Programming Paradigms for Machine Learning

Wannes Meert
Research Manager @ DTAI research group

Leuven.inc Visionary seminar
22 March 2016

KU LEUVEN

Outline

- Introduction: What is a programming language?
- Part I : Machine Learning as a Programming Language
- Part II : Machine Learning using a Programming Language

KU LEUVEN

What is a programming language?

"A formal constructed language designed to communicate instructions to a machine"

Declarative/Functional Languages
Express goals, not algorithms

```
Border(a2,a1) ⇒
  Color(a1) ≠ Color(a2)
```

Compilation

General purpose language
Expresses high-level operations
CPU independent

```
program helloworld
      print *, "Hello world!"
end program helloworld
```

Compilation

Assembly language
Strongly tied to chipset
custom and low-level

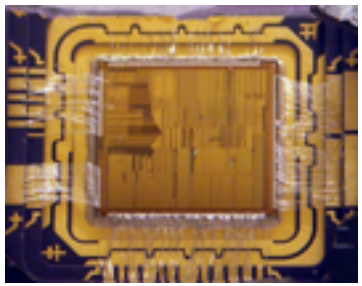
```
cmpl   $1, %edx
jle    endloop
add    $1, %ecx
movl   %edx, %eax
```

Software

Hardware

Compilation

CPU
Bare metal
'Physical' programming



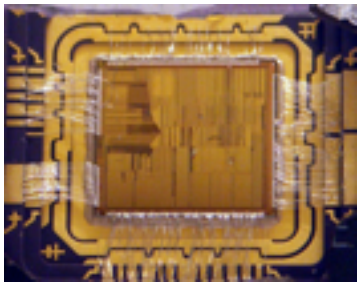
KU LEUVEN

What is a programming language?

Compilation

Dynamic instance of

CPU
Bare metal
'Physical' programming



And - Or - Not
A circuit representing a logical formula

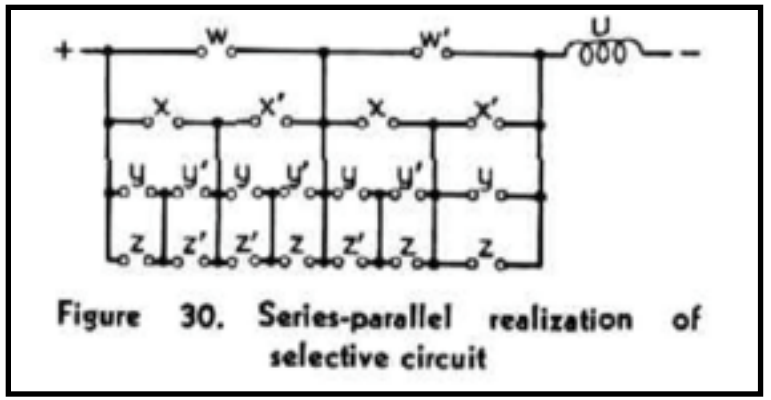


Figure 30. Series-parallel realization of selective circuit

A Symbolic Analysis of Relay and Switching Circuits*

*Claude E. Shannon***

One of the first mentions of 'automated reasoning' (1937)

