Reliable and Interpretable Artificial Intelligence

Martin Vechev
ETH Zurich

Fall 2017

Today

Motivation for material

What will we learn

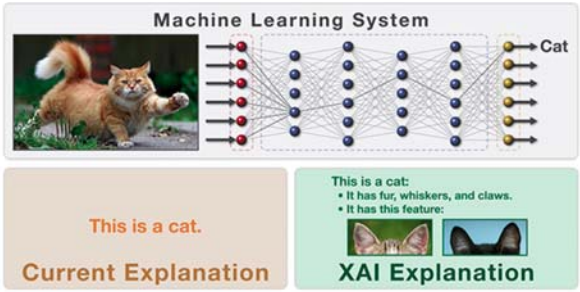
Course organization

DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY ABOUT US / OUR RESEARCH /

Defense Advanced Research Projects Agency > Program Information

Explainable Artificial Intelligence (XAI)

Mr. David Gunning



How AI detectives are cracking open the black box of deep learning

By Paul Voosen | Jul. 6, 2017 , 2:00 PM

Artificial intelligence pioneer says we need to start over (Sept 15, 2017)

Intelligent Machines

The Dark Secret at the Heart of AI

No one really knows how the most advanced what they do. That could be a problem.

How well can we get along with machines that are unpredictable and inscrutable?

AI programs exhibit racial and gender biases, research reveals

Machine learning algorithms are picking up deeply ingrained race and gender prejudices concealed within the patterns of language use, scientists say

European Union regulations on algorithmic decision-making and a "right to explanation"

Bryce Goodman, Seth Flaxman

(Submitted on 28 Jun 2016 (v1), last revised 31 Aug 2016 (this version, v3))

We summarize the potential impact that the European Union's new General Data Protection Regulation will have on the routine use of machine learning algorit effect as law across the EU in 2018, it will restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors affect" users. The law will also effectively create a "right to explanation," whereby a user can ask for an explanation of an algorithmic decision that was made a that while this law will pose large challenges for industry, it highlights opportunities for computer scientists to take the lead in designing algorithms and evaluati avoid discrimination and enable explanation.

Comments: presented at 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016), New York, NY
Subjects: Machine Learning (stat.ML); Computers and Society (cs.CY); Learning (cs.LG)
Cite as: arXiv:1606.08813 [stat.ML]

"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

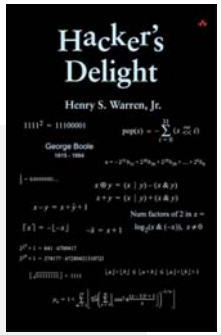
- Traditional synthesis derives programs from full formal specifications. This is a difficult, slow, and manual process

- 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

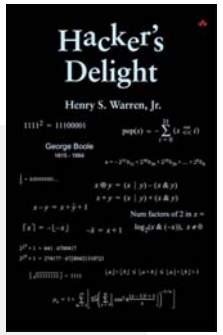
SKETCH: isolate rightmost 0




```
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)

SKETCH: isolate rightmost 0



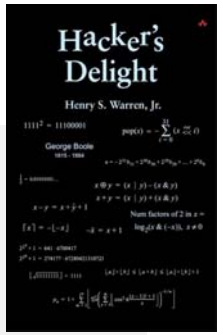

