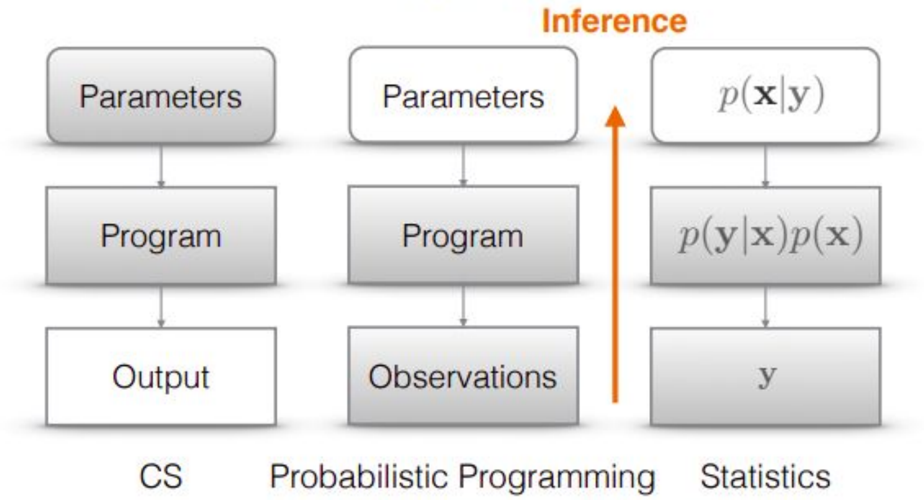
Probabilistic Programming with Pyro

Introduction

What is **Probabilistic Programming** (PP)?

- A common misconception: PP is **not** about writing software that behaves probabilistically.
- Rather, PP is a method of **Bayesian statistical modelling** – based on representing causal models as executable programs.

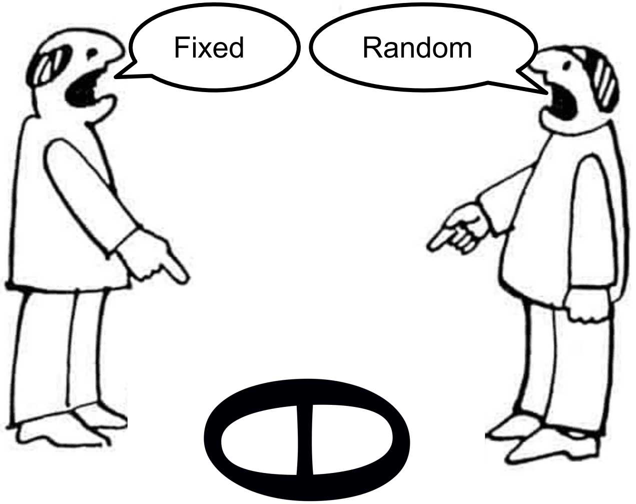


https://towardsdatascience.com/intro-to-probabilistic-programming-b47c4e926ec5

Background: **Frequentist** vs. **Bayesian** statistics

Frequentists view the:
- true parameters θ as fixed
- data **X** as **random**
- estimators f(**X**) as **random**

Bayesians view the:
- true parameters **θ** as **random**
- data X as fixed
- estimators f(X) as fixed



Background: **Frequentist** vs. **Bayesian** statistics

Frequentists view the:

- true parameters $\underline{\theta}$ as fixed
- data $\mathbf{X}$ as **random**
- estimators $f(\mathbf{X})$ as **random**

Confidence Interval:

$P\{\, l(\mathbf{X}) < \underline{\theta} < u(\mathbf{X}) \,\} = 95\%$

*"When sampling **X** from a population many times, the estimated lower bound l(**X**) and upper bound u(**X**) will contain the true parameter value $\underline{\theta}$ 95% of the time."*

Bayesians view the:

- true parameters $\boldsymbol{\theta}$ as **random**
- data $\underline{X}$ as fixed
- estimators $f(\underline{X})$ as fixed

Credible Interval:

$P\{\, l(\underline{X}) < \boldsymbol{\theta} < u(\underline{X}) \,\} = 95\%$

*"When sampling **θ** from a distribution many times, the sampled value will be between lower bound l($\underline{X}$) and upper bound u($\underline{X}$) 95% of the time."*

Many statisticians see both approaches as useful depending on the type of question.

