Probabilistic **P**rogramming for **A**dvancing **M**achine **L**earning (**PPAML**)

Program Kick-Off

Providence, RI

November 13-15, 2013

Dr. Kathleen Fisher

I2O Program Manager

Machine Learning is Ubiquitous and Very Useful

ISR

DARPA Grand Challenge
Fully autonomous ground vehicles competition

Nuclear Test Ban Treaty Compliance: Deduce set of seismic events given detections and misdetections on a network of stations

Image Search/Activity Detection: Find and identify objects and actions in video

Object Tracking: Follow vehicles as they move through a city and are recorded in multiple video streams (DARPA CZTS)

Patterns of Life: Process wide area aerial surveillance data and associated tracks to infer location types and object dynamics

Bird Migration Patterns: Model spatio-temporal distribution of birds (by species); involves large-scale sensor integration

DARPA LAGR: Vision-based robot navigation

Google Glasses: Perform searches based on images taken by user cell phone cameras

Natural Language Processing

© Apple

Siri
Voice recognition and Natural Language Processing (NLP)

Watson: Computer system capable of answering questions posed in natural language

Topic Models: Statistical model for discovering the abstract "topics" that occur in a collection of documents

Distributed Topic Models: Asynchronous distributed topic discovery

Citation Analysis: Given citations, extract author, title, and venue strings and identify co-reference citations

Entity Resolution: Discovering entities that are mentioned in a large corpus of news articles

NLP Sequence Tagging: Tagging parts of speech and recognizing named entities in newspaper articles

Predictive Analytics →

© Netflix

Netflix Challenge
Predict user ratings for films based on previous ratings

Microsoft Matchbox: Match players based on their gaming skill set

Predictive Database: Understand information based on causal relationships in data

Bing Image Search: Search for images on the web by selecting text in word document

Amazon Recommendation Engine: Recommend items based on consumer data

Cyber and Other

ORNL's Attack Variant Detector: Discover compromised systems

Yahoo's Bayesian Spam Filtering: Self-adapting system based on word probabilities

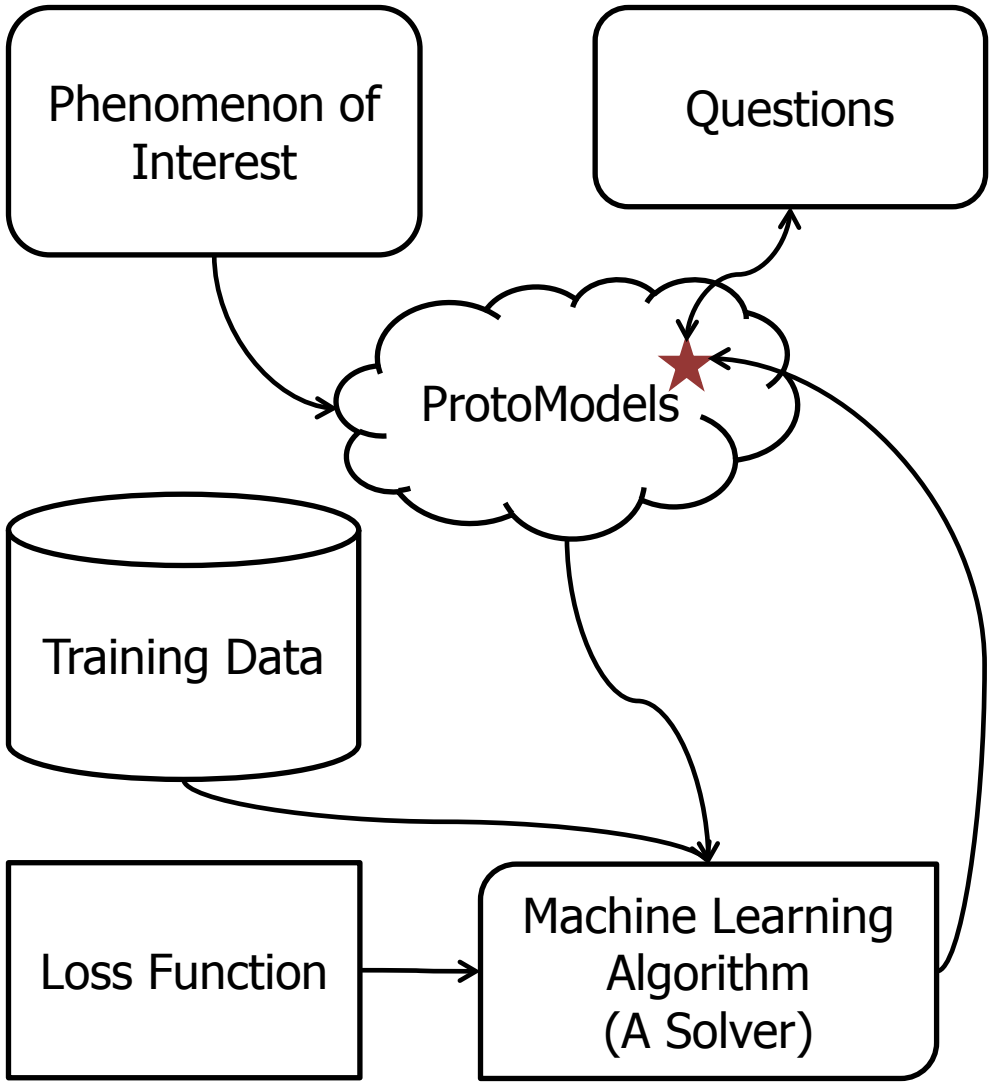
Cyber Genome Lineage: Reverse engineer malware samples to find shared "genetic" features between different malware samples

