Reliable and Interpretable Artificial Intelligence

Martin Vechev
ETH Zurich

Fall 2017
Today

Motivation for material

What will we learn

Course organization
"My view is throw it all away and start again"

"I don't think it's how the brain works," he said. "We clearly don't need all the labeled data."

"The future depends on some graduate student who is deeply suspicious of everything I have said."
This course: A glimpse into latest AI research...

Part I: Program Synthesis/Induction: a new frontier in AI where the computer learns an interpretable program from user-provided examples. We will cover the latest theory (CEGIS, SMT, lower bounds) and systems (e.g., neural synthesis).

Part II: Robustness of Deep Learning: we will study the latest methods for finding adversarial examples in neural nets, methods to automatically prove the network is robust, as well as techniques for learning interpretable robustness specs of the network.

Part III: Probabilistic Programming: Probabilistic programming is an emerging direction whose goal is democratize the construction of probabilistic models. Here we will study inference, semantics, and synthesis.
Traditional Synthesis (bird’s eye view)

Automatically synthesize a program that is correct-by-construction from a (higher-level) specification.
Program Synthesis

• Traditional synthesis derives programs from full formal specifications. This is a difficult, slow, and manual process.

• Modern Program Synthesis: new, emerging area. Really, a way to generate interpretable models…many new flavors in the last 10 years, but roughly 3 main kinds:
  
  – Constraint-based synthesis: CEGIS
  – Synthesis from examples: Excel’s FlashFill
  – Statistical synthesis with machine learning: SLANG, Deep3

Lets see an example of each kind
SKETCH: isolate rightmost 0

```c
bit[W] isolate0 (bit[W] x) { // W: word size
    bit[W] ret=0;
    for (int i = 0; i < W; i++)
        if (!x[i]) { ret[i] = 1; break; }
    return ret;
}
```

Naïve implementation (spec)
bit[W] isolate0 (bit[W] x) { // W: word size
    bit[W] ret=0;
    for (int i = 0; i < W; i++)
        if (!x[i]) { ret[i] = 1; break; }
    return ret;
}

bit[W] isolate0Sketched(bit[W] x) implements isolate0 {
    return ~(x + ??) & (x + ??);
}
**SKETCH: isolate rightmost 0**

```c
bit[W] isolate0 (bit[W] x) { // W: word size
    bit[W] ret=0;
    for (int i = 0; i < W; i++)
        if (!x[i]) { ret[i] = 1; break; }
    return ret;
}
```

```c
bit[W] isolate0Fast (bit[W] x) implements isolate0 {
    return ~x & (x+1);
}
```

```c
bit[W] isolate0Sketched (bit[W] x) implements isolate0 {
    return ~(x + ??) & (x + ??);
}
```

Solved with Counter-example guided inductive synthesis (CEGIS)
**Discrete Learning from examples**

**Technical Analysis:** Predict price direction using current prices

**Patterns:** Special forms that signal whether to buy or sell

**Goal:** Synthesize a program (a model) detecting a pattern

\[
P_0 = \text{LLV}(\text{Close}, W);
BP_0 = \text{LLVBars}(\text{Close}, W);
P_1 = \text{HHV}(\text{Close}, BP_0);
P_2 = \text{LLV}(\text{Close}, BP_1);
BP_2 = \text{HHVBars}(\text{Close}, BP_0);
P_3 = \text{HHV}(\text{Close}, BP_2);
P_4 = \text{LLV}(\text{Close}, BP_3);
BP_4 = \text{LLVBars}(\text{Close}, BP_3);
P_5 = \text{HHV}(\text{Close}, BP_4);
P_6 = \text{LLV}(\text{Close}, BP_5);
\]

**Filter:**

\[
P_0 < P_1 \text{ AND } P_2 < P_1 \text{ AND } P_1 < P_3 \text{ AND } P_5 < P_3 \text{ AND } P_4 < P_5 \text{ AND } P_6 < P_5;
\]

**Idea:** Learn the exact pattern from charts

**Challenges:**

Which questions to ask? How to reduce their number?

**Solution:**

Algorithm that will ask at most \(|F| \cdot \text{OPT}(F_Y)\) membership queries where \(F\) is the set of possible features, \(\text{OPT}(F_Y)\) is the minimum worst case number of membership queries for \(F_Y\).
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AUDIO_SOURCE_MICROPHONE);
r.setVideoSource(MediaRecorder.VIDEO_SOURCE_CAMERA);
r.setOutputFormat(MediaRecorder.OUTPUT_FORMAT_MPEG_4);
r.setProfile(MediaRecorder.PROFILE_HIGH_QUALITY_VIDEO);
r.set rtcStartThreshold(5000);
Deep Learning for Synthesis

Statistical language models
Deep learning (i.e. RNN-40)

Symbolic Analysis

More here: http://plml.ethz.ch/
Synthesis vs. Machine Learning

Synthesis has elaborate techniques to learn **interpretable functions** over **discrete spaces** (e.g., over a DSL), creative ways to express intent, and often with **few examples**!