(Read more: https://www.stat.umn.edu/geyer/3701/notes/mcmc-bayes.html)

Background: **Frequentist** vs. **Bayesian** statistics

- For the Bayesian, θ being random means that it has a probability distribution.
  - **Prior:** P(θ) *"What distribution do we think θ has before evidence?"*
  - **Posterior:** P(θ | X) *"What distribution does θ has after evidence?"*

- Related through **Likelihood:** L(θ ; X) ≡ P(X | θ)

- Bayes rule:

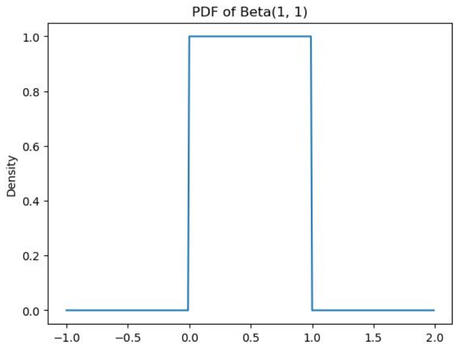
$$\underbrace{P(\theta|X)}_{\text{A posterior probability}} = \frac{\overbrace{P(X|\theta)}^{\text{Likelihood}}\,\overbrace{p(\theta)}^{\text{A prior probability}}}{\underbrace{P(X)}_{\text{Marginal probability}}}$$

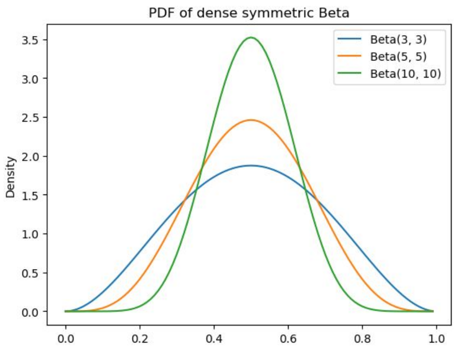
Background: simple example

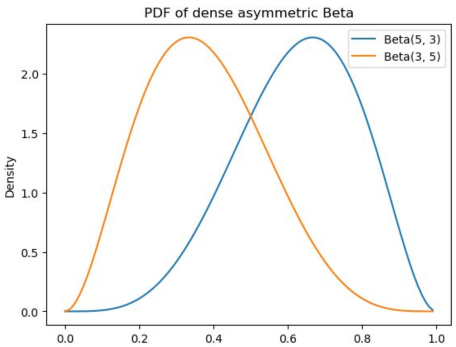
- Imagine flipping a coin N times to determine if biased or not:
  - $x_i \sim \text{Bernoulli}(p)$
- What's a reasonable **prior** distribution for the parameter p?

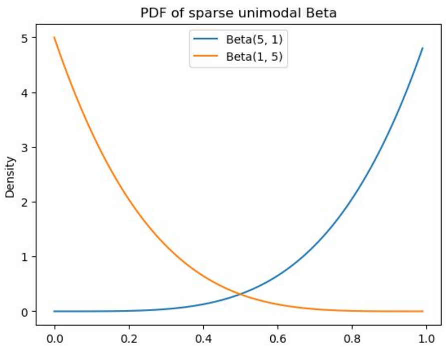
Background: simple example

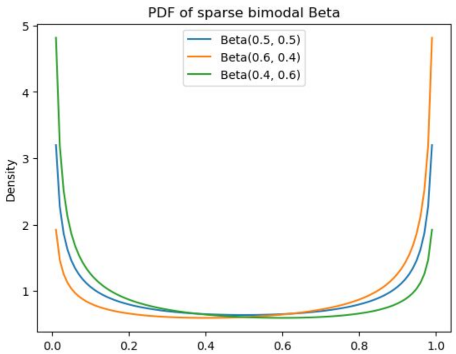
- Introducing the Beta distribution: p ~ Beta(α, β)
- Can be considered:
  - A generalization of the uniform distribution
  - A probability distribution over *probabilities* (i.e., samples will be in [0, 1])