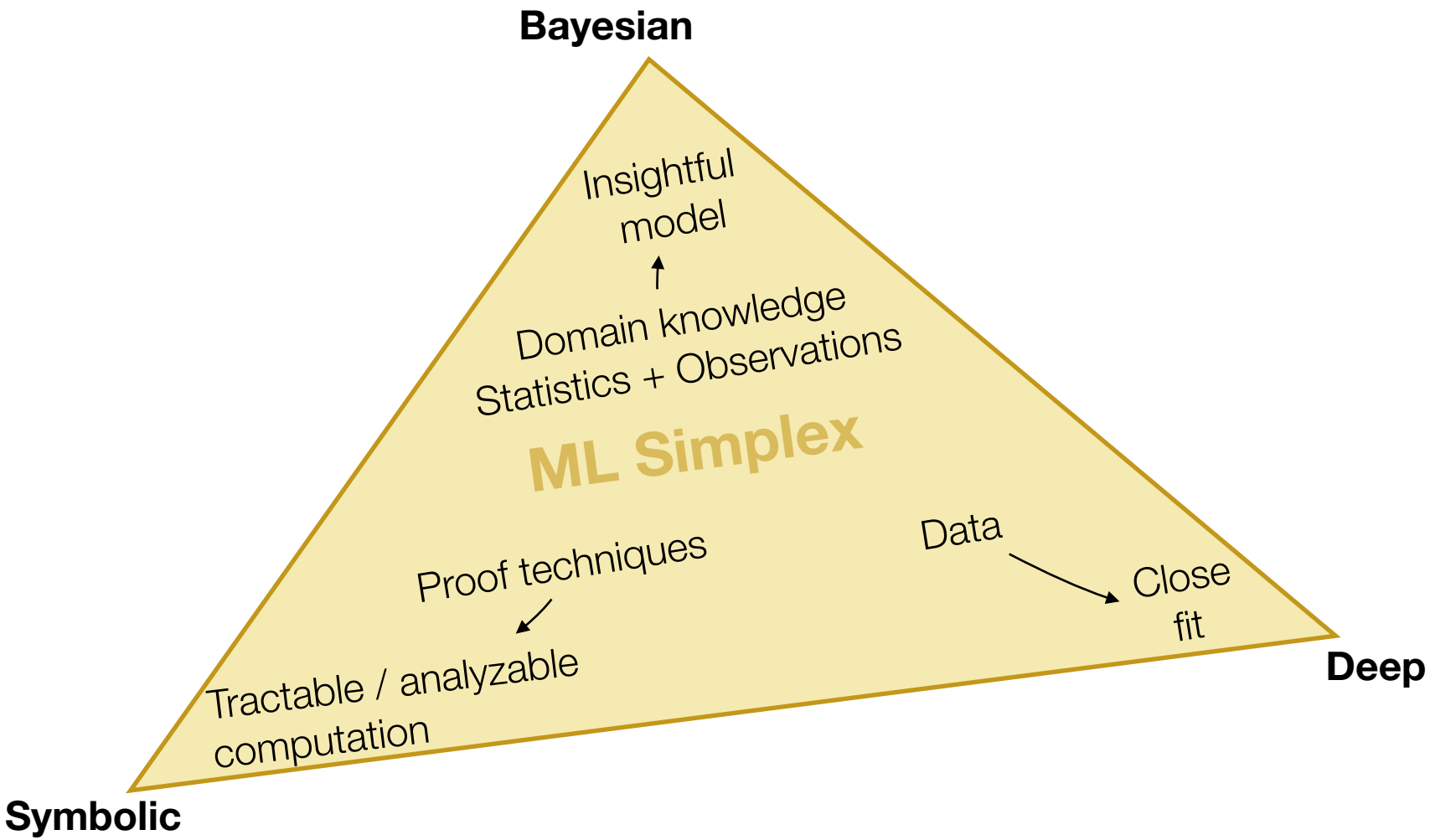
KU LEUVEN

Part I
Machine Learning as a Programming Language

What is the ‘assembly language’ you use?

KU LEUVEN

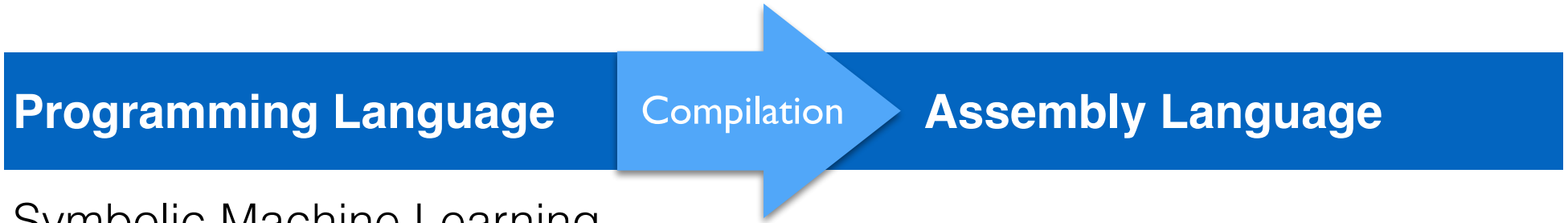
Cultures of Machine Learning



[Breiman 2001, Eisner 2015, Domingos 2015]

KU LEUVEN

Types of Languages



Symbolic Machine Learning
Logic reasoning, Declarative modelling
SAT, Constraint Programming, Decision trees

First-Order Logic

Bayesian Machine Learning
Statistics, Probabilistic Graphical Models

Factor Graphs

Connectionist Machine Learning
Neuroscience, Deep learning

Artificial Neural Networks

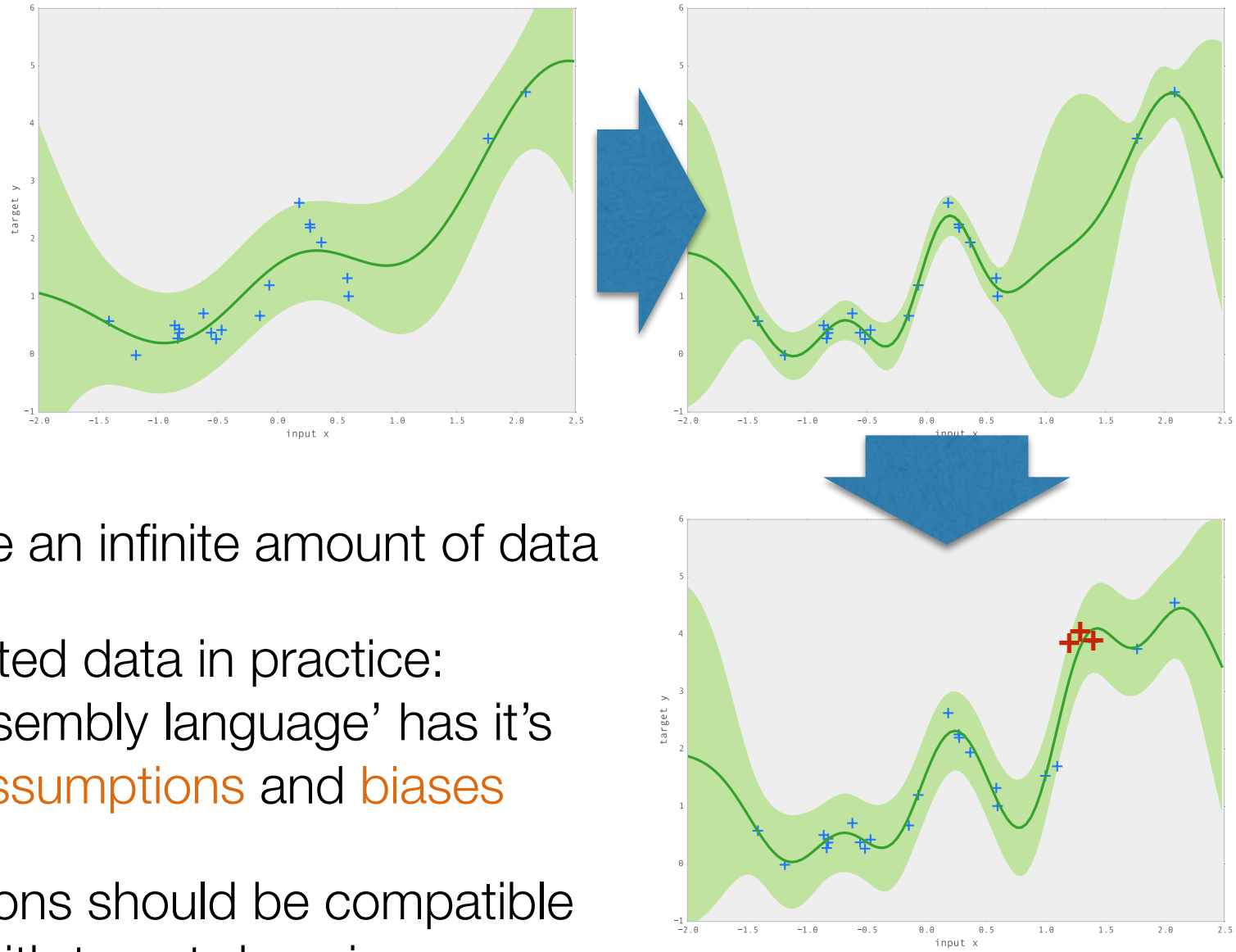
Interpretable

Efficient / Standardised

KU LEUVEN

Method

Adapt the parameters/structure until the objective function is learned perfectly



Can require an infinite amount of data

Limited data in practice:
each ‘assembly language’ has it’s own assumptions and biases

