CS6462

Probabilistic and Explainable AI

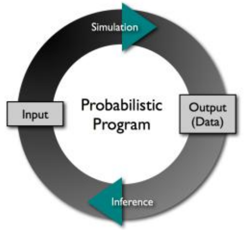
Lesson 8

Principles of Probabilistic Programming

by Emil Vassev

February 7, 2022

Bayesian Inference



Inference:

- use statistics to deduce properties about a probability distribution from data

Bayesian Inference:

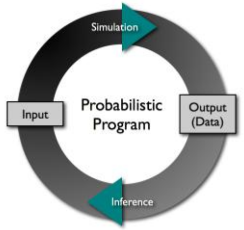
- using Bayes' Theorem to do statistics inference
- Bayes' Theorem works not on events but on distributions:
  - ***Θ*** – set of parameters
  - *prior distribution* ***P(Θ)*** – distribution of our belief about the true value of ***Θ***
  - *posterior distribution* ***P(Θ|data)*** - distribution of our belief about ***Θ*** after we have taken the observed data into account
  - *likelihood distribution* ***P(data|Θ)*** - measures the degree to which data supports ***Θ***

$$P(\Theta|data) = \frac{P(data|\Theta)*P(\Theta)}{P(data)}; \quad L(\Theta|data) = P(data|\Theta)$$

normalizing factor

What kind of probability distributions should we use to model a probability?

Bayesian Inference (cont.)



Steps:

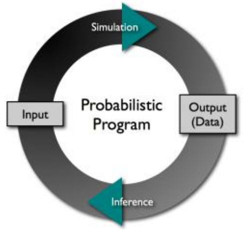
- Step 1. **[Prior]** Choose a PDF to model your parameters ***Θ*** and the prior distribution ***P(Θ)***. This is your best guess about parameters before seeing ***data***.
- Step 2. **[Likelihood]** Choose a PDF for ***P(data|Θ)***. Basically you are modeling how ***data*** will look like given the parameters ***Θ***.
- Step 3. **[Posterior]** Calculate the posterior distribution ***P(Θ|data)*** and pick up the ***Θ*** that has the highest ***P(Θ|data)*.**

  And the posterior becomes the new prior. Repeat step 3 as you get more data.

Probability Distributions:

- *Normal distribution* - has two parameters: mean ***μ*** and standard deviation ***σ***
- *Beta distribution*
- *Poisson distribution*

Bayesian Inference (cont.)



Normalizing factor ***P(data)***:

Why *P(data)* is important?

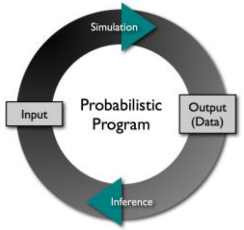
- the probability number that comes out is a normalizing factor
- recall: a necessary conditions for a probability distribution - the sum of all possible outcomes ***S*** of an event is equal to 1, i.e., ***P(S) = 1***

  example: total probability of rolling a 1, 2, 3, 4, 5 or 6 on a die is equal to 1
- the normalizing factor makes sure that the resulting posterior distribution is a true probability distribution by ensuring that the sum of the distribution is equal to 1

Ignoring ***P(data):***

- could be ignored when the focus is on the peak of the distribution, regardless of whether the distribution is normalized or not

Bayesian Inference (cont.)



Posterior:

- The goal of Bayesian inference is to update our prior beliefs ***P(Θ)*** by taking into account ***data*** that we observe.
- If we assume that in any particular inference problem, data is fixed, we are interested in only the terms which are functions of ***Θ***, i.e., ignore ***P(data)***:

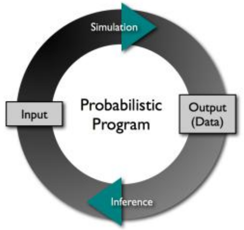
  $\boldsymbol{P(\Theta \mid data) \sim P(data \mid \Theta) * P(\Theta)}$, or:

  Posterior ~ Likelihood × Prior (~ posterior prob. distr., e.g., using Poisson distr.)

  asymptotic (approximately equal)

- Our final beliefs about ***Θ*** combine both the relevant information we had a priori and the knowledge we gained a posteriori by observing data

Bayesian Inference (cont.)



Example: The probability of a certain medical test being positive is 90%, if a patient has disease . 1% of the population has the disease, and the test records a false positive 5% of the time. If you receive a positive test, what is your probability of having D?

Likelihood function   Prior probability

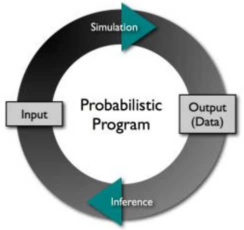
Posterior probability

$$P\,(D|+) = \frac{P(+|D)\ *\ P(D)}{P(+)} = \frac{P(+|D) * P(D)}{P(+|D) * P(D) + P(+|D^c) * P(D^c)}$$

Result:

- *posterior probability* of having the disease ***P (Disease|+)*** given that the test was positive depends on the *prior probability* of the disease ***P(Disease)***
- ***P(+|D)=0.9***, ***P(D)=0.01***, ***P(+|D^c^)=0.05***, ***P(D|+) = ?***
- Substituting in the numbers : ***P(D|+) = 0.15***
- Final beliefs: ***Posterior ~ Likelihood × Prior = 0.9 * 0.01 = 0.009***

Bayes’ Theorem & Probabilistic Programming



Bayesian Inference:

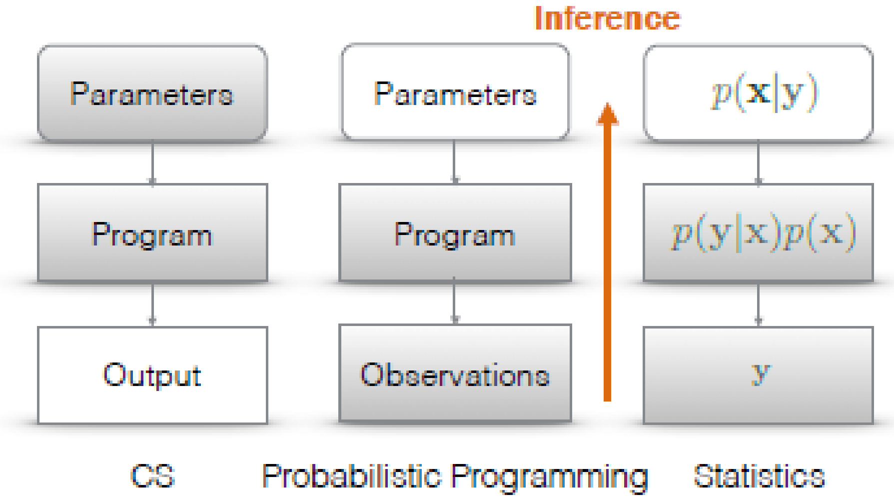
- Bayes' Theorem is used by statistical inference to update the probability for a hypothesis to cope with new information that has become available

Probabilistic Programming:

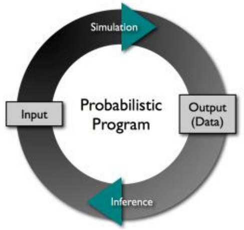
- about automating Bayesian inference
- syntax and semantics for languages that denote *conditional inference problems*
- formal semantics for building evaluators of models and applications from machine learning using the inference algorithms and theory from statistics

Example:

- ***y*** – data (observations) output
- ***p(y|x), p(x)*** – probabilistic models (data & parameters)
- ***p(x|y)*** – posterior distribution result of inference techniques



Probabilistic Programming



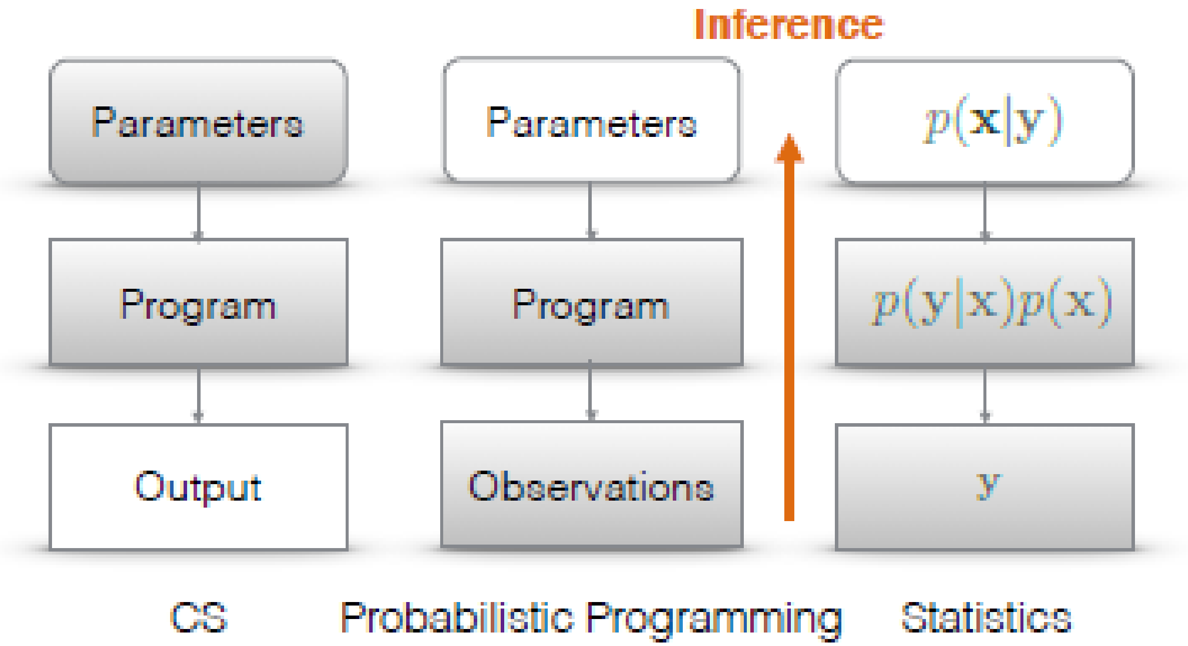
Example: disease test

- ***y (data):*** *%* population has the disease, test records
- ***p(y|x), p(x)*** **(**probabilistic models) - use Poisson distribution to build models for ***P(+|D) and P(D)***
- ***p(x|y)*** (posterior distribution) –

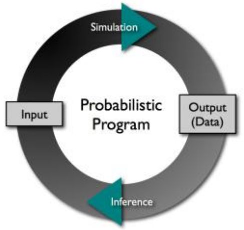
  Posterior ~ Likelihood × Prior

Challenges:

- building the probabilistic modelss computational: though Bayes'
- implementing an algorithm following the theoretical model, so to computationally characterize the posterior distribution



Probabilistic Programming (cont.)

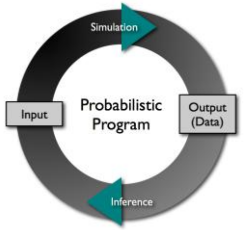


Features of probabilistic programs:

- probabilistic programs are usual functional or imperative programs with two added constructs [Gordon et al.]:
  - ability to draw values at random from distributions
  - ability to *condition* values of variables in a program via observations
- a probabilistic program simultaneously denotes a joint and conditional distribution:
  - indicates the conditioning
  - indicates what random variable values will be observed
- probabilistic programming languages support syntactic constructs for conditioning and evaluators that implement conditioning

Gordon, A. D., T. A. Henzinger, A. V. Nori, and S. K. Rajamani (2014), 'Probabilistic programming'. In: *Proceedings of the on Future of Software Engineering. pp. 167–181.*

Existing Probabilistic Languages



*By research communities *:*

- Anglican
- BLOG
- BayesDB
- Venture
- Probabilistic-C
- Church
- WebPPL
- CPProb
- Augur
- FOPPL
- Hakaru

Recursive Multi-Agent Reasoning
probabilistic programming used to study the mutually recursive reasoning among multiple agents

** "An Introduction to Probabilistic Programming", book by J.W. van de Meent, B. Paige, H. Yang, F. Wood]*

Probabilistic Program - Example

Example: reasoning about the bias of a coin

data - outcome heads or tails of one coin flip

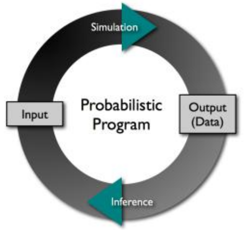
model - Beta-Bernoulli model - a coin output and its bias are generated according to the model and then the coin flip outcome is observed and analyzed under this model

- ***y :*** data **-** heads ***a*** and tails ***b***
- ***p(x) ~ Beta(a, b)*** – prior probability distribution function
- ***p(y|x) = p(a,b|x) ~ Bernoulli(x)*** – likelihood function
- ***p(x|y) = p(x|a, b) ~ Bernoulli(x) * Beta(a, b) -*** posterior distribution

```
(let [prior (beta a b)
      x (sample prior)
      likelihood (bernoulli x)
      y 1]
  (observe likelihood y)
  x)
```

prior (beta a b) - function call that creates a prior distribution (heads and tails)
x (sample prior) - function call that creates a sample of x [0..1]
likelihood (bernouli x) – the likelihood function
y 1 – ***y*** is assigned value 1 (heads)
(***observe likelihood y)*** – posterior ***p(y|likelihood)***

Summary



Bayesian inference:

- using Bayes' Theorem to do statistics inference
- update our prior beliefs by taking into account data that we observe

  1) Posterior ~ Likelihood × Prior (~ posterior prob. distr., e.g., using Poisson distr.)

  2) posterior becomes the new prior

Probabilistic Programming:

- about automating Bayesian inference
- syntax and semantics for languages that denote *conditional inference problems*
- formal semantics for building evaluators of models and applications from machine learning using the inference algorithms and theory from statistics

Challenges

Next Lesson – Lesson: First-Order Probabilistic Programming Language (FOPPL)

Thank You!

Questions?