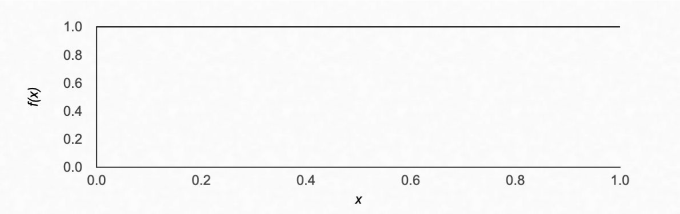






Background: simple example

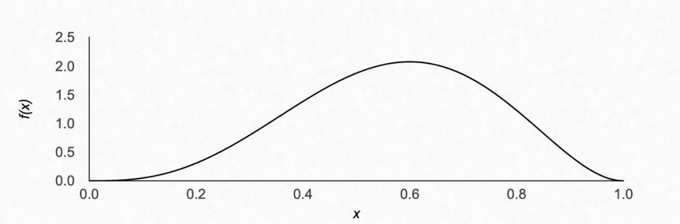
- Let's assume that the prior P(p) = Beta(1,1)



- Now we flip the coin 5 times and observe **3** heads and **2** tails.

- It can be shown* that the posterior P(p | X) will also be a Beta distribution:

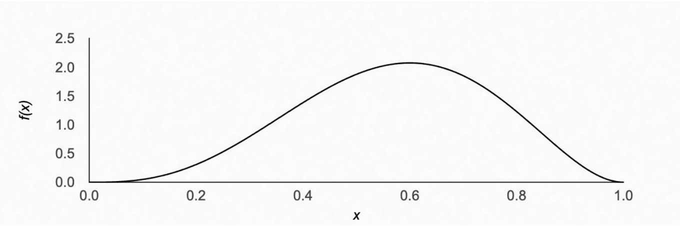
  P(p | X) = Beta(1 + **3**, 1 + **2**)



- The posterior has a lot of uncertainty.
  - 0.5 is well within the credible interval.

* https://en.wikipedia.org/wiki/Conjugate_prior

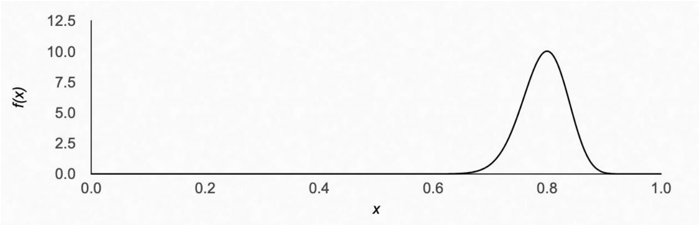
Background: simple example



- Now we flip the coin 95 more times and observe **77** heads and **18** tails.

- Updating the posterior again:

  $P(p \mid X) = \text{Beta}(1 + \mathbf{80}, 1 + \mathbf{20})$



- We are now quite certain that the coin is biased!

Background: Back to PP…

Some problems can be solved analytically like this, but:

- Bayesian models can get indefinitely complex.
  - What if multiple coins with different unknown probabilities?
  - What if we wanted priors for the α, β in the Beta distribution (hyperpriors)?
  - What if we had latent variables?

- Calculating the posterior can be mathematically intractable.

Background: Back to PP…

Instead of doing that… what if we "implement" the model as a data-generating program and use automatic tools to estimate the posterior?

Coin example as pseudocode:

```
Program generate_data(N, α, β):

    p := Beta(α, β).sample()

    For 0 ≤ i < N:
        x_i := Bernoulli(p).sample()

    return x
```

What is **Pyro**?

Combines PP with Deep Learning (PyTorch as a backend).

Question: When would full Bayesian inference be useful in ML?

What is **Pyro** (nuts and bolts)?

1. Methods and context managers for implementing Bayesian models.
2. Tools for statistical inference, given a data-generating model function:
    a. Stochastic Variational Inference (SVI)
    b. Markov Chain Monte Carlo (MCMC)
3. A lot of other useful tools that I won't discuss today.

How does Pyro implement models?

Three main components:

1. **pyro.sample** samples a value with a given name (or “site”) from a dist.
   a. A special “obs” keyword argument must be used to indicate *observed* variables.
2. **pyro.param** is used to declare a tunable parameter.
   a. Stored in a global *parameter store* dict for future access.
   b. Allows for constraints on parameter values (e.g., must be greater than 0).
3. **pyro.plate** is used for independently and identically distributed (iid) samples.
   a. Think of it as replacing for-loops, like in the coin example.

