Learning Programs from Noisy Data

Veselin Raychev
Pavol Bielik
Martin Vechev
Andreas Krause

ETH Zurich
Why learn programs from examples?

Input/output examples

often easier to provide examples than specification (e.g. in FlashFill)
Why learn programs from examples?

Input/output examples

often easier to provide examples than specification (e.g. in FlashFill)

learn a function

\[ p \] such that

\[ p(\bullet) = \text{red} \]
\[ p(\circ) = \text{brown} \]

\[ \ldots \ ? \]
Why learn programs from examples?

Input/output examples

often easier to provide examples than specification (e.g. in FlashFill)

learn a function

\[ p \text{ such that } \]

\[ p(\bullet) = \square \]

\[ p(\bullet) = \blacksquare \]

the user may make a mistake in the examples

\[ \ldots \quad ? \]
Why learn programs from examples?

Input/output examples

often easier to provide examples than specification (e.g. in FlashFill)

learn a function

p such that

p(●) = □
p(○) = □

the user may make a mistake in the examples

Actual goal: produce p that the user really wanted and tried to specify
Why learn programs from examples?

Input/output examples

often easier to provide examples than specification (e.g. in FlashFill)

learn a function

p such that

p(○) = □
p(●) = □

the user may make a mistake in the examples

Actual goal: produce p that the user really wanted and tried to specify

Key problem of synthesis: overfits, not robust to noise
Learning Programs from Data: Defining Dimensions

Handling errors in the dataset

Number of Examples

Learned program complexity
Learning Programs from Data: Defining Dimensions

- Handling errors in the dataset: no
- Number of Examples: tens
- Learned program complexity: interesting programs

Program synthesis (PL)
Learning Programs from Data: Defining Dimensions

<table>
<thead>
<tr>
<th>Program synthesis (PL)</th>
<th>Deep learning (ML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handling errors in the dataset</td>
<td>yes</td>
</tr>
<tr>
<td>Number of Examples</td>
<td>millions</td>
</tr>
<tr>
<td>Learned program complexity</td>
<td>interesting programs</td>
</tr>
</tbody>
</table>
Learning Programs from Data: Defining Dimensions

<table>
<thead>
<tr>
<th>Handling errors in the dataset</th>
<th>Program synthesis (PL)</th>
<th>This paper bridges a gap</th>
<th>Deep learning (ML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>tens</td>
<td>millions</td>
<td>millions</td>
</tr>
<tr>
<td>tens</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>interesting programs</td>
<td>interesting programs</td>
<td>simple, but unexplainable functions</td>
<td></td>
</tr>
</tbody>
</table>
Learning Programs from Data: Defining Dimensions

<table>
<thead>
<tr>
<th>Handling errors in the dataset</th>
<th>Program synthesis (PL)</th>
<th>This paper bridges a gap</th>
<th>Deep learning (ML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Program synthesis (PL)</td>
<td>This paper bridges a gap</td>
<td>Deep learning (ML)</td>
</tr>
<tr>
<td>Tens</td>
<td>no</td>
<td>yes</td>
<td>simple, but unexplainable functions</td>
</tr>
<tr>
<td>Interesting programs</td>
<td>interesting programs</td>
<td>interesting programs</td>
<td>interesting programs</td>
</tr>
<tr>
<td>Expands capabilities of existing synthesizers</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This paper bridges a gap in that it expands the capabilities of existing synthesizers.
Learning Programs from Data: Defining Dimensions

<table>
<thead>
<tr>
<th>Handling errors in the dataset</th>
<th>Program synthesis (PL)</th>
<th>This paper bridges a gap</th>
<th>Deep learning (ML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>tens</td>
<td>tens</td>
<td>millions</td>
<td>millions</td>
</tr>
<tr>
<td>interesting programs</td>
<td>interesting programs</td>
<td>simple, but unexplainable functions</td>
<td></td>
</tr>
</tbody>
</table>

- This paper bridges a gap for programming tasks.
- Expands capabilities of existing synthesizers.
- New state-of-the-art precision for programming tasks.
Learning Programs from Data: Defining Dimensions