Assumptions should be compatible with target domain

Shared property of all ML assembly languages

Assumptions should

Allow (approximate) reasoning in all directions

y=f(x): Easy

Every x leads to one y

```
def f(x1,x2,x3):
  g = 9.8
  y = g*x1**2
  y /= 2*(x2*cos(x3))**2
  y += x1*tan(x3)
  return y
```

x=g(y): Hard

Every y can lead to multiple x

Forward + Backward Reasoning

- Reasoning in both directions is required to learn from data
  - Reason back from the classification outcome of a model to tune its parameters (numeric knowledge)
  - Reason back over the decisions to verify with expert knowledge (symbolic knowledge)

- Machine learning =
  - A language that allows reasoning over the impact of its parameters and structure (why)
  - A heuristic to change parameters and structure to fit the observations better (how)

KU LEUVEN

Symbolic Machine Learning

Relational Probabilistic Database

Induction of logic rules from data:

```
Cancer(id1) IF
  Birads_category(id1, B5) AND
  MassPAO(id1, Present) AND
  MassesDensity(id1, High) AND
  HO_breastCA(id1, HxDCorLC) AND
  In_same_mammogram(id1, id2 ) AND
  Calc_pleomorphic(id2, NotPresent) AND
  Calc_punctate(id2, NotPresent)
```

General language: first-order logic / AND-OR-NOT
Forward reasoning: logic deduction
Backward reasoning: logic abduction

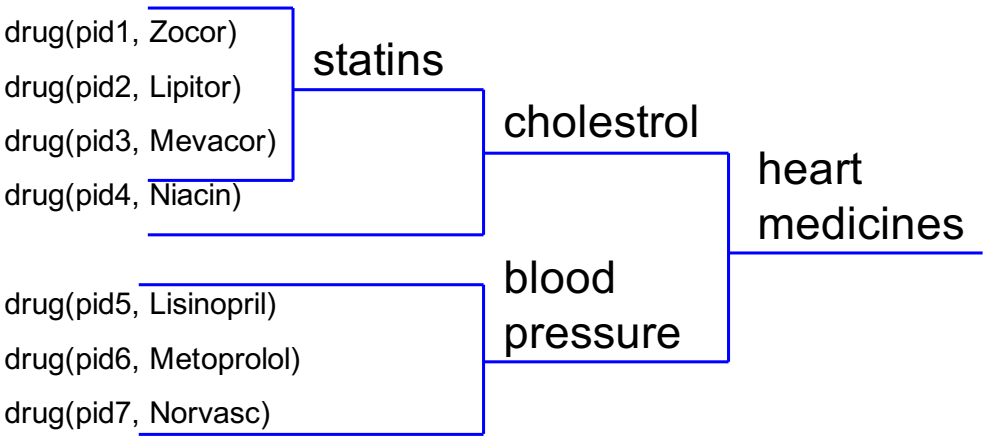
[Burnside et al. 2005]

KU LEUVEN

Symbolic Machine Learning

```
cluster(zocor, statins).
cluster(lipitor, statins).
cluster(lisinopril, bp).
cluster(statins, cholesterol).
cluster(niacin, cholesterol).
…
group(X, Group) IF
  cluster(X, Group) OR
  cluster(X, Y) AND group(Y, Group)
```

```
¬group(X, Group) v
  cluster(X, Group) v
  cluster(X, Y)
¬group(X, Group) v
  cluster(X, Group) v
  group(Y, Group)
…
```



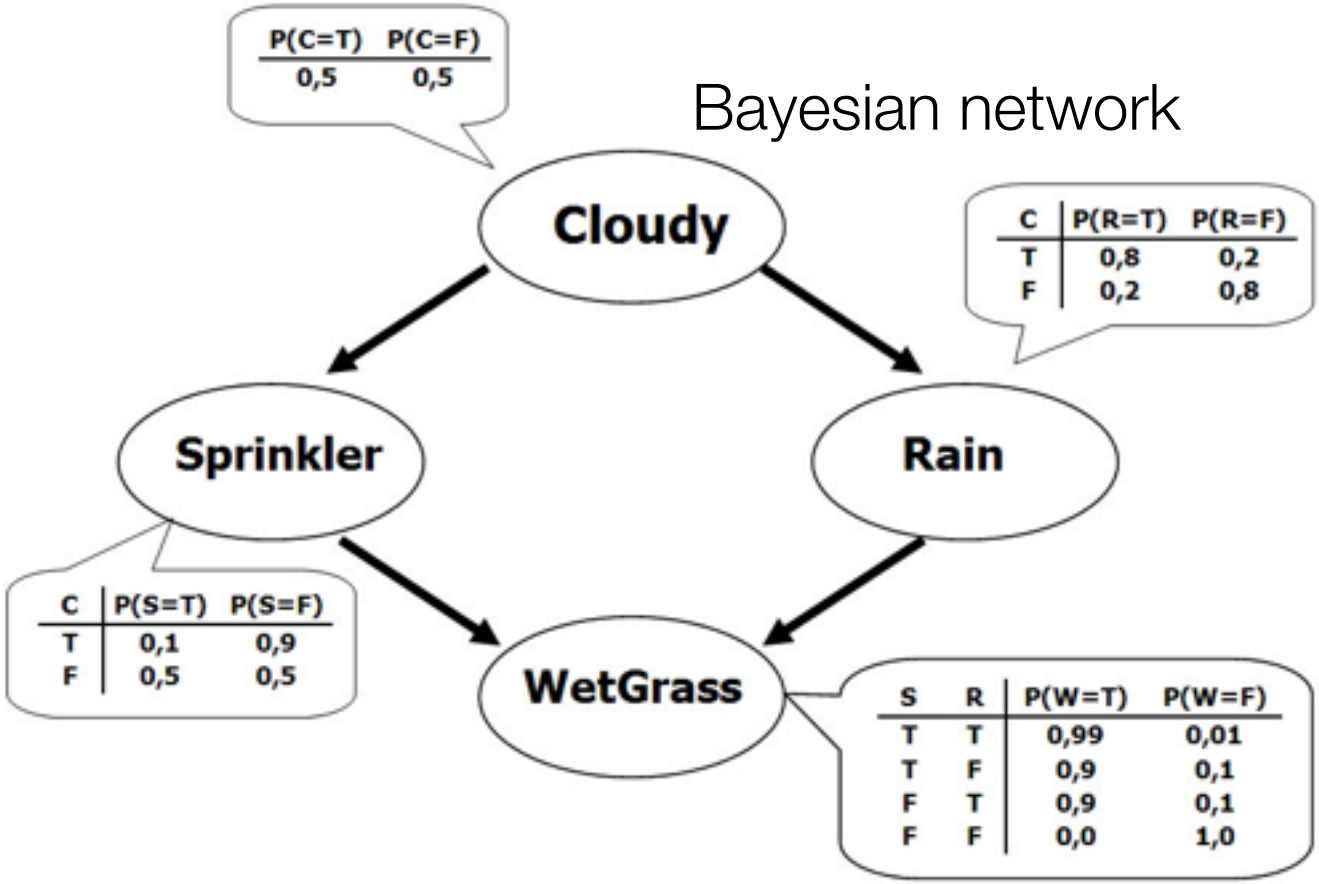
Deduction from logic rules

```
group(zocor, statins).
group(zocor, cholesterol).
group(zocor, heartmedicines).
group(lipitor, cholesterol).
…
```

