Exploring Visual Programming Concepts for Probabilistic Programming Languages

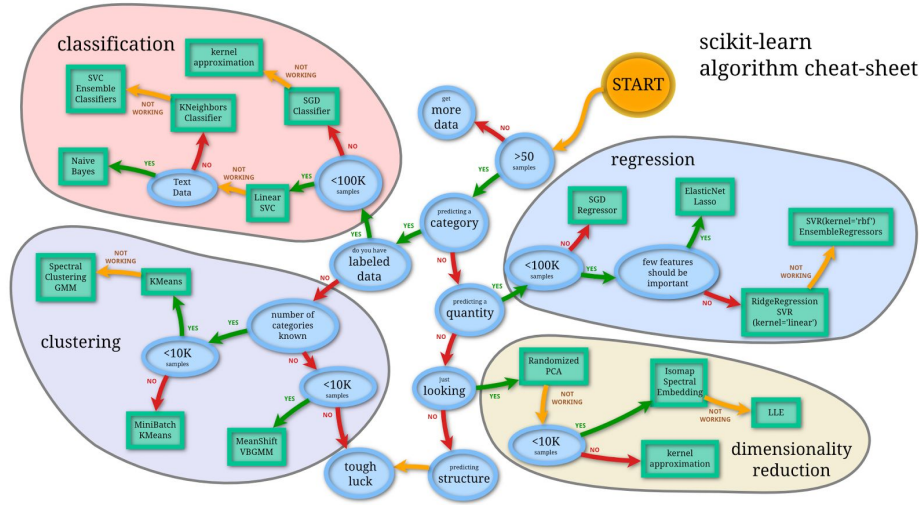
Gabriel Candal

Reasoning under uncertainty

Required in some domains [intml]:

- Computer Vision
- Cryptography
- Biology
- Fraud detection
- Internet ads bid/placement
- Recommender systems

Traditional approach [sp]: **prepackaged** and **very generic** machine learning **models**



Traditional approach

Problems?

- Doesn’t fully leverage domain knowledge [sp]
- May be hard to explain the results [b]

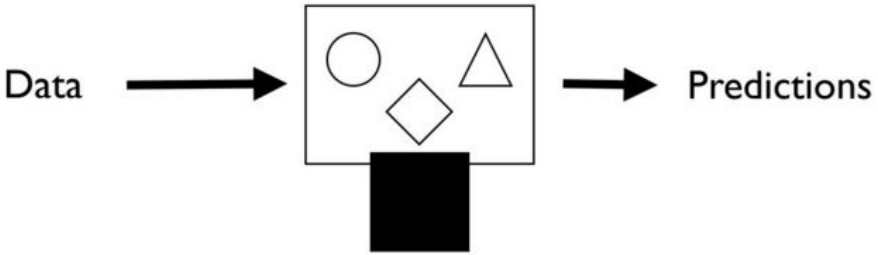
Solution?

- Build your own model!

How?

- Bayesian reasoning + PPLs

- Openbox models
- Blackbox inference engine

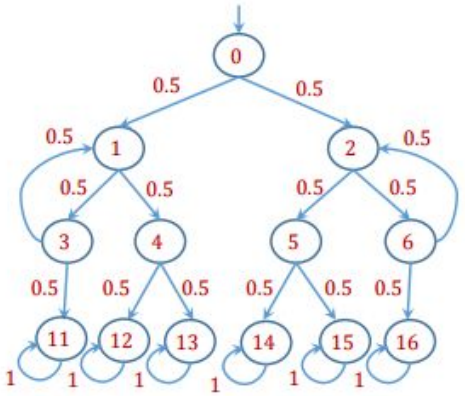


From Olivier Grisel’s SciPy 2013 Keynote

Probabilistic Programming Languages (PPLs)

Why PPLs

- **Unambiguous** way to communicate a model
- **Re-use** inference engine
- Use **different inference** methods easily
- **More general** than graphical models (such as Bayesian and Markov Networks) [intpp]




```
int x = 0;
while (x < 11) {
  bool coin = Bernoulli(0.5);
  if(x=0)
    if (coin) x = 1 else x = 2;
  else if (x=1)
    if (coin) x = 3 else x = 4;
  else if (x=2)
    if (coin) x = 5 else x=  6;
  else if (x=3)
    if (coin) x = 1 else x = 11;
  else if (x=4)
    if (coin) x = 12 else x = 13;
  else if (x=5)
    if (coin) x = 14 else x = 15;
  else if (x=6)
    if (coin) x = 16 else x = 2;
}
return (x);
```

