UNIVERSITY OF MARYLAND

MSML610: Advanced Machine Learning

Probabilistic Programming

Instructor: GP Saggese, PhD - gsaggese@umd.edu

References:

- AIMA (Artificial Intelligence: a Modern Approach)
  - Chap 15: Probabilistic programming

Probabilistic programming

- **Probabilistic programming**
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - Posterior-based decisions
  - Posterior predictive checks
  - Robust inference
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

EDA vs inference

- **Exploratory data analysis**
  - Summarize, interpret, check data
  - Visually inspect the data
  - Compute descriptive statistics
  - Communicate results
- **Inferential statistics / inference**
  - Draw insights from a limited set of data
  - Make statements beyond the current data
  - Make predictions for future unobserved data points
  - Understand a phenomenon
  - Choose among competing explanations for the same observations

Good vs Bad Way to Do Statistics

- **Bad** 😈
  - Learn a collection of "statistical recipes"
    - Make assumption / approximate to make math workable
  - Given data and problem, pick one recipe, try until you get a "low" p-value
  - For machine learning, iterate until you get a "good" fit on out-of-sample data
- **Good** 🙂
  - Bayesian statistics
    - Provide a general approach to statistical inference
    - Remove limitations from closed analytical form
  - Probabilistic approach gives a unified view of seemingly disparate methods
    - E.g., statistical methods and machine learning
    - There is a deep unity of all different recipes
  - Modern tools (e.g., PyMC3) allow defining and solving models that were unsolvable a few years ago

Data

- Data comes from:
  - Experiments
  - Simulations
  - Surveys
  - Field observations
- Data is stochastic due to uncertainty
  - Ontological: system is intrinsically stochastic
  - Technical: measurement precision is limited or noisy
  - Epistemic: conceptual limitations in understanding
- Collecting data is costly
  - Consider questions before collecting data
  - Experiment design is a branch of statistics for data collection
- Data is rarely clean and tidy
- Data is interpreted through mental and formal models

Models

- Models are simplified descriptions of a given system/process
  - A more complex model is not always a better one
- Goal:
  - Capture the most relevant aspects of the system
  - Ignore minor details

Bayes' theorem: Recap

- Bayes' theorem posits that for model parameters θ and data X

$$\Pr(\theta|X) = \frac{\Pr(X|\theta) \cdot \Pr(\theta)}{\Pr(X)}$$

where:

- $\Pr(\theta|X)$
  - Posterior: probability for parameters θ after seeing data X
- $\Pr(X|\theta)$
  - Likelihood/statistical model: plausibility of data X given parameters θ
- $\Pr(\theta)$
  - Prior: knowledge about parameter θ before any data
- $\Pr(X)$
  - Marginal likelihood/evidence: probability of observing data X
  - "Marginal" as it averages over all possible parameter values

- In other words:

$$\text{Posterior} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Evidence}}$$

Single-parameter inference

- Probabilistic programming
  - **Single-parameter inference**
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - Posterior-based decisions
  - Posterior predictive checks
  - Robust inference
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

Bayesian Models

- **Probabilities** measure uncertainty about parameters
- **Bayes' theorem** updates the probabilities in light of new data (hopefully) reducing the uncertainty

$$\Pr(hyp|data) = \frac{\Pr(data|hyp)\Pr(hyp)}{\Pr(data)}$$

- **Bayesian model workflow**

1. Given data and assumptions on how data could have been generated, design a model by combining probability distributions
   - A model can be crude approximation
2. Use Bayes' theorem to add data to the model
   - Aka "conditioning" the model on our data
3. Check whether the model makes sense according to:
   - The data
   - Our expertise on the subject
   - Other related models

- Often these steps are not done in a linear fashion, but with backtracking:
  - There was a coding mistake
  - Improve the model
  - Collect more or different data

Coin example: problem

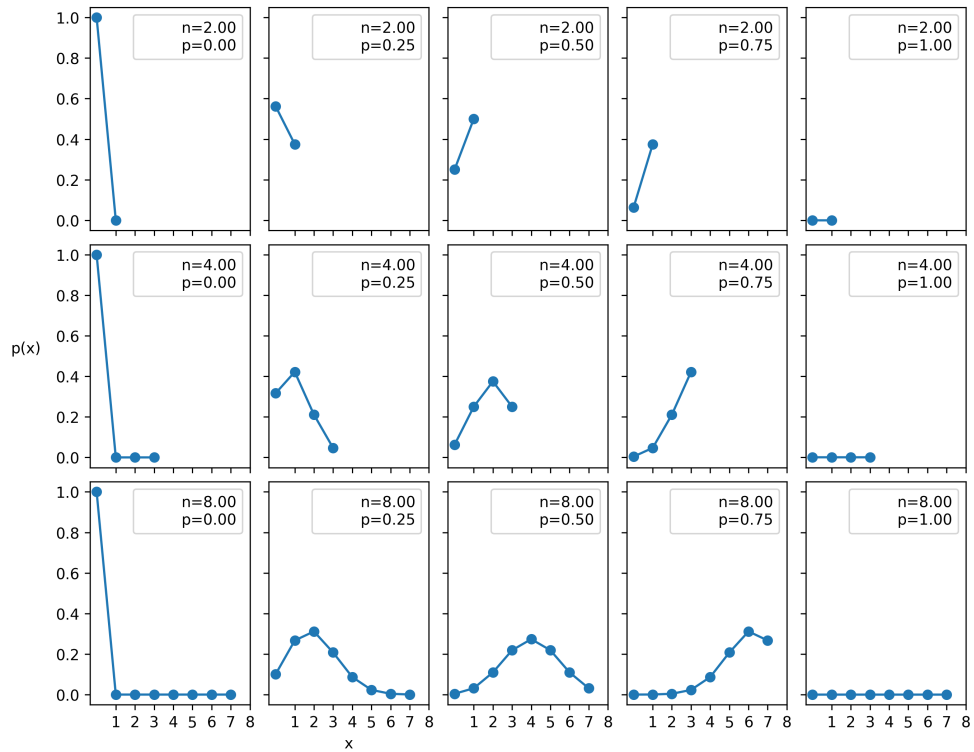
- **Problem**:
  - Toss a coin N times
  - Record the number of heads Y and tails $N - Y$
  - How biased is the coin?
- There is true uncertainty
  - An underlying parameter exists
  - θ represents the coin bias
    - 0: always tails
    - 1: always heads
    - 0.5: half tails, half heads
  - It is related to observed realizations, but not deterministically
- **Modeling assumptions**:
  - Independent Identically Distributed (IID)
    - Independence: a coin toss does not affect other tosses
    - Identically distributed: the coin's bias is constant
  - The likelihood $Y|\theta$ is modeled as a binomial distribution
    - Probability of Y heads out of N tosses, given θ
  - Prior is modeled as a beta distribution
    - It can adopt several shapes
    - Beta is the conjugate prior of the binomial distribution

