
<td><code>getDefault</code> <code>result</code> <code>divideMessage</code> <code>sendTextMessage</code></td>
<td>0.0016</td>
</tr>
<tr>
<td><code>getDefault</code> <code>result</code> <code>sendTextMessage</code></td>
<td>0.0073</td>
</tr>
<tr>
<td><code>divideMessage</code> <code>result</code> <code>sendMultipartTextMessage</code></td>
<td>0.0821</td>
</tr>
<tr>
<td><code>length</code> <code>length</code></td>
<td>0.0132</td>
</tr>
<tr>
<td><code>length</code> <code>split</code></td>
<td>0.0080</td>
</tr>
<tr>
<td><code>length</code> <code>sendTextMessage</code> <code>param3</code></td>
<td>0.0017</td>
</tr>
</tbody>
</table>

Not a feasible solution: completions disagree on selected method

The solution must satisfy program constraints
### Abstract object `smsMgr`

<table>
<thead>
<tr>
<th>Operation Sequence</th>
<th>Completion Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDefault</code> <code>result</code> <code>divideMessage</code> <code>sendMultipartTextMessage</code></td>
<td>0.0033</td>
</tr>
<tr>
<td><code>getDefault</code> <code>result</code> <code>divideMessage</code> <code>sendTextMessage</code></td>
<td>0.0016</td>
</tr>
<tr>
<td><code>getDefault</code> <code>result</code> <code>sendTextMessage</code></td>
<td>0.0073</td>
</tr>
<tr>
<td><code>getDefault</code> <code>result</code> <code>sendMultipartTextMessage</code></td>
<td>0.0010</td>
</tr>
<tr>
<td><code>divideMessage</code> <code>result</code> <code>sendMultipartTextMessage</code> <code>param3</code></td>
<td>0.0821</td>
</tr>
<tr>
<td><code>length</code> <code>length</code></td>
<td>0.0132</td>
</tr>
<tr>
<td><code>length</code> <code>split</code></td>
<td>0.0080</td>
</tr>
<tr>
<td><code>length</code> <code>sendTextMessage</code> <code>param3</code></td>
<td>0.0017</td>
</tr>
</tbody>
</table>
## Statistical Code Synthesis

### Abstract object `smsMgr`:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Completion Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDefaults</code> → <code>divideMessage</code> → <code>sendMultipartTextMessage</code></td>
<td>0.0033</td>
</tr>
<tr>
<td><code>getDefaults</code> → <code>divideMessage</code> → <code>sendTextMessage</code></td>
<td>0.0016</td>
</tr>
<tr>
<td><code>getDefaults</code> → <code>sendTextMessage</code></td>
<td>0.0073</td>
</tr>
<tr>
<td><code>getDefaults</code> → <code>sendMultipartTextMessage</code></td>
<td>0.0010</td>
</tr>
<tr>
<td><code>divideMessage</code> → <code>sendMultipartTextMessage</code> → <code>param3</code></td>
<td>0.0821</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Completion Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>length</code> <code>length</code></td>
<td>0.0132</td>
</tr>
<tr>
<td><code>length</code> <code>split</code></td>
<td>0.0080</td>
</tr>
<tr>
<td><code>length</code> <code>sendTextMessage</code> → <code>param3</code></td>
<td>0.0017</td>
</tr>
</tbody>
</table>
smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
    list = smsMgr.divideMessage(message);
    smsMgr.sendMultipartTextMessage(...list...);
} else {
    smsMgr.sendTextMessage(...message...);
}
The SLANG System

Completion phase

Incomplete program

Completed sentences

Completed program

Completion phase

Sentences with holes

Query

Combine

~100MB

Language model

84 testing samples

82

Training phase

Sentences

Program Analysis

Train Language Model

~700MB

~100MB

1M Java methods

Program Analysis

Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();

MediaRecorder rec = new MediaRecorder();
rec.setCamera(camera);
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
rec.setAudioEncoder(1);
rec.setVideoEncoder(3);
rec.setOutputFile("file.mp4");
...
Lessons

• Connection to PL
  • decide what to `\`compile`” into the word
  • Program Analysis need not be sound

• N-gram Models
  • Easy and cheap to train
  • Important to deal with sparsity via smoothing
  • Can be tricky to capture long distance relationships

• Recurrent Networks
  • Can capture longer distance relationships
  • More expensive to train
  • Experimentally similar to language models
Importance of Static Analysis

**Precision vs. % of Data used**

- **no alias analysis**
- **with alias analysis**

Static Analysis Benefit = 10x more Data
Models of Full Language

• So far:
  – N-gram and RNNs
  – Modelling specific elements (APIs)

• How can be build statistical language models of full programming language?
Tutorial Outline

• Motivation
  • Potential applications

• Statistical Language Models
  • N-gram, Recurrent Networks, PCFGs
  • Application: code completion

• Graphical Models
  • Markov Networks, Conditional Random Fields
  • Inference & Learning in Markov Networks
  • Application: predicting names and types

• Learning Features for Programs
  • Combining Program Synthesis + Machine Learning
Applications

Computer Vision  Natural Language Processing  Programming

the dog saw a man
in the park

fun chunkData(e, t)
    var n = [];
    var r = e.length;
    var i = 0;
    for (; i < r; i += t)
        ...
    return n;
Challenges

Facts to be predicted are dependent

Millions of 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
[V. Raychev, M. Vechev, A. Krause, ACM POPL’15]

First connection between Programs and Conditional Random Fields
# Machine Learning for Programs