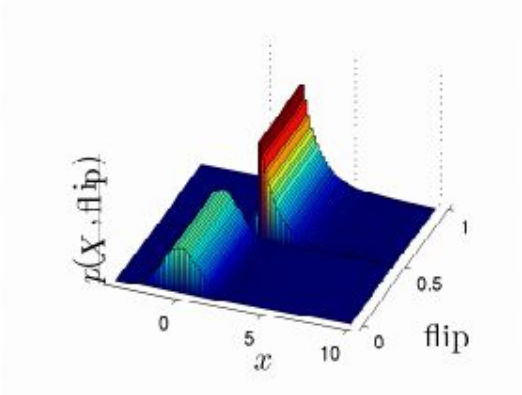
From: Andrew D. Gordon and Thomas A. Henzinger and Aditya V. Nori and Sriram K. Rajamani. Probabilistic Programming. International Conference on Software Engineering (ICSE Future of Software Engineering), 2014.

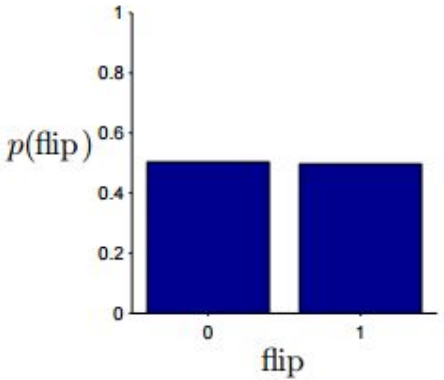
Why PPLs - example

```
flip = rand < 0.5
if flip
    x = randg + 2    % Random draw from Gamma(1,1)
else
    x = randn        % Random draw from standard Normal
end
```

Inference engine

Implied distributions over variables





From: David Duvenaud and James Lloyd. Introduction to probabilistic programming. University of Cambridge, 2013.

Problem definition - PPLs' shortcomings

- Forces the users to learn **yet another syntax**
- Interface **doesn't resemble common data analysis tools** (Excel, RapidMiner, Weka Knowledge Flow, ...)

Simple model

$$\begin{cases} \alpha,\ \beta & \sim & \text{NORMAL}(0, 10^{-6}) \\ \tau & \sim & GAMMA(10^{-3}, 10^{-3}) \end{cases}, \quad \begin{cases} Y_i & \sim & \text{NORMAL}(\mu_i, \tau) \\ \mu_i & = & \alpha + \beta(x_i - \bar{x}) \end{cases}$$

From: Ciprian Crainiceanu. A Short Introduction to WinBUGS. 2014.

WinBUGS PPL

Complicated textual repres.

```
model
    {for (i in 1:N)
        {Y[i]~dnorm(mu[i],tau)
         mu[i]<-alpha+beta*(x[i]-mean(x[]))}
    alpha~dnorm(0,1.0E-6)
    beta~dnorm(0,1.0E-6)
    tau~dgamma(1.0E-3,1.0E-3)
    sigma<-1/sqrt(tau)
    }
```

- **Negatively affects productivity** [jr]
- **Slows down PPL adoption** [dp]

State of the art - PPLs

Modern PPLs:

- Figaro (Scala) - Scala library
- Infer.NET - language that compiles to C#
- Church - inspired in Lisp
- WebPPL - written in Javascript as learning tool and proof-of-concept

All of these:

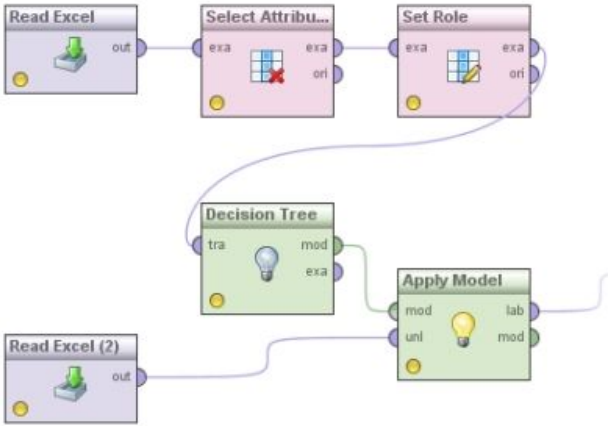
- Provide **high-level abstractions**
- ... but as other programming languages, still have a **steep learning curve** for unexperienced programmers [cu][ca]