Binomial distribution

- Probability of k heads out of N tosses given θ is

$$X \sim Binomial(n, p)$$

$$\Pr(k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

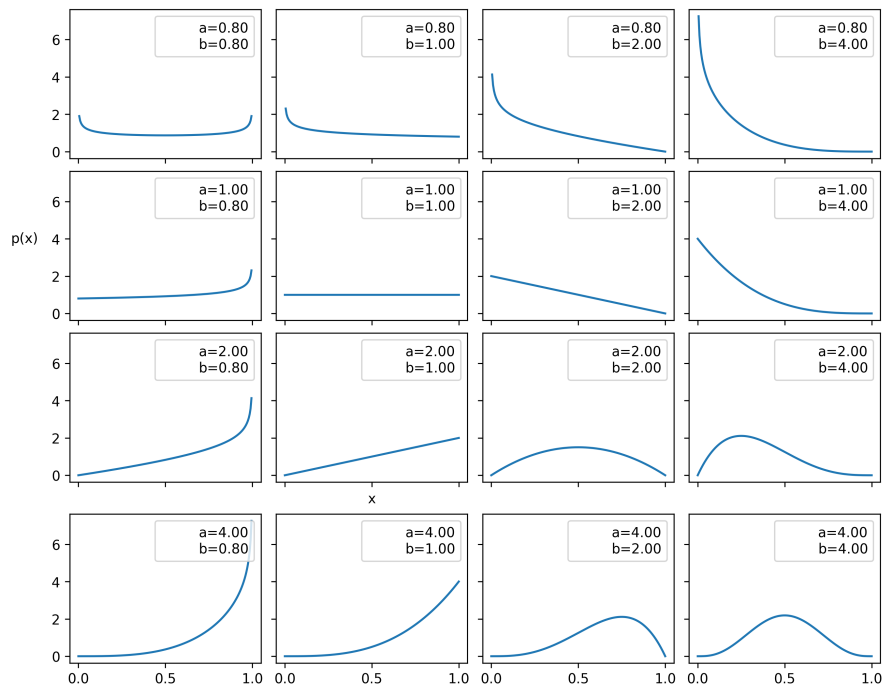


Beta distribution

- Continuous prob distribution defined in [0, 1]
- It can adopt several shapes (e.g., uniform, increasing, decreasing, Gaussian-like, U-like)
  - α represents "success" parameter
  - β represents "failure" parameter
  - When $\alpha > \beta$, the distribution skews toward 1, indicating a higher probability of success
  - When $\alpha = \beta$, the distribution is symmetric and centered around 0.5
- It is useful to model probability or proportion
  - E.g., the probability of success in a Bernoulli trial θ
- Beta is the conjugate prior of the binomial distribution

$$X \sim Beta(\alpha, \beta)$$

$$\Pr(\theta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}$$



Coin example: analytical solution

- The **posterior** is proportional to **likelihood** $\times$ **prior**

$$\Pr(\theta|y) \propto \Pr(y|theta)\Pr(theta)$$

- Substituting the **likelihood** with a Binomial and the **prior** with a Beta

$$\begin{aligned}
&\Pr(\theta \mid Y) \\
&= \underbrace{\frac{N!}{y!(N-y)!}\theta^{y}(1-\theta)^{N-y}}_{\text{likelihood}} \underbrace{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}}_{\text{prior}} \\
&\propto \underbrace{\theta^{y}(1-\theta)^{N-y}}_{\text{likelihood}} \underbrace{\theta^{\alpha-1}(1-\theta)^{\beta-1}}_{\text{prior}} \\
&= \theta^{y+\alpha-1}(1-\theta)^{N-y+\beta-1} \\
&= \text{Beta}\left(\alpha_{\text{prior}}+y, \beta_{\text{prior}}+N-y\right)
\end{aligned}$$

- This is how to update the posterior given the data

Conjugate prior of a likelihood

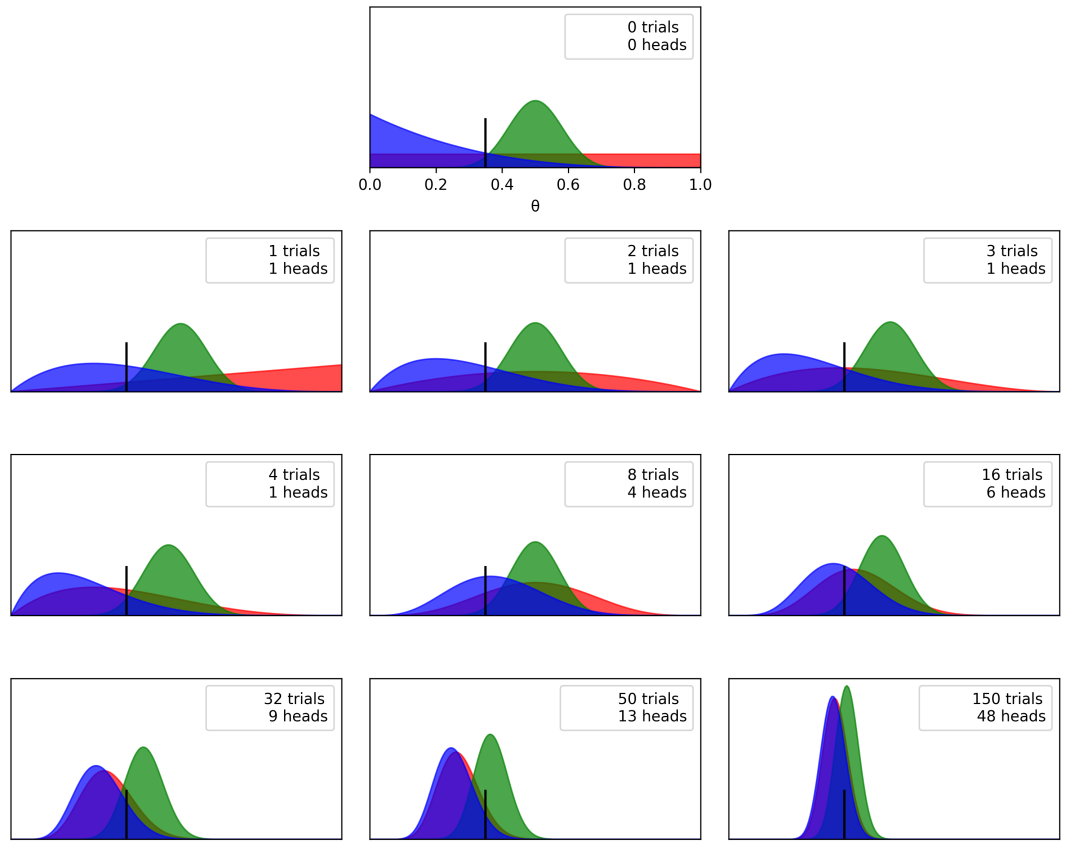
- **Conjugate prior** is a prior that when used in combination with a given likelihood, returns a posterior with the same functional form as the prior
- E.g.,

Prior	**Likelihood**	**Posterior**
Beta	Binomial	Beta
Normal	Normal	Normal

- Conjugacy ensures tractability of the posterior
  - Prior and posterior have the same distribution
  - Posterior has a closed analytical form (that we know in advance)
  - Use data to update parameters from the prior in multiple iterations

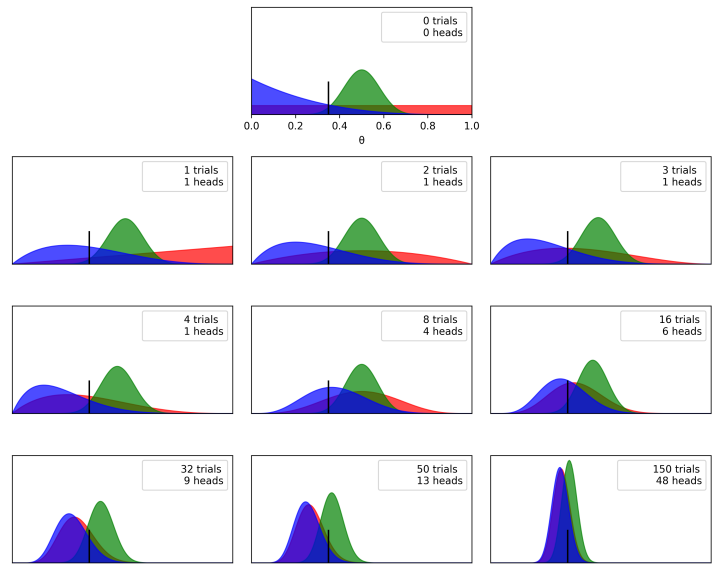
Coin example: effect of priors (1/2)

- Start with 3 priors:
  - Red: uniform prior (all bias values equally probable)
  - Green: Gaussian-like prior around 0.5 (coin mostly unbiased)
  - Blue: skewed towards tail (coin biased)
- Apply data to update the posterior distribution



Coin example: effect of priors (2/2)

1. The outcome of a Bayesian analysis is a posterior distribution, not a single value
2. The spread of the posterior is proportional to the uncertainty
3. The spread decreases with more data
4. The spread decreases faster if aligned with the prior
5. With enough data, models with different priors tend to converge to the same result
6. Applying the posterior sequentially or at once yields the same result