[Davis et al 2007]

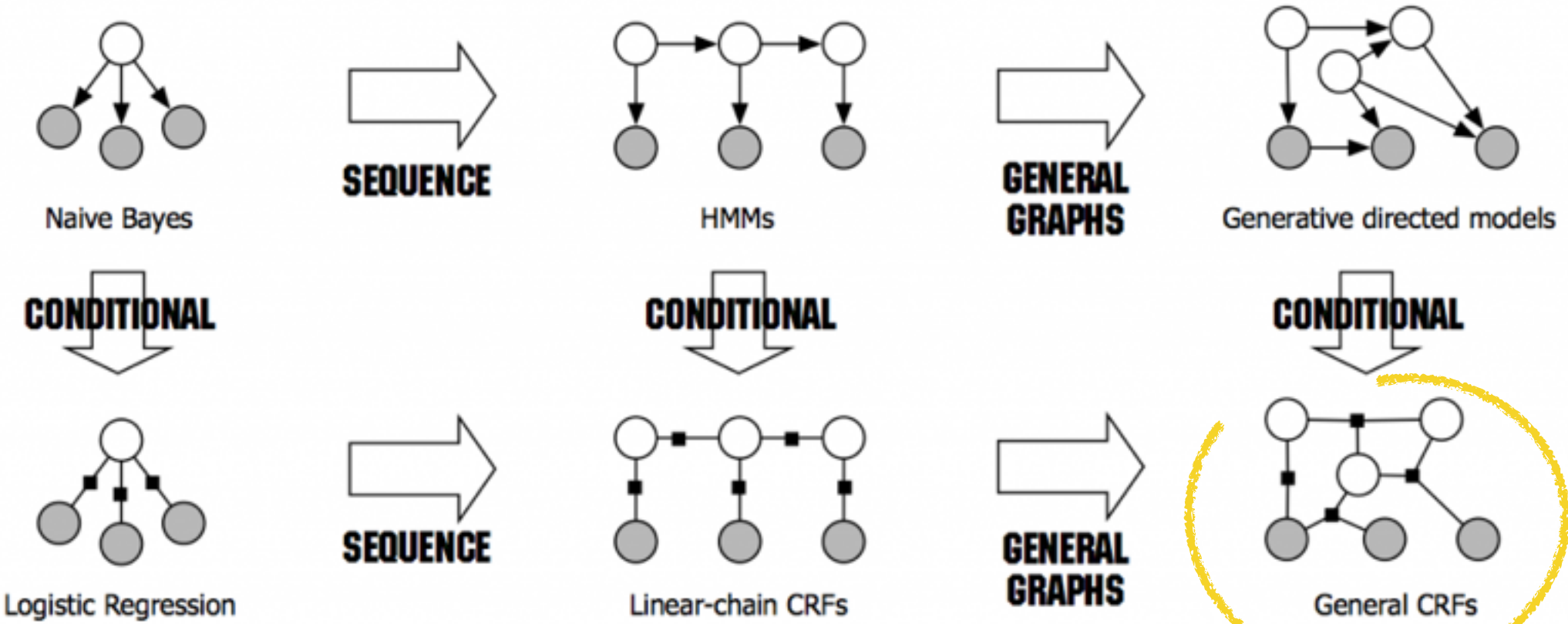
KU LEUVEN

Bayesian Machine Learning



General language: Factor Graphs
Forward reasoning: Conditional probabilities
Backward reasoning: Bayes' rule