<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>Markov Networks</td>
<td>Graphical Models (CRFs)</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 $P(y \mid x)$</td>
<td>$y \in \Omega$</td>
<td>Greedy MAP inference</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# Machine Learning for Programs

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deobfuscation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intermediate Representation</th>
<th>Sequences (sentences)</th>
<th>Translation Table</th>
<th>Graphical Models (CRFs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trees</td>
<td>Markov Networks</td>
<td>Log-linear Models</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyze Program (PL)</th>
<th>typestate analysis</th>
<th>control-flow analysis</th>
<th>alias analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>scope analysis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Train Model (ML)</th>
<th>Neural Networks</th>
<th>SVM</th>
<th>Structured SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N-gram language model</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Model (ML)</th>
<th>argmax $P(y \mid x)$</th>
<th>Greedy MAP inference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$y \in \Omega$</td>
<td></td>
</tr>
</tbody>
</table>

## Machine Learning for Programs

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code completion</td>
<td>Deobfuscation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intermediate Representation</th>
<th>Sequences (sentences)</th>
<th>Translation Table</th>
<th>Graphical Models (CRFs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typestate analysis</td>
<td></td>
<td></td>
<td>Log-linear Models</td>
</tr>
<tr>
<td>Scope analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control-flow analysis</td>
<td></td>
<td></td>
<td>Alias analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyze Program (PL)</th>
<th>Neural Networks</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-gram language model</td>
<td></td>
<td>Structured SVM</td>
</tr>
</tbody>
</table>

| Train Model (ML) | | |
|------------------|---------------------|
| SVM | Greedy MAP inference |

| Query Model (ML) | argmax $P(y | x)$ |
|------------------|------------------|
| $y \in \Omega$ | |

Defining a Global Probabilistic Model

Joint probability distribution

\[ P(A, B, C, D) \]
Defining a Global Probabilistic Model

Joint probability distribution

P(A, B, C, D)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>P(a0, b0, c0, d0) = 0.04</th>
<th>P(a0, b0, c0, d1) = 0.04</th>
<th>P(a0, b0, c1, d0) = 0.04</th>
<th>P(a0, b0, c1, d1) = 4.1 \times 10^{-6}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table of joint probabilities
Defining a Global Probabilistic Model

Joint probability distribution

\[ P(A, B, C, D) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>( P(a_0, b_0, c_0, d_0) = 0.04 )</th>
<th>( P(a_0, b_0, c_0, d_1) = 0.04 )</th>
<th>( P(a_0, b_0, c_1, d_0) = 0.04 )</th>
<th>( P(a_0, b_0, c_1, d_1) = 4.1 \times 10^{-6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table of joint probabilities

\( n \) variables \( \rightarrow O(2^n) \) entries!
Joint probability distribution

\[ P(A, B, C, D) = F(A, B, C, D) \]

\[ F(A, B, C, D) = \phi_1(A, B) \times \phi_2(B, C) \times \phi_3(C, D) \times \phi_4(D, A) \]

Define the global model by combining a set of local interactions between variables.
Factors

• A factor (or potential) \( \varphi \) is a function from a set of random variables \( \mathcal{D} \) to a real number \( \mathbb{R} \)

\[
\varphi: \mathcal{D} \rightarrow \mathbb{R}
\]

• The set of variables \( \mathcal{D} \) is the scope of the factor \( \varphi \)
  – we are typically concerned with non-negative factors

• Intuition: captures affinity, agreement, compatibility of the variables in \( \mathcal{D} \)
Factors: Example

• Example factors
  
  – $\varphi_1 (0,0) = 30$ says we believe A and B agree on 0 with belief 30

\[
\varphi_1 (A,B)
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>
Factors: Example

• Example factors
  – $\varphi_1 (0,0) = 30$ says we believe A and B agree on 0 with belief 30
  – $\varphi_3 (C,D)$ says that C and D argue all the time

<table>
<thead>
<tr>
<th>$\varphi_1$ (A,B)</th>
<th>$\varphi_2$ (B,C)</th>
<th>$\varphi_3$ (C,D)</th>
<th>$\varphi_4$ (D,A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>value</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>
Factors: Example

- Example factors
  - $\varphi_1 (0,0) = 30$ says we believe A and B agree on 0 with belief 30
  - $\varphi_3 (C,D)$ says that C and D argue all the time

<table>
<thead>
<tr>
<th>$\varphi_1$ (A,B)</th>
<th>$\varphi_2$ (B,C)</th>
<th>$\varphi_3$ (C,D)</th>
<th>$\varphi_4$ (D,A)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td><strong>B</strong></td>
<td><strong>value</strong></td>
<td><strong>B</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

Key point: (Local) factors have no probabilistic interpretation
Defining a Global Probabilistic Model

Joint probability distribution

\[ P(A, B, C, D) = F(A,B,C,D) \]

\[ F(A, B, C, D) = \phi_1 (A,B) \times \phi_2 (B,C) \times \phi_3 (C,D) \times \phi_4 (D,A) \]
Defining a Global Probabilistic Model

\[ P(A, B, C, D) = \frac{F(A, B, C, D)}{Z(A, B, C, D)} \]

\[ F(A, B, C, D) = \varphi_1(A, B) \times \varphi_2(B, C) \times \varphi_3(C, D) \times \varphi_4(D, A) \]

\[ Z(A, B, C, D) = \sum_{a \in A, b \in B, c \in C, d \in D} F(a, b, c, d) \]
### P(A,B,C,D)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Unnormalized (F)</th>
<th>Normalized (P = F/Z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>300,000</td>
<td>0.04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>300,000</td>
<td>0.04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>300,000</td>
<td>0.04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>30</td>
<td>4.1 \times 10^{-6}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ Z(A,B,C,D) = 7,201,840 \]