Frequentist approach vs priors

- Detractors of Bayesian approach complain that:
  - *"One should let the data speak"*
  - The prior doesn't let the data speak for itself
- **Counterpoints** ⚖️
- In reality *"Data doesn't speak, but murmurs"*
  - We can only make sense of data made in context of models (priors, mental models, mathematical models)
- Every statistical model has a prior, even if not set explicitly
  - E.g., maximum likelihood estimate (MLE) in frequentist approach corresponds to a uniform prior and using the mode of the posterior
  - Problems ⚠️
    - Assumptions are unclear
    - MLE is a point-estimate, not a distribution of plausible values

Advantages of using prior

1. All the assumptions are clear and specified
2. The prior encourages deeper analysis of the problem and the data
   - It forces you to understand the problem before seeing data
3. The posterior, averaged over priors, is more robust and less prone to overfitting
4. Spread of the distribution is linked to uncertainty
5. A well-chosen prior can simplify and speed up inference
   - *"When you encounter computational problems, there's often an issue with your model"*

How to choose priors

- **Weakly-informative priors** (aka "flat", "vague", "diffuse priors")
  - Provide minimal information with minimal impact on the analysis, e.g.,
    - Coefficient of linear regression is centered around 0 with little restriction: $\beta \sim \mathit{Normal}(0, 10)$
- **Regularizing priors**
  - Known information about the parameter, e.g.,
    - Parameter is positive: $\sigma \sim \mathit{HalfCauchy}(0, 5)$
    - Parameter is close to zero, above/below a certain number, or in a range
    - $\beta \sim \mathit{Laplace}(0, 1)$ (lasso prior) encourages sparsity
    - $\beta \sim \mathit{Normal}(0, 1)$ discourages extreme values
- **Informative priors**
  - Strong priors conveying a lot of information from previous knowledge (expert opinion, studies), e.g.,
    - From experimental data: $\beta_1 \sim \mathit{Normal}(2.5, 0.5^2)$
    - From previous data, about 5% of cases were positive: $p \sim \mathit{Beta}(2, 38)$
- **Prior elicitation**
  - Involves computing the least informative distribution given certain constraints
    - E.g., estimate distribution using maximum entropy to satisfy constraints
    - E.g., beta distribution with 90% of mass between 0.1 and 0.7

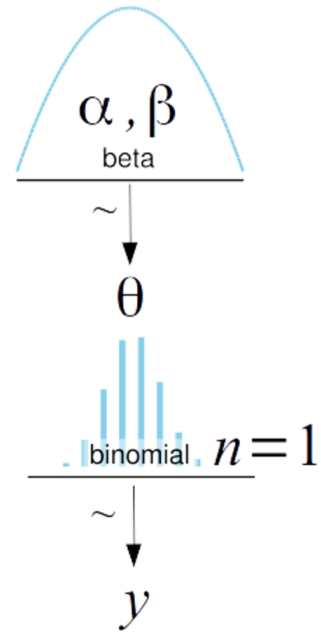
Communicating a Bayesian analysis

- Probabilistic programming
  - Single-parameter inference
  - **Communicating a Bayesian analysis**
  - Probabilistic programming
  - Posterior-based decisions
  - Posterior predictive checks
  - Robust inference
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

Communicating the model of a Bayesian analysis

- You need to communicate the probabilistic models used
  - E.g., for the coin-flip the distributions are described as:

$$y \sim \mathit{Binomial}(n = 1, p = \theta)$$
$$\theta \sim \text{Beta}(\alpha, \beta)$$



  - Kruschke diagram
- The result of a Bayesian analysis is a posterior distribution, summarized with location and dispersion, e.g.,
  - Mean (or mode, median)
  - Std dev
    - Can be misleading for skewed distributions
  - Highest-posterior density (HPD)
    - Shortest interval containing a given portion of the probability density (e.g., 95% or 50%)
    - The amount is arbitrary (E.g., `ArviZ` defaults to 94%)

Confidence intervals vs Credible intervals

- People confuse frequentist **confidence intervals** with Bayesian **credible intervals**
- In the frequentist framework, there is a true (but unknown) value of a parameter
  - A (frequentist) **confidence interval** can contain or not contain the true value of a parameter
  - Interpretation of a 95% confidence interval
    - No: *"There is a 95% probability that the true value is in this interval"*
    - Yes: *"If we repeated this process many times, 95% of the intervals we construct would contain the true value"*
- In the Bayesian framework, parameters are random variables and there is a probability of having a parameter inside an interval
  - Bayesian **credible interval** is intuitive
  - *"There is a 95% probability that the true parameter lies within this interval, given the observed data"*

Probabilistic programming

- Probabilistic programming
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - **Probabilistic programming**
  - Posterior-based decisions
  - Posterior predictive checks
  - Robust inference
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

Bayesian statistics

- Given:
  - The **"knows"**
    - Model structure (modeled as a graph of probability distributions)
    - Data, observations (modeled as constants)
  - The **"unknowns"**
    - Model parameters (modeled as probability distributions)
- We use Bayes' theorem to condition **unknowns** to **knowns**, hoping to reduce the uncertainty about the unknowns
- **Problem** 
  - Most probabilistic models are analytically intractable
- **Solution** 
  - Probabilistic programming
    - Specify a probabilistic model using code
    - Solve models using numerical techniques

Probabilistic programming languages

- **Steps**:
  1. Specify models using code
  2. Numerical models solve the inference problem without requiring user understanding (i.e., "universal inference engines")
     - `PyMC3`: a flexible Python library for probabilistic programming, which relies on `NumPy` and `Theano`
     - `Theano`: a library to define, optimize, evaluate mathematical expressions using multidimensional arrays (tensors) efficiently (using C, CPU, GPU)
     - `ArviZ`: a library to interpret results of probabilistic models
- **Pros**:
  - Compute results, even without an analytical closed form
  - Solving the model can be considered a black box
  - Focus on model design, evaluation, and interpretation, rather than solving the model
- Probabilistic programming languages offer a similar impact as Fortran had on scientific computing
  - Focus on building algorithms, ignoring computational details

Coin example: numerical solution 1/3

1. Assume we know the true value of θ, which is not true in general
2. Observe samples of the variable y
3. Model the prior θ and the likelihood $y|\theta$
   - Binomial with $n = 1$ is a Bernoulli RV
   - Beta with $\alpha = \beta = 1$ is a uniform RV

$$\theta \sim \text{Beta}(\alpha = 1, \beta = 1)$$
$$Y \sim \text{Binomial}(n = 1, p = \theta)$$

4. Run inference
5. Generate samples of the posterior
6. Summarize posterior
   - E.g., Highest-Posterior Density (HPD)

Coin example: numerical solution 2/3

[18]:
```
np.random.seed(123)
n = 4
# Unknown value.
theta_real = 0.35

# Generate some observational data.
data = stats.bernoulli.rvs(p=theta_real, size=n)
data
```

[18]:
```
array([1, 0, 0, 0])
```

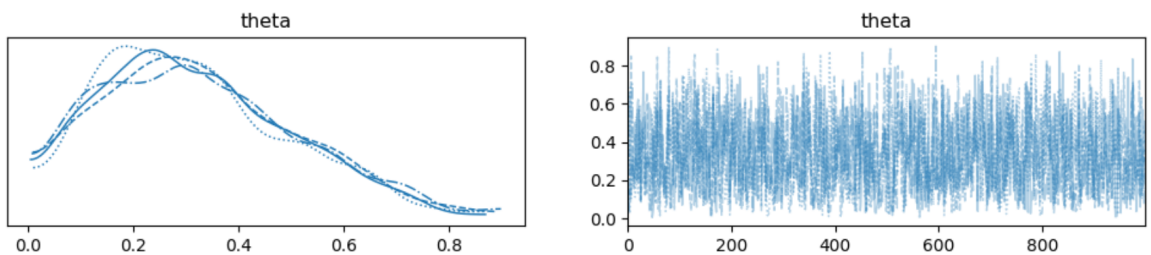
[19]:
```
with pm.Model() as our_first_model:
    # Prior.
    theta = pm.Beta('theta', alpha=1., beta=1.)
    # Likelihood.
    y = pm.Bernoulli('y', p=theta, observed=data)
    # (Numerical) Inference to estimate the posterior distribution through samples.
    idata = pm.sample(1000, random_seed=123)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [theta]
```

```
Sampling 4 chains, 0 divergences ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 1 seconds.
```