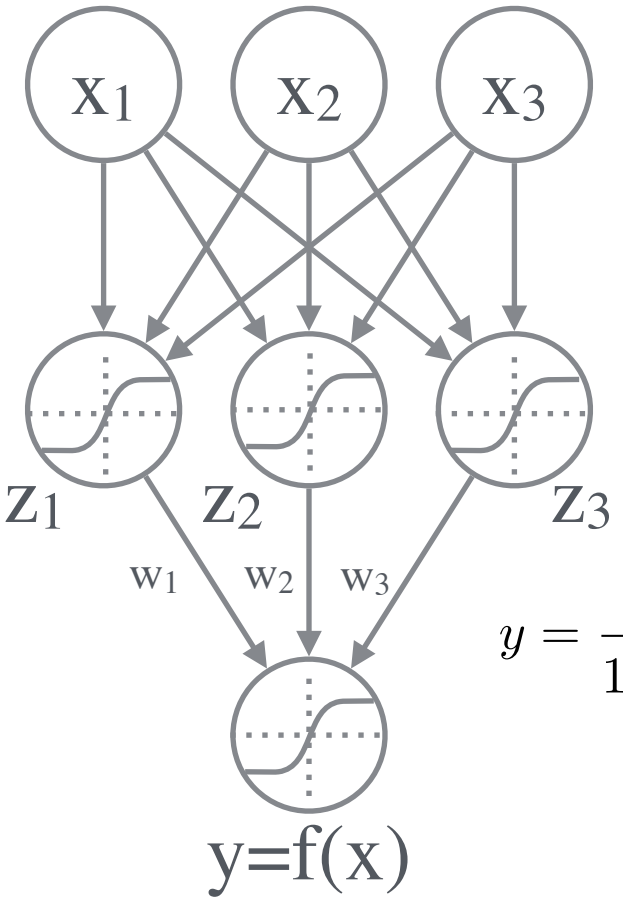
Bayesian Machine Learning



Factor Graph

[Sutton & McCallum 2010]

Connectionist Machine Learning

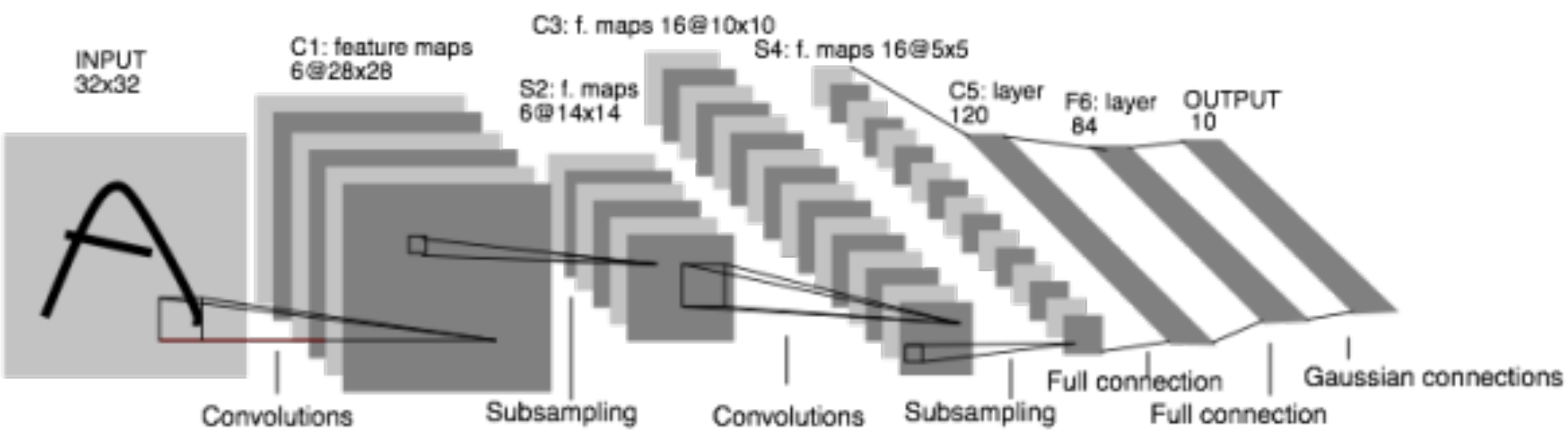


$$y = \frac{1}{1 + e^{-\sum_j w_j \cdot z_i}}$$

General language: Neural net
Forward reasoning: Activation function + convolution
Backward reasoning: Backpropagation + dropout

Connectionist Machine Learning

```
layer {
  name: "conv1"
  type: "Convolution"
  param { lr_mult: 1 }
  param { lr_mult: 2 }
  convolution_param {
    num_output: 20
    kernel_size: 5
     weight_filler {
       type: "xavier"
…
```



[LeCun 1989]

Conclusion Part I

- The ‘assembly language’ chosen has a large impact on the models that can be represented and the tasks that can be handled
- Hardware that knows how to handle the ML model allows for improved inference and learning
- A typical real-world system uses a combination of different models

- Most machine learning models require that the user has deep knowledge of the underlying ‘assembly language’

KU LEUVEN

Part II
Machine Learning using a Programming Language

Use the programming paradigm you prefer, not the ‘assembly language’

KU LEUVEN

Brief history of Bayesian networks

- 1962: Tree based networks with AND/OR and using maximisation as approximation (Fault Tree Analysis)
- 1980: Tree based networks with conditional probability tables and exact inference (~Bayesian networks)
- 1988: Arbitrary networks with limited treewidth (Bayesian networks)
- 1993: Networks with independent causation (noisy-or)
- 2000: Networks with limited contextual treewidth
- 2005: Networks with any type of local context
- 2010: Probabilistic programming (arbitrary relations/functions)

Our compilers and target language improved over time

KU LEUVEN

Probabilistic Programming

- Introduce stochastic variables into a programming language

```
>>> choice(0.5)
True
>>> choice(0.5)
False
>>> choice(0.5)
False
```

- Any type of distribution and interactions
- Embed in your favourite ecosystem (SciPy, R, Scala, …)
- The user can express the problem without restrictions
- Automatically find the best ‘assembly language’ given the program