Program synthesis (PL)
- Handling errors in the dataset
- Number of Examples
- Learned program complexity

This paper bridges a gap

Deep learning (ML)
- Number of Examples
- Learned program complexity

Bridges gap between ML and PL
Advances both areas

expands capabilities of existing synthesizers
new state-of-the-art precision for programming tasks
In this paper

● A general framework that handles
  ○ errors in training dataset
  ○ learns statistical models on data
  ○ handles synthesis with millions of examples

● Instantiated with two synthesizers
  ○ generalize existing works
Contributions

Handling noise

Input/output examples

- blue
- red
- green
- black
- orange
- blue
- purple

incorrect examples

New probabilistic models

1. synthesize $p$

2. use probabilistic model parametrized with $p$

Handling large datasets

Representative dataset sampler

Program generator
Contributions

Handling noise

Input/output examples

1. synthesize $p$

New probabilistic models

2. use probabilistic model parametrized with $p$

Handling large datasets

Representative dataset sampler

Program generator

incorrect examples
Synthesis with noise: usage model

Input/output examples

- Blue
- Red
- Green
- Brown
- Orange
- Purple
- Black
Synthesis with noise: usage model

Input/output examples

Domain
Specific
Language
Synthesis with noise: usage model

Input/output examples

synthesizer

Domain
Specific
Language
Synthesis with noise: usage model

Input/output examples

synthesizer

Domain Specific Language

\[ p \text{ such that } p(\bullet) = \square \]
\[ p(\bullet) = \blacksquare \]
\[ \ldots \]
Synthesis with noise: usage model

Input/output examples

synthesizer

Such that

Domain Specific Language

Incorrect example (e.g. a typo)

\( p \)

\( p(\bullet) = \)
Synthesis with noise: usage model

Input/output examples

incorrect example (e.g. a typo)

synthesizer

Domain Specific Language

\[ p \text{ such that } p(\bigcirc) = \square \]
\[ p(\bigcirc) = \blacksquare \]
\[ \ldots \]
\[ p(\bigcirc) \neq \heartsuit \]
Synthesis with noise: usage model

Input/output examples

✔️  ✔️  ✔️
✔️  ✔️  ✔️
❌  ✔️  ✔️

incorrect example (e.g. a typo)

synthesizer

p such that
p(●) =
p(●) =

new kind of feedback from synthesizer
Synthesis with noise: usage model

Input/output examples

- ✔/✔ ✔/✔ ✔/✔ ✔/✔
- ✗/✗ ✗/✗ ✗/✗ ✗/✗

Incorrect example (e.g. a typo)

- Tell user to remove suspicious example, or
- Ask for more examples

New kind of feedback from synthesizer

\[
p \text{ such that } p(\bigcirc) = \square \]
\[
p(\bigcirc) = \blacksquare \]
\[
\ldots \]
\[
p(\bigcirc) \neq \Box \]
Handling noise: problem statement

D: Input/output examples

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) \]

Too long program, hardcodes the input/outputs. Synthesis must penalize such answers

Our problem formulation:

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

total solution cost

number of instructions
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

- total solution cost
- number of instructions
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

- \( \text{err}_1 = \text{if } p(\text{●}) = \text{●} \text{ then } 0 \text{ else } 1 \)
- \( \text{err}_2 = \text{if } p(\text{POINTS}) = \text{POINTS} \text{ then } 0 \text{ else } 1 \)
- \( \text{err}_3 = \text{if } p(\text{●}) = \text{●} \text{ then } 0 \text{ else } 1 \)

\[ \text{errors} = \text{err}_1 + \text{err}_2 + \text{err}_3 \]

- \( p \in P_r \text{ (with } r \text{ instructions)} \)

\[ \Psi \]

Total solution cost

Number of instructions

Formula given to SMT solver
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