<table>
<thead>
<tr>
<th>( \varphi_1 ) (A,B)</th>
<th>( \varphi_2 ) (B,C)</th>
<th>( \varphi_3 ) (C,D)</th>
<th>( \varphi_4 ) (D,A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>value</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
P(A,B,C,D)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Unnormalized (F)</th>
<th>Normalized (P = F/Z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>300,000</td>
<td>0.04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>300,000</td>
<td>0.04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>300,000</td>
<td>0.04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>30</td>
<td>4.1 × 10⁻⁶</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ Z(A,B,C,D) = 7,201,840 \]

We can now answer probability queries on P(A, B, C, D):

For example: \[ P(A = 0, B = 0) = \sum_{c \in C, d \in D} P(0, 0, C, D) \] \[ \approx 0.125 \]
Markov Network

Let $X = \{X_1, \ldots, X_n\}$ be a set of random variables and $D_1, \ldots, D_m \subseteq X$

Then, a Markov Network is defined as:

$$P(X_1, X_2, \ldots, X_n) = \frac{\varphi_1(D_1) \times \varphi_2(D_2) \times \cdots \times \varphi_m(D_m)}{Z(X_1, X_2, \ldots, X_n)}$$

$$Z(X_1, X_2, \ldots, X_n) = \sum \varphi_1(D_1) \times \varphi_2(D_2) \times \cdots \times \varphi_m(D_m)$$
Graphs vs. Probability Distributions

We will next relate graphs and probability distributions. This is important for two reasons.

First, it allows us to determine properties (e.g., independence of variables) of a probability distribution directly from the graph.

Second, it allows us to answer queries (e.g., MAP inference) on the probability distribution by working with the graph.
Graphs vs. Probability Distributions

Which variables are independent?

\[ P(A, B, C, D) = \varphi_1 (A, B) \times \varphi_2 (B, D) \times \varphi_3 (D, C) \times \varphi_4 (C, A) / Z \]
Graphs vs. Probability Distributions

Intuitively, variables are nodes and factors introduce edges

\[
P(A, B, C, D) = \varphi_1(A, B) \times \varphi_2(B, D) \times \varphi_3(D, C) \times \varphi_4(C, A) / Z
\]
A distribution $P$ equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph $G$ if each $D_i$ is a complete subgraph of $G$. 
A distribution $P$ equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph $G$ if each $D_i$ is a complete subgraph of $G$.

\[
P(A, B, C, D) = \varphi_1(A, B) \times \varphi_2(B, D) \times \varphi_3(D, C) \times \varphi_4(C, A) / Z
\]
Graph Factorization

A distribution $P$ equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph $G$ if each $D_i$ is a complete subgraph of $G$.

$P(A, B, C, D) = \varphi_1(A, B) \times \varphi_2(B, D) \times \varphi_3(D, C) \times \varphi_4(C, A) / Z$

Here, $P$ factorizes over $G$. 

![Graph with nodes A, B, C, D connected by edges]
A distribution $P$ equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph $G$ if each $D_i$ is a complete subgraph of $G$.

$$P(A, B, C, D) = \frac{\varphi_1(A, B) \times \varphi_2(B, D) \times \varphi_3(D, C) \times \varphi_4(C, A)}{Z}$$

Here, $P$ factorizes over $G$.
Graph Factorization

A distribution $P$ equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph $G$ if each $D_i$ is a complete subgraph of $G$.

\[
P(A, B, C, D) = \varphi_1(A, B) \times \varphi_2(B, D) \times \varphi_3(D, C) \times \varphi_4(C, A) / Z
\]

Here, $P$ factorizes over $G$.

\[
P(A, B, C, D) = \varphi_1(A, B, C) \times \varphi_2(B, D, C) / Z
\]

Here, $P$ does not factorize over $G$. 
A distribution $P$ equipped with a set of factors $\varphi_1(D_1), \ldots, \varphi_m(D_m)$ factorizes over a graph $G$ if each $D_i$ is a complete subgraph of $G$.

\[ P(A, B, C, D) = \frac{\varphi_1(A, B) \times \varphi_2(A, C) \times \varphi_3(A, D) \times \varphi_4(B, C) \times \varphi_5(B, D) \times \varphi_6(C, D)}{Z} \]
Graphs vs. Probabilities

There is an important connection between a probability distribution that factorizes over a graph and the properties of the graph.

In particular, we can discover various independence properties of the probabilistic distribution directly from the graph.
Graph Separation

Two sets of disjoint nodes $A$ and $B$ in a graph $G$ are separated by a set $S$, if every path between $A$ and $B$ goes through a node in $S$. $A$ and $B$ are also separated if no path between $A$ and $B$ exists regardless of set $S$.

The **Global Markov Property** says that if $A$ and $B$ are separated by $S$, then

$$A \perp B \mid S \quad \text{that is} \quad P(A, B \mid S) = P(A \mid S) \times P(B \mid S)$$
Graph Separation: Example
Graph Separation: Example

Example of global independencies found in the above graph:

\[ \{A\} \perp \{G\} \mid \{D\} \]
\[ \{B\} \perp \{F\} \mid \{E,G\} \]

\[ P(A, G \mid D) = P(A \mid D) \times P(G \mid D) \]
Graphical Models: So far

So far we learned what a Markov Network is, what it means for a probability distribution to factor over a graph and how to extract information about independence of variables from the graph.