[20]:
```
az.plot_trace(idata);
```



[21]:
```
az.summary(idata)
```

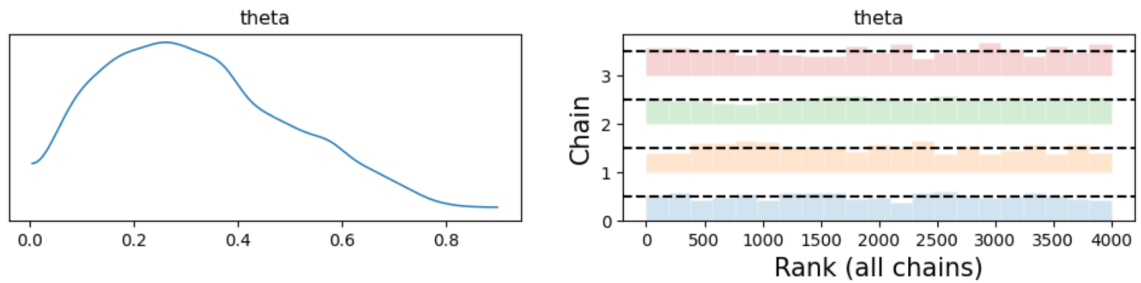
[21]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
theta	0.324	0.179	0.031	0.653	0.005	0.003	1500.0	1737.0	1.0

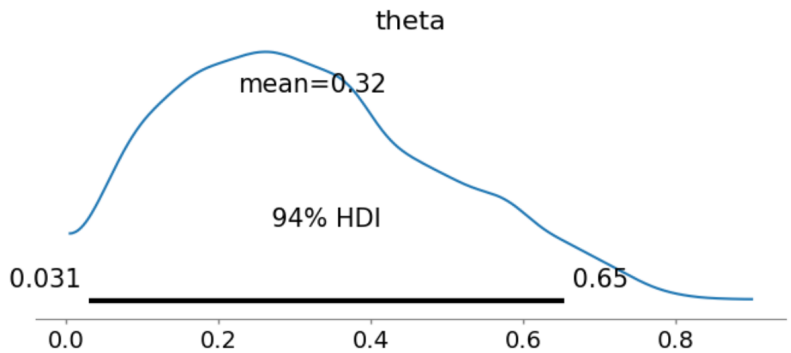
- Generate some data from the (ground truth) model
- Build a PyMC model which is one-to-one to the mathematical model
- PyMC uses a NUTS sampler, computes 4 chains (since there are 4 cores)
- No trace is diverging
- Kernel density estimation (KDE) for the posterior (they should be Beta!)
- Traces look "noisy" and non-diverging (good!)
- Numerical summary of the posterior, e.g., mean, std dev, HDI
- $\mathbb{E}[\hat{\theta}] \approx 0.324$
- $\Pr(\hat{\theta} \in [0.031, 0.653]) = 0.94$

Coin example: numerical solution 3/3

```
[22]: az.plot_trace(idata, kind="rank_bars", combined=True);
```




```
[23]: az.plot_posterior(idata);
```



- Compute a single KDE for all chains
- Rank plot to check sanity of results
- Histograms should look uniform so that they are exploring different regions of the posterior (ideally all posterior)
- Single KDE with more statistics

Posterior-based decisions

- Probabilistic programming
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - **Posterior-based decisions**
  - Posterior predictive checks
  - Robust inference
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

Posterior-based decisions

- Sometimes describing the posterior is not enough, but we need to make decisions based on our inference
- E.g., is the coin fair ($\theta = 0.5$) or biased?
  - Since $\mathbb{E}[\hat{\theta}] = 0.324$ it seems that the coin is biased
  - We can't rule out the possibility that the coin in unbiased since $HPI = [0.03, 0.65]$ and 0.5 is inside HPI
- If we want a sharper decision, we need to:
  - Collect more data to reduce the spread of the posterior
  - Define a more informative prior

Savage-Dickey density ratio

- The Savage-Dickey ratio tests point null hypotheses in Bayesian inference
- **Idea**: compare prior and posterior densities at a single point θ_0

$$BF_{01} = \frac{p(\theta_0|H_1)}{p(\theta_0|\mathcal{D}, H_1)}$$

where:

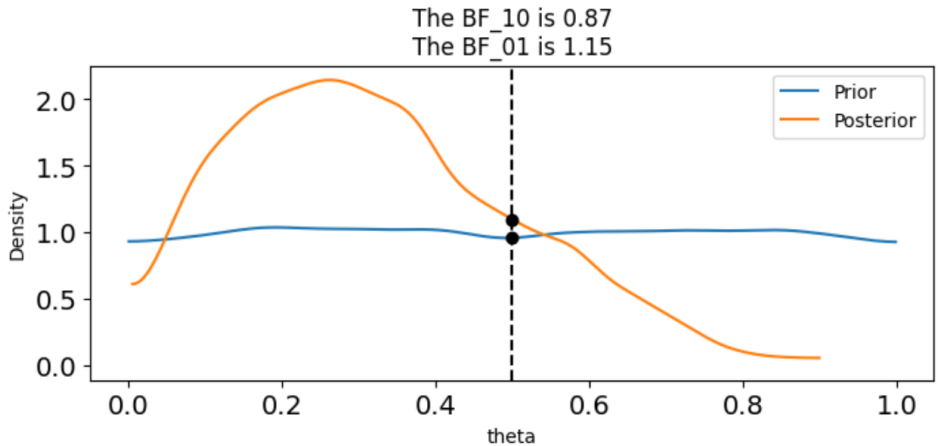
  - $p(\theta_0|H_1)$ is the *prior* density θ under the alternative hypothesis H_1, evaluated at θ_0
  - $p(\theta_0|\mathcal{D}, H_1)$ is the *posterior* density θ under H_1 evaluated at θ_0

Bayes Factor (BF)	**Interpretation**
1 - 3	Not enough evidence
3 - 10	Substantial evidence
10 - 100	Strong evidence
> 100	Decisive evidence

- **Intuition**: this ratio tells us how much the data has changed our belief about θ_0
  - If the posterior density at θ_0 is much smaller than the prior density, the Bayes factor suggests strong evidence against H_0

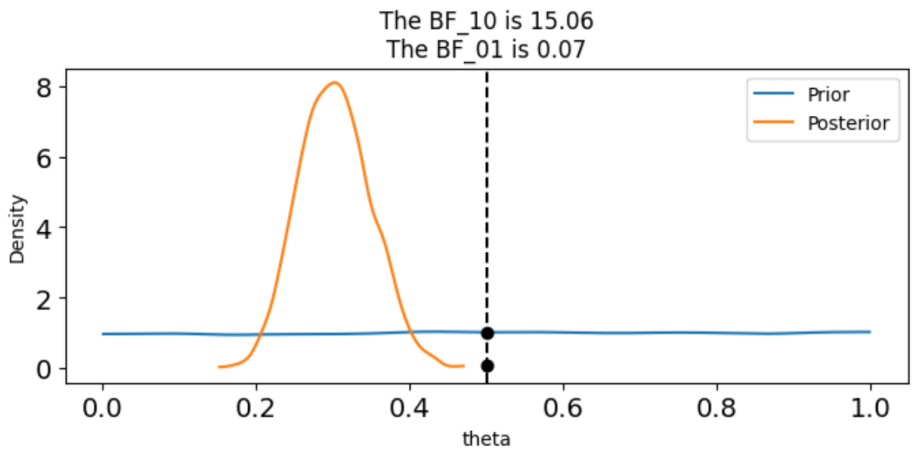
Savage-Dickey density ratio: example

```
[29]: az.plot_bf(idata1, var_name="theta", prior=np.random.uniform(0, 1, 10000), ref_val=0.5);
```




```
[30]: az.plot_bf(idata2, var_name="theta", prior=np.random.uniform(0, 1, 10000), ref_val=0.5);
```

```
The reference value is outside of the posterior. This translate into infinite support for H1
```