- \( \text{errors} = \text{err}_1 + \text{err}_2 + \text{err}_3 \)
- \( \text{err}_1 = \begin{cases} 0 & \text{if } p(\circ) = \text{red} \\ 1 & \text{else} \end{cases} \)
- \( \text{err}_2 = \begin{cases} 0 & \text{if } p(\bullet) = \text{brown} \\ 1 & \text{else} \end{cases} \)
- \( \text{err}_3 = \begin{cases} 0 & \text{if } p(\diamond) = \text{blue} \\ 1 & \text{else} \end{cases} \)

Ask a number of SMT queries in increasing value of solution cost

e.g. for \( \lambda = 0.6 \)

costs are

<table>
<thead>
<tr>
<th>( r )</th>
<th>( 0 )</th>
<th>( 1 )</th>
<th>( 2 )</th>
<th>( 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6</td>
<td>1.6</td>
<td>2.6</td>
<td>3.6</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>2.2</td>
<td>3.2</td>
<td>4.2</td>
</tr>
<tr>
<td>3</td>
<td>1.8</td>
<td>2.8</td>
<td>3.8</td>
<td>4.8</td>
</tr>
</tbody>
</table>

\( p \in P_r \) (with \( r \) instructions)

\( \Psi \)

formula given to SMT solver

total solution cost

number of instructions
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

\[
\begin{align*}
\text{err}_1 &= \text{if } p(\bigcirc) = \text{red} \text{ then } 0 \text{ else } 1 \\
\text{err}_2 &= \text{if } p(\bigotimes) = \text{orange} \text{ then } 0 \text{ else } 1 \\
\text{err}_3 &= \text{if } p(\bullet) = \text{blue} \text{ then } 0 \text{ else } 1 \\
\text{errors} &= \text{err}_1 + \text{err}_2 + \text{err}_3 \quad p \in P_r \quad \text{(with } r \text{ instructions)}
\end{align*}
\]

Ask a number of SMT queries in increasing value of solution cost

e.g. for \( \lambda = 0.6 \) costs are

<table>
<thead>
<tr>
<th>( r )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UNSAT</td>
<td>1.6</td>
<td>2.6</td>
<td>3.6</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>2.2</td>
<td>3.2</td>
<td>4.2</td>
</tr>
<tr>
<td>3</td>
<td>1.8</td>
<td>2.8</td>
<td>3.8</td>
<td>4.8</td>
</tr>
</tbody>
</table>

\( \Psi \) formula given to SMT solver
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

\( \text{errors} = \text{err}_1 + \text{err}_2 + \text{err}_3 \)

\( \text{err}_1 = \text{if } p(\circ) = \bullet \text{ then } 0 \text{ else } 1 \)
\( \text{err}_2 = \text{if } p(\circ) = \square \text{ then } 0 \text{ else } 1 \)
\( \text{err}_3 = \text{if } p(\circ) = \square \text{ then } 0 \text{ else } 1 \)

Ask a number of SMT queries in increasing value of solution cost

e.g. for \( \lambda = 0.6 \)

<table>
<thead>
<tr>
<th>( r )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UNSAT</strong></td>
<td>1.6</td>
<td>2.6</td>
<td>3.6</td>
</tr>
<tr>
<td><strong>UNSAT</strong></td>
<td>2.2</td>
<td>3.2</td>
<td>4.2</td>
</tr>
<tr>
<td>1</td>
<td>1.8</td>
<td>2.8</td>
<td>3.8</td>
</tr>
</tbody>
</table>

formal given to SMT solver
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

- \[ \text{err}_1 = \text{if } p(\bigcirc) = \text{red} \text{ then 0 else 1} \]
- \[ \text{err}_2 = \text{if } p(\bullet) = \text{brown} \text{ then 0 else 1} \]
- \[ \text{err}_3 = \text{if } p(\bullet) = \text{blue} \text{ then 0 else 1} \]

\[ \text{errors} = \text{err}_1 + \text{err}_2 + \text{err}_3 \quad p \in P_r \quad (\text{with } r \text{ instructions}) \]

Ask a number of SMT queries in increasing value of solution cost

E.g. for \( \lambda = 0.6 \)