So far we have been discussing the joint probability distribution $P(X_1,\ldots,X_n)$. We next focus on conditional distributions.
So Far: Generative Models
Next: Discriminative Models

Predict *unknown* facts given some *known* facts
Next: Discriminative Models

Predict unknown facts given some known facts

do not model distribution over known facts
Conditional Random Field

Let $X \cup Y$ be a set of random variables where $X = \{X_1, \ldots, X_n\}$, $Y = \{Y_1, \ldots, Y_n\}$. Here, $X$ are the observed variables and $Y$ are the target variables.

Let $D_1, D_2, \ldots, D_m \subseteq X \cup Y$ where $D_i \not\subseteq X$

A Conditional Random Field is defined as:

$$P(Y_1, \ldots, Y_n \mid X_1, \ldots, X_n) = \frac{\prod \varphi_i(D_i)}{Z(X_1, \ldots, X_n)}$$

$$Z(X_1, \ldots, X_n) = \sum_Y \prod \varphi_i(D_i)$$

Note that the $Z$ function ranges over variables in $X$ only.
Conditional Random Field

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

Key advantage: avoids encoding distribution over the variables $X$. Thus, we can use many observed variables with complex dependencies without needing to model any joint distributions over them.

\[
P(Y_1, \ldots, Y_n \mid X_1, \ldots, X_n) = \frac{\varphi_1(D_1) \times \varphi_2(D_2) \times \cdots \times \varphi_m(D_m)}{Z(X_1, \ldots, X_n)}
\]

\[
Z(X_1, \ldots, X_n) = \sum_Y \varphi_1(D_1) \times \varphi_2(D_2) \times \cdots \times \varphi_m(D_m)
\]
Problem Statement:
Given a sentence, label each word with whether it is a person or a location
Problem Statement:
Given a sentence, label each word with whether it is a person or a location.

5 Labels: B-per, I-per, B-loc, I-loc, O
Conditional Random Field: Example
Text Analytics (Named Entity Recognition)

Problem Statement:
Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, O

Given:
Mr. Smith came today to Toronto
Conditional Random Field: Example
Text Analytics (Named Entity Recognition)

Problem Statement:
Given a sentence, label each word with whether it is a person or a location

5 Labels: B-per, I-per, B-loc, I-loc, O

Predict:
B-per  I-per  O  O  O  B-loc

Given:
Mr.  Smith  came  today  to  Toronto
Conditional Random Field: Example
Text Analytics (Named Entity Recognition)

Problem Statement:
Given a sentence, label each word with whether it is a person or a location.

5 Labels: B-per, I-per, B-loc, I-loc, O

Mr. Smith came today to Toronto
Conditional Random Field: Example
Text Analytics (Named Entity Recognition)

Problem Statement:
Given a sentence, label each word with whether it is a person or a location.

5 Labels: B-per, I-per, B-loc, I-loc, O
Problem Statement:
Given a sentence, label each word with whether it is a person or a location.

5 Labels: B-per, I-per, B-loc, I-loc, O

Two kinds of factors for each word $X_t$:

\[ \varphi_1^t (Y_t, Y_{t+1}) \quad \varphi_2^t (Y_t, X_1, \ldots, X_t) \]

Captures relationship between neighboring predictions.

Relationship between prediction and the neighbors of the word.

Mr. Smith came today to Toronto
Problem Statement:
Given a sentence, label each word with whether it is a person or a location.

5 Labels: B-per, I-per, B-loc, I-loc, O

Two kinds of factors for each word $X_t$: $\varphi^1_t (Y_t, Y_{t+1})$ $\varphi^2_t (Y_t, X_1, \ldots, X_t)$

Mr. Smith came today to Toronto.
Conditional Random Field: Example
Text Analytics (Named Entity Recognition)

Problem Statement:
Given a sentence, label each word with whether it is a person or a location

\[
P(Y_1, \ldots, Y_7 \mid X_1, \ldots, X_7) = \\
\sum Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7
\]

\[
\frac{\varphi_1(Y_1, Y_2) \times \cdots \times \varphi_6(Y_6, Y_7) \times \varphi_1(Y_1, X_1, X_2) \times \cdots \times \varphi_6(Y_6, X_6, X_7)}{\varphi_1(Y_1, Y_2) \times \cdots \times \varphi_6(Y_6, Y_7) \times \varphi_1(Y_1, X_1, X_2) \times \cdots \times \varphi_6(Y_6, X_6, X_7)}
\]

Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7

Mr. Smith came today to Toronto

X_1 X_2 X_3 X_4 X_5 X_6 X_7
Conditional Random Field: Example

Text Analytics (Named Entity Recognition)

Problem Statement:
Given a sentence, label each word with whether it is a person or a location.

\[ P(Y_1, \ldots, Y_7 \mid X_1, \ldots, X_7) = \]

\[ \sum Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7 \]

\[ \varphi^1_1(Y_1, Y_2) \times \ldots \times \varphi^7_6(Y_6, Y_7) \times \varphi^2_1(Y_1, X_1, X_2) \times \ldots \times \varphi^2_7(Y_7, X_6, X_7) \]
Learning from Data

We next look at equivalent representation of Markov Networks and Conditional Random Fields that are amenable to learning from data (e.g., log-linear form).
Another representation of a Markov Network is that of a log-linear model.