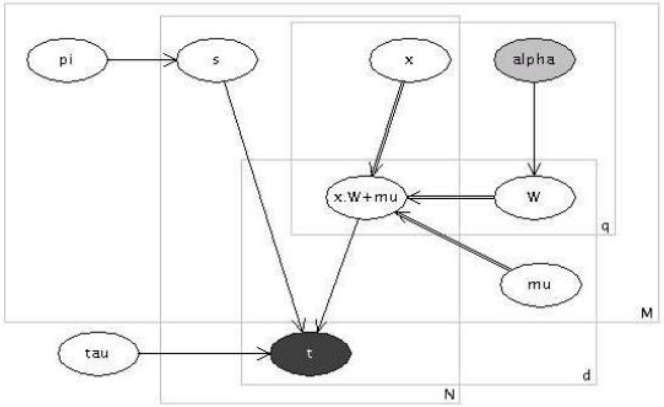
Visual Programming Concepts for PPLs

State of the art - Visual Programming

- **Purely visual** programming languages: **executable** graphical representation [bu]

E.g.: RapidMiner or VIBES/WinBUGS (w/inf. for bayesian networks)

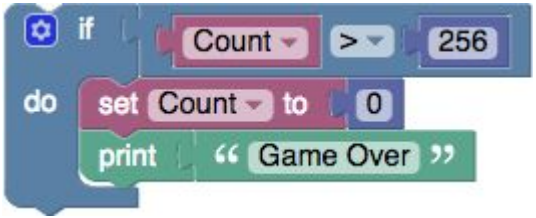




State of the art - Visual Programming

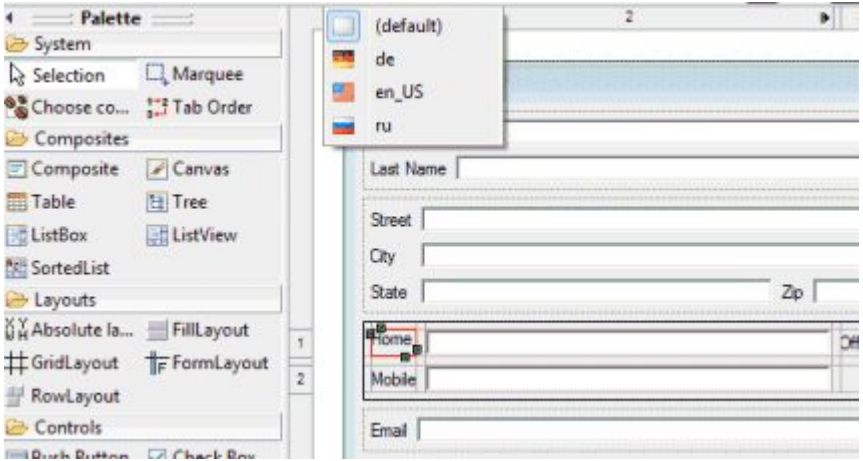
- **Visual** programming **environment**: **translate** graphical representation **into code** [bo]

E.g.: Blockly



or

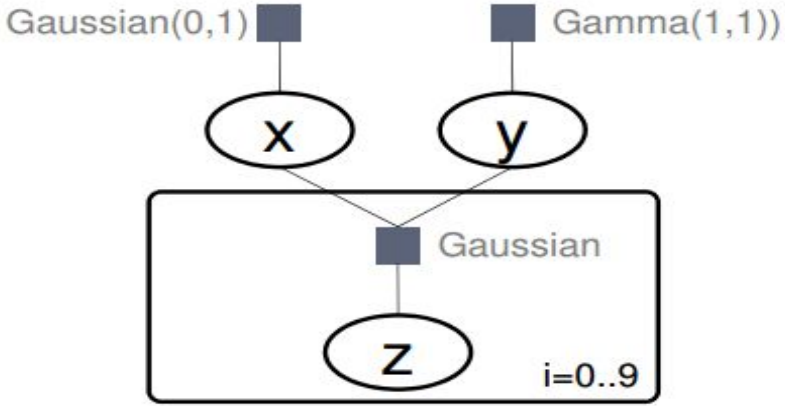
Eclipse Plugin WindowBuilder



Proposed Solution - VPE for PPLs

How?