Costs are

<table>
<thead>
<tr>
<th>( r )</th>
<th>cost</th>
<th>number of errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UNSAT</td>
<td>2.6 3.6</td>
</tr>
<tr>
<td>2</td>
<td>UNSAT</td>
<td>2.2 3.2 4.2</td>
</tr>
<tr>
<td>3</td>
<td>1.8</td>
<td>2.8 3.8 4.8</td>
</tr>
</tbody>
</table>
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

\[ \text{errors} = \text{err}_1 + \text{err}_2 + \text{err}_3 \]

\[ p \in P_r \text{ (with } r \text{ instructions)} \]

- \text{err}_1 = \text{if } p(\text{blue}) = 1 \text{ then } 0 \text{ else } 1
- \text{err}_2 = \text{if } p(\text{red}) = 1 \text{ then } 0 \text{ else } 1
- \text{err}_3 = \text{if } p(\text{yellow}) = 1 \text{ then } 0 \text{ else } 1

Ask a number of SMT queries in increasing value of solution cost

e.g. for \( \lambda = 0.6 \) costs are

<table>
<thead>
<tr>
<th>( r )</th>
<th>cost</th>
<th>number of errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UNSAT</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>UNSAT</td>
<td>2.6</td>
</tr>
<tr>
<td>2</td>
<td>UNSAT</td>
<td>2.2</td>
</tr>
<tr>
<td>3</td>
<td>UNSAT</td>
<td>2.8</td>
</tr>
<tr>
<td>1</td>
<td>UNSAT</td>
<td>3.6</td>
</tr>
<tr>
<td>2</td>
<td>UNSAT</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>UNSAT</td>
<td>3.8</td>
</tr>
<tr>
<td>1</td>
<td>UNSAT</td>
<td>4.2</td>
</tr>
<tr>
<td>2</td>
<td>UNSAT</td>
<td>4.8</td>
</tr>
</tbody>
</table>

ψ

Noisy synthesis using SMT

encoding

total solution cost

number of instructions

formula given to SMT solver
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

\[
\begin{align*}
\text{err}_1 &= \text{if } p(\bullet) = \blacksquare \text{ then 0 else 1} \\
\text{err}_2 &= \text{if } p(\odot) = \blacksquare \text{ then 0 else 1} \\
\text{err}_3 &= \text{if } p(\circ) = \blacksquare \text{ then 0 else 1}
\end{align*}
\]

\[ \text{errors} = \text{err}_1 + \text{err}_2 + \text{err}_3 \]

\[ p \in P_r \quad (\text{with } r \text{ instructions}) \]

Ask a number of SMT queries in increasing value of solution cost

e.g. for \( \lambda = 0.6 \) costs are

<table>
<thead>
<tr>
<th>( r )</th>
<th>\text{cost}</th>
<th>\begin{align*} \text{number of errors} \end{align*}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\text{UNSAT}</td>
<td>2.6, 3.6</td>
</tr>
<tr>
<td>2</td>
<td>\text{UNSAT}</td>
<td>\text{SAT}</td>
</tr>
<tr>
<td>3</td>
<td>\text{UNSAT}</td>
<td>2.8, 3.8, 4.8</td>
</tr>
</tbody>
</table>
Noisy synthesis using SMT

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]

\begin{align*}
\text{err}_1 &= \text{if } p(\bigcirc) = \blacksquare \text{ then 0 else 1} \\
\text{err}_2 &= \text{if } p(\bullet) = \blacksquare \text{ then 0 else 1} \\
\text{err}_3 &= \text{if } p(\bigcirc) = \bigcirc \text{ then 0 else 1} \\
\text{errors} &= \text{err}_1 + \text{err}_2 + \text{err}_3 \\
p \in P_r \text{ (with } r \text{ instructions)}
\end{align*}