Here, we have a set of feature functions $f_1(D_1), \ldots, f_k(D_k)$ where each $D_i$ is a complete subgraph of $G$. An $f_i(D_i)$ can return any value, i.e., can be negative.

$$P(X_1, \ldots, X_n) = \frac{exp^{-w_1 \times f_1(D_1) \times \ldots \times exp^{-w_k \times f_k(D_k)}}}{Z(X_1, \ldots, X_n)}$$

$$Z(X_1, \ldots, X_n) = \sum \exp^{-\sum_{i=1}^{k}(w_i \times f_i(D_i))}$$

Any Markov Network whose factors are positive can be converted to a log-linear model.
Log-Linear Representation: Benefits

\[ P(X_1, \ldots, X_n) = \frac{\exp^{-w_1 \times f_1(D_1)} \times \cdots \times \exp^{-w_k \times f_k(D_k)}}{Z(X_1, \ldots, X_n)} \]

\[ Z(X_1, \ldots, X_n) = \sum \exp^{-\sum_{i=1}^{k}(w_i \times f_i(D_i))} \]

Log-linear models have few benefits over factors. They:

1. Make certain relationships more explicit

2. Offer a much more compact representation for many distributions, especially for variables with large domains (e.g., names in JSNice)

3. They are useful for learning where we are given the feature functions and the learning phase simply figures out the weights.
Feature Function: Example

For example, suppose that for the factor $\varphi_1$:

$\varphi_1 (A, B) = 100$ \quad \text{if} \quad A = B$

$\varphi_2 (A, B)$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

If A and B range over m and n values respectively, we have $m \times n$ possible values to store in order to keep this factor.
Feature Function: Example

For example, suppose that for the factor $\varphi_1$:

$$\varphi_1 (A, B) = 100 \quad \text{if} \quad A = B$$
$$1 \quad \text{otherwise}$$

If $A$ and $B$ range over $m$ and $n$ values respectively, we have $m \times n$ possible values to store in order to keep this factor.

Instead, we can use a single (indicator) feature function $f_1(A, B)$ where

$$f_1(A, B) = 1 \quad \text{if} \quad A = B$$
$$0 \quad \text{else}$$

and $w_1 = -4.61$

Indicator functions can be directly extracted from data during learning (later)
Graphical Models: So far

In the last segment we learned about log-linear models and indicator functions.

Log-linear models are important because they allow to capture factors more concisely, the indicator functions can be directly extracted from data, and the weights can be learned (to fit the optimization objective, discussed later).
Hands-on
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;
}
Structured Prediction for Programs

```javascript
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;
```

Unknown facts:
- t
- r
- i
- ...

Known facts:
- length
- ...

```
<table>
<thead>
<tr>
<th>i</th>
<th>t</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>step</td>
<td>0.5</td>
</tr>
<tr>
<td>j</td>
<td>step</td>
<td>0.4</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>i</th>
<th>r</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>len</td>
<td>0.6</td>
</tr>
<tr>
<td>i</td>
<td>length</td>
<td>0.3</td>
</tr>
</tbody>
</table>
```

```
argmax \mathbf{w}^T f(i, t, r, length)
```

```
<table>
<thead>
<tr>
<th>r</th>
<th>length</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>length</td>
<td>0.5</td>
</tr>
<tr>
<td>len</td>
<td>length</td>
<td>0.3</td>
</tr>
</tbody>
</table>
```
Structured Prediction for Programs

```javascript
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;
```

Unknown facts:

- t
- r
- i
- ...

Known facts:

- length
- ...