- Prototype visual programming environment that compiles to PPL syntax
- Exploratory analysis of which visual concepts are compatible with the PPLs' domain




```
x = Gaussian(0, 1)

y = Gamma(1, 1)

for(i in 0..9)
  z = Gaussian(x, y)
```

From: John Winn and Tom Minka. Probabilistic Programming. Machine Learning Summer School, 2009.

Proposed Solution - Validating

Hypothesis:

- More intuitive
- Easier to learn
- Faster to develop in

Validation:

- **Convert examples** of a program written in a PPL **to a graphical form**
- **Compare** both the textual and graphical form

Open issues:

- Functions
- Recursion
- Arrays of arrays of ...
- Mutation: x=x+1
- Objects?
- Immediate visual feedback?

Work Plan

		February		March				April				May					June				July
		15th - 21th	22th - 28th	29th - 6th	7th - 13th	14th - 20th	21th - 27th	28th - 3rd	4th - 10th	11th - 17th	18th - 24th	25th - 1st	2nd - 8th	9th - 15th	16th - 22th	23th - 29th	30th - 5th	6th - 12th	13th - 19th	20th - 26th	27th - 1rd
	Tarefas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Dissertation	Writting																				
Development	Analyze frontend alternatives																				
	Prototype frontend																				
	Pick a target PPL																				
	Write code generator																				
	Define visual grammar and features to include																				
	Grather examples and convert to visual form																				
Conclusion	Tutorial																				
	Study differences between representations																				

References

- scikit-learn. Choosing the right estimator. http://scikit-learn.org/stable/tutorial/machine_learning_map/
- Rob Knies. Infer.NET: Machine Learning Tailor-Made. Inside Microsoft Research, 2013.
- John Winn and Tom Minka. Probabilistic Programming. Machine Learning Summer School, 2009.
- Andrew D. Gordon and Thomas A. Henzinger and Aditya V. Nori and Sriram K. Rajamani. Probabilistic Programming. International Conference on Software Engineering (ICSE Future of Software Engineering), 2014.
- [sp] Olivier Grisel. SciPy 2013 Keynote: Trends in Machine Learning and the SciPy community.
- [intml] Ethem Alpaydin. Introduction to machine learning. MIT Press. 2010.
- [intpp] David Duvenaud and James Lloyd. Introduction to probabilistic programming. University of Cambridge, 2013.

References

- [jr] Jamal, Rahman and Wenzel. The Applicability of Visual Programming to Large Real-World Applications. Igarss. 2014.
- [dp] Suresh Jagannathan. Probabilistic Programming for Advancing Machine Learning (PPAML). 2013.
- [b] Allen Downey. Think Bayes. Green Tea Press. 2012.
- [cu] Nancy Cunniff and Robert Taylor and John Black. Does programming language affect the type of conceptual bugs in beginners' programs? A comparison of FPL and Pascal. ACM SIGCHI Bulletin. 1986.
- [ca] Martin Carlisle, and Terry Wilson and Jeffrey Humphries and Steven Hadfield. Raptor. SIGCSE '05 2005.
- [bu] Margaret Burnett. Visual programming. Wiley Encyclopedia of Electrical and Electronics Engineering. 1999.
- [bo] Marat Boshernitsan and M.S. Downes.Visual programming languages: A survey. Control. 2004.

References

- Avi Pfeffer and Brian Ruttenberg and Michael Howard and Allison O'Connor. Figaro Tutorial. Online, accessed 07/02/16, https://www.cra.com/sites/default/files/pdf/Figaro_Tutorial.pdf.
- Noah Goodman and Andreas Stuhlmueller. The Design and Implementation of Probabilistic Programming Languages. Online, accessed 07/02/16, http://dippl.org. 2014.
- N. D. Goodman and J. B. Tenenbaum. Probabilistic Models of Cognition. Online, accessed 07/02/16, http://probmods.org.