Gene Sequencing: Determine order of nucleotides in a DNA molecule

Disclaimer: Images of specific products are used for illustration only. Use of these images does not imply endorsement of inherent technical vulnerabilities.

Why Is It Hard?

- Brittleness of implementations & lack of reusable tools.
  PHY decoder: $100M/standard;
  Infer.NET: max 20% on model
- High level of required expertise
  10K solvers,
  100s of grad student hours per model
- Painfully slow & unpredictable solvers
  Massive data sets, complex algorithms, tricky coding for graph traversal and numeric stability
- Challenges constructing models
  Limited modeling vocabulary; models entwined with solvers



The Probabilistic Programming Revolution

Traditional Programming	*Probabilistic Programming*	
• Application	• Model	Code models capture how the data was generated using random variables to represent uncertainty
• Code Libraries	• Model Libraries	Libraries contain common model components: Markov chains, deep belief networks, etc.
• **Programming Language**	• **Probabilistic Programming Language**	PPL provides probabilistic primitives & traditional PL constructs so users can express model, queries, and data
• Compiler	• Inference Engine	Inference engine analyzes probabilistic program and chooses appropriate solver(s) for available hardware
• Hardware	• Hardware	Hardware can include multi-core, GPU, cloud-based resources, GraphLab, UPSIDE/Analog Logic results, etc.

High-level programming languages facilitate building complex systems
Probabilistic programming languages facilitate building rich ML applications