Structured Prediction

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

- i
- t
- w
  - i: step 0.5
  - j: step 0.4

- i: length
  - i: len 0.6
  - i: length 0.3

- r
- length
- w
  - length: length 0.5
  - len: length 0.3
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(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;
```
Tutorial Outline

• Motivation
  • Potential applications

• Statistical Language Models
  • N-gram, Recurrent Networks, PCFGs
  • Application: code completion

• Graphical Models
  • Markov Networks, Conditional Random Fields
  • Inference & Learning in Markov Networks
  • Application: predicting names and types

• Learning Features for Programs
  • Combining Program Synthesis + Machine Learning
Queries: MAP vs. Max Marginals

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

**Max 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. Max Marginals

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

VS.

Max 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>0.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.35</td>
</tr>
</tbody>
</table>
Queries: MAP vs. Max Marginals

**MAP Inference:**
\[(y_1, ..., y_n)^{\text{MAP}} = \underset{y_1, ..., y_n}{\text{argmax}} \ P(y_1, ..., y_n)\]

**VS.**

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

\[y_1^{\text{ML}} = \underset{y_1}{\text{argmax}} \ P(y_1) \quad \text{......} \quad y_n^{\text{ML}} = \underset{y_n}{\text{argmax}} \ P(y_n)\]

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>0</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.35</td>
</tr>
</tbody>
</table>

\[\underset{A, B}{\text{argmax}} \ P(A, B) \quad \underset{A}{\text{argmax}} \ P(A) \quad \underset{B}{\text{argmax}} \ P(B)\]
Queries: MAP vs. Max Marginals

**MAP Inference:**

\[(y_1, \ldots, y_n)^{MAP} = \underset{y_1, \ldots, y_n}{\text{argmax}} \ P(y_1, \ldots, y_n)\]

**Max Marginals:**

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

\[y_1^{ML} = \underset{y_1}{\text{argmax}} \ P(y_1) \quad \ldots \quad y_n^{ML} = \underset{y_n}{\text{argmax}} \ P(y_n)\]

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>0.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.35</td>
</tr>
</tbody>
</table>
Queries: MAP vs. Max Marginals

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

VS.

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

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>25.3%</td>
</tr>
<tr>
<td>Max Marginals</td>
<td>54.1%</td>
</tr>
<tr>
<td>MAP Inference</td>
<td>63.4%</td>
</tr>
</tbody>
</table>
Bad news: computing the argmax is in general NP-hard (Max-SAT)

Good news: many approximate methods exists

- Gibbs sampling
- Expectation Maximization
- Variational methods
- Elimination Algorithm
- Junction-Tree algorithm
- Loopy Belief Propagation
Bad news: computing the argmax is in general NP-hard (Max-SAT)

Good news: many approximate methods exist

- Gibbs sampling
- Expectation Maximization
- Variational methods
- Elimination Algorithm
- Junction-Tree algorithm
- Loopy Belief Propagation

Bad news: still too slow for learning $O(N \cdot M^N)$

$N$: number of neighbors
$M$: number of states
 MAP Inference

**Bad news:** computing the argmax is in general NP-hard (Max-SAT)

**Good news:** many approximate methods exists

- Gibbs sampling
- Expectation Maximization
- Variational methods
- Elimination Algorithm
- Junction-Tree algorithm

**Bad news:** still too slow for learning

\[ O(N \cdot M^N) \]

\( N \): number of neighbors
\( M \): number of states

We designed an approximate one to fit our needs, i.e., deal with many values
Structured SVM Training

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

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

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

**Optimization objective** (max-margin training):

\[
\forall j \ \forall y \ \Sigma w_i f_i(x^{(j)}, y^{(j)}) \geq \Sigma w_i f_i(x^{(j)}, y) + \Delta(y, y^{(j)})
\]
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: \( D = \{ x^j, y^j \}_{j=1..n} \) 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 \( Z(x) \)
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)

Learning Phase
- program analysis
- SSVM learning
- max-margin training
- alias, call analysis
- 7M feature functions for names
- 70K feature functions for types

Prediction Phase
- program analysis
- MAP inference
- transform

Time: milliseconds

30 nodes, 400 edges

Names: 63%
Types: 81%
(helps typechecking)

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

150MB

Conditional Random Field

\[ P(y | x) \]
Tutorial Outline

• Motivation
  • Potential applications

• Statistical Language Models
  • N-gram, Recurrent Networks, PCFGs
  • Application: code completion

• Graphical Models
  • Markov Networks, Conditional Random Fields
  • Inference & Learning in Markov Networks
  • Application: predicting names and types

• Learning Features for Programs
  • Combining Program Synthesis + Machine Learning
Models of Full Language

• So far:
  – N-gram and RNNs
  – Modelling specific elements (APIs)

• How can be build statistical language models of full programming language?
Program Representation

```javascript
elem.notify({
    position: 'top',
    autoHide: false,
    delay: 100
});
```

Sequences

```javascript
elem, ., notify, (, {},
    position, :, 'top', ',',
    autoHide, :, false, ',',
    delay, :, 100,
}, ), ;
```
Program Representation

elem.notify(
    {
        position: 'top',
        autoHide: false,
        delay: 100
    });

Sequences

elem, ., notify, (, {
    position, :, 'top', ,
    autoHide, :, false, ,
    delay, :, 100,
}, ), ;

Learned Model

?, ?, ?, (, {
    ?, :, ?, ,
    ?, :, ?, ,
    ?, :, ?, ,
    ?, :, ?,
}, ?, ;
Statistical Code Synthesis: Existing Approaches

**Syntactic**
- [Hindle et al., 2012]
- [Allamanis et al., 2015]

```
arg max \( P(x | \text{(features)}) \)
```

**Semantic**
- [Nguyen et al., 2013]
- [Allamanis et al., 2014]
- [Raychev et al., 2014]