Ask a number of SMT queries in increasing value of solution cost

- e.g. for \( \lambda = 0.6 \)
- costs are

<table>
<thead>
<tr>
<th>( r )</th>
<th>\text{cost}</th>
<th>\text{number of errors}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UNSAT</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>UNSAT</td>
<td>SAT</td>
</tr>
<tr>
<td>3</td>
<td>UNSAT</td>
<td>UNSAT</td>
</tr>
</tbody>
</table>

Best program is with two instructions and makes one error
Noisy synthesizer: example

Take an actual synthesizer and show that we can make it handle noise
Implementation: BitSyn

For BitStream programs, using Z3
similar to Jha et al.[ICSE’10] and Gulwani et al.[PLDI’11]

Example program:

```plaintext
function check_if_power_of_2(int32 x){
    var o = add(x, 1)
    return bitwise_and(x, o) ← synthesized, short loop-free programs
}
```
Implementation: BitSyn

For BitStream programs, using Z3

similar to Jha et al.[ICSE’10] and Gulwani et al.[PLDI’11]

Example program:

```plaintext
function check_if_power_of_2(int32 x){
    var o = add(x, 1)
    return bitwise_and(x, o)
}
```

**Question:** how well does our synthesizer discover noise? (in programs from prior work)
Implementation: BitSyn

For BitStream programs, using Z3 similar to Jha et al. [ICSE’10] and Gulwani et al. [PLDI’11]

Example program:

```plaintext
function check_if_power_of_2(int32 x) {
    var o = add(x, 1)
    return bitwise_and(x, o)
}
```

Question: how well does our synthesizer discover noise? (in programs from prior work)
Implementation: BitSyn

Example program:

```c
function check_if_power_of_2(int32 x) {
    var o = add(x, 1)
    return bitwise_and(x, o)
}
```

Question: how well does our synthesizer discover noise? (in programs from prior work)
So far... handling noise

- Problem statement and regularization
- Synthesis procedure using SMT
- Presented one synthesizer

Handling noise enables us to solve new classes of problems beyond normal synthesis
Contributions

Handling noise

Input/output examples

incorrect examples

New probabilistic models

1. synthesize $p$

2. use probabilistic model parametrized with $p$

Handling large datasets

Representative dataset sampler

Program generator
Contributions

Handling noise

Input/output examples

- Blue
- Red
- Green
- Black
- Orange
- Purple
- Incorrect examples

New probabilistic models

1. Synthesize $p$

2. Use probabilistic model parametrized with $p$

Handling large datasets

Representative dataset sampler

Program generator
Large number of examples:

\[ p_{\text{best}} = \arg \min_{p \in P} \text{cost}(D, p) \]
Fundamental problem

Large number of examples:

\[ p_{\text{best}} = \arg \min_{p \in P} \text{cost}(D, p) \]
Fundamental problem

Large number of examples:

\[ p_{\text{best}} = \arg\min_{p \in \mathcal{P}} \text{cost}(D, p) \]

Computing \( \text{cost}(D, p) \):

\[ O(|D|) \]

 Millions of input/output examples
Fundamental problem

Large number of examples:

\[
p_{\text{best}} = \arg \min_{p \in P} \text{cost}(D, p)
\]

Computing \( \text{cost}(D, p) \) is \( O(|D|) \).

Synthesis: practically intractable

Millions of input/output examples
Fundamental problem

Large number of examples:

\[ p_{\text{best}} = \arg \min_{p \in \mathcal{P}} \text{cost}(D, p) \]

- Computing \( \text{cost}(D, p) \) is \( O(|D|) \)

**Synthesis:** practically intractable

**Key idea:** iterative synthesis on fraction of examples
Our solution: two components

\[
p_{\text{best}} = \arg \min_{p \in P} \text{cost}(d, p)
\]

given dataset d, finds best program
Our solution: two components

Given dataset $d$, finds best program $p_{\text{best}} = \arg \min_{p \in P} \text{cost}(d, p)$

- Program generator
- Dataset sampler

Picks dataset $d \subseteq D$

Synthesizer for small number of examples

We introduce representative dataset sampler
Generalize a user providing input/output examples
In a loop

Program generator

Representative dataset sampler
Start with a small random sample $d \subseteq D$

Iteratively generate programs and samples.
In a loop

Start with a small random sample $d \subseteq D$

Iteratively generate programs and samples.
In a loop

Start with a small random sample $d \subseteq D$

Iteratively generate programs and samples.
In a loop

Start with a small random sample $d \subseteq D$

Iteratively generate programs and samples.
In a loop

Start with a small random sample \( d \subseteq D \)

Iteratively generate programs and samples.
In a loop

Start with a small random sample $d \subseteq D$

Iteratively generate programs and samples.