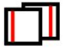
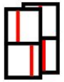
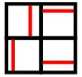
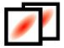
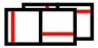
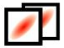
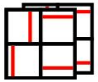
How does Pyro represent tensors?

Sample shape: the *iid* dimensions of the tensor (plate)

Batch shape: the *conditionally independent* dimensions of the tensor

Event shape: the *conditionally dependent* dimensions of the tensor

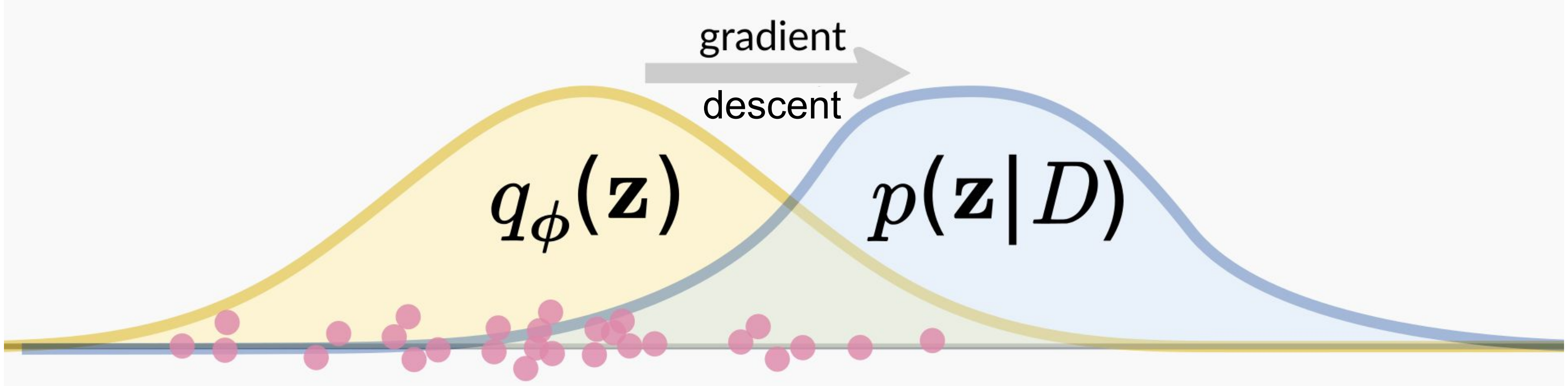
Always safe to assume dependence, but declaring independence when appropriate can make computations faster.

Distribution	Draws	Event Shape	Batch Shape	Sample Shape	Plain English
		( )	( )	( )	one draw from one normal
		( )	( )	(2,)	two draws from one normal
		( )	(2,)	( )	one draw from two normals
		( )	(2,)	(2,)	two draws from two normals
		(2,)	( )	( )	one draw from a bivariate normal
		(2,)	( )	(2,)	two draws from a bivariate normal
		(2,)	(2,)	( )	one draw from two bivariate normals
		(2,)	(2,)	(2,)	two draws from two bivariate normals

https://ericmjl.github.io/blog/2019/5/29/reasoning-about-shapes-and-probability-distributions/

Inference methods: SVI

- Approximate posterior $P_\theta(Z|X)$ with a simpler distribution $Q_\varphi(Z)$
  - Q is called the **guide** in Pyro.
  - Can be defined manually, or with an AutoGuide function.
- Use stochastic gradient descent on both θ and φ to move Q closer to P.