```
arg max \( P(x | \text{label, conditioning context}) \)
```
- defer
- reject
- promise

Bad fit for programs

Hard-coded heuristics
Task & Language specific
elem.notify({
    position: 'top',
    autoHide: false,
    delay: 100
});

Trees
Probabilistic Context Free Grammars

- $N$ is a finite set of non-terminal symbols
- $\Sigma$ is a finite set of terminal symbols
- $R$ is a finite set of production rules $\alpha \rightarrow \beta_1 \ldots \beta_n$
- $S \in N$ is a start symbol
- $q: R \rightarrow \mathbb{R}^{(0,1)}$ is a conditional probability of choosing given production rule $\alpha \in N \quad \beta_i \in (N \cup \Sigma)$
Probabilistic Context Free Grammars

CallExpr → MemberExpr ObjectExpr        [ elem.notify({...}) ]
CallExpr → MemberExpr                  [ elem.notify() ]
CallExpr → MemberExpr Identifier       [ elem.notify(x) ]
CallExpr → Identifier                   [ notify() ]
Probabilistic Context Free Grammars

\[ \begin{align*}
CallExpr & \rightarrow \text{MemberExpr ObjectExpr} & \quad [\text{elem.notify(\{\ldots\})}] \\
CallExpr & \rightarrow \text{MemberExpr} & \quad [\text{elem.notify()}] \\
CallExpr & \rightarrow \text{MemberExpr Identifier} & \quad [\text{elem.notify(x)}]
\end{align*} \]

\[ q(\alpha \rightarrow \beta_1 \ldots \beta_n) \approx \frac{\#(\alpha \rightarrow \beta_1 \ldots \beta_n)}{\#(\alpha)} \]

\[ \forall \alpha \sum_{\alpha \rightarrow \beta_1 \ldots \beta_n \in R} q(\alpha \rightarrow \beta_1 \ldots \beta_n) = 1 \]

valid probability distribution
Probabilistic Context Free Grammars

CallExpr → MemberExpr ObjectExpr  [ elem.notify({...}) ]
CallExpr → MemberExpr               [ elem.notify() ]
CallExpr → MemberExpr Identifier    [ elem.notify(x) ]

↑
poor context for prediction

→
non-trivial to obtain good production rules

[Gvero & Kuncak, OOPSLA’15]
[Maddison & Tarlow, ICML’14]
[Liang et.al., ICML’10]

[Allamanis & Sutton, FSE’14]
Lessons

• Probabilistic Grammars
  + Easy to implement and cheap to train
  + Defined over generic AST representation of programs
    - Important to deal with sparsity via smoothing
    - Non-trivial to define good production rules
    - Production rules selected based on limited context (e.g., parent non-terminal symbol)
Lessons

• Probabilistic Grammars
  + Easy to implement and cheap to train
  + Defined over generic AST representation of programs
  - Important to deal with sparsity via smoothing
  - Non-trivial to define good production rules
  - Production rules selected based on limited context (e.g., parent non-terminal symbol)

PCFG are still poor at modelling full programming language but are a step towards the solution.
Our Goal

Existing Programs → Learning → Model

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

Probabilistic Model
Key Idea

So far:
Context defined manually for each model

\[ P( w_i \mid context ) \]
Key Idea

So far:
Context defined manually for each model

\[ P( w_i \mid \text{context} ) \]

N-grams
\[ \rightarrow \]
RNNs
\[ \rightarrow \]
CRFs
\[ \rightarrow \]
PCFGs

Goal:
Use program synthesis and machine learning techniques
to learn best context for given query \( w_i \)
PHOG: Concepts

Program synthesis learns a function that explains the data. The function returns a conditioning context for a given query.

Use function to build a probabilistic model. Generalizes PCFGs to allow conditioning on richer context.
Generalizing PCFG

Context Free Grammar

\[ \alpha \rightarrow \beta_1 \ldots \beta_n \]

\[ P \]

Property \( \rightarrow x \) 0.05
Property \( \rightarrow y \) 0.03
Property \( \rightarrow \) promise 0.001
PHOG: Generalizes PCFG

Context Free Grammar
\[ \alpha \rightarrow \beta_1 \ldots \beta_n \]

<table>
<thead>
<tr>
<th>Property</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property → x</td>
<td>0.05</td>
</tr>
<tr>
<td>Property → y</td>
<td>0.03</td>
</tr>
<tr>
<td>Property → promise</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Higher Order Grammar
\[ \alpha[y] \rightarrow \beta_1 \ldots \beta_n \]

<table>
<thead>
<tr>
<th>Property[reject, promise] → promise</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property[reject, promise] → notify</td>
<td>0.12</td>
</tr>
<tr>
<td>Property[reject, promise] → resolve</td>
<td>0.11</td>
</tr>
</tbody>
</table>
Conditioning on Richer Context

\[ \alpha[\gamma] \rightarrow \beta_1 \cdots \beta_n \]

What is the best conditioning context?
Conditioning on Richer Context

\[ \alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n \]

What is the best conditioning context?

- APIs
- Identifiers
- Control Structures
- Fields
- Constants
- ...

184
Conditioning on Richer Context

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

What is the best conditioning context?

- APIs
- Fields
- Identifiers
- Constants
- Control Structures
- ...

Source Code $\rightarrow$ ? $\rightarrow$ Conditioning Context
Higher Order Grammar

Production Rules R:
\[ \alpha[\gamma] \rightarrow \beta_1 \cdots \beta_n \]

Function:
\[ f: \rightarrow \gamma \]

Parametrize the grammar by a function used to dynamically obtain the context
Higher Order Grammar

Production Rules R:
\[ \alpha[\gamma] \rightarrow \beta_1 \cdots \beta_n \]

Function:
\[ f: \text{AST} \rightarrow \gamma \]

Parametrize the grammar by a function used to dynamically obtain the context.
Higher Order Grammar

Production Rules R:
\[ \alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n \]

Function:
\[ f: \text{AST} \rightarrow \gamma \]
Function Representation

In general:
Unrestricted programs (Turing complete)

Our Work:
TCond Language for navigating over trees and accumulating context

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

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

\[
\text{WriteOp} ::= \text{WriteValue, WriteType, WritePos}
\]
Expressing functions: TCond Language

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

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

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

Query

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

TCond

\( \gamma \)
Example

Query

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

TCond

Left
WriteValue

γ

{ }
{hide}
Example

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

TCond

Left
WriteValue
Up
WritePos

\gamma

{}  
{hide}  
{hide, 3}
elem.notify(
  ...
  ...
  {
    position: 'top',
    hide: false,
    ?
  }
);
Example

Query

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

TCond