Program generator $d$ 

Representative dataset sampler $p_{\text{best}}$

Program generator $p_1$

Representative dataset sampler $p_1, p_2$

Program generator $p_1, p_2$
In a loop

Start with a small random sample $d \subseteq D$

Iteratively generate programs and samples.

Algorithm generalizes synthesis by examples techniques
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 the full dataset

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

\[
d = \arg \min_{d \subseteq D} \max_{i \in 1..n} | \text{cost}(d, p_i) - \text{cost}(D, p_i) |
\]
Representative dataset sampler

**Idea:** pick a small dataset $d$ for which a set of already generated programs $p_1,...,p_n$ behave like on the full dataset

$$d = \arg \min_{d \subseteq D} \max_{i \in 1..n} | \text{cost}(d, p_i) - \text{cost}(D, p_i) |$$
**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 the full dataset $D$.

$$d = \arg \min_{d \subseteq D} \max_{i \in 1..n} | \text{cost}(d, p_i) - \text{cost}(D, p_i) |$$
**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 the full dataset $D$.

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

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

Theorem: this sampler shrinks the candidate program search space

In evaluation: significant speedup of synthesis
So far... handling large datasets

- Iterative combination of synthesis and sampling
- New way to perform approximate empirical risk minimization
- Guarantees (in the paper)
Contributions

Handling noise

Input/output examples

1. synthesize
2. use probabilistic model parametrized with \( p \)

Handling large datasets

Representative dataset sampler

Program generator
Contributions

Handling noise

Input/output examples

Incorrect examples

New probabilistic models

1. synthesize p

2. use probabilistic model parametrized with p

Handling large datasets

Representative dataset sampler

Program generator
Statistical programming tools

A new breed of tools:

Learn from large existing codebases (e.g. Big Code) to make predictions about programs
Statistical programming tools

A new breed of tools:

Learn from large existing codebases (e.g. Big Code) to make predictions about programs

1. Train machine learning model
Statistical programming tools

A new breed of tools:

Learn from large existing codebases (e.g. Big Code) to make predictions about programs

1. Train machine learning model
2. Make predictions with model
Statistical programming tools

A new breed of tools:

Learn from large existing codebases (e.g. Big Code) to make predictions about programs

1. Train machine learning model
2. Make predictions with model
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Hindle et al.[ICSE’12]

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Hindle et al.[ICSE’12]

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Hindle et al.[ICSE’12]
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Learn a mapping

Model will predict slice when it sees it after “\(+\) name .”

This model comes from NLP
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Hindle et al.[ICSE’12]

Learn a mapping

Model will predict slice when it sees it after “+ name .”

This model comes from NLP

Raychev et al.[PLDI’14]
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Learn a mapping

Model will predict `slice` when it sees it after "+ name ."

This model comes from NLP
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Learn a mapping

Model will predict slice when it sees it after “+ name .”

This model comes from NLP
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Learn a mapping

Model will predict slice when it sees it after “+ name.”

This model comes from NLP

Raychev et al.[PLDI’14]

Hindle et al.[ICSE’12]
Existing machine learning models

Essentially remember mapping from context in training data to prediction (with probabilities)

Hindle et al. [ICSE’12]

```
function d3_vendorSymbol(object, name) {
  if (name in object) return name;
  name = name.charAt(0).toUpperCase() + name.slice(1);
  for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++)
    var prefixName = d3_vendorPrefixes[i] + name;
```

Learn a mapping

```
+ name .
```

Model will predict slice when it sees it after “+ name .”