- H_0: "coin is fair"
- The prior for H_0 is 0.87
- The posterior for H_0 is 1.15
- $BF_{10} = $

ROPE: Region Of Practical Equivalence

- A **ROPE** is an interval for a parameter where all values inside are considered "equivalent", e.g.,
  - A H_0: *"coin is fair"* iff $\theta = 0.5$ is impractical
  - For a coin, the interval $\theta \in [0.45, 0.55]$ is equivalent to 0.5
- **Hypothesis testing with ROPE and HPI**
  - Compare Region Of Practical Equivalence (ROPE) with Highest-Posterior Interval (HPI)
    - If the HPI is within ROPE, no effect: H_1 is rejected
    - If the HPI is outside ROPE, there is an effect: H_0 is rejected
    - If the HPI overlaps with ROPE, the result is inconclusive
- The ROPE should be decided before analysis based on domain knowledge
  - Picking it after analysis is like picking the p-value threshold after seeing the p-value

Loss function: motivation

- For many problems, the cost of a decision is asymmetric
  - E.g., the cost of a bad decision is much greater than the benefit of a good decision
  - E.g., vaccines can cause an overreaction, but the benefits outweigh the risk significantly
- To make the best decision, measure:
  - Benefits of a correct decision
  - Cost of a mistake
  - Decide the trade-off using a loss function
  - Use loss function to make decisions
- The loss captures *"how bad is to make an estimation mistake?"*
  - The larger the loss, the worse the estimation

Loss function

- Aka "cost function"
  - The inverse is known as "objective", "fitness", "utility function"
- Use a function to measure the difference between:
  - The true value θ; and
  - The estimated value $\hat{\theta}$

Loss	**Expression**	**Point estimate**
Quadratic loss	$(\theta - \hat{\theta})^2$	Mean of posterior
Absolute loss	$\lvert\theta - \hat{\theta}\rvert$	Median of posterior
1-0 loss	$I(\theta \neq \hat{\theta})$	Mode of posterior

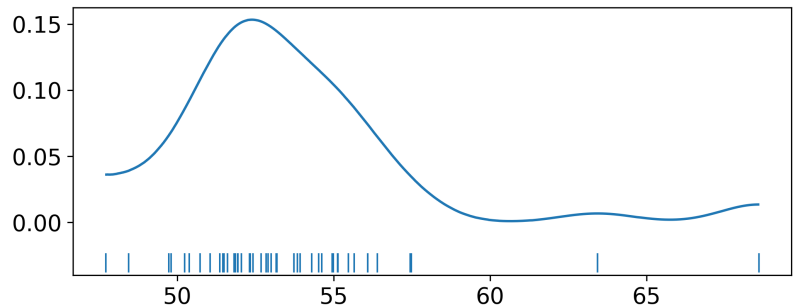
- Making decisions in Bayesian statistics using loss function
  - Goal: pick a single value $\hat{\theta}$
  - We don't know the true value θ
  - Estimate θ in terms of the posterior distribution
  - Find the value $\hat{\theta}$ that minimizes the expected loss function

Use of Gaussians in Bayesian analysis

- Aka "normal" distribution
- Gaussians are easy to work with and abundant in nature
  1. The average of something using a big enough sample size tends to be a Gaussian (CLT)
  2. Many phenomena can be approximated using Gaussians, since they are average of effects
  3. Conjugate prior of Gaussian is Gaussian
- On the other hand, it's important to relax the assumption of Gaussianity

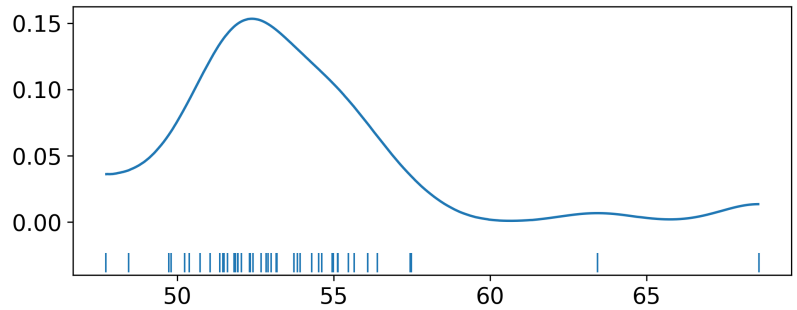
Chemical shift

- Nuclear magnetic resonance (NMR)
  - Used to study molecules of living things
  - Measures observable quantities related to unobservable molecular properties (e.g., chemical shift)
- Data looks Gaussian with a couple of outliers



Chemical shift: example

- Assume Gaussian is a decent approximation of the data



- The likelihood $y|\mu, \sigma$ comes from a normal distribution:

$$Y \sim N(\mu, \sigma)$$

- Priors for mean and sigma of Y

1. The mean comes from a uniform distribution with boundaries l and h:

$$\mu \sim U(l, h)$$

   - Set μ as a uniform distribution larger than the data range, E.g., $[40, 75]$
2. The std dev comes from a half-normal distribution (i.e., a regular normal but restricted to non-negative values):

$$\sigma \sim HalfNormal(0, \sigma_\sigma)$$

   - If we don't know possible values of μ and σ, set $\sigma_\sigma = 10$ as a large value

Chemical shift: PyMC

[87]:
```
with pm.Model() as model_g:
    # The mean is Uniform in [40, 70] (which is larger than the data).
    mu = pm.Uniform("mu", lower=40, upper=70)
    # The std dev is half normal with a large value (which is a large value based on the data).
    sigma = pm.HalfNormal("sigma", sigma=10)
    # The model is N(mu, sigma).
    y = pm.Normal("y", mu=mu, sigma=sigma, observed=data)
    # Sample.
    idata_g = pm.sample(1000)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mu, sigma]
```

```
Sampling 4 chains, 0 divergences ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 1 seconds.
```

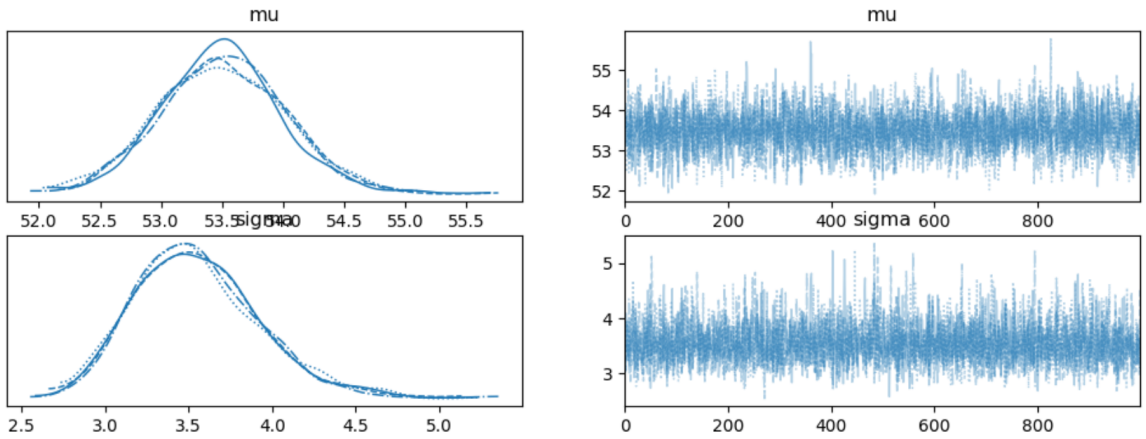
- The code is one-to-one with the model:

$$Y \sim N(\mu, \sigma)$$
$$\mu \sim U(l = 40, h = 75)$$
$$\sigma \sim HalfNormal(0, \sigma_\sigma = 10)$$