```
Left
WriteValue
Up
WritePos
Up
DownFirst
DownLast
WriteValue
γ

{ hide, 3, notify }
```

{ Previous Property, Parameter Position, API name }
Learning PHOG

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

**TCond Language**

- **TCond**: $\varepsilon | \text{WriteOp TCond} | \text{MoveOp TCond}
- **MoveOp**: Up, Left, Right, ...
- **WriteOp**: WriteValue, WriteType, ...

**Existing Dataset**
Synthesis: Practically Intractable

- Millions ($\approx 10^8$) of input/output examples

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

computing cost $\rightarrow O(|D|)$

**Synthesis**: practically intractable

**Key Idea**: iterative synthesis on fraction of examples
Solution: Two Components

Program Generator

\[ f_{best} = \arg \min_{f \in TCond} \text{cost}(d, f) \]

Dataset Sampler

representative dataset sampler
picks dataset
\[ d \subseteq D \]
In a Loop

• Start with small number of samples \( d \subseteq D \)
• Iteratively generate programs and samples

Program Generator

\( d_1 \subseteq D \)

\( p_1 \)

\( d_2 \subseteq D \)

\( p_1, p_2 \)

Dataset Sampler

Dataset Sampler

Dataset Sampler

\( p_{\text{best}} \)
Representative Dataset Sampler

Idea: pick a small dataset $d$ for which a set of already generated programs $p_1 \ldots p_n$ behave like on full dataset $D$

$$d = \arg\min_{d \subseteq D} \max_{i \in 1 \ldots n} |\text{cost}(d, p_i) - \text{cost}(D, p_i)|$$
Representative Dataset Sampler

**Idea:** pick a small dataset $d$ for which a set of already generated programs $p_1 \ldots p_n$ behave like on full dataset $D$

$$d = \arg \min_{d \subseteq D} \max_{i \in 1 \ldots n} |\text{cost}(d, p_i) - \text{cost}(D, p_i)|$$

**Theorem:** The sampler shrinks search space of candidate programs
Results

Probabilistic Model of JavaScript Language

20k Learning 100k Training 50k Blind Set

GitHub
### Results: JavaScript APIs

<table>
<thead>
<tr>
<th>Conditioning program p</th>
<th>Accuracy</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>Program synthesis</td>
<td>46.3%</td>
</tr>
<tr>
<td>Program synthesis + dataset sampler</td>
<td>50.4%</td>
</tr>
</tbody>
</table>
## Results: JavaScript Structure

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFG</td>
<td>51.1%</td>
</tr>
<tr>
<td>N-gram</td>
<td>71.3%</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>44.2%</td>
</tr>
<tr>
<td>SVM</td>
<td>70.5%</td>
</tr>
<tr>
<td>Program synthesis</td>
<td>81.5%</td>
</tr>
</tbody>
</table>
## Results: JavaScript Values

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Accuracy</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>62%</td>
<td><code>contains = jQuery ...</code></td>
</tr>
<tr>
<td>Property</td>
<td>65%</td>
<td><code>start = list.length;</code></td>
</tr>
<tr>
<td>String</td>
<td>52%</td>
<td><code>'[ ' + attrs + ']'</code></td>
</tr>
<tr>
<td>Number</td>
<td>64%</td>
<td><code>canvas(xy[0], xy[1], ...)</code></td>
</tr>
<tr>
<td>RegExp</td>
<td>66%</td>
<td>`line.replace(/( </td>
</tr>
<tr>
<td>UnaryExpr</td>
<td>97%</td>
<td>`if (!events</td>
</tr>
<tr>
<td>BinaryExpr</td>
<td>74%</td>
<td><code>while (++index &lt; ...)</code></td>
</tr>
</tbody>
</table>
| LogicalExpr  | 92%      | `frame = frame | | ...`
Tutorial Outline

• Motivation
  • Potential applications

• Statistical Language Models
  • N-gram, Recurrent Networks, PCFGs
  • Application: code completion

• Graphical Models
  • Markov Networks, Conditional Random Fields
  • Inference & Learning in Markov Networks
  • Application: predicting names and types

• Learning Features for Programs
  • Combining Program Synthesis + Machine Learning
# Machine Learning for Programs

<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></td>
<td></td>
</tr>
<tr>
<td>Representation</td>
<td>typestate analysis</td>
<td>scope analysis</td>
<td>control-flow analysis</td>
<td>alias analysis</td>
<td></td>
</tr>
<tr>
<td>Analyze Program (PL)</td>
<td>Neural Networks</td>
<td>SVM</td>
<td>Structured SVM</td>
<td>Greedy MAP inference</td>
<td></td>
</tr>
<tr>
<td>Train Model (ML)</td>
<td>N-gram language model</td>
<td>argmax $P(y</td>
<td>x)$</td>
<td>y $\in \Omega$</td>
<td>More information and tutorials at:</td>
</tr>
</tbody>
</table>
Statistical Programming Tools

Write new code [PLDI’14]:
Code Completion

Port code [ONWARD’14]:
Programming Language Translation

Understand code/security [POPL’15]:
JavaScript Deobfuscation
Type Prediction

Debug code:
Statistical Bug Detection

Camera camera = Camera.open();
camera.SetDisplayOrientation(90);

... for x in range(a):
  print a[x]

All of these benefit from the probabilistic model for code.

www.jsnice.org
# Machine Learning for Programs

<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>Graphical Models (CRFs)</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>Greedy MAP inference</td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>argmax $P(y \mid x)$</td>
<td>$y \in \Omega$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Materials

Dagstuhl Seminar on Big Code Analytics, Nov 2015
http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=15472
  – ML, NLP, PL, SE
  – Link to materials, people in the general area

http://learningfrombigcode.org
  – data sets, tools, challenges

http://nice2predict.org
  – fully open source, online demo, build your own tool
Probabilistic Learning from Big Code

Probabilistically likely solutions to problems impossible to solve otherwise

Publications

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

Publicly Available Tools

http://JSNice.org
statistical de-obfuscation

http://Nice2Predict.org
structured prediction framework

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