The Promise of Probabilistic Programming Languages

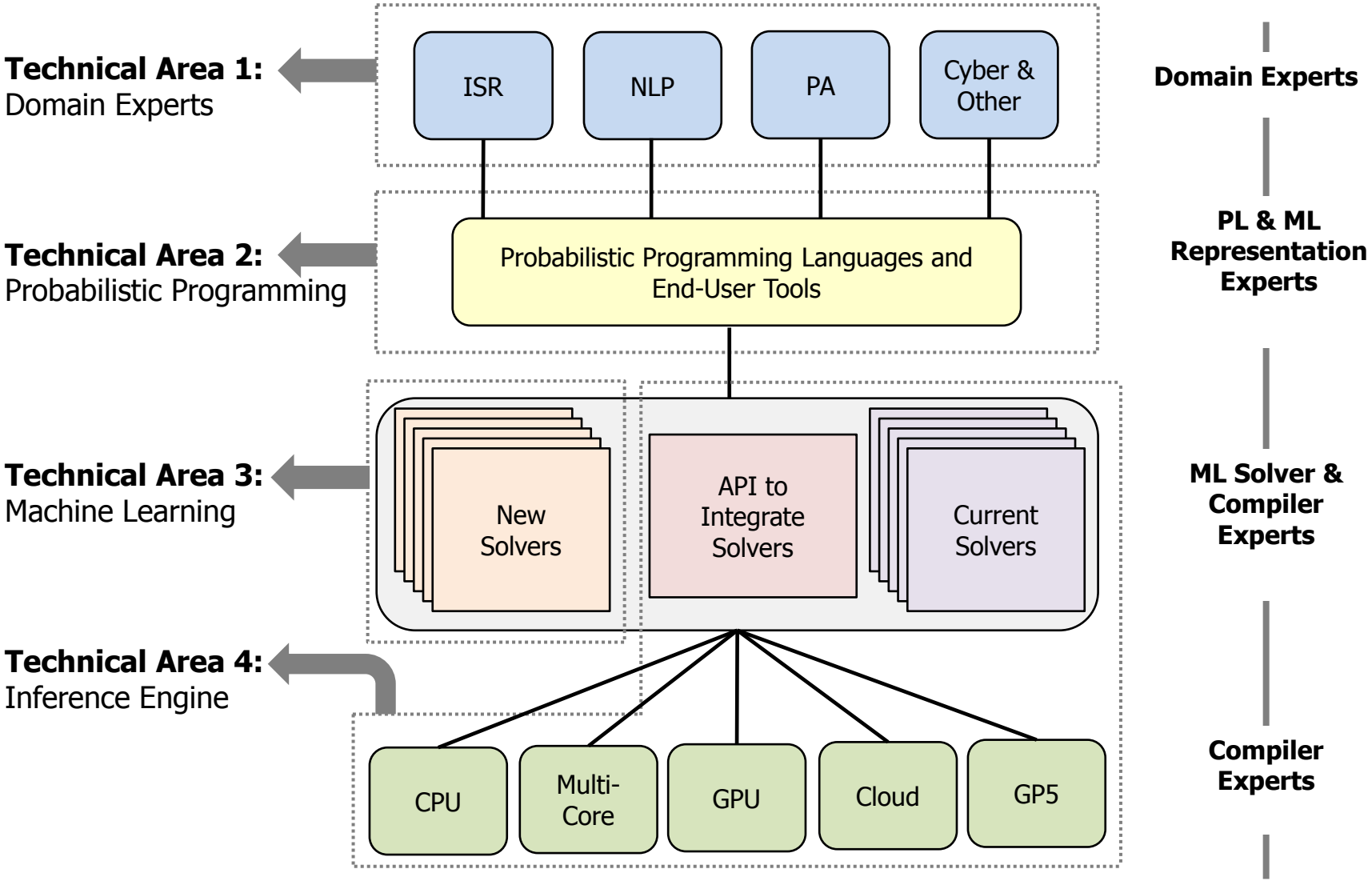
- **Shorter**: Reduce LOC by 100x for machine learning applications
  - Seismic Monitoring: 28K LOC in C vs. 25 LOC in BLOG
  - Microsoft MatchBox: 15K LOC in C# vs. 300 LOC in Fun
- **Faster**: Reduce development time by 100x
  - Seismic Monitoring: Several years vs. 1 hour
  - Microsoft TrueSkill: Six months for competent developer vs. 2 hours with Infer.Net
  - Enable quick exploration of many models
- **More Informative**: Develop models that are 10x more sophisticated
  - Enable surprising, new applications
  - Incorporate rich domain-knowledge
    - Produce more accurate answers
    - Require less data
    - Increase robustness with respect to noise
    - Increase ability to cope with contradiction

Sources:
- Bayesian Data Analysis, Gelman, 2003
- Pattern Recognition and Machine Learning, Bishop, 2007
- Science, Tanenbaum et al, 2011

- **With less expertise**: Enable 100x more programmers
  - Separate the model (the program) from the solvers (the compiler), enabling *domain experts* without machine learning PhDs to write applications

Probabilistic Programming could empower domain experts *and* ML experts

Proposed Program Structure



Performers

TA 1: Domain Experts

Galois*/OSU/Kitware/Raytheon Company & others (TBD)
Evaluator; Challenge Programs; Summer Schools

TAs 2 & 4: Probabilistic Programming & Inference Engine

BAE*/Tufts/ Stanford/ Northeastern
Create Open PP Platform (OP^3); intensional contracts

Gamelan*/ Lyric Labs/ Stanford/ Freer, LLC
Feedback about model likelihood and inference convergence

Stanford* (TA 4 only)
Efficient back-end solvers by applying opt. at multiple levels of abstraction

CRA*/UC-Berkeley/ UCI
Develop HLLs (i.e. BLOG) and compile to Figaro; algebra of solutions

MIT*/ U. Oxford/ Cambridge/ Continuum Analytics
Venture PPS; SIVM; advance theory of PP

Indiana*/ McMaster
Symbolic probabilistic intermediate language

Provide support to TAs 2 & 4 teammates to integrate the novel algorithms, representations or analyses discovered into the developed PPS

TA 3: Machine Learning & Programming Languages

Princeton*
Generic variational inference; streaming computation; model diagnosis and fitness