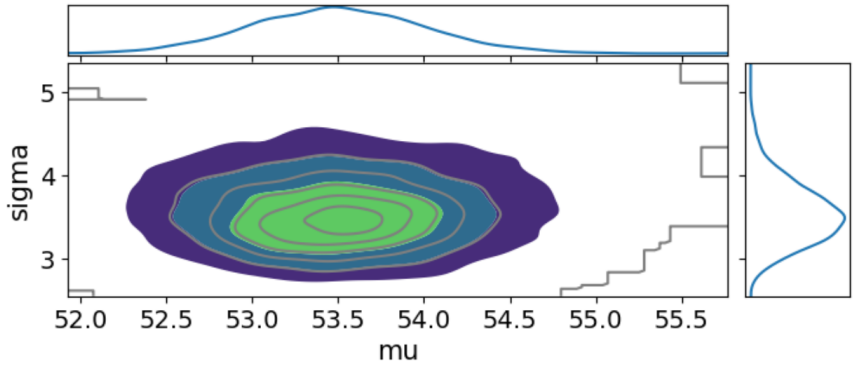
- Get 1000 samples from the posterior

Chemical shift: PyMC

[92]:
```
# There are 4 traces for 2 variables.
az.plot_trace(idata_g);
```



[93]:
```
# The posterior distribution of the params is bi-dimensional, since it has mu and sigma.
az.plot_pair(idata_g, kind='kde', marginals=True);
```



[94]:
```
# Report a summary of the inference.
az.summary(idata_g, kind="stats").round(2)
```

[94]:

	mean	sd	hdi_3%	hdi_97%
mu	53.50	0.51	52.55	54.46
sigma	3.55	0.38	2.86	4.23

- Compute 4 traces for the 2 variables μ, σ

Posterior predictive checks

- Probabilistic programming
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - Posterior-based decisions
  - **Posterior predictive checks**
  - Robust inference
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

Samples from posterior distribution

- Given a posterior distribution $\Pr(\theta|y)$, you can generate predictions $\tilde{y}$ based on the data y and the estimated parameters $\hat{\theta}$:

$$\Pr(\tilde{y}|y) = \int \Pr(\tilde{y}|\hat{\theta}) \Pr(\theta|y) d\theta = \int \text{model} \times \text{posterior}$$

- This is called the "**posterior predictive distribution**" as it predicts future data using the posterior distribution
- **Conceptually:**
  - Sample a value of θ from the posterior $\Pr(\theta|y)$
  - Feed the value of θ to the likelihood $\Pr(y|\theta)$
  - Obtain $\tilde{y}$
- This process has two sources of uncertainty:
  - Parameter uncertainty
    - Captured by the posterior $\Pr(\theta|y)$
  - Sampling uncertainty
    - Captured by the likelihood $\Pr(y|\theta)$

Posterior predictive check (PPC)

- PPC approach:
  - Generate predictions $\tilde{y}$ with observed data y
  - Check for consistency
- **Intuition**: can the model reproduce observed data?
- Models should always be checked
  - Differences can arise by mistakes or limitations of the problem/data
  - E.g., the model works well for average behavior but fails to predict rare values

A Bayesian workflow

1. We have data from a process
   - True distribution of the data is unknown / unknowable
2. Sample
   - Get a finite sample y through sampling
   - E.g., experiment, survey, simulation)
3. Inference
   - From the observed sample y we build a probabilistic model using prior $\Pr(\theta)$ and likelihood $\Pr(y|\theta)$ to get a posterior distribution $\Pr(theta|y)$
   - The posterior distribution is a distribution of the parameters of the models θ given the data
4. Predictive distribution
   - From posterior distribution we can compute predictions (posterior predictive distribution)
   - The posterior predictive distribution is a distribution of predicted samples averaged over the posterior distribution
5. Validation
   - Validate the model by comparing the original samples vs the predicted samples

Robust inference

- Probabilistic programming
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - Posterior-based decisions
  - Posterior predictive checks
  - **Robust inference**
  - Groups comparison
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

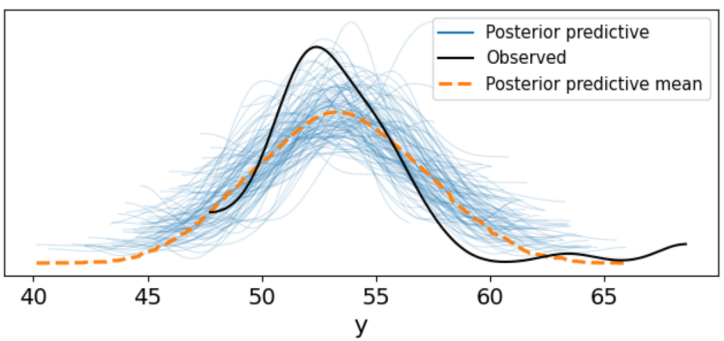
Posterior predictive check: Chemical shift example

```
[95]: # Compute 100 posterior predictive samples.
      y_pred_g = pm.sample_posterior_predictive(idata_g, model=model_g)
```

```
Sampling: [y]
```

```
Sampling ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
[96]: # Black: KDE of the data (observed)
      # Blue: KDEs of the posterior predictive samples
      # Orange: KDE of the posterior predictive mean
      az.plot_ppc(y_pred_g, mean=True, num_pp_samples=100);
```

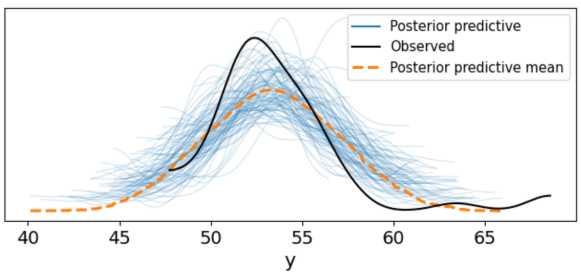


- **Is the PPC model good?**
- No
  - The mean of the posterior is more to the right than the data
  - The std dev of the posterior is larger