```
bit[W] isolate0 (bit[W] x) { // W: word size
  bit[W] ret=0;
  for (int i = 0; i < W; i++)
    if (!x[i]) { ret[i] = 1; break; }
  return ret;
}
```

Naïve implementation (spec)

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

Sketch – provides intent, restricts hypothesis space

SKETCH: isolate rightmost 0




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

Sketch – provides intent, restricts hypothesis space

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

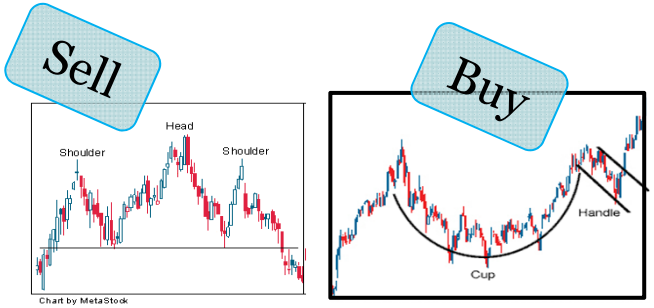
Discovered hypothesis that obeys spec

Solved with Counter-example guided inductive synthesis (CEGIS)

Discrete Learning from examples

Technical Analysis: Predict price direction using current prices

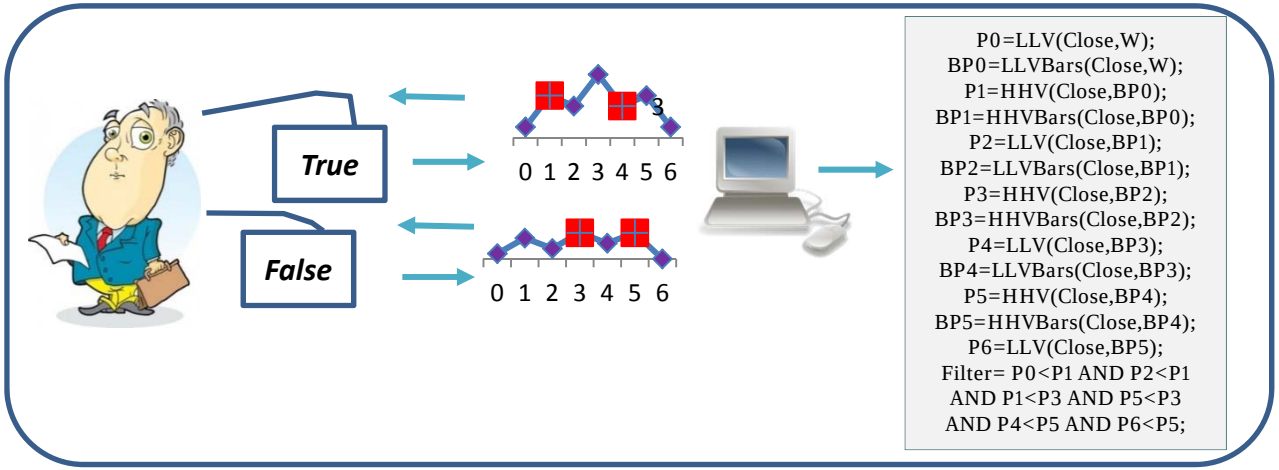
Patterns: Special forms that signal whether to buy or sell



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

```
P0=LLV(Close,W);
BP0=LLVBars(Close,W);
P1=HHV(Close,BP0);
BP1=HHVBars(Close,BP0);
P2=LLV(Close,BP1);
BP2=LLVBars(Close,BP1);
P3=HHV(Close,BP2);
BP3=HHVBars(Close,BP2);
P4=LLV(Close,BP3);
BP4=LLVBars(Close,BP3);
P5=HHV(Close,BP4);
BP5=HHVBars(Close,BP4);
P6=LLV(Close,BP5);
Filter= P0<P1 AND P2<P1 AND P1<P3 AND
P5<P3 AND P4<P5 AND P6<P5;
```

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



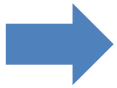

```
P0=LLV(Close,W);
BP0=LLVBars(Close,W);
P1=HHV(Close,BP0);
BP1=HHVBars(Close,BP0);
P2=LLV(Close,BP1);
BP2=LLVBars(Close,BP1);
P3=HHV(Close,BP2);
BP3=HHVBars(Close,BP2);
P4=LLV(Close,BP3);
BP4=LLVBars(Close,BP3);
P5=HHV(Close,BP4);
BP5=HHVBars(Close,BP4);
P6=LLV(Close,BP5);
Filter= P0<P1 AND P2<P1
AND P1<P3 AND P5<P3
AND P4<P5 AND P6<P5;
```

Challenges:
Which questions to ask?
How to reduce their number?

Solution:
Algorithm that will ask at most $|F| \cdot OPT(F_V)$ membership queries where F is the set of possible features, $OPT(F_V)$ is the minimum worst case number of membership queries for F_V

Deep Learning for Synthesis

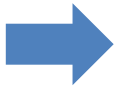
```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
```



```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);
r.setOutFormat(MediaRecorder.MPEG4);
```

Deep Learning for Synthesis

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



```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);
r.setOutFormat(MediaRecorder.MPEG4);
```

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:

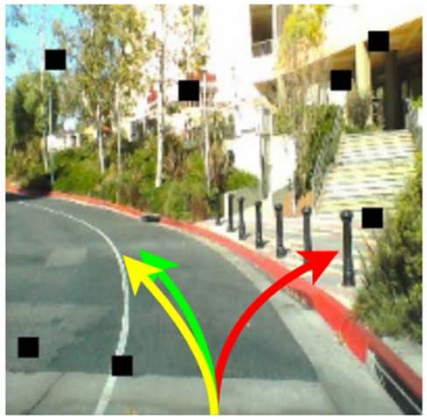
"Program Synthesis for Character Level Language Modeling", Bielik, Raychev, V., ICLR'2017
https://openreview.net/forum?id=ry_sjFqgx&noteId=ry_sjFqgx

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)

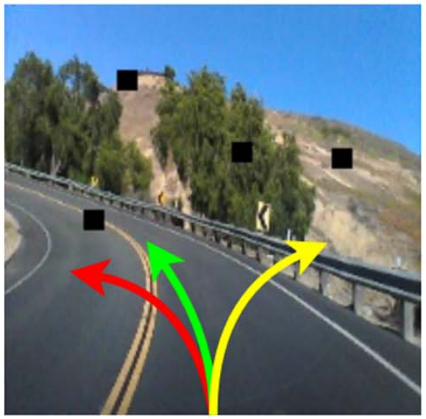
Robustness of Neural Networks (Part II)

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



DRV_C1: right

DRV_C2: right



DRV_C3: right

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

[1] Pei et. al., DeepXplore: Automated Whitebox Testing of Deep Learning Systems

Attacks on Machine Learning…

Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms

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

Attacking Machine Learning with Adversarial Examples

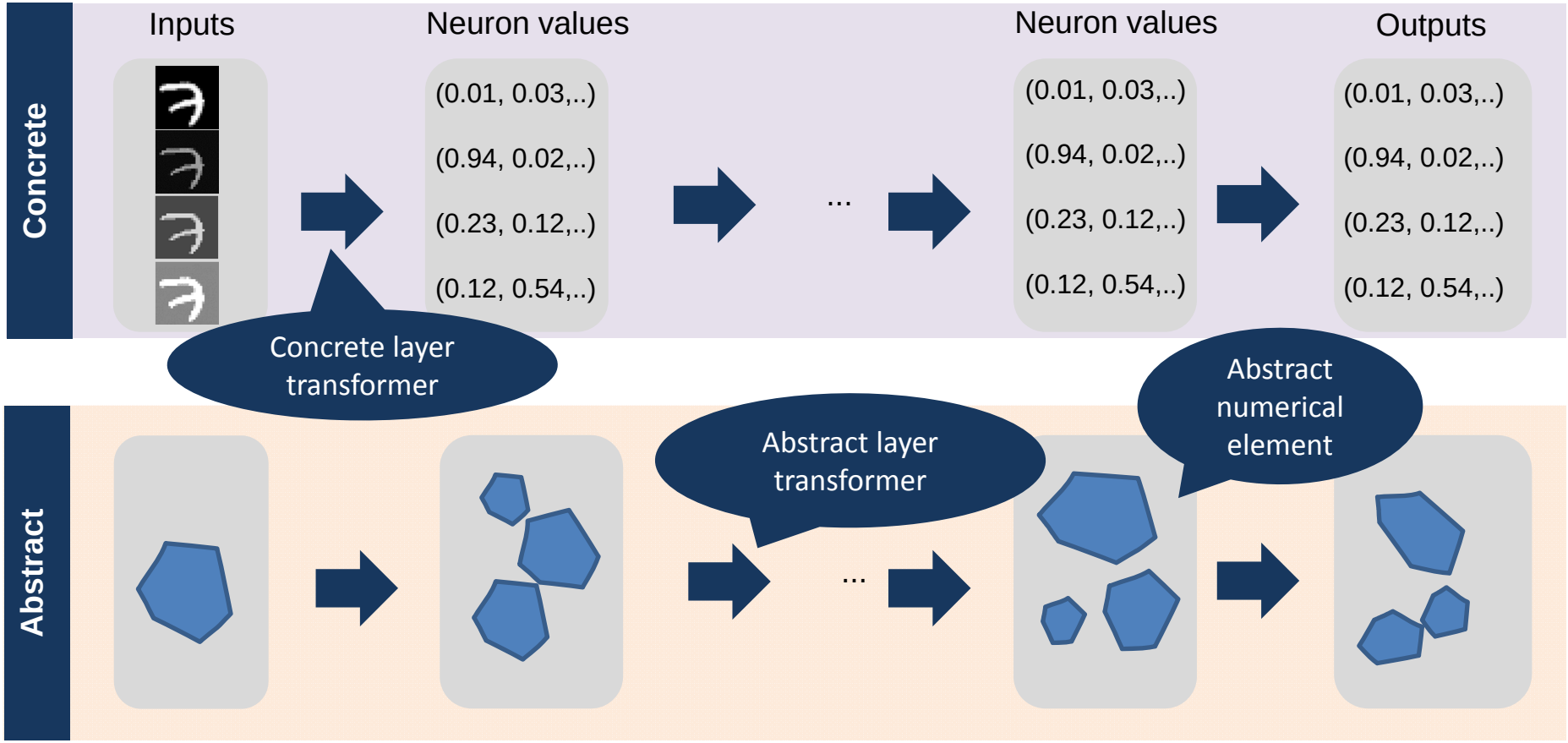
FEBRUARY 24, 2017

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)

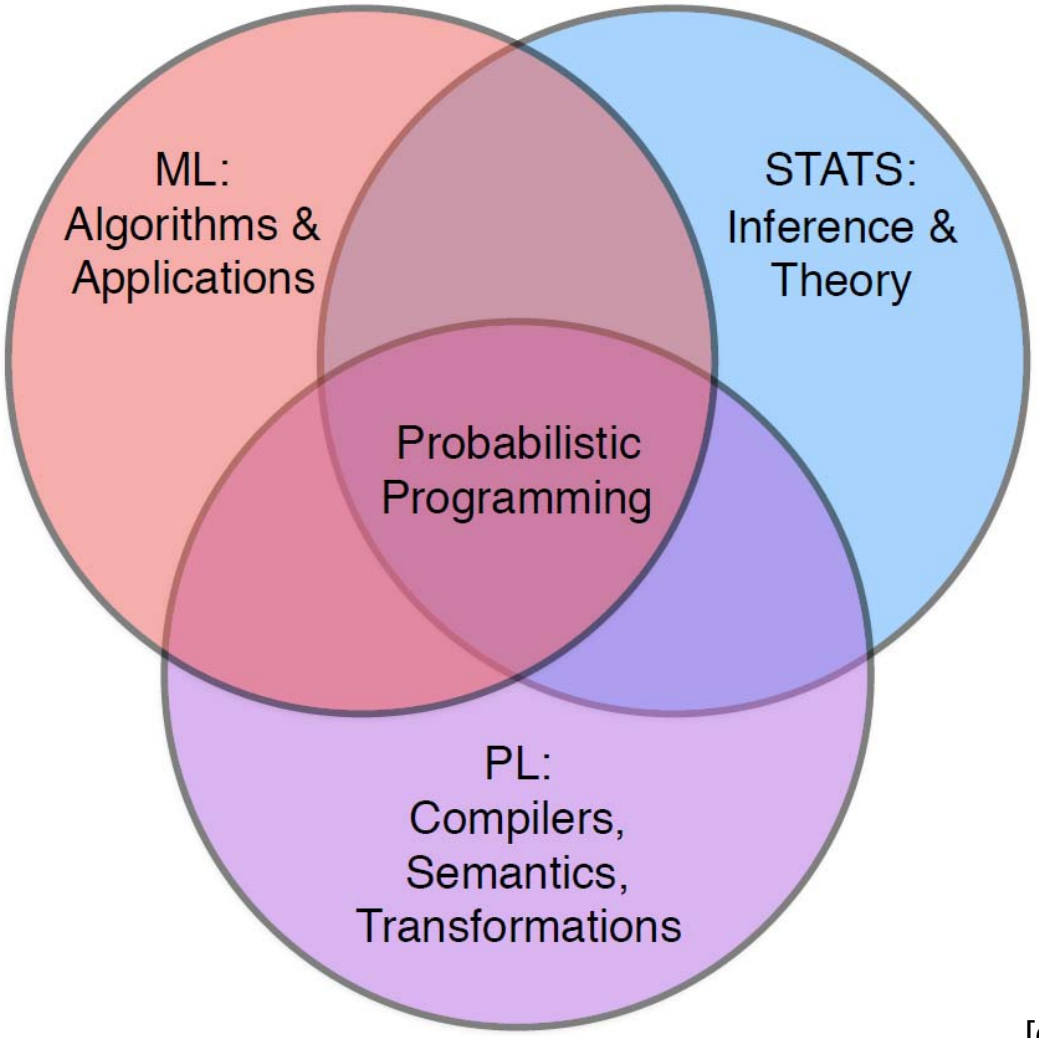
- Basics: fully connected nets, convolutional nets, activation functions
- Finding adversarial examples: finding examples where small perturbations to the input cause misclassification. Connections with GAN neural networks and CEGIS
- Proving robustness w.r.t adversarial examples: leveraging SMT solvers 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



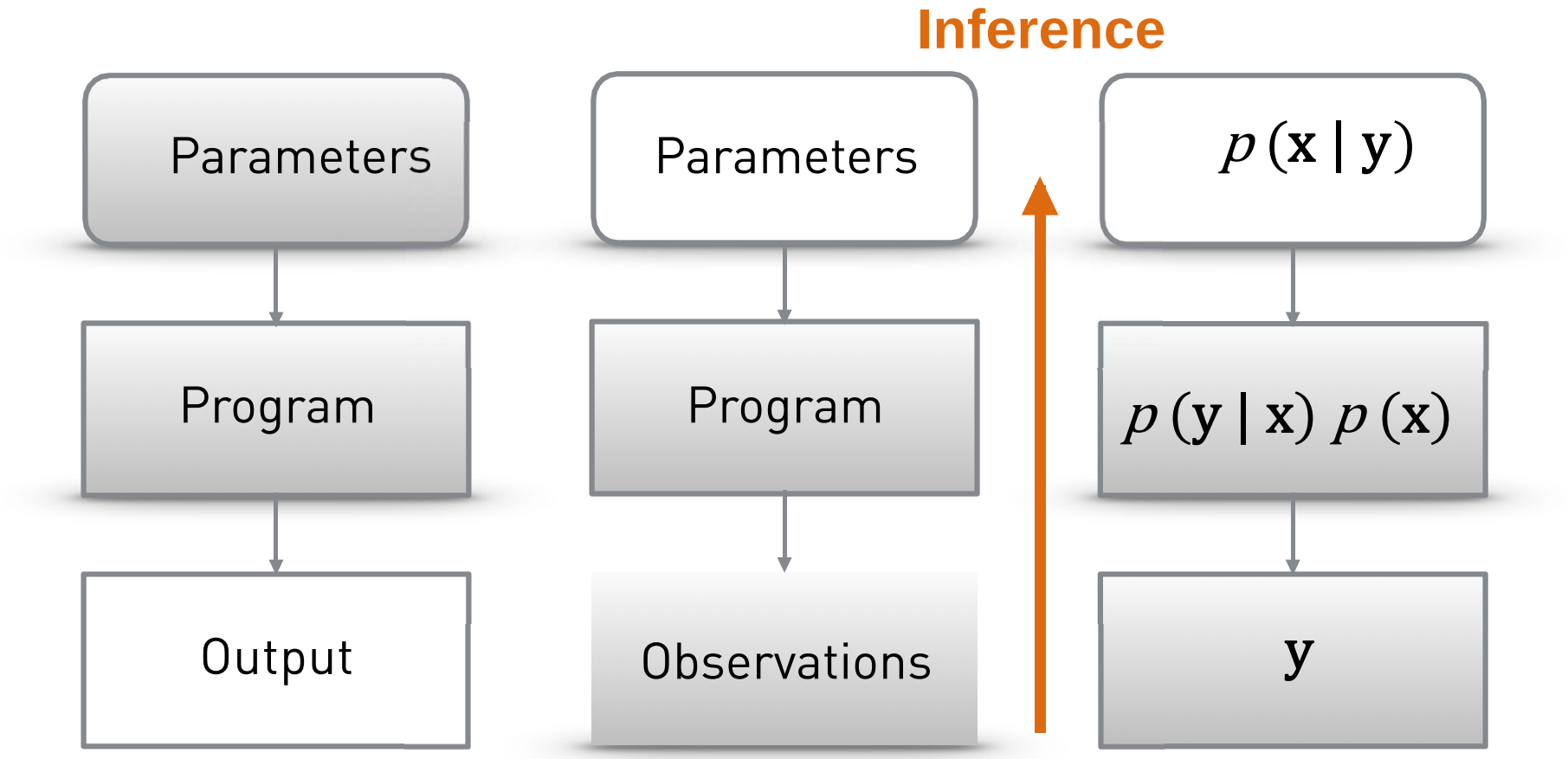
Abstract analysis allows one to reason about an **infinite set** of possible images **at once**!

Probabilistic Programming



[credit: Frank Wood]

Intuition



[credit: Frank Wood]

Examples of Models

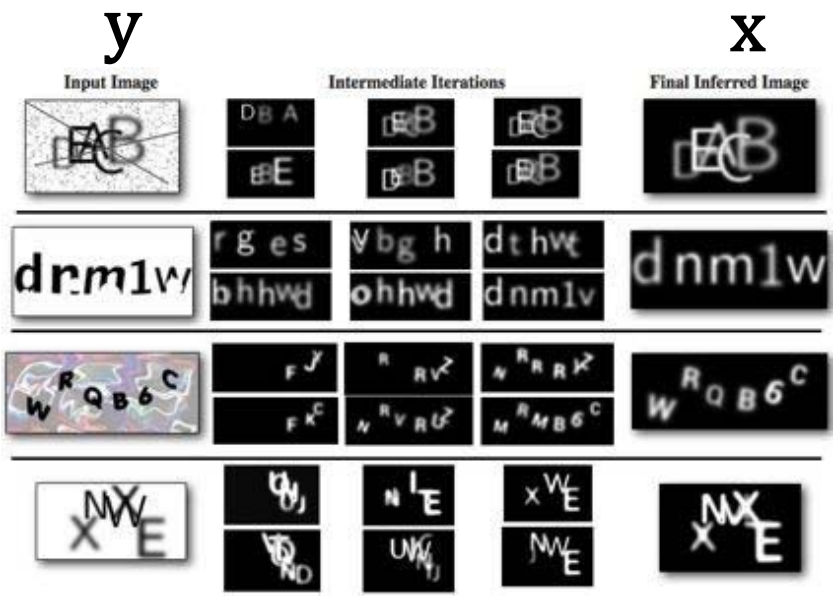
$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})\ p(\mathbf{x})}{p(\mathbf{y})}$$

$\mathbf{x}$	$\mathbf{y}$
program source code	program output
scene description	image
policy and world	rewards
cognitive process	behavior
simulation	constraint

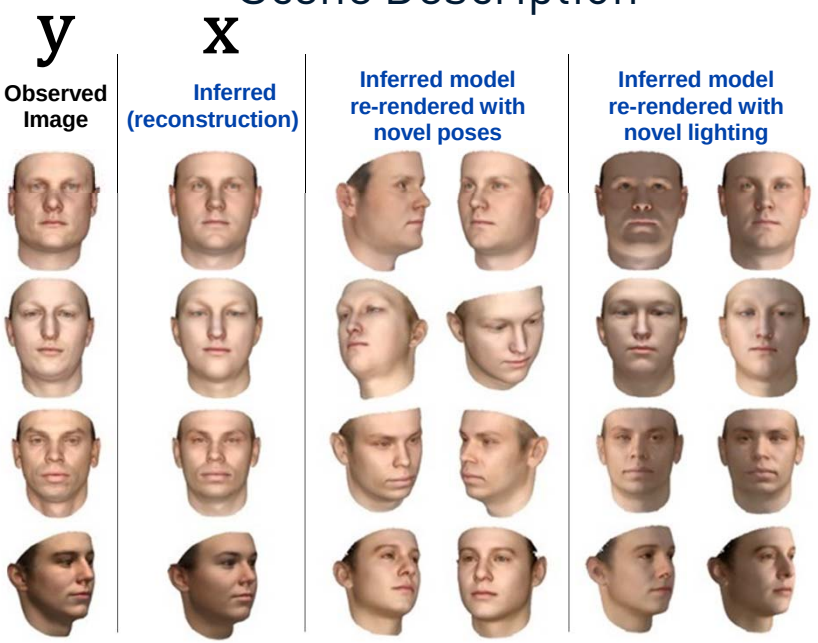
[credit: Frank Wood]

Perception / Inverse Graphics

Captcha Solving



Scene Description



x — scene description

y — image

Mansinghka,, Kulkarni, Perov, and Tenenbaum. "Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

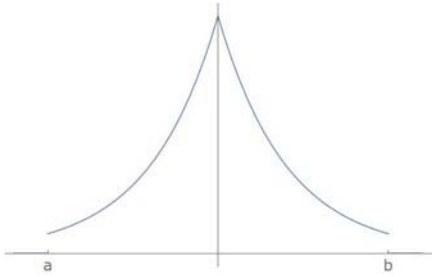
Kulkarni, Kohli, Tenenbaum, Mansinghka "Picture: a probabilistic programming language for scene perception." CVPR (2015)

How About Sum of Three Variables?

$Z = X1 + X2 + X3$

X1, X2, X3
i.i.d.

Truncated Laplace



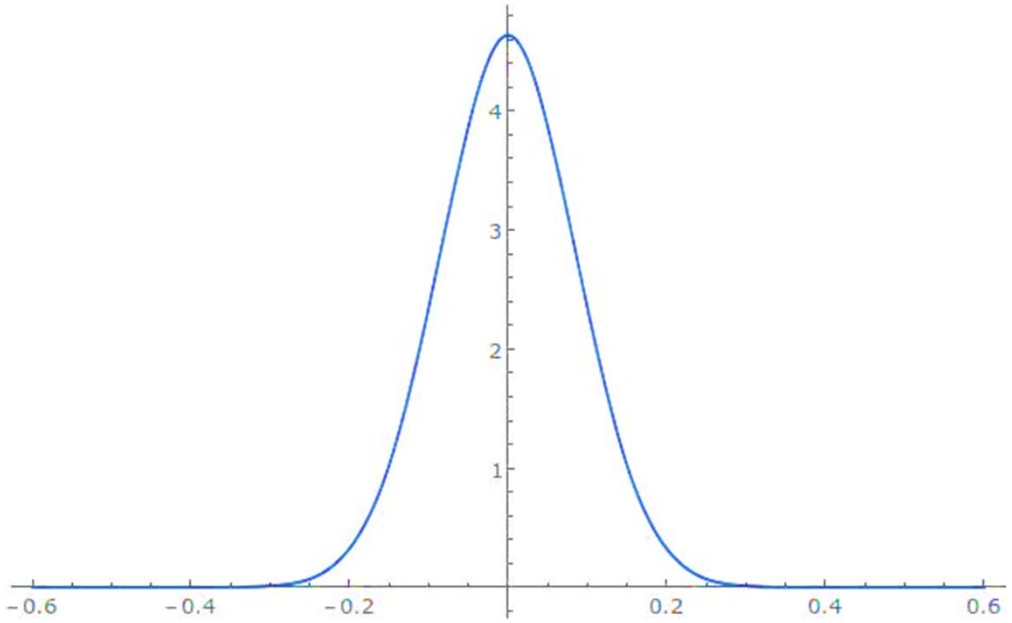
Z

?

Exact Final Distribution (PSI)

$ psi sum_laplace.prb

$$p(x) = -[x\neq 0]\cdot[x\leq 0]\cdot e\text{^}x\cdot x\cdot{}^1\!/_{16}+-[x\leq 0]\cdot e\text{^}x\cdot x\cdot{}^1\!/_{8}+3/32\cdot[-x\leq 0]\cdot[x\neq 0]\cdot{}^1\!/_{e}\text{^}x+3/32\cdot[-x\leq 0]\cdot{}^1\!/_{e}\text{^}x+3/32\cdot[x\neq 0]\cdot[x\leq 0]\cdot e\text{^}x+3/32\cdot[x\leq 0]\cdot e\text{^}x+[-x\leq 0]\cdot[x\neq 0]\cdot x\cdot{}^1\!/_{16}\cdot{}^1\!/_{e}\text{^}x+[-x\leq 0]\cdot x^2\cdot{}^1\!/_{16}\cdot{}^1\!/_{e}\text{^}x+[x\leq 0]\cdot x\cdot{}^1\!/_{8}\cdot{}^1\!/_{e}\text{^}x+[x\leq 0]\cdot e\text{^}x\cdot x^2\cdot{}^1\!/_{16}$$



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

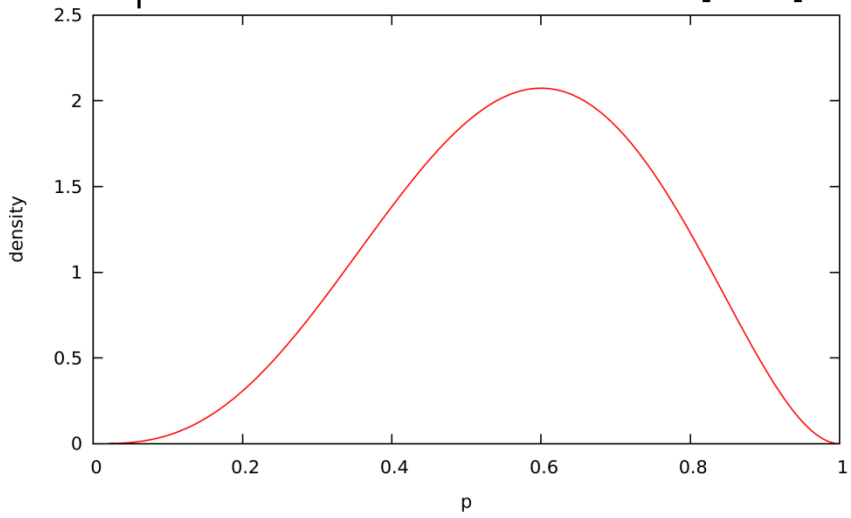
Applications: reliable machine learning (bias), robotics, networks, security

Probabilistic Program

```
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:

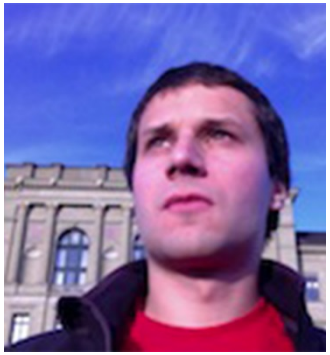
- do not agree with some of the objectives
- do not want to work during the semester (e.g., homework, reading papers)

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/riai.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)
ACM Doctoral Dissertation, 2016
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.