https://matsen.fhcrc.org/general/2019/08/24/vbpi.html

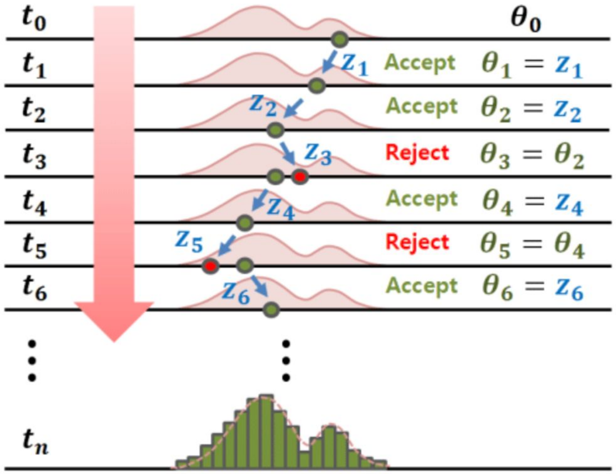
Inference methods: SVI

- Approximate posterior $P_\theta(Z|X)$ with a simpler distribution $Q_\varphi(Z)$
  - Q is called the **guide** in Pyro.
  - Can be defined manually, or with an AutoGuide function.
- Use stochastic gradient descent on both θ and φ to move Q closer to P.
- Uses a special loss function called the “evidence lower bound” (ELBO):

$$\mathrm{ELBO} \equiv \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\right]$$

Inference methods: MCMC

- The general idea of MCMC is that you can create a Markov Chain of samples that will eventually converge to the posterior dist.
  - Start with an initial sample θ_0.
  - Create a *proposal* by sampling θ_i from some simpler distribution $Q(\theta_i \mid \theta_{i-1})$
  - Accept the proposal proportionally to the *ratio* of $P(\theta_i \mid X)$ to $P(\theta_{i-1} \mid X)$
  - Continue indefinitely



https://www.researchgate.net/publication/334001505_Adaptive_Markov_chain_Monte_Carlo_algorithms_for_Bayesian_inference_recent_advances_and_comparative_study

Inference methods: MCMC

- The general idea of MCMC is that you can create a Markov Chain of samples that will eventually converge to the posterior dist.
    - Start with an initial sample θ_0.
    - Create a *proposal* by sampling θ_i from some simpler distribution $Q(\theta_i \mid \theta_{i-1})$
    - Accept the proposal proportionally to the *ratio* of $P(\theta_i \mid X)$ to $P(\theta_{i-1} \mid X)$
    - Continue indefinitely
- In practice, Pyro's MCMC uses a more complex algorithm called NUTS.
    - Based on Hamiltonian Monte Carlo (HMC), which automatically computes proposal dist. Q.
    - Need only provide model function and other hyperparameters.

SVI vs MCMC

SVI

+ Faster
+ Scalable

- Biased estimator
- May get stuck in local optimum and not converge to true posterior

MCMC

+ Unbiased estimator
+ Guaranteed convergence with enough samples

- Slower
- Requires a large number of samples

Rule of thumb: MCMC when working with very small data and need best estimate possible; SVI if working with large(r) dataset.

Supplementary: connections between ML and statistics

- Maximum Likelihood Estimation (MLE):
  - Frequentist approach to finding a point estimate of optimal parameters
  - Corresponds to a basic loss function.
- Maximum A Posteriori Estimation (MAP):
  - Bayesian approach to point estimation that incorporates a prior
  - Corresponds to loss with regularization.
- The gap filled by Pyro is when one wants to perform full Bayesian inference.
  - I.e., compute posterior distribution rather than a point estimate.

MLE $\theta^* = \text{argmax } P(X\|\theta)$	**MAP** $\theta^* = \text{argmax } P(X\|\theta)\, P(\theta)$	**Bayesian Inference** $P(\theta\|X) = P(X\|\theta)\, P(\theta) / P(X)$
Ordinary loss $\mathcal{L} = \text{error}(\hat{y}, y)$	**Loss with regularization** $\mathcal{L} = \text{error}(\hat{y}, y) + \lambda R(w)$	**??**

Supplementary: connections between ML and statistics

- Common ML loss functions and the equivalent likelihood/prior distributions:

Cross-entropy loss: $\mathcal{L} = - \Sigma\, y \log(\hat{y})$	Logistic likelihood
Absolute error loss (L1): $\mathcal{L} = \|y - \hat{y}\|$	Laplace likelihood
OLS loss (L2): $\mathcal{L} = (y - \hat{y})^2$	Gaussian likelihood
Lasso loss (OLS with L1 reg)	Gaussian likelihood w/ Laplace prior
Ridge loss (OLS with L2 reg)	Gaussian likelihood w/ Gaussian prior

- Note: all probability distributions have a corresponding loss function, but not all loss functions correspond to a valid probability distribution!