Standard machine learning typically learns **non-interpretable functions** over **continuous spaces** (e.g., weights, feature functions, deep learning)

The two areas are **slowly merging** (e.g., combining discrete with continuous search, dealing with noise systematically) with many exciting opportunities for interesting connections!

A sample paper bridging the two areas:

“Program Synthesis for Character Level Language Modeling”, Bielik, Raychev, V., ICLR’2017
https://openreview.net/forum?id=ry_sjFqqx&noteId=ry_sjFqqx
Program Synthesis (Part I): Topics

- **Programming-by-example (PBE):** representing sets of programs succinctly (e.g., via version spaces)

- **Constraint-based synthesis:** counterexample-guided inductive synthesis (CEGIS) to reduce size of labeled data set.

- **Combinations of machine learning and synthesis:** deep learning + symbolic analysis, neural program synthesis (NTMs, NPIs)
Robustness of Neural Networks (Part II)

Neural networks are *not* robust to input perturbations (e.g., image rotation / change of lighting)

Misclassifications in neural networks deployed in self-driving cars [1]
In each picture one of the 3 networks makes a mistake...

---

Attacks on Machine Learning...

Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms

By Evan Ackerman
Posted 4 Aug 2017 | 18:00 GMT

Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake; they're like optical illusions for machines. In this post we'll show how adversarial examples work across different mediums, and will discuss why securing systems against them can be difficult.
Robustness of Neural Networks (Part II)

- **Basics**: fully connected nets, convolutional nets, activation functions

- **Finding adversarial examples**: finding examples where small perturbations to the input cause misclassification. Connections with GAN neural networks and CEGIS

- **Proving robustness w.r.t adversarial examples**: leveraging SMT solvers and abstraction techniques to prove the network is free of robustness violations

- **Spec inference**: inferring a spec which determines when the network is free of robustness violations
Example: Analysis of Neural Networks

Concrete

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Neuron values</th>
<th>Neuron values</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Concrete Image]</td>
<td>(0.01, 0.03,..)</td>
<td>(0.94, 0.02,..)</td>
<td>(0.01, 0.03,..)</td>
</tr>
<tr>
<td>![Concrete Image]</td>
<td>(0.23, 0.12,..)</td>
<td>(0.23, 0.12,..)</td>
<td>(0.94, 0.02,..)</td>
</tr>
<tr>
<td>![Concrete Image]</td>
<td>(0.12, 0.54,..)</td>
<td>(0.12, 0.54,..)</td>
<td>(0.23, 0.12,..)</td>
</tr>
<tr>
<td>![Concrete Image]</td>
<td>(0.12, 0.54,..)</td>
<td>(0.12, 0.54,..)</td>
<td>(0.12, 0.54,..)</td>
</tr>
</tbody>
</table>

Concrete layer transformer

Abstract

Abstract layer transformer

Abstract numerical element

Abstract analysis allows one to reason about an infinite set of possible images at once!
Probabilistic Programming

ML: Algorithms & Applications

 STATS: Inference & Theory

Probabilistic Programming

PL: Compilers, Semantics, Transformations

[credit: Frank Wood]
Intuition

\[ p(x | y) \]

Inference

Programming

Probabilistic Programming

Statistics

[credit: Frank Wood]
Examples of Models

\[
p(x | y) = \frac{p(y | x) \ p(x)}{p(y)}
\]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>program source code</td>
<td>program output</td>
</tr>
<tr>
<td>scene description</td>
<td>image</td>
</tr>
<tr>
<td>policy and world</td>
<td>rewards</td>
</tr>
<tr>
<td>cognitive process</td>
<td>behavior</td>
</tr>
<tr>
<td>simulation</td>
<td>constraint</td>
</tr>
</tbody>
</table>
Perception / Inverse Graphics

Captcha Solving

\[ y \]
- Input Image
  - DRMLW
  - WRQBC
  - NWYE

\[ x \]
- Intermediate Iterations
  - DBA
  - EBE
  - DBB
  - RBB
- Final Inferred Image
  - EBA

Scene Description

\[ y \]
- Observed Image
  - Inferred model re-rendered with novel poses
  - Inferred model re-rendered with novel lighting

\[ x \]
- Inferred (reconstruction)
  - Faces in novel poses
  - Faces in novel lighting

\[ y \]
- Captcha description
  - NWYE
  - NWYE

\[ x \]
- Scene description

Mansinghka, Kulkarni, Perov, and Tenenbaum.