KU LEUVEN

Probabilistic Programming

Many variations exist, both for exact and approximate answers

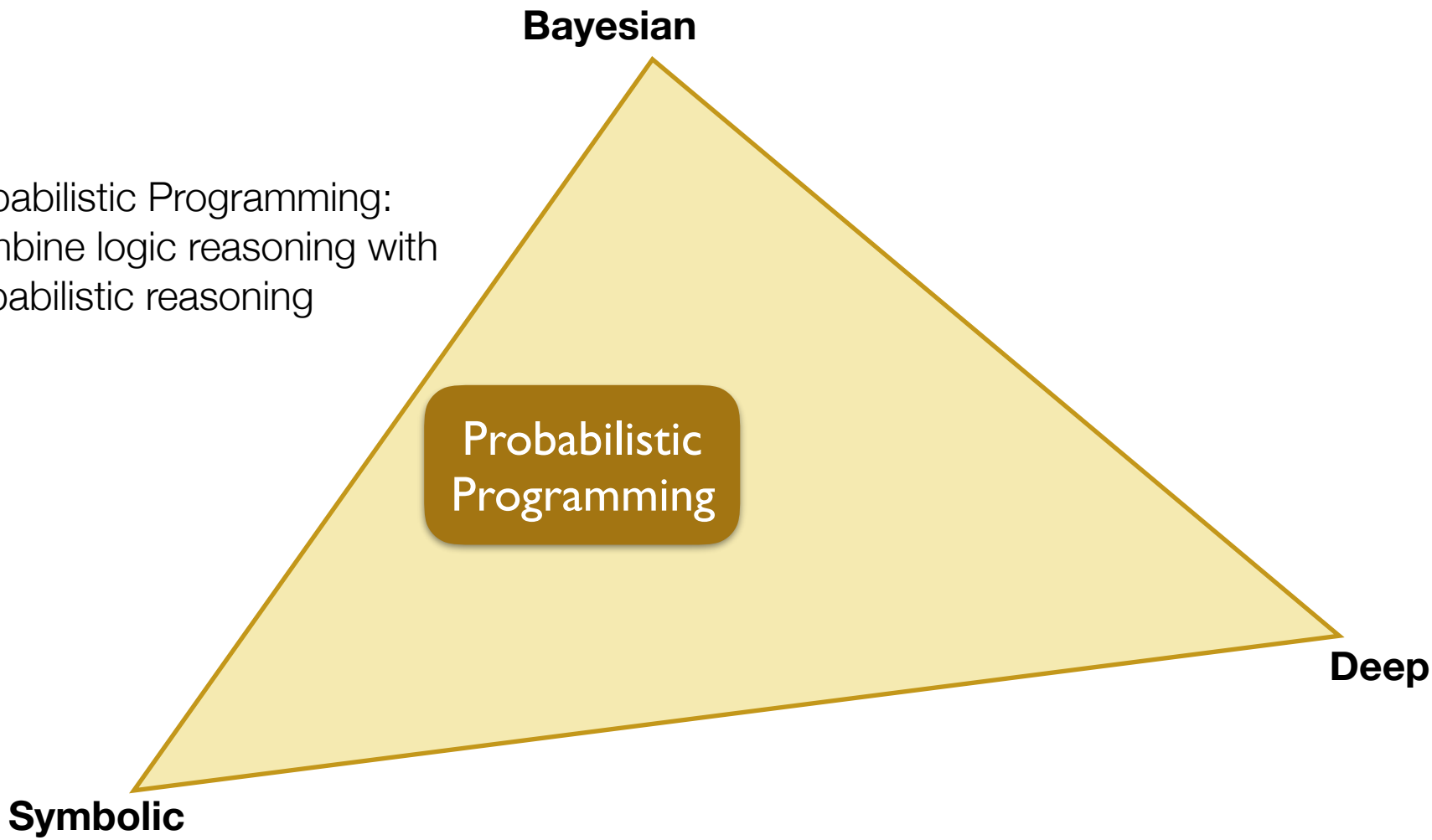
2000	**Balios**	**Prolog**	**University Freiburg / KU Leuven**
2002	PRISM	Prolog	University of Tokyo
2003	BLOG	DSL	UC Berkeley
2005	Alchemy	First-Order Logic	University of Washington
2006	**ProbLog**	**Prolog, Python**	**KU Leuven**
2008	Church	Scheme	MIT
2008	infer.NET	C#	Microsoft
2008	Factorie	Scala	University of Massachusetts Amherst
2009	Figaro	Scala	Charles River Analytics
2011	Stan	R, Python, Matlab, Julia	Columbia University
2011	**WFOMC**	**First-Order Logic, Scala**	**KU Leuven / UCLA**
2014	Anglican	Clojure	Oxford University
2015	Venture	Python	MIT
	…	…	…

A probabilistic program is an interpretation

- A probabilistic program is rather a set of interpretations than an algorithm
- A learning algorithm uses the interpretations it requires to perform the task at hand (inference, learning, decisions, …)