SRI*/ UT-Dallas/ NYU
Research advances in lifted inference and relational representation

UC-Riverside*
Inference engines for continuous-time (asynchronous event) systems

ACS*/ Columbia
Discriminative solvers to solve discriminative tasks specified in probabilistic programs

11 Total Primes (*)

Galois - Technical Area 1

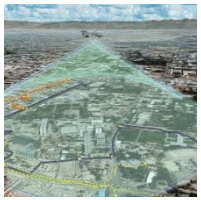
Subs: OSU, Kitware, Raytheon Company, and others

1. Develop Challenge Problems (CPs)

- Apply formal procedure for selecting CPs: a) solicitation via Wiki and b) evaluation based on "required" and "diversity" criteria
- Initial set of CPs: Quad-Rotor Sensor Fusion; Tracking Bird Migration; Wide Area Motion Imagery Track Linking

© Gaui

© Audobon of FL



Source: DARPA.mil

2. Evaluate PPSs w.r.t. each CP

Source: carolmunro.com

- Provide teams with problem specific APIs
- Evaluate each team's solutions on a "fresh" evaluation data set
- Apply standard and non-standard evaluation metrics
- Conduct benchmarking studies
- Use PL specific evaluation metrics to evaluate expressiveness and flexibility of PPLs
- Generate evaluation report before each PI meeting

3. Run Summer Schools

- Recruit participants from within and outside the PPAML community
- Develop curriculum and define tutorial problems with help from TA2-4 teams
- Generate evaluation reports, detailing the performance of each PPS

Source: smithsystem.com

4. Foster Collaboration

- Create Internal Wiki: program documents; materials for CP selection process; data and code for CPs; evaluation API materials, etc.
- Create External Wiki: publicly released CPs and educational materials and pointers to publications
- Organize workshops co-located with academic conferences (e.g., NIPS, ICML, POPL, etc.)

Source: technorati.com

MIT – Technical Areas 2 and 4

Subs: Cambridge University, U. of Oxford, Continuum Analytics

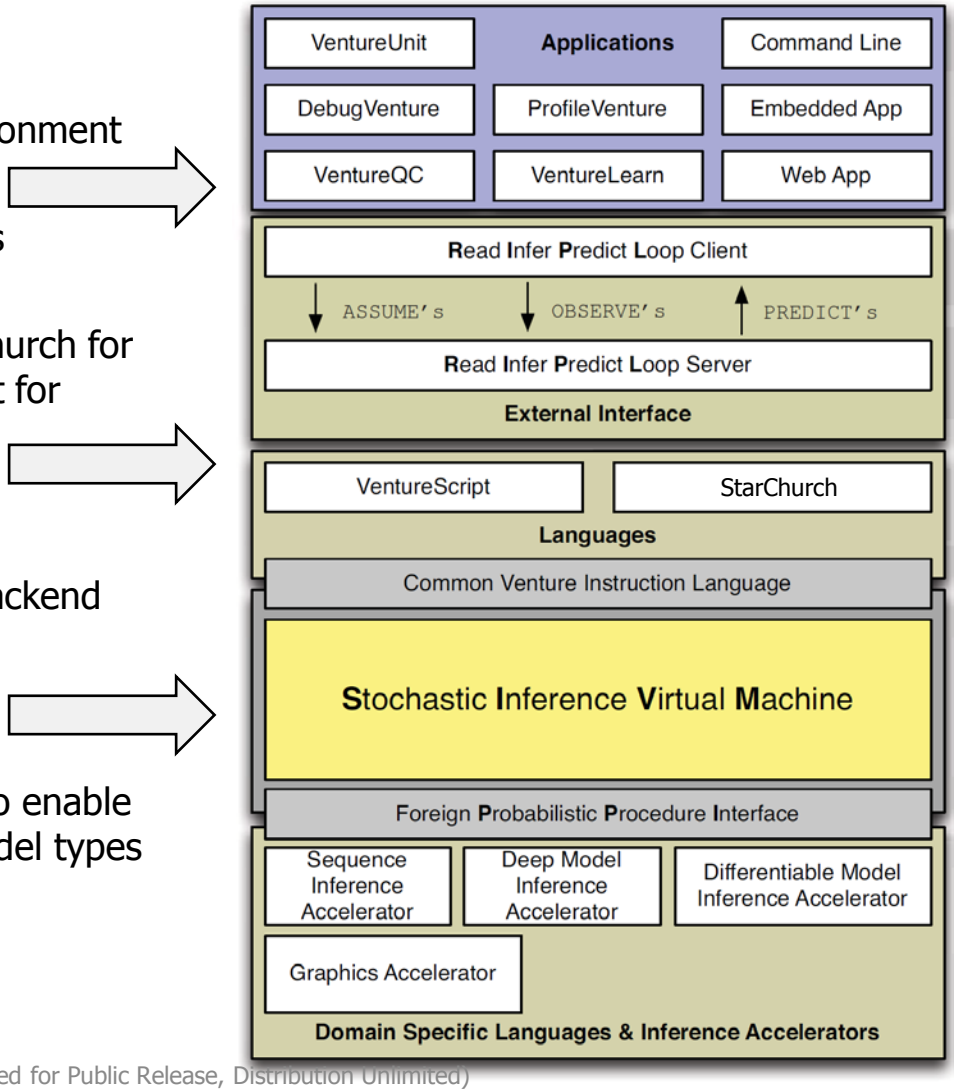
Goal: Make the *Venture PPS* useable by a wide array of people for a wide array of ML tasks