Kulkarni, Kohli, Tenenbaum, Mansinghka

How About Sum of Three Variables?

\[ Z = X_1 + X_2 + X_3 \]

X1, X2, X3 i.i.d.
Exact Final Distribution (PSI)

\[ p(x) = -[x \neq 0] \cdot [x \leq 0] \cdot e^x \cdot x \cdot \sqrt{32} + [x \leq 0] \cdot e^x \cdot x \cdot \sqrt{16} + [-x \leq 0] \cdot e^x + \frac{3}{32} \cdot [-x \leq 0] \cdot e^x + \frac{3}{32} \cdot [x \neq 0] \cdot [x \leq 0] \cdot e^x + [-x \leq 0] \cdot [x \neq 0] \cdot x \cdot \sqrt{16} \cdot e^x + [-x \leq 0] \cdot x^2 \cdot \sqrt{16} \cdot e^x + [x \leq 0] \cdot x \cdot \sqrt{8} \cdot e^x + [x \leq 0] \cdot e^x \cdot x^2 \cdot \sqrt{16} \]
Probabilistic Programming

http://probabilistic-programming.org/

**Idea:** express probabilistic model as a program, in a **probabilistic programming language** as a program, let inference be done by underlying system

**Promise:** simplifies construction and querying of probabilistic models, leads to better interpretability and reliability of the machine learning model

**Applications:** reliable machine learning (bias), robotics, networks, security

---

**Probabilistic Program**

```python
def main()
    p := Uniform(0,1);
    r := [1,1,0,1,0];

    for i in [0..r.length] {
        observe(Bernoulli(p) == r[i]);
    }
    return p;
```

---

**Expresses a Probabilistic Model [PDF]**
Probabilistic Programming (Part III): Topics

Theory & Applications

• Theory: probabilistic semantics, open problems
• Applications: probabilistic computer networks, bias in machine learning, probabilistic security enforcement

Reasoning approaches:

• **Exact symbolic inference**: discrete, continuous and mixed distributions, higher order exact inference
• **Approximate inference**: MCMC sampling, combinations with exact inference, combinations with deep learning
Learning Objectives

- Understand some of the latest advancements in interpretable A.I.

- Be able to build systems based on the concepts in class

- Understand the open research problems in the area
Should I take this course?

Take this course if you agree with the objectives

Do not take this course if you:

– do not agree with some of the objectives
– do not want to work during the semester
  (e.g., homework, reading papers)
Organization of the Course

Teaching Assistants

Dr. Petar Tsankov  Dr. Dana Drachsler-Cohen  Dimitar Dimitrov  Matthew Mirman

Course web site: http://www.srl.inf.ethz.ch/ria.php

All information posted there: lectures notes, exercises, etc.
More lectures from

Timon Gehr
Main author of PSI
(http://psisolver.org/)

Dr. Veselin Raychev
(CTO, DeepCode.AI)
honorable mention winner

Prof. Dr. Swarat Chaudhuri
Professor at Rice, USA
Synthesis/ML
Course Organization

Prerequisites
- Course is self-contained
- Basic math background helps
- 4 credits: 2h lecture + 1h exercises.

Grading
- 100% final exam (make sure you do the homework)
Exercises

- Come with questions/solutions
  - The TA will go over the homework

- Exercises are not graded, but if you do them, you will have an advantage on the final exam
Q&A (from lecture)

Q: How much background is assumed in the course?
A: Course generally does not require special background. Basic familiarity with Python and Linear Algebra helps.

Q: Are generative models covered in the lecture?
A: Yes.

Q: Will course help me in deciding when to use neural nets?
A: Not directly. However, we will look at several applications and limitations, so this may be helpful.

Q: Will exercises contain coding or pen and paper only?
A: Mostly pen and paper, also reading papers occasionally. However, there will be some exercises with coding.

Q: How does course relate to the Deep Learning course and Probabilistic A.I.?
A: All 3 courses are complementary and there is not much overlap between them.

Q: What are examples of areas we will not cover?
A: We will not cover optimizations of gradient descent, sparsifying networks, or legal aspects of machine learning. We will however mention references to these topics if someone would like to pursue them.