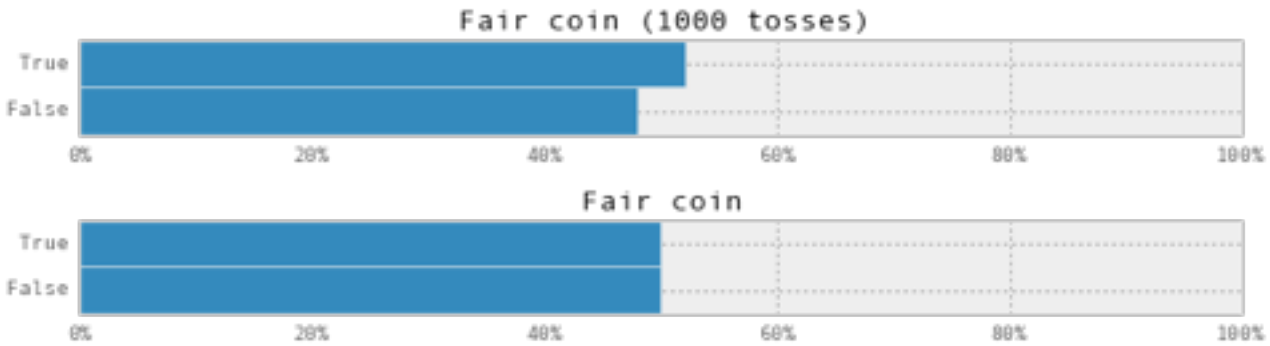
ML Simplex

Probabilistic Programming:
Combine logic reasoning with probabilistic reasoning

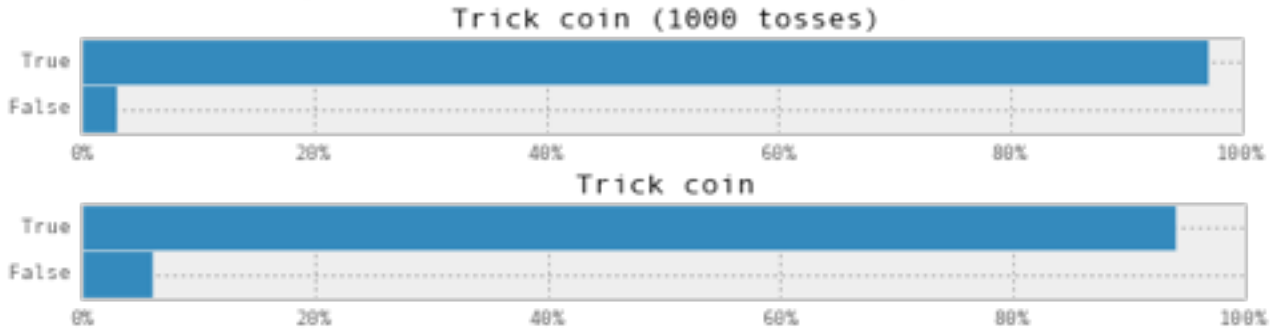


Example: Coin Toss

```
def fair_coin():
  return choice(0.5)
```




```
def trick_coin():
  return choice(0.95)
```



KU LEUVEN

Example: Sum of 2 dice throws

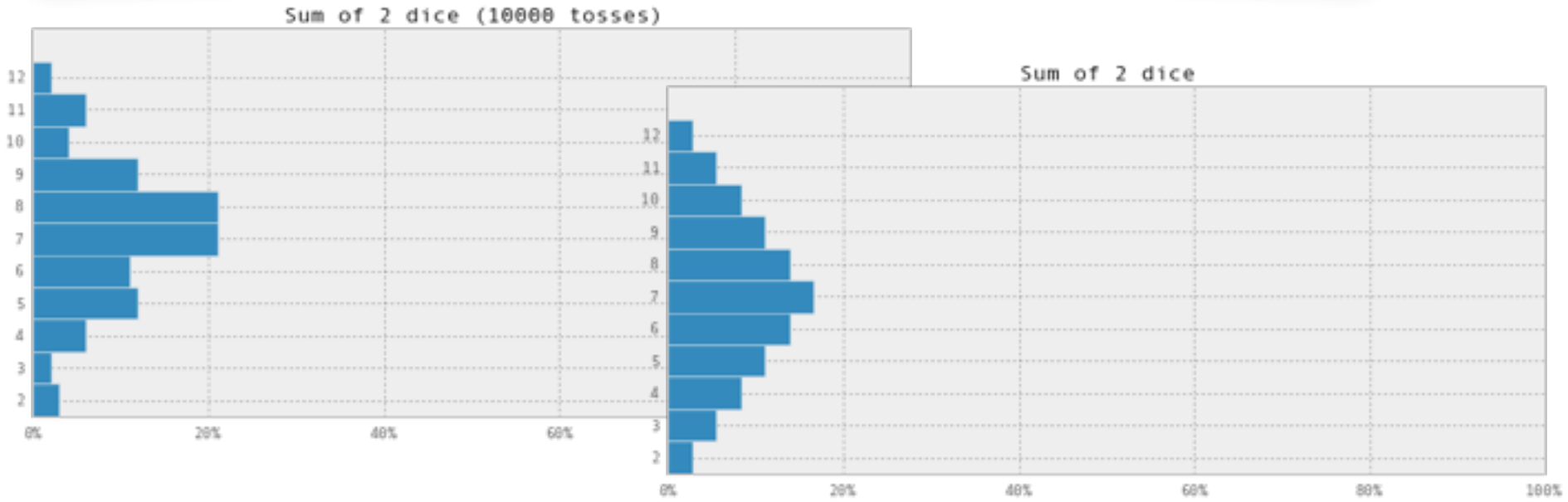
```
number_of_dice = 2

def dice():
  choice((1/6,1), (1/6,2), (1/6,3), (1/6,4), (1/6,5), (1/6,6))

def dice_sum():
  return sum([dice() for i in range(0,number_of_dice)])
```

Probability distribution

Symbolic rule



Backward reasoning: If I observe more times 4 than 10, what is the probability the die is tampered with?

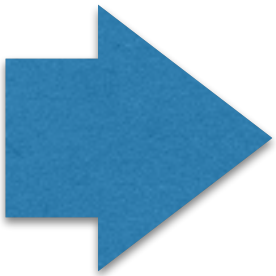
KU LEUVEN

Example: Mastermind

Write down how to place black and white pegs

```
// Code to play mastermind

…
```



Infer what the best inputs are

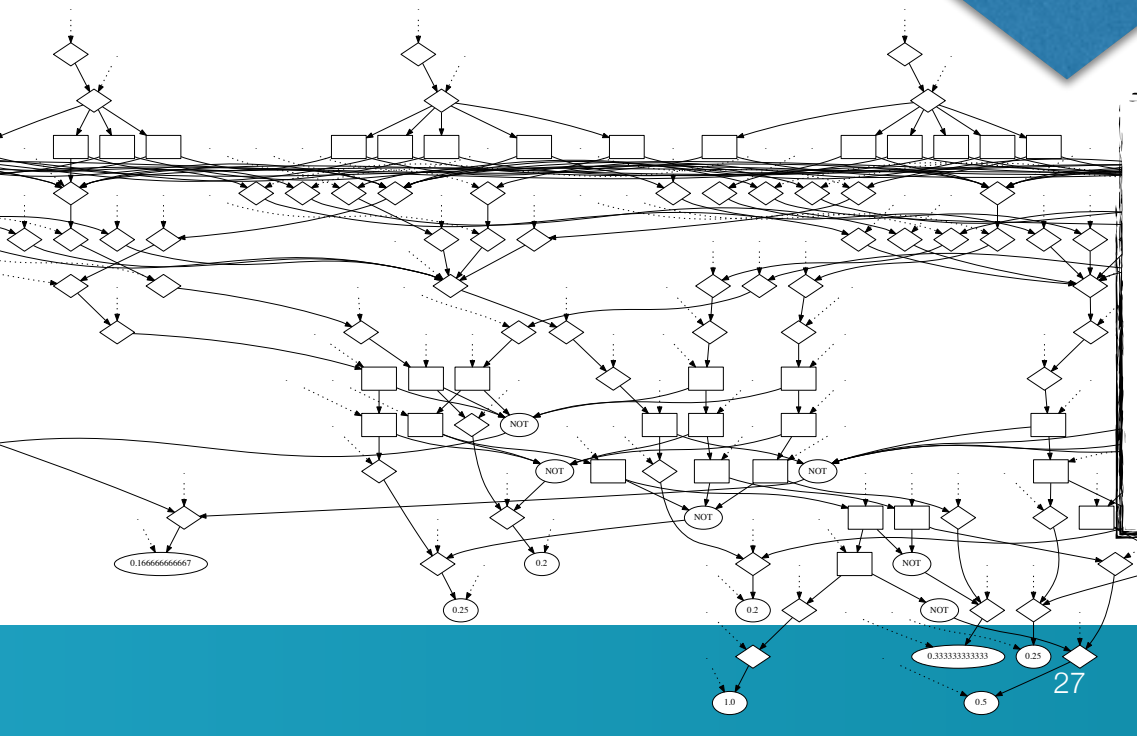
[Vlasselaer et al 2015, 2016]