This model comes from NLP

Raychev et al. [PLDI’14]

```
function d3_vendorSymbol(object, name) {
  if (name in object) return name;
  name = name.charAt(0).toUpperCase() + name.slice(1);
  for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++)
    var prefixName = d3_vendorPrefixes[i] + name;
```

Learn a mapping

```
charAt
```

Model will predict slice when it sees it after “charAt”

Relies on static analysis
Problem of existing systems

Precision. They rarely predict the next statement

Hindle et al.[ICSE’12]

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++)
        var nprefixName = d3_vendorPrefixes[i] + name;
}
```

Learn a mapping

```
+ name .
```

Model will predict `slice` when it sees it after “+ name .”

This model comes from NLP

Raychev et al.[PLDI’14]

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++)
        var nprefixName = d3_vendorPrefixes[i] + name;
}
```

Learn a mapping

```
charAt
```

Model will predict `slice` when it sees it after “charAt”

Relies on static analysis
Problem of existing systems

Precision. They rarely predict the next statement.

Hindle et al. [ICSE'12]

Very low precision

Raychev et al. [PLDI'14]

Learn a mapping

Model will predict slice when it sees it after "+ name ."

This model comes from NLP

Learn a mapping

Model will predict slice when it sees it after "charAt"

Relies on static analysis
Problem of existing systems

Precision. They rarely predict the next statement

Hindle et al. [ICSE’12]

Very low precision

Raychev et al. [PLDI’14]

Low precision for JavaScript

Learn a mapping

+ name . → slice

Model will predict slice when it sees it after “+ name .”

This model comes from NLP

Learn a mapping

charAt → slice

Model will predict slice when it sees it after “charAt”

Relies on static analysis
Problem of existing systems

Precision. They rarely predict the next statement

Core problem:
Existing machine learning models are limited and not expressive enough

Very low precision

Low precision for JavaScript
Key idea: second-order learning

Learn a program that parametrizes a probabilistic model that makes predictions.
Key idea: second-order learning

Learn a program that parametrizes a probabilistic model that makes predictions.

1. Synthesize program describing a model
Key idea: second-order learning

Learn a program that parametrizes a probabilistic model that makes predictions.

1. Synthesize program describing a model

2. Train model
   i.e. learn the mapping
Key idea: second-order learning

Learn a program that parametrizes a probabilistic model that makes predictions.

1. Synthesize program describing a model

2. Train model

i.e. learn the mapping

3. Make predictions with this model
Key idea: second-order learning

Learn a program that parametrizes a probabilistic model that makes predictions.

prior models are described by simple hard-coded programs

Our approach: learn a better program

2. Train model  
i.e. learn the mapping

3. Make predictions with this model

element.className = this.options.className  
element.style.width = this.options.width  
element.style.
Training and evaluation

Training example:

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n;)
        var prefixName = d3_vendorPrefixes[i] + name;
}
```
Training and evaluation

Training example:

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```

Compute context with program p

input / slice / output
Training and evaluation

Training example:

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; 
        var prefixName = d3_vendorPrefixes[i] + name;
```
Training and evaluation

Training example:

```
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name;
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```

Evaluation example:

```
(/
cc, word) => {
acc + ' ' + word[0].toUpperCase() + word.
```
Training and evaluation

Training example:

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name;
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```

Evaluation example:

```javascript
(() => {  
    cc, word => {  
        acc + ' ' + word[0].toUpperCase()) + word;
    }
```

Learn a mapping

- `toUpperCase`
- `slice`
Training and evaluation

Training example:

```javascript
function d3_vendorSymbol(object, name) {
    // Compute context with program p
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```

Evaluation example:

```javascript
cc, word) => {
    acc + ' ' + word[0].toUpperCase() + word.slice(1)
}
```
Training and evaluation

Training example:

```javascript
function d3_vendorSymbol(object, name) {
    if (name in object) return name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    for (var i = 0, n = d3_vendorPrefixes.length; i < n; i++) {
        var prefixName = d3_vendorPrefixes[i] + name;
    }
}
```

Evaluation example:

```javascript
/
cc, word) => {
    acc + ' ' + word[0].toUpperCase() + word.slice(1) + ' ' + word;
}
```
Observation

Synthesis of probabilistic model can be done with the same optimization problem as before!