Chemical shift: model critique

```
[95]: # Compute 100 posterior predictive samples.
      y_pred_g = pm.sample_posterior_predictive(idata_g, model=model_g)

      Sampling: [y]

         Sampling ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

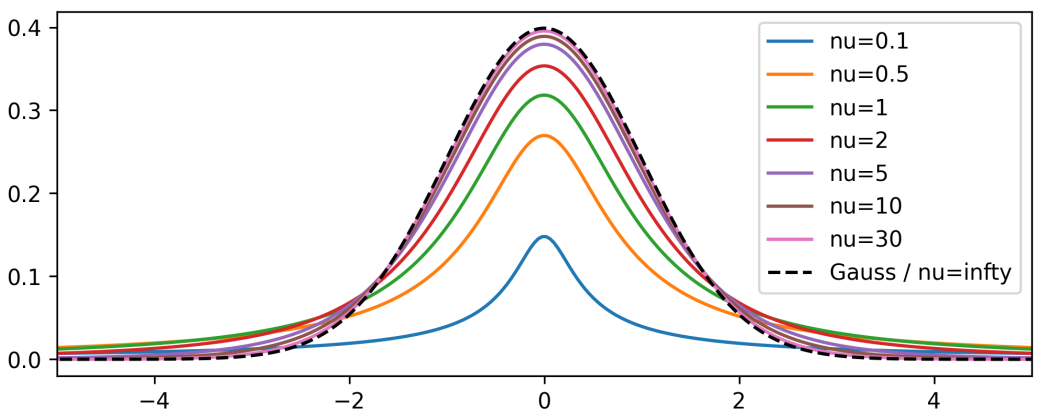
```
[96]: # Black: KDE of the data (observed)
      # Blue: KDEs of the posterior predictive samples
      # Orange: KDE of the posterior predictive mean
      az.plot_ppc(y_pred_g, mean=True, num_pp_samples=100);
```



- **Problem**
  - Two data points lie on the tails of the distribution
  - The normal distribution is "surprised" by these points and "reacts" by adjusting the mean towards them and increasing the standard deviation
  - A normal distribution for the chemical shift creates issues
- **Solution**
  - We could declare these points outliers and discard them
    - E.g., it's malfunction of the equipment (we need evidence!)
  - We can change the model
- In general Bayesians prefer to "encode" assumptions:
  - Into the model (e.g., in priors and likelihoods)
  - Rather than ad-hoc heuristics (e.g., outlier removal rules)

Student's t-distribution: Recap

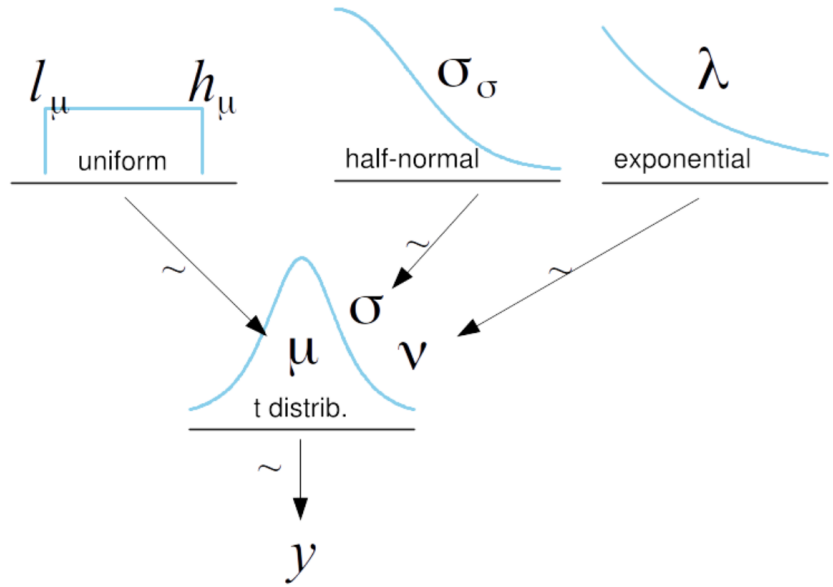
- Student's t-distribution has 3 params:
  1. Mean μ
     - It doesn't always exist
  2. Scale σ
     - Similar to std dev, but it doesn't always exist
  3. Degrees of freedom $\nu \in [0, \infty]$
     - Aka "normality parameter", since it controls how "normal" is the distribution
     - With $\nu = 1$: heavy tails and no mean (Cauchy)
     - With $\nu \to \infty$ we recover the Gaussian
- Heavy tails (high kurtosis) means "it is more likely to find values away from the mean compared to a Normal"



Chemical shift: use Student's t-dist (1/3)

- Let's use a Student's t-distribution model, instead than Normal

$$\mu \sim U(l, h)$$
$$\sigma \sim HalfNornal(0, \sigma)$$
$$\nu \sim Exp(\lambda)$$
$$y \sim StudentT(\mu, \sigma, \nu)$$

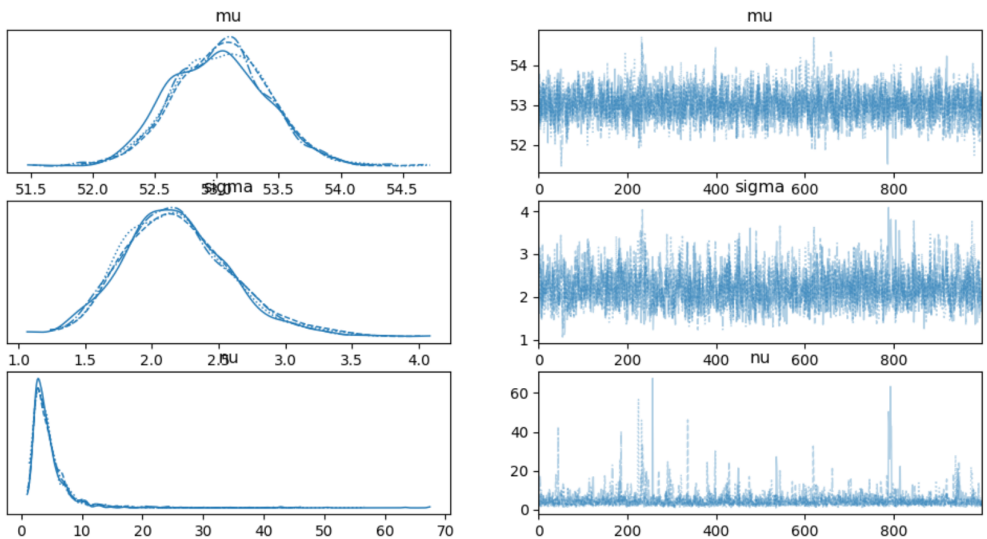


Chemical shift: use Student's t-dist (2/3)

[102]:
```
# Use a Student-T model.
with pm.Model() as model_t:
    mu = pm.Uniform("mu", 40, 75)
    sigma = pm.HalfNormal("sigma", sigma=10)
    # A student with nu = 30 is close to a Gaussian.
    nu = pm.Exponential("nu", 1/30)
    #
    y = pm.StudentT("y", mu=mu, sigma=sigma, nu=nu, observed=data)
    idata_t = pm.sample(1_000)
```

```
Auto-assigning NUTS sampler... •••
```

```
az.plot_trace(idata_t); •••
```



[104]:
```
az.summary(idata_t, kind="stats").round(2)
```

[104]:

	mean	sd	hdi_3%	hdi_97%
mu	53.03	0.38	52.35	53.76
sigma	2.19	0.40	1.45	2.95
nu	4.65	3.92	1.20	9.20

- The outliers decrease the value of ν (i.e., less Gaussian) instead of increasing mean and standard deviation
  - μ is similar to the estimate with a Gaussian
  - σ is smaller
  - $\nu \approx 5$ (not very Gaussian)
- The estimation is more robust since outliers have less effects

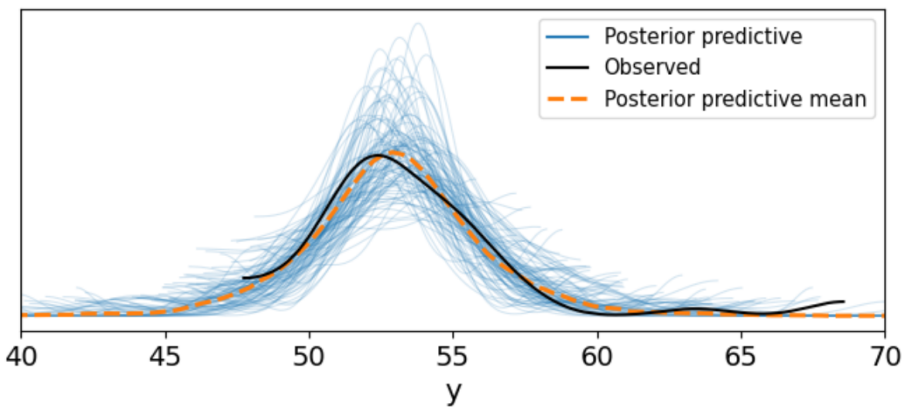
Chemical shift: use Student's t-dist (3/3)

[105]:
```
# Compute 100 posterior predictive samples.
y_ppc_t = pm.sample_posterior_predictive(idata_t, model_t);
```

```
Sampling: [y]
```

```
Sampling ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

[106]:
```
ax = az.plot_ppc(y_ppc_t, num_pp_samples=100, mean=True)
ax.set_xlim(40, 70);
```



- The PPC (posterior predictive check) is a much better fit than with Normal model
- The plot is "hairy" because the KDE is estimated only in the actual interval of the data and it's 0 outside

Groups comparison

- Probabilistic programming
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - Posterior-based decisions
  - Posterior predictive checks
  - Robust inference
  - **Groups comparison**
  - Hierarchical models
- Simple linear model
- Comparing models
- Inference engines

Group comparison

- **Group comparison** is hypothesis testing for statistically significant results when comparing a "treatment" to a "control group"
- E.g., 
  - How well patient respond to a new drug (vs a group receiving nothing or a placebo)?
  - Is there a reduction of car accidents after new traffic regulation?
  - Does college students' performance improves if they don't have cellphone at school?
- **Effect size** quantifies the difference between two groups
  - It moves from questions like "does it work?" (hypothesis testing) to "how well does it work?" (estimate effect size)