KU LEUVEN

Toolchain

`.py` `.pl` `.scala` `...` + Task (inference, learning, decisions)

Probabilistic + Symbolic Reasoning



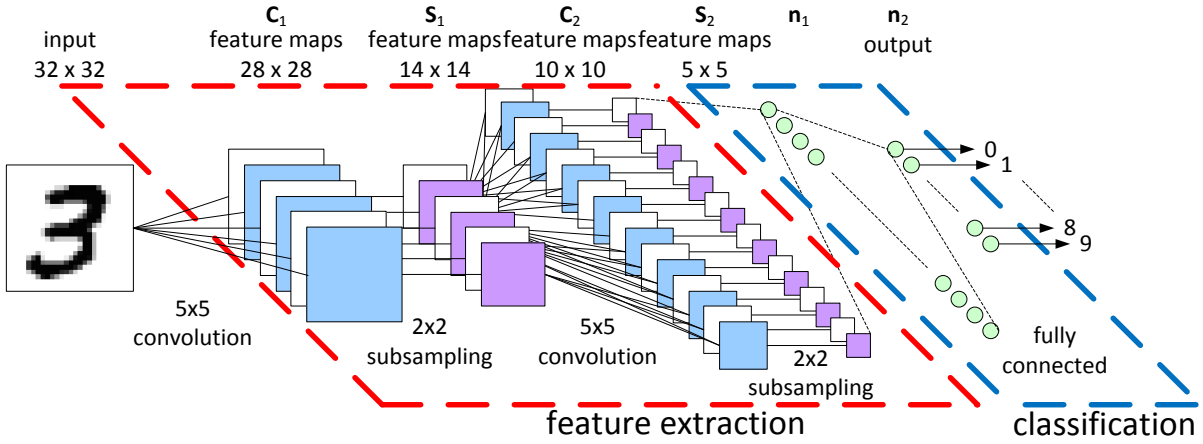
Assembly language for deterministic and probabilistic reasoning:
Graph of sums and products

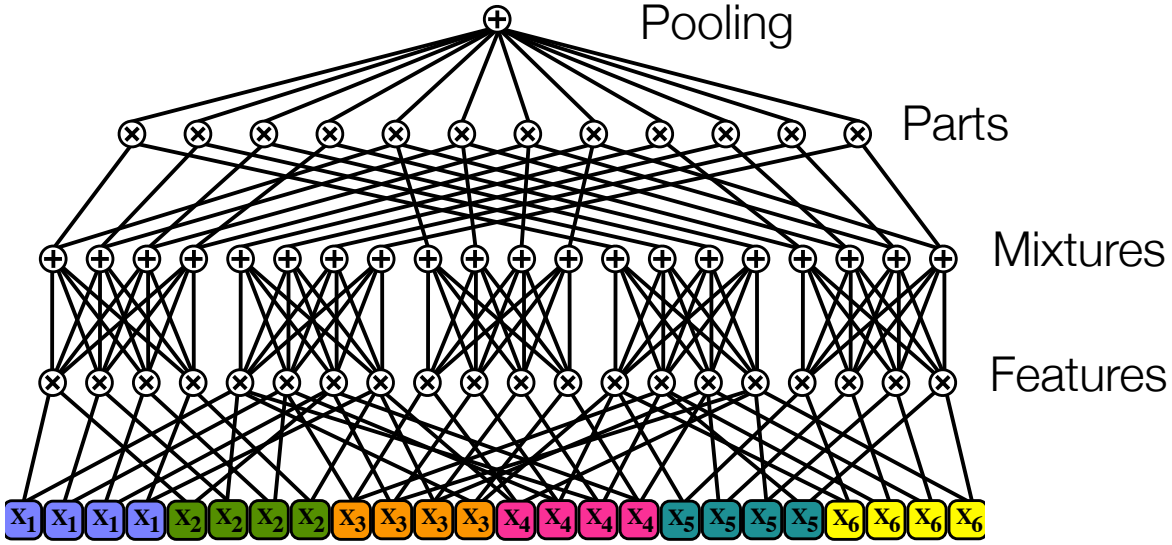
If regular: tensor/matrix multiplications
If irregular: arithmetic circuits

KU LEUVEN

Future

Bayesian+symbolic learning Connectionist learning





Integrate feature extraction into the model: learn both simultaneously

- Now: digital features
- Future: analog+digital features
- Challenge:
  - Find relevant information early in the chain

- Now: highly structured graphs
- Future: arbitrary graphs
- Challenge:
  - Find most simple graph
  - Find approximate graph
  - Find repetitive subgraphs
  - Take into account hardware architectures

Active research areas at KU Leuven

[Peemen 2015, Gens 2014]

KU LEUVEN

Conclusions Part II

- Write the model you understand
- Let the computer figure out the best ‘assembly language’
- Requires smart optimisation on all layers
  - Detect task
  - Detect problem type
  - Detect recurrent structure
  - Map to efficient hardware
  - …
- First generation probabilistic programming systems are available
  https://dtai.cs.kuleuven.be/problog

KU LEUVEN

Supported by 
EluciDATA.be
Data innovation network

wannes.meert@cs.kuleuven.be
https://dtai.cs.kuleuven.be

KU LEUVEN