Technical Approach:

- Develop debugging and profiling tools
- Develop a web-based interactive user environment
- Develop a library of reusable models
- Develop a suite of on-line training materials

- Extend current front-end languages (StarChurch for machine learning experts and VentureScript for domain experts)
- Advance the state of the theory of PP

- Improve the runtime performance of the backend (Stochastic Inference Virtual Machine)
  - Target parallel hardware
  - Adopt hybrid solver techniques
  - Enable a "foreign function interface" to enable calls to custom solvers for specific model types
  - Develop a benchmark suite to drive performance improvements

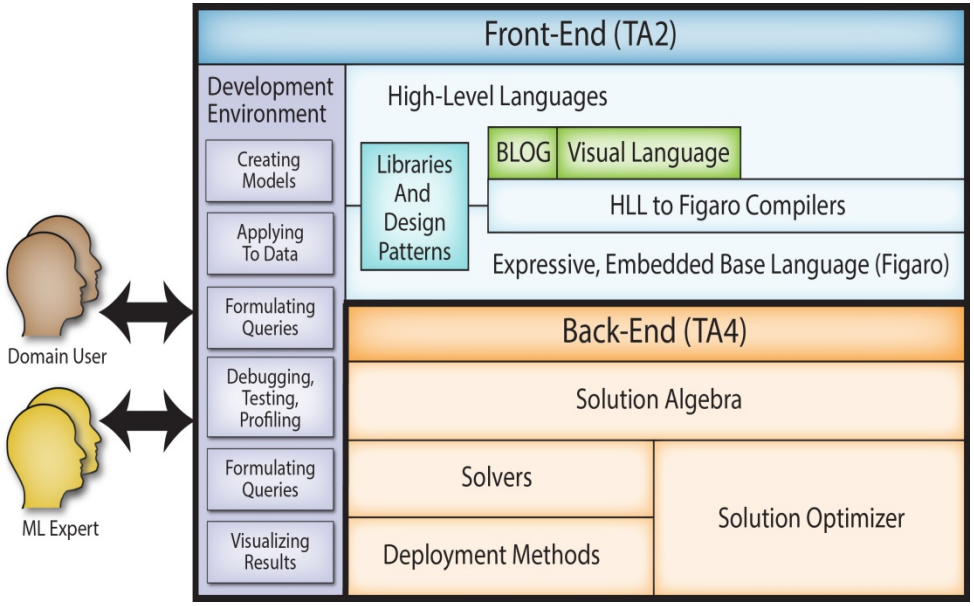


Charles River Analytics – Technical Areas 2 and 4

Subs: UC Berkeley, UC Irvine

Goal: Develop a fully integrated PPS that allows users to create rich models and automatically derive efficient implementations using:

- high-level languages (HLLs);
- compositional inference; and a
- development environment



Approach and Architecture

Technical Approach:

Front-End (TA2):

- Develop *HLLs* (BLOG & visual language)
- Improve *Figaro* as a compilation target IR for HLLs
- Develop *compilers* for HLLs into Figaro
- Build *model libraries and design patterns* for HLLs and Figaro

Back-End (TA4):

- Develop inference and data API based on *solution algebra*
- Develop *solution optimizer*
- Develop *deployment methods* to efficiently deploy solvers on HW
- Investigate new *solvers*

Development Environment:

- Develop *debugging & verification tools*
- Develop *profiling tools*
- Develop *query and results interface*

www.darpa.mil