

# Probabilistic Machine learning

## Introduction to probabilistic programming languages (PPLs)

Andrés Masegosa and Thomas Dyhre Nielsen

[Material partly made by Antonio Salmeron and Helge Langseth]

- **Day 1: Probabilistic Modeling**

- Introduction to probabilistic modeling and inference
- Introduction to probabilistic programming
- Probabilistic programming in Pyro

- **Day 2 : Variational Inference for Probabilistic Machine Learning**

- Introduction to Variational inference
- Variational inference in Pyro
- Variational AutoEncoders

## What is NOT Covered in This PhD Course

To set clear expectations, the following topics are **not** included in this introductory course on Probabilistic ML:

- **Advanced Bayesian Nonparametrics:** Dirichlet Process, Chinese Restaurant Process, Hierarchical Bayesian models
- **Causal Inference/Discovery:** Structural causal models, counterfactual reasoning, etc
- **Advanced Deep Probabilistic Models:** Normalizing Flows, Energy-based models, GANs, etc.
- **Diffussion Models:** Image generation, protein structures generation, etc.
- **Probabilistic Robotics and Control:** SLAM, Kalman filters in control, Probabilistic model predictive control, Reinforcement Learning, etc.

**For further learning, consider attending the** Nordic Probabilistic AI School.

To get credits for the course, you need to hand in a deliverable/assignment with the following contents:

- Describe a problem/dataset for which a probabilistic model can be used for analysis. Examples include:
  - Based on your own PhD research (preferred!)
  - Taken from a public repository such as [www.kaggle.com](http://www.kaggle.com) or <https://archive.ics.uci.edu>
- Design a probabilistic model for solving the problem.
- Implement your model in *Pyro*.
- Use the implementation to perform an analysis, for example:
  - Learn and apply a classifier
  - Perform Bayesian inference to compute distributions over quantities of interest (e.g., parameters, latent variables, etc.)
- Document your approach and findings in a short report (3-5 pages).

**Deadline: 30th April 2025**

A majestic oil painting of a raccoon Queen wearing red French royal gown. The painting is hanging on an ornate wall decorated with wallpaper.



## Imagen

unprecedented photorealism × deep level of language understanding

Google Research, Brain Team

Generating images from written descriptions

# Examples: ML for Self-driving cars



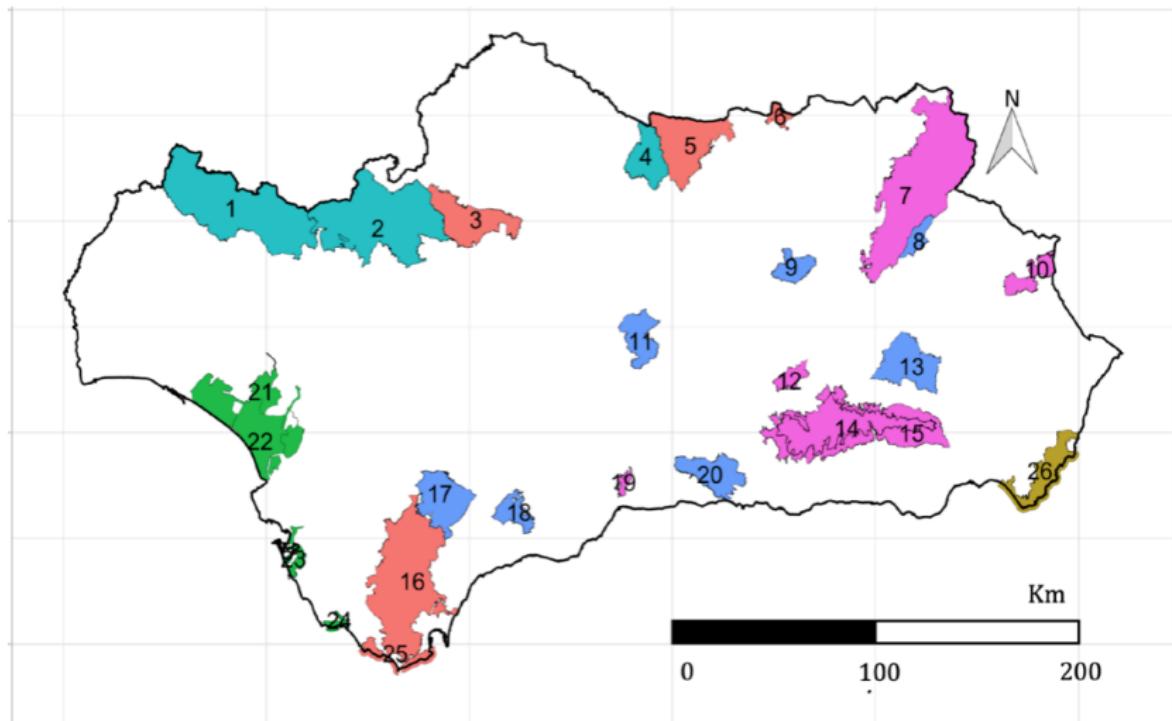
Can I predict an event in advance so that I can avoid it?

# Examples: ML for Predictive medicine



What should I do to prevent players from being injured?

## Examples: ML for Land use



Monitoring protected areas



"I always wondered how it would be if a Superior species landed on Earth and showed us How they played chess. Now I know it."

Peter Heine Nielsen.

Chess Grand Master and Magnus Carlsen's coach

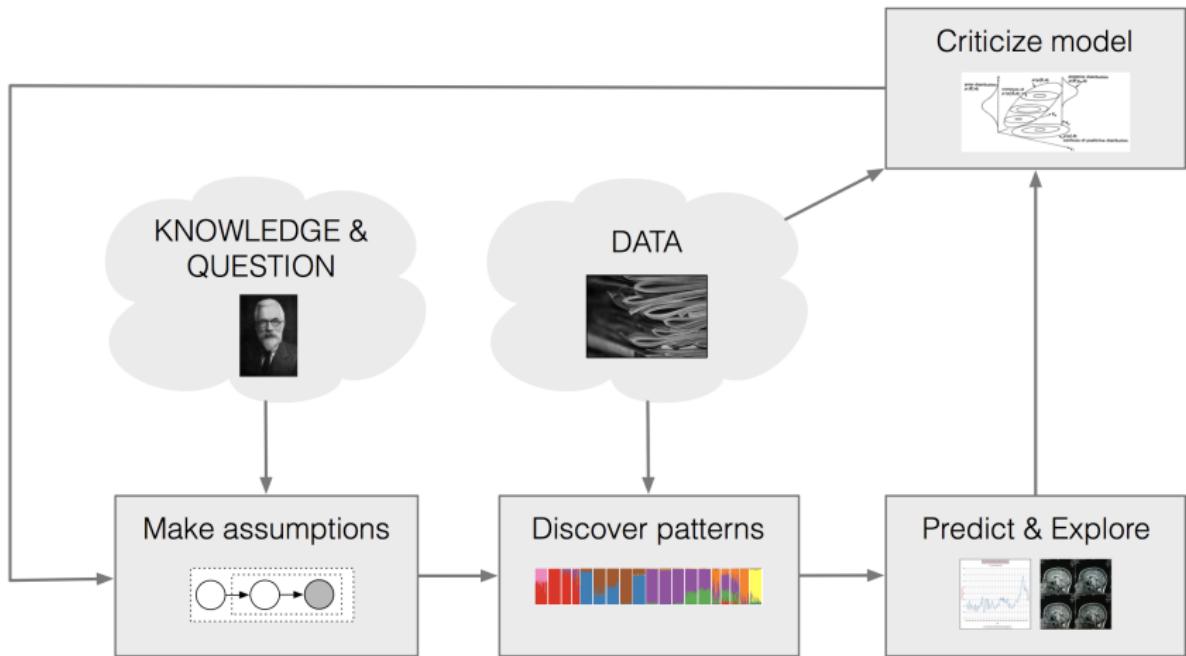
## Definition:

- **Probabilistic Machine Learning** integrates **probability theory** with **machine learning** to model **uncertainty** about predictions, parameters, and data-generating processes.
- By treating unknown quantities as **random variables**, it quantifies **confidence** in inferences and predictions - an important aspect for real-world decision-making.

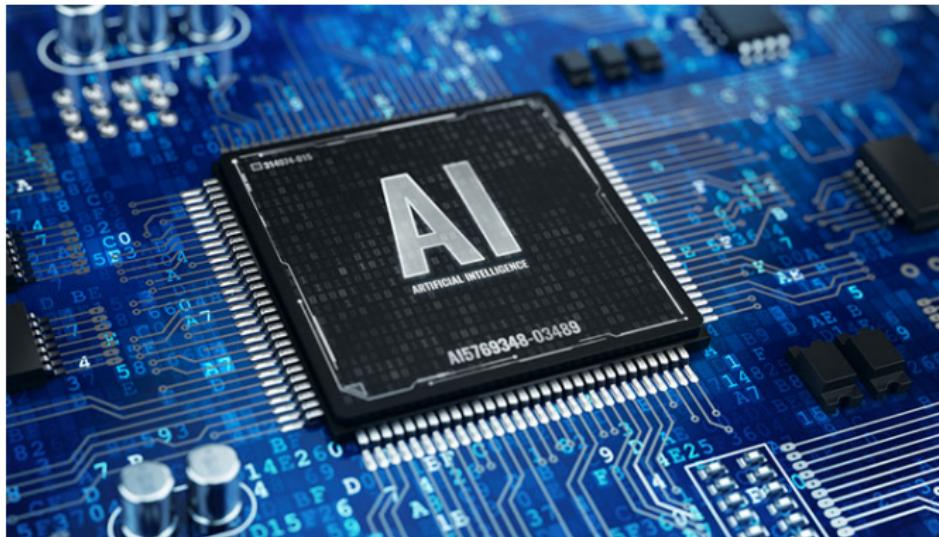
## Motivating Example: Medical Diagnosis

- Instead of making a **binary decision** (e.g., "the patient has the disease" or "the patient does not"), a **probabilistic approach** provides a **distribution over possible diagnoses**.
- For example, a **probabilistic model** determines that a patient has a **70% chance** of having a particular disease based on symptoms and test results, helping in making **informed medical decisions**.

# The probabilistic modeling cycle



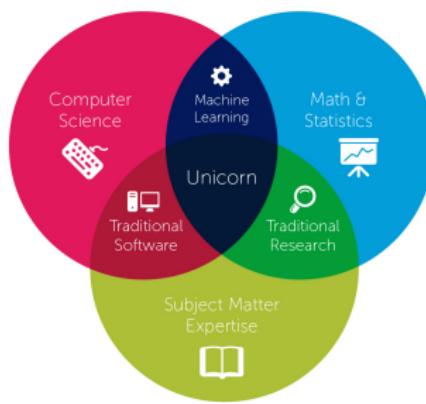
[Box, 1980; Rubin, 1984; Gelman+ 1996; Blei, 2014]



The development of **machine learning systems** requires enormous efforts.

- It can be a **highly technical** task.

## Data Science

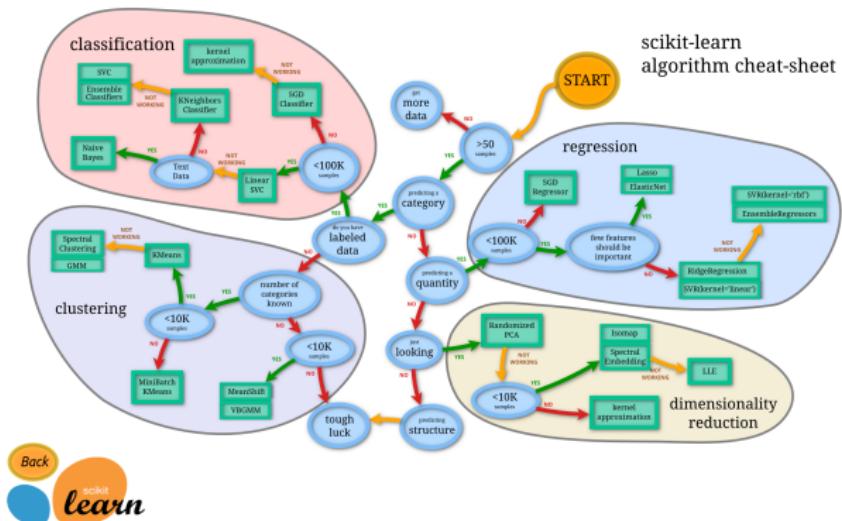


Copyright © 2014 by Steven Geringer Raleigh, NC.  
Permission is granted to use, distribute, or modify this image,  
provided that this copyright notice remains intact.

The development of **machine learning systems** requires enormous efforts.

- It can be a **highly technical** task.
- It requires of **highly qualified experts**.

# Machine Learning Systems

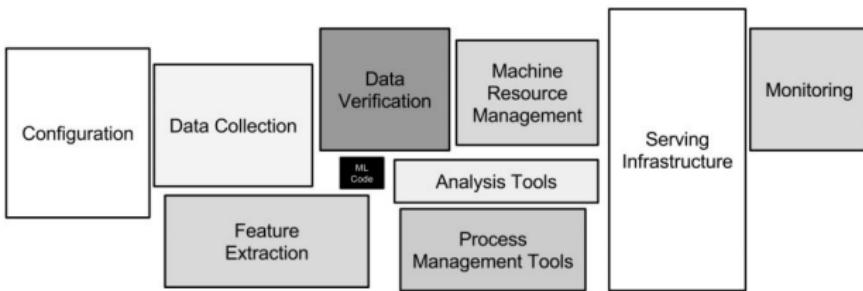


The development of **machine learning systems** requires enormous efforts.

- It can be a **highly technical** task.
- It requires of **highly qualified experts**.
- It is difficult to find the **ML model most suitable** for an application.

## Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips  
{dsculley, gholt, dg, edavydov, toddphillips}@google.com  
Google, Inc.



The development of **machine learning systems** requires enormous efforts.

- It can be a **highly technical** task.
- It requires of **highly qualified experts**.
- It is difficult to find the **ML model most suitable** for an application.
- Programming a ML model is a **complex task**; many problems are intermingled.

# Machine Learning Libraries



Claire D. Costa. Best Python Libraries for Machine Learning and Deep Learning.

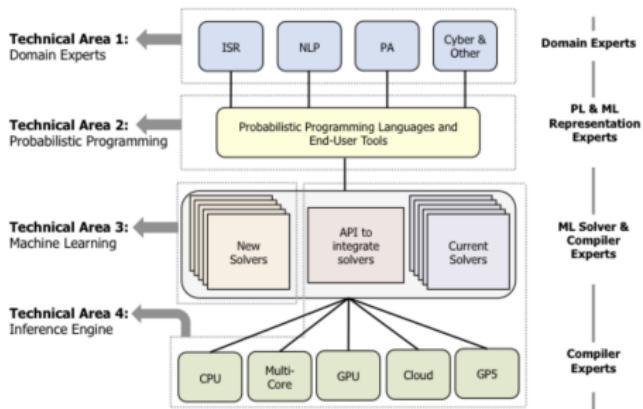
<https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c>

## Machine Learning Libraries:

- **High-quality**, well-maintained, and open-source libraries
- Provide **high-level abstractions**.
- Hide **low level details** under the hood.
- Increase the **adoption** of the technologies.

Which are the "high-level libraries" in Probabilistic AI?

# Programming languages (PPLs)

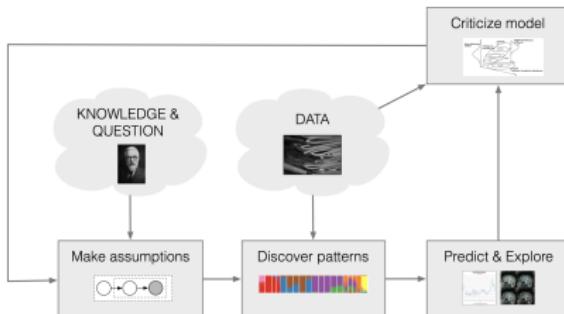


PPLs as high-level programming languages for **probabilistic ML systems**:

- Stacked architecture.
- Different Domain Experts can code their models using the same language.
- ML experts can focus on the development of ML methods/algorithms (ML solvers).
- Compiler experts can focus on running these ML solvers on specialized hardware.

# Why probabilistic programming languages (PPLs)?

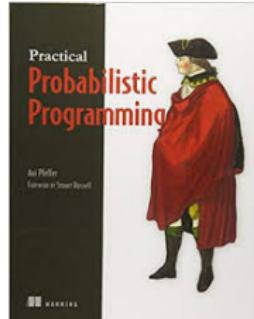
## Box's Loop



[Box, 1980; Rubin, 1984; Gelman+ 1996; Blei, 2014]

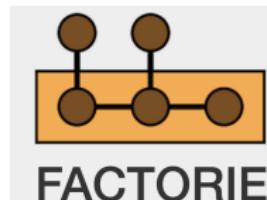
## Benefits of PPLs for developing probabilistic machine learning systems:

- Simplify probabilistic machine learning model code.
- Reduce development time and cost to encourage experimentation.
- Reduce the necessary level of expertise.
- “Democratization” of the development of probabilistic ML systems.



## 1st Generation of PPLs :

- Bugs, WinBugs, Jags, Figaro, etc.
- Turing-complete probabilistic programming languages. (i.e. they can represent any computable probability distribution).
- Inference engine based on Monte Carlo methods.
- Not able to scale to large data samples/high-dimensional models.



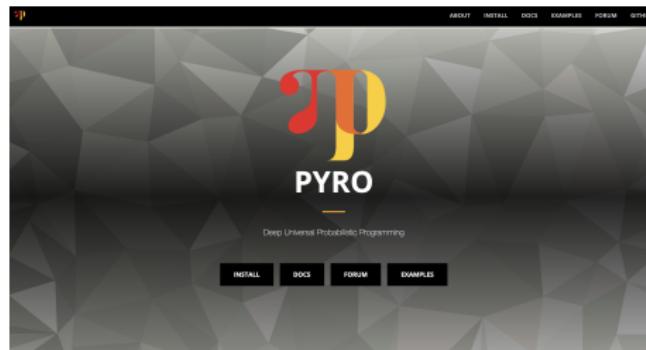
## 2nd Generation of PPLs :

- Infer.net, Factorie, Amidst, etc.
- Inference engine based on message passage algorithms and/or variational inference methods.
- Scale to large data samples/high-dimensional models.
- Restricted probabilistic model families (i.e. factor graphs, conjugate exponential family, etc.)



### 3rd Generation of PPLs :

- Pyro, Stan, PyMC, InferPy, etc.
- Black Box Variational Inference and Hamiltonian Monte-Carlo.
- Scale to large data samples/high-dimensional models.
- Turing-complete probabilistic programming languages.
  - Strong focus on probabilistic models with **deep neural networks**.
- Most rely on deep learning frameworks (Pytorch, JAX, TensorFlow, etc).
  - Specialized hardware like GPUs, TPUs, etc.
  - Automatic differentiation methods.



## Pyro's main features ([www.pyro.ai](http://www.pyro.ai)) :

- Initially developed by UBER (the car riding company).
- Community of contributors and a dedicated team at Broad Institute (US).
- Rely on Pytorch (Deep Learning Framework).
- Enable GPU acceleration and distributed learning.

# Probabilistic Graphical Models

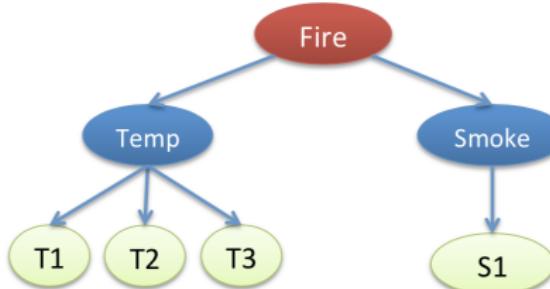
# Bayesian networks

A Bayesian Network is a **directed acyclic graph (DAG)** where:

- **Nodes represent random variables** (e.g., Fire, Temp., Smoke, and Sensors).
- **Edges represent probabilistic dependencies** between variables. If there is an edge from node *A* to node *B*, it means that *A* has a direct influence on *B*.

**Example:** The network structure captures (**causal**) **conditional relationships** between events. In our case:

- **Fire** is the root cause, influencing both **Temperature** and **Smoke**.
- **Temperature** affects three independent temperature sensors (**T1, T2, T3**).
- **Smoke** affects a single smoke sensor (**S1**).



$$p(f, t, s, t_1, t_2, t_3, s_1) = p(t_1|t)p(t_2|t)p(t_3|t)p(s_1|s)p(t|f)p(s|f)p(f)$$

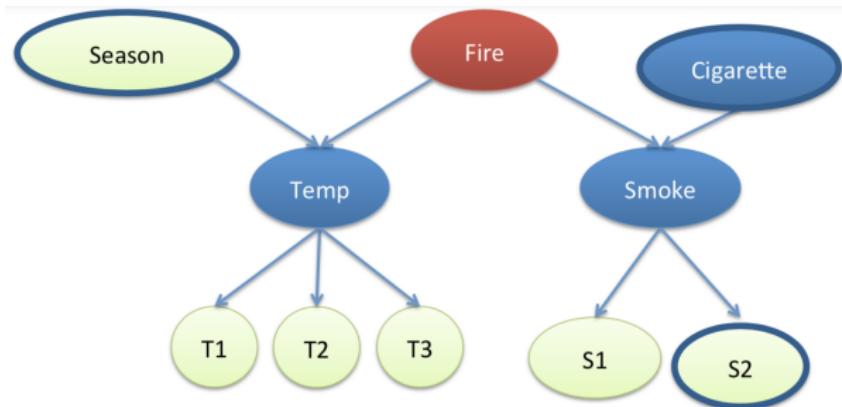
A Bayesian network over random variables  $X_1, \dots, X_n$  consists of

- A DAG  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $\mathcal{V} = \{X_1, \dots, X_n\}$
- A set of local conditional distributions  $\mathcal{P} = \{p(X_i|pa(X_i)), X_i \in \mathcal{V}\}$  where  $pa(X_i)$  denotes the parents of  $X_i$  according to  $\mathcal{E}$

Every Bayesian network encodes a joint distribution factorized as

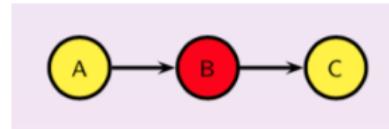
$$p(X_1, \dots, X_n) = \prod_{i=1}^n p(X_i|pa(X_i))$$

# Bayesian networks: modular structure

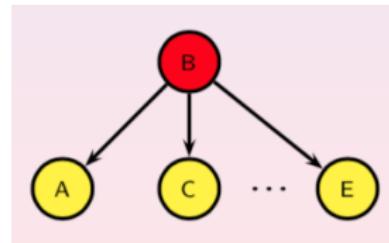


# Interpreting Bayesian network structures: $d$ -separation

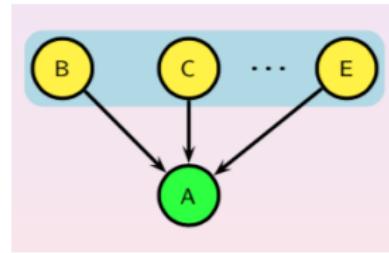
- Serial connection



- Diverging connection



- Converging connection



# Inference

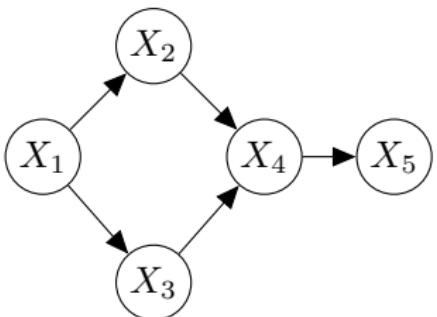
Assume a Bayesian network over variables  $\mathbf{X} = \{X_1, \dots, X_n\}$

$$\left. \begin{array}{c} \text{Bayesian network} \\ + \\ \text{Evidence } (\mathbf{X}_E) \end{array} \right\} \Rightarrow P(\mathbf{X}_I | \mathbf{X}_E)?$$

## Inference methods

- Exact
  - Brute force: compute  $P(\mathbf{X}, \mathbf{X}_E)$  and marginalize out  $\mathbf{X} \setminus \mathbf{X}_I$
  - Take advantage of the network structure
- Approximate
  - Sampling
  - Deterministic

## Exact inference: Variable elimination



- We are interested in  $X_5$
- All variables are discrete
- $E = \emptyset$

$$\begin{aligned} p(x_5) &= \sum_{x_1 \dots x_4} p(x_1, x_2, x_3, x_4, x_5) \\ &= \sum_{x_1 \dots x_4} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_4) \\ &= \sum_{x_2 \dots x_4} \sum_{x_1} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_4) \\ &= \sum_{x_2 \dots x_4} p(x_4|x_2, x_3)p(x_5|x_4) \boxed{\sum_{x_1} p(x_1)p(x_2|x_1)p(x_3|x_1)} \\ &= \sum_{x_2 \dots x_4} p(x_4|x_2, x_3)p(x_5|x_4) \textcolor{red}{h(x_2, x_3)} \end{aligned}$$

We have reached a similar problem as initially, but with one variable less.

## Input

$\mathcal{P}$ : Conditional distributions in the network

$\mathbf{X}$ : Variables in the network

$X_I$ : Target variable

$\mathbf{X}_E$ : Observed variables

$\mathbf{Y}$ : All variables in  $\mathbf{X}$  except  $X_I$  ( $\mathbf{Y} = \mathbf{X} \setminus \{X_I\}$ ).

- ① Restrict all the distributions in  $\mathcal{P}$  to the evidence  $\mathbf{X}_E = \mathbf{x}_E$
- ② For each  $Y \in \mathbf{Y}$ ,
  - ① Let  $\mathcal{P}_Y$  be the set of distributions in  $\mathcal{P}$  that contain  $Y$ .
  - ②  $q := \prod_{p \in \mathcal{P}_Y} p$ .
  - ③  $r := \sum_y q$ .
  - ④  $\mathcal{P} := \mathcal{P} \setminus \mathcal{P}_Y \cup \{r\}$ .
- ③  $p(x_I) = \prod_{p \in \mathcal{P}} p$ .
- ④ Normalize  $p$

## Input

$\mathcal{P}$ : Conditional distributions in the network

$\mathbf{X}$ : Variables in the network

$X_I$ : Target variable

$\mathbf{X}_E$ : Observed variables

$\mathbf{Y}$ : All variables in  $\mathbf{X}$  except  $X_I$  ( $\mathbf{Y} = \mathbf{X} \setminus \{X_I\}$ ).

- ① Restrict all the distributions in  $\mathcal{P}$  to the evidence  $\mathbf{X}_E = \mathbf{x}_E$
- ② For each  $Y \in \mathbf{Y}$ 
  - ① Let  $\mathcal{P}_Y$  be the set of distributions in  $\mathcal{P}$  that contain  $Y$
  - ②  $q := \prod_{p \in \mathcal{P}_Y} p \implies \text{COMPLEXITY}$
  - ③  $r := \sum_y q$
  - ④  $\mathcal{P} := \mathcal{P} \setminus \mathcal{P}_Y \cup \{r\}$
- ③  $p(x_I) = \prod_{p \in \mathcal{P}} p$
- ④ Normalize  $p$

- Product of functions raises complexity
  - Exponentially in the case of **discrete** variables
- Complexity also depends on the **elimination order**
- Representation of densities turns out to be relevant
  - **Closed-form** solutions to product and marginalization are preferable

- A Bayesian network is a representation of a joint probability distribution over  $\mathbf{X} \Rightarrow$  it describes some **population** consisting of all the possible configurations of  $\mathbf{X}$
- If the entire population was available, the **inference problem** could be solved exactly, basically by **counting cases**
- **Problem:** Population size can be huge or even infinite.
- **Monte Carlo** methods operate by drawing an artificial **sample** from it using some random mechanism
- The sample (**much smaller than the population**), is used to estimate the distribution of each variable of interest.

- A Bayesian network is a representation of a joint probability distribution over  $\mathbf{X} \Rightarrow$  it describes some **population** consisting of all the possible configurations of  $\mathbf{X}$
- If the entire population was available, the **inference problem** could be solved exactly, basically by **counting cases**
- **Problem:** Population size can be huge or even infinite.
- **Monte Carlo** methods operate by drawing an artificial **sample** from it using some random mechanism
- The sample (**much smaller than the population**), is used to estimate the distribution of each variable of interest.

Key issues in a Monte Carlo inference algorithm:

- ① The sampling mechanism
- ② The functions (estimators) which compute the probabilities from the sample

## Importance sampling. General setting

- Assume we have a random variable  $X$  with density  $p(x)$
- Importance sampling is a technique designed for estimating the expected value of a function  $f(X)$ . It is based on the following transformation:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int \frac{p(x)}{p^*(x)}f(x)p^*(x)dx = \mathbb{E}_{p^*}\left[\frac{p(x)}{p^*(x)}f(x)\right],$$

where  $p^*$  is a density function called the sampling or proposal distribution, s.t.  $p^*(x) > 0$  whenever  $p(x) > 0$ .

## Importance sampling. General setting

- Assume we have a random variable  $X$  with density  $p(x)$
- Importance sampling is a technique designed for estimating the expected value of a function  $f(X)$ . It is based on the following transformation:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int \frac{p(x)}{p^*(x)}f(x)p^*(x)dx = \mathbb{E}_{p^*}\left[\frac{p(x)}{p^*(x)}f(x)\right],$$

where  $p^*$  is a density function called the sampling or proposal distribution, s.t.  $p^*(x) > 0$  whenever  $p(x) > 0$ .

- Therefore,  $\mathbb{E}_p[f(x)]$  can be estimated by drawing a sample  $x^{(1)}, \dots, x^{(m)}$  from  $p^*$  and computing

$$\hat{\mathbb{E}}_p[f(x)] = \frac{1}{m} \sum_{j=1}^m \frac{p(x^{(j)})}{p^*(x^{(j)})} f(x^{(j)}),$$

which is specially convenient if  $p^*$  is easier to handle than  $p$

- We start with an initial configuration of the variables in the network
- A new item in the sample is generated conditional on the previous configuration
- The elements of the sample are no longer independent, instead they form a **Markov chain**

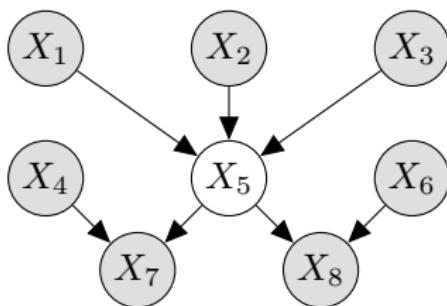
- We start with an initial configuration of the variables in the network
- A new item in the sample is generated **conditional on the previous configuration**
- The elements of the **sample** are **no longer independent**, instead they form a **Markov chain**

## Gibbs sampling

A new item in the sample is generated by simulating each variable at a time conditional on the value of the other variables sampled so far:

- Assume  $\mathbf{X} = \{X_1, \dots, X_n\}$
- Let  $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_n^{(i)})$  be the current configuration
- A new configuration,  $\mathbf{x}^{(i+1)}$  is generated by sampling each variable  $X_k$ ,  $k = 1, \dots, n$  with

$$p(x_j^{(i+1)} | x_1^{(i+1)}, \dots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \dots, x_n^{(i)})$$



$$p^*(X_5) \propto p(X_5|X_1, X_2, X_3)p(X_7|X_4, X_5)p(X_8|X_5, X_6)$$

## Problems

- Generating an initial configuration with positive probability can be hard
- The dependence between elements of the sample can make it **converge slowly**

# Learning From Data

## Thumbtack example

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the probability  $\theta$  of pin up ( $\rightsquigarrow$  model that best fits the observations/data):



### Thumbtack example

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the probability  $\theta$  of pin up ( $\rightsquigarrow$  model that best fits the observations/data):



We can measure how well a model fits the data using the likelihood:

$$\begin{aligned} p(\text{data}|\theta) &= p(\text{pin up}, \text{pin up}, \text{pin down}, \dots, \text{pin up}|\theta) \\ &= p(\text{pin up}|\theta) \cdot p(\text{pin up}|\theta) \cdot p(\text{pin down}|\theta) \cdot \dots \cdot p(\text{pin up}|\theta) \end{aligned}$$

## Thumbtack example

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the probability  $\theta$  of pin up ( $\rightsquigarrow$  model that best fits the observations/data):

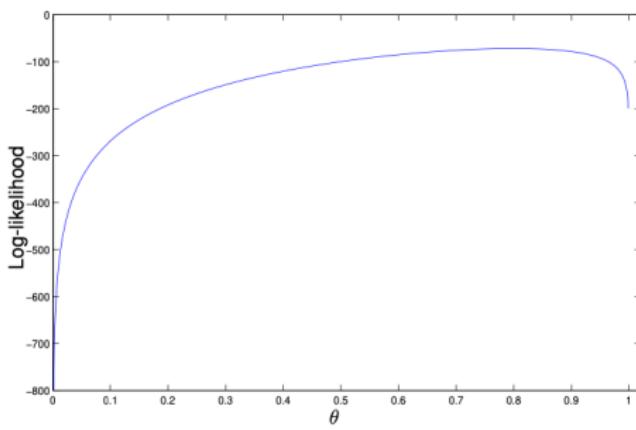


We select the parameter  $\hat{\theta}$  that maximizes:

$$\hat{\theta} = \arg \max_{\theta} p(\text{data} | \theta) \quad (\text{or } \log p(\text{data} | \theta))$$

$$= \arg \max_{\theta} \prod_{i=1}^{100} p(x_i | \theta)$$

$$= \arg \max_{\theta} \theta^{80} (1 - \theta)^{20}.$$



### Thumbtack example

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the probability  $\theta$  of pin up ( $\rightsquigarrow$  model that best fits the observations/data):



By setting:

$$\frac{d}{d\theta} \theta^{80} (1 - \theta)^{20} = 0 \quad \left( \text{or } \frac{d}{d\theta} \log(\theta^{80} (1 - \theta)^{20}) = 0 \right)$$

we get the *maximum likelihood estimate*:

$$\hat{\theta} = 0.8.$$

- A random variable  $X$
- A probability density/mass function  $p$  associated with  $X$
- A parameter  $\theta$ , whose value is fixed but unknown
- We assume  $p$  is known except for parameter  $\theta$ , and denote this fact as  $p(x|\theta)$
- For a sample  $X_1, \dots, X_n$  drawn from  $f(x|\theta)$ , the likelihood of the data is

$$p(x_1, \dots, x_n | \theta) = \prod_{i=1}^n p(x_i | \theta)$$

i.e. the joint density/mass of the sample regarded as a function of the parameter

- Learning problem is to find the parameter that maximize the log-likelihood of the data:

$$\max_{\theta} \ln p(x_1, \dots, x_n | \theta)$$

# Bayesian parameter learning

## Basis

- Parameters are modeled as random variables and information about them can be included prior to observing data
- By Bayes' rule, the prior information is combined with the likelihood, yielding a posterior distribution that is used for making inferences about the parameter

**More formally** Parameter learning amounts to calculating the posterior distribution over the parameters given the data  $\mathcal{D} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ :

$$p(\boldsymbol{\theta} | \mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} = \frac{p(\boldsymbol{\theta})p(\mathcal{D} | \boldsymbol{\theta})}{p(\mathcal{D})}$$

$$p(\mathcal{D} | \boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\theta}) \quad [\text{Assuming independently sampled data}]$$

## Posterior predictive distribution

For making predictions about future cases we use

$$p(\mathbf{X} | \mathcal{D}) = \int_{\boldsymbol{\theta}} p(\mathbf{X} | \boldsymbol{\theta})p(\boldsymbol{\theta} | \mathcal{D})d\boldsymbol{\theta}$$

## Basis

- Parameters are modeled as random variables and information about them can be included prior to observing data
- By Bayes' rule, the prior information is combined with the likelihood, yielding a posterior distribution that is used for making inferences about the parameter

**More formally** Parameter learning amounts to calculating the posterior distribution over the parameters given the data  $\mathcal{D} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ :

$$p(\boldsymbol{\theta} | \mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} = \frac{p(\boldsymbol{\theta})p(\mathcal{D} | \boldsymbol{\theta})}{p(\mathcal{D})}$$

$$p(\mathcal{D} | \boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\theta}) \quad [\text{Assuming independently sampled data}]$$

## Basis

- Parameters are modeled as random variables and information about them can be included prior to observing data
- By Bayes' rule, the prior information is combined with the likelihood, yielding a posterior distribution that is used for making inferences about the parameter

**More formally** Parameter learning amounts to calculating the posterior distribution over the parameters given the data  $\mathcal{D} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ :

$$p(\boldsymbol{\theta} | \mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} = \frac{p(\boldsymbol{\theta})p(\mathcal{D} | \boldsymbol{\theta})}{p(\mathcal{D})}$$

$$p(\mathcal{D} | \boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\theta}) \quad [\text{Assuming independently sampled data}]$$

## Posterior predictive distribution

For making predictions about future cases we use

$$p(\mathbf{X} | \mathcal{D}) = \int_{\boldsymbol{\theta}} p(\mathbf{X} | \boldsymbol{\theta})p(\boldsymbol{\theta} | \mathcal{D})d\boldsymbol{\theta}$$

## Model and data

Consider the variables  $X_1, X_2, \dots, X_N$  with  $P(X_i|\theta) = \theta$ :

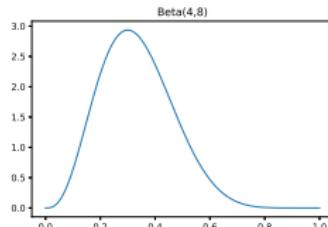
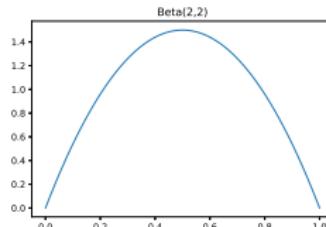
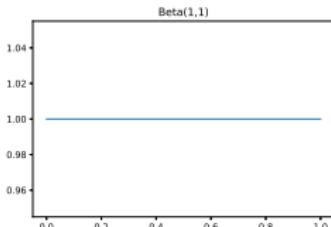
- $X_i = 1$  represents pin up and  $X_i = 0$  represents pin down
- $\theta \in [0, 1]$  is the probability of pin up

### Prior

The beta distribution  $\text{Beta}(\theta | a, b)$  over  $\theta$ :

$$p(\theta) \propto \theta^{a-1} (1 - \theta)^{b-1},$$

where  $a$  and  $b$  are hyper parameters.



## Model and data

Suppose that  $X_1, X_2, \dots, X_N \sim \text{Ber}(\theta)$

- $N_1$  are the number of pin up
- $N_0$  are the number of pin down ( $N = N_0 + N_1$ )

## Model and data

Suppose that  $X_1, X_2, \dots, X_N \sim \text{Ber}(\theta)$

- $N_1$  are the number of pin up
- $N_0$  are the number of pin down ( $N = N_0 + N_1$ )

## Prior

The beta distribution  $\text{Beta}(\theta | a, b)$  over  $\theta$ :

$$p(\theta) \propto \theta^{a-1} (1 - \theta)^{b-1}$$

## Model and data

Suppose that  $X_1, X_2, \dots, X_N \sim \text{Ber}(\theta)$

- $N_1$  are the number of pin up
- $N_0$  are the number of pin down ( $N = N_0 + N_1$ )

## Prior

The beta distribution  $\text{Beta}(\theta | a, b)$  over  $\theta$ :

$$p(\theta) \propto \theta^{a-1} (1 - \theta)^{b-1}$$

## Posterior

$$p(\theta | \mathcal{D}) = \text{Beta}(\theta | a + N_1, b + N_0)$$

## Model and data

Suppose that  $X_1, X_2, \dots, X_N \sim \text{Ber}(\theta)$

- $N_1$  are the number of pin up
- $N_0$  are the number of pin down ( $N = N_0 + N_1$ )

## Prior

The beta distribution  $\text{Beta}(\theta | a, b)$  over  $\theta$ :

$$p(\theta) \propto \theta^{a-1}(1-\theta)^{b-1}$$

## Posterior

$$p(\theta | \mathcal{D}) = \text{Beta}(\theta | a + N_1, b + N_0)$$

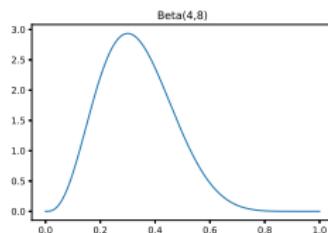
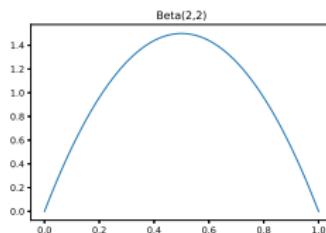
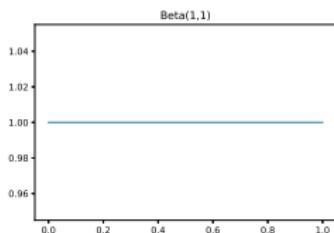
## Posterior predictive distribution

$$\begin{aligned} p(X_{N+1} = 1 | \mathcal{D}) &= \int_0^1 p(X_{N+1} = 1 | \theta) p(\theta | \mathcal{D}) d\theta \\ &= \int_0^1 \theta \text{Beta}(\theta | a + N_1, b + N_0) d\theta = \frac{a + N_1}{a + b + N} \end{aligned}$$

# Beta posterior: example

Assume that  $N_0 = 5$  and  $N_1 = 9$ .

## Prior



## Posterior