Bogus control groups

- When something is claimed to be harder/better/faster/stronger, ask for the baseline used for comparison
- **E.g.,**
  - Sell sugary yogurts to boost the immune system by comparing it to using milk
  - A better control group would be a less sugary yogurt
- **Placebo** is a psychological phenomenon where a patient experiences improvements after receiving an inactive treatment
  - Using a placebo is better than "no treatment"
  - It shows how difficult it is to account for all factors in an experiment

Group Comparison Bayesian-style

- The frequentist approach is to compare the p-value of difference of means in each group
- Bayesian approach is to compare the posterior distribution of the means between groups using:
    - Plot of posterior as reference
    - Cohen's d
    - Probability of superiority

Sample size effect

- **Sample size effect** is the impact of the number of observations (i.e., sample size) has on statistical results (e.g., p-values, confidence intervals)
  - With a small sample, a large mean difference might not be statistical significant (low p-value)
  - With a large sample, a tiny mean difference can be highly significant (small p-value) but be meaningless
  - E.g., Cohen's d

Cohen's d

- **Cohen's d** is the difference of the means with respect to the pooled standard deviation of both groups

$$\frac{\mu_2 - \mu_1}{\sqrt{(\sigma_1^2 + \sigma_2^2)/2}}$$

  - It normalizes the effect by the variability for the sample size for a pooled std dev
  - Bayesian analysis directly estimates the actual std dev
- The variability of each group normalizes the difference between means
  - It's like a Z-score, i.e., number of std dev by which the values differ
- Compute posterior distribution of means and std
- Compute the distribution of Cohen's d or use the formula

Probability of superiority

- $=$ probability that a data point taken randomly from one group is larger than a random point from the other group
- We can compute the

Hierarchical models

- Probabilistic programming
  - Single-parameter inference
  - Communicating a Bayesian analysis
  - Probabilistic programming
  - Posterior-based decisions
  - Posterior predictive checks
  - Robust inference
  - Groups comparison
  - **Hierarchical models**
- Simple linear model
- Comparing models
- Inference engines

Hierarchical models

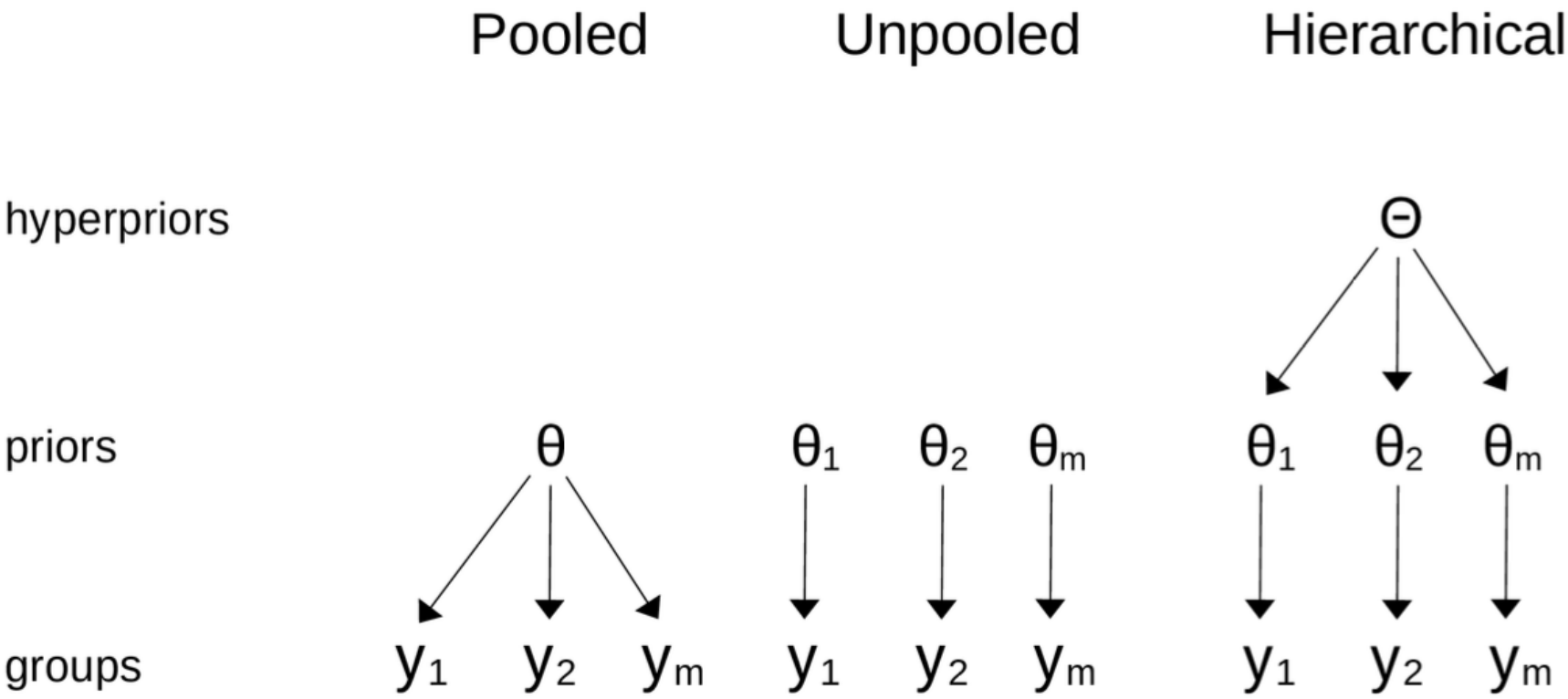
- Aka “multilevel”, “nested”, “mixed-effects” models
- **Observation**: Data points share some common structure, but also have their own variations
  - Data can be grouped together
    - E.g., sales in different cities: each city is a different market, but there are common trends
  - There is a hierarchical structure
    - E.g., students within a school: each student is different, but there are common educational factors
  - Repeated measurements on the same objects
- **Approach**: 
  - Model shares information between groups and allow differences between groups
  - The parameters of the prior distributions have also a prior distribution, aka “hyper-priors”

Hierarchical models:

- Many data structures lend themselves to hierarchical descriptions
- E.g.,
  - Medical research:
    - We want to estimate effectiveness of drugs in treating a disease
    - Categorize the patients based on demographic information, disease severity
    - Estimate probability of cure for each subgroup
  - Market research
    - We want to understand the purchasing behavior of consumers
    - Categorize consumers by age, gender, income, education

Un-pooled, pooled, hierarchical models

- Unpooled
  - Groups have different priors
- Pooled
  - Groups have the same priors
- Hierarchical
  - Groups have different priors which come from a common prior



Hierarchical models: Chemical shift

- Proteins are made out of 20 amino acids
  - Proteins can be studied with nuclear magnetic resonance
  - One of the quantity measured is "chemical shift"
- The data looks like:

	ID	aa	theo	exp	diff
0	1BM8	ILE	61.18	58.27	2.91
1	1BM8	TYR	56.95	56.18	0.77
2	1BM8	SER	56.35	56.84	-0.49
3	1BM8	ALA	51.96	51.01	0.95
4	1BM8	ARG	56.54	54.64	1.90
...	...	...	...	...	...
1771	1KS9	LYS	55.79	57.51	-1.72
1772	1KS9	ARG	58.91	59.02	-0.11
1773	1KS9	LYS	59.49	58.92	0.57
1774	1KS9	GLU	59.48	58.36	1.12
1775	1KS9	SER	58.07	60.55	-2.48

- ID: code of the protein
- aa: name of the amino acid (e.g., ILE, TYR, SER, . . . , PRO)
- theo: theoretical values of chemical shift
- exp: experimental value
- diff: difference between theoretical and experimental value

Hierarchical models: Chemical shift

- We have experimental measures of chemical shifts vs theoretical values
  - We want to evaluate the model using different "styles" of modeling
  1. **Unpooled**
     - There are 20 amino acids and so we should fit 20 Gaussians
     - More detailed analysis, but with less accuracy
  2. **Pooled**
     - Compute the difference between estimates and measures and fit a Gaussian
     - More accurate estimates (since we have more data), but we loose information about different amino acids
  3. **Hierarchical**
     - Build a model to estimate groups assuming they belong to a common population

Chemical shift: unpooled model