Our problem formulation:

\[ p_{\text{best}} = \arg \min_{p \in P} \text{errors}(D, p) + \lambda r(p) \]
Observation

Synthesis of probabilistic model can be done with the same optimization problem as before!

Our problem formulation:

\[
p_{\text{best}} = \arg\min_{p \in P} \text{errors}(D, p) + \lambda r(p)
\]
So far...

Handling noise
Synthesizing a model
Representative dataset sampler

Techniques are generally applicable to program synthesis

Next, application for “Big Code” called DeepSyn
DeepSyn: Training

Trained on 100’000 JavaScript files from GitHub
DeepSyn: Training

Trained on 100’000 JavaScript files from GitHub

Program synthesizer

Representative dataset sampler

Program generator

Train model on full data and best program p
DeepSyn: Evaluation

50’000 evaluation files (not used in training or synthesis)

API completion task
DeepSyn: Evaluation

50’000 evaluation files (not used in training or synthesis)

API completion task

<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 per object, Raychev et al.[PLDI’14]</td>
<td>30.4%</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td></td>
</tr>
<tr>
<td>Program synthesis with noise</td>
<td>46.3%</td>
</tr>
<tr>
<td>Program synthesis with noise + dataset sampler</td>
<td>50.4%</td>
</tr>
</tbody>
</table>
DeepSyn: Evaluation

50'000 evaluation files (not used in training or synthesis)

API completion task

<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 per object, Raychev et al.[PLDI’14]</td>
<td>30.4%</td>
</tr>
<tr>
<td><strong>Program synthesis with noise</strong></td>
<td>46.3%</td>
</tr>
<tr>
<td><strong>Program synthesis with noise + dataset sampler</strong></td>
<td>50.4%</td>
</tr>
</tbody>
</table>

We can explain best program. It looks at API preceding completion position and at tokens prior to these APIs.
Handling noise

Input/output examples

- Blue square, red square
- Green square, black square
- Brown square, orange square
- Black square, blue square

Incorrect examples

Extending synthesizers to handle noise

Second-order learning

1. Synthesize $p$

2. Use probabilistic model parametrized with $p$

Handling large datasets

Representative dataset sampler

Program generator

Scalability
Synthesis of probabilistic models

Handling noise

Input/output examples

- Correct examples
- Incorrect examples

Extending synthesizers to handle noise

Second-order learning

1. synthesize $p$

2. use probabilistic model parametrized with $p$

Bridges gap between ML and PL
Advances both areas

Handling large datasets

- Representative dataset sampler
- Program generator

Scalability
What did we synthesize?

Left PrevActor WriteAction WriteValue PrevActor WriteAction PrevLeaf WriteValue PrevLeaf WriteValue

\[ p_{\approx \text{best}} = \]
\[
\begin{align*}
\text{Left} & \quad \text{PrevActor} \quad \text{WriteAction} \quad \text{WriteValue} \\
\text{PrevActor} & \quad \text{WriteAction} \quad \text{PrevLeaf} \\
\text{WriteValue} & \quad \text{PrevLeaf} \quad \text{WriteValue}
\end{align*}
\]

(a) TCOND program

if (show) {
  var cws = document.querySelectorAll(...);
  for (var i = 0, slide; slide = cws[i]; i++) {
    slide.classList.add("hidden");
  }
  var iap = document.querySelectorAll(...);
  for (var i = 0, slide; slide = iap[i]; i++) {
    slide.classList.add("hidden");
  }
  var dart = document.
  ...
}

(b) JavaScript code snippet

(c) Execution of \( p_{\approx \text{best}} \) on the AST representation of the code snippet from (b)