Probabilistic Programming With PyMC3

John Kimball

9/26/19

Overview

- Why do we need Probabilistic Programming (PP)?
- What is PP?
- Four simple examples of PP using PyMC3
  - A bit of theory as needed
- How PP is being used today
- How to get started

The goal is a high-level overview, not a detailed tutorial

Why Do We Need Probabilistic Programming (PP)?

- An explosion of data
  - Web commerce, Internet of Things, human genome, etc.
  - Often the data is uncertain
- Enormous value is theoretically available from the data
- Very difficult to interpret the data (draw inferences)
  - Advanced and specialized mathematical knowledge is required
  - Inference methods are unique to each problem
  - It is difficult to experiment and make changes to the inference systems
- See video:
  - https://www.oreilly.com/ideas/probabilistic-programming

Probabilistic Programming to the Rescue

- Two major components of a PP system:
  - **The modeling system is a high level programming language (e.g. Python)**
    - All the flexibility of the host language for expressing data structures
    - **Probability distributions as high level constructs**
  - **An automatic general purpose inference engine**
    - Independent from the model
    - Enables "running the model backward" to infer causes from data
- The PP system provides:
  - High-level probability constructs => shorter code, reduced development time
  - Host language flexibility => more sophisticated models
  - Automatic inference => greatly reduced statistical expertise, rapid experimentation
- Goal:
  - Domain experts can learn from their own data
  - Statistics/Machine Learning experts can increase their power

Component 1: Probabilistic Modeling in PyMC3

- Model setup
- Probability distributions
  - Continuous: Normal Distribution
  - Discrete: Bernoulli Distribution
- Sampling data from the model
- Inspecting model output using Arviz
- See Jupyter Notebook:
  - Part1-ProbabilisticModeling.ipynb

Running Models in Reverse

- Bayes' Theorem:

$$\boxed{P(H|D) = \frac{P(D|H)P(H)}{P(D)}}$$

$P(H|D)$ means the probability of H given D

- Bayes' Theorem gives us a way of updating a hypothesis (H) using data (D)

Example: Ferrari Detector

- We are designing a Ferrari detector.
- The sensor works by detecting the color of the car.
- If our sensor detects red, how likely is it that the car is a Ferrari?
- We know that
  - 50% of Ferraris are red
  - 5% of all cars are red
  - .02% of all cars are Ferraris

Let:
R be True if the car is red (**D**ata)
F be True if the car is a Ferrari (**H**ypothesis)

$$P(F|R) = \frac{P(R|F)P(F)}{P(R)} = \frac{(0.5)(0.0002)}{0.05} = 0.2\%$$

- Prior to any evidence, there is a 0.02% probability of a car being Ferrari
- The Likelihood of data (Red) given the hypothesis (Ferrari)
- After (Posterior to) the evidence that the car is red, there is a 0.2% probability that the car is a Ferrari

Terminology and Modeling Procedure[1]

1. Define the **likelihood function**

2. Define the **prior(s)**

3. Run the inference engine to get the **posterior** (updated) probability distribution

$$P(F|R) = \frac{P(R|F)P(F)}{P(R)} = \frac{(0.5)(0.0002)}{0.05} = 0.2\%$$

We'll treat this a normalization factor and ignore it

- Note: in most models the the priors, likelihoods and posterior will be in terms of *probability distributions*

1. Martin, Osvaldo, *Bayesian Analysis with Python*, 2nd. Edition, Packt Publishing, 2018.

Coin Toss

- The "Hello, World!" of probabilistic programming
- We'll:
  - Generate some data with known bias (outside of PyMC3)
  - Create a PyMC3 model of the coin toss process
  - Pass the data to our model
  - Attempt to determine the bias in the coin
- See Jupyter Notebook:
  - Part2-FairCoin.ipynb

A/B Testing

- Given:
  - Two versions of a website (A and B) differing in a single feature
    - We expect B to be an improvement on the original A
- Then:
  - Randomly present A and B to different groups
  - Observe which users purchase the product
  - Use PP to infer the effectiveness of the change from A to B
- See Jupyter Notebook:
  - Part3-ABTesting.ipynb

Space Shuttle Challenger Disaster

- Given:
  - O-ring damage and temperature for 23 launches
- Then:
  - Model the probability of O-Ring damage vs. temperature
  - Infer model parameters from data
  - Display the expected probability of damage
    - Include 95% confidence intervals
  - Evaluate probability of damage at Challenger launch (31F)
- See Jupyter Notebook:
  - Part4-Challenger.ipynb

Inference Algorithms

- Non-Markovian
  - Brute force: Grid method
    - Define a grid of points at which to evaluate each parameter
    - Multiply the prior and the likelihood at each point to define the posterior at that point
    - Very inefficient; spends much time in areas that don't matter
  - Automatic Differentiation Variational Inference (ADVI)
    - Approximates posterior as a simpler function
    - Becomes an optimization problem
    - Uses automatic differentiation gradients for efficient optimization
- Markov Chain Monte Carlo (MCMC)
  - Evaluates the prior and likelihood function pointwise (like grid method)
  - Much more efficient than grid method, since spends time in most important areas

See Martin, Osvaldo, *Bayesian Analysis with Python: Introduction to Statistical Modeling and Probabilistic Programming Using PyMC3 and ArviZ, Chapter 8*

Other Areas Probabilistic Programming is Being Used

- PyMC3
  - According to Thomas Wiecki, A | B testing is the most common use case
  - Quantitative Finance (e.g. Quantopian to evaluate trading algorithms)
  - Earthquake Analysis (e.g. BEAT https://github.com/hvasbath/beat)
  - Supply Chain Optimization (https://twiecki.io/blog/2019/01/14/supply_chain/
- Probabilistic Programming in General
  - Astrophysics: to infer black hole parameters from LIGO data (2017 Nobel Prize in Physics)
  - Search and Rescue
    - https://en.wikipedia.org/wiki/Search_and_Rescue_Optimal_Planning_System
  - Drug Discovery
    - "…allows for predicting complex phenotypes by casting light on the causal molecular underpinnings of disease. This method combines the use of **ensemble deep learning with probabilistic programming that applies Bayesian models to test the causal dependencies** of millions of possible interactions between proteins to elucidate the core biological network connecting the observed phenotype with the mechanism of disease."
    - https://www.wuxinextcode.com/genomic-insights/a-i-breakthroughs-in-drug-rd-probabilistic-programming-biological-context-and-quantum-a-i/

There are countless other applications of probabilistic programming

More Python Libraries for Probabilistic Programming

- PyStan – Python interface to the popular Stan PP platform
- Edward – "a Python library for probabilistic modeling, inference, and criticism. … Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.
- Pyro (Uber AI Labs) – "Pyro is a universal probabilistic programming language (PPL) written in Python … enables flexible and expressive deep probabilistic modeling, unifying the best of modern deep learning and Bayesian modeling.
- MIT Probabilistic Computing Project - http://probcomp.csail.mit.edu/software/
- PyCBC – specialized for LIGO, which won 2017 Nobel Prize for Physics
  - "PyCBC was used in the first direct detection of gravitational waves (GW150914) by LIGO and is used in the ongoing analysis of LIGO and Virgo data." http://pycbc.org/pycbc/latest/html/
  - MCMC Bayesian inference like PyMC3 http://pycbc.org/pycbc/latest/html/inference.html

References

- What is probabilistic programming?
  - https://www.oreilly.com/ideas/probabilistic-programming
- PyMC3 Online Documentation
  - https://pymc3.readthedocs.io/en/latest/index.html
- Martin, Osvaldo, *Bayesian Analysis with Python: Introduction to Statistical Modeling and Probabilistic Programming Using PyMC3 and ArviZ*, 2nd edition, Packt Publishing, 2018.
- Davidson-Pilon, Cameron, *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference*, Addison Wesley, 2016.
  - Also available online:
    - https://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/
    - https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers
- The Python Podcast.__init__ (April 29, 2019 with guest Thomas Wiecki)
  - https://www.podcastinit.com/feed/mp3/
- Pfeffer, Avi, Practical Probabilistic Programming, Manning Publications, 2016.
  - Uses Figaro, a Scala-based PP system.

Additional Learning Resources

- Kruschke, John K., *Doing Bayesian Data Analysis: A Tutorial with R and BUGS*, Academic Press, 2011.
  - PyMC3 port of examples:
    - https://github.com/aloctavodia/Doing_bayesian_data_analysis
- Thomas Wiecki (PyMC3 developer) blog:
  - https://twiecki.io
- McGrayne, Sharon Bertsch, *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy*, Yale University Press, 2011.
- VanderPlas, Jake, *Frequentism and Bayesianism: A Practical Introduction*
  - http://jakevdp.github.io/blog/2014/03/11/frequentism-and-bayesianism-a-practical-intro

How to install PyMC3 and ArviZ

- This presentation (including Jupyter Notebooks) is on GitHub:
  - https://github.com/NH-Python/pymc3
  - **Installation instructions are given in the README.md file**
    - Installing using PyMC3 using the official directions can be difficult
    - Using the Docker image used to run the Jupyter Notebooks during the presentation is much easier
- Getting started with Python Scientific/Data Science stacks:
  - If working primarily in a Jupyter Notebook, the Docker image referenced above includes the major scientific/data science Python (and R and Julia) tools.
  - If writing code using an editor/IDE, the Anaconda Distribution is recommended.
    - https://www.anaconda.com/distribution/
    - PyMC3 is an optional package in the Anaconda Distribution

Additional Information

The Python Data Science Stack

- A great overview by Jake VanderPlas (though a bit dated)
  - https://speakerdeck.com/jakevdp/pythons-data-science-stack-jsm-2016
- Some other libraries used by PyMC3 and this presentation:
  - Theano: a Python library that lets you to define, optimize, and evaluate mathematical expressions
    - Automatically generated C code
    - Automatic differentiation of mathematical expressions
    - PyMC4 will use TensorFlow Probability instead of Theano
    - Theano is being maintained by the PyMC team
  - Arviz: a Python package for exploratory analysis of Bayesian models.
    - Much easier plotting that Matplotlib

Examples of ML Problems in Probabilistic Programming

- Bayesian Linear Regression
- Logistic Regression
- Naïve Bayes
- K-Means Clustering
- Latent Dirichlet Allocation (LDA)
- Correlated Topic Models (CTM)
- Autoregressive Integrated Moving Average (ARIMA)
- Hidden Markov Models
- Matrix Factorization
- Sparsity and Sparse Bayes
- Conditional Random Fields

PP makes many machine learning tasks available to a broad audience

https://www.zinkov.com/posts/2012-06-27-why-prob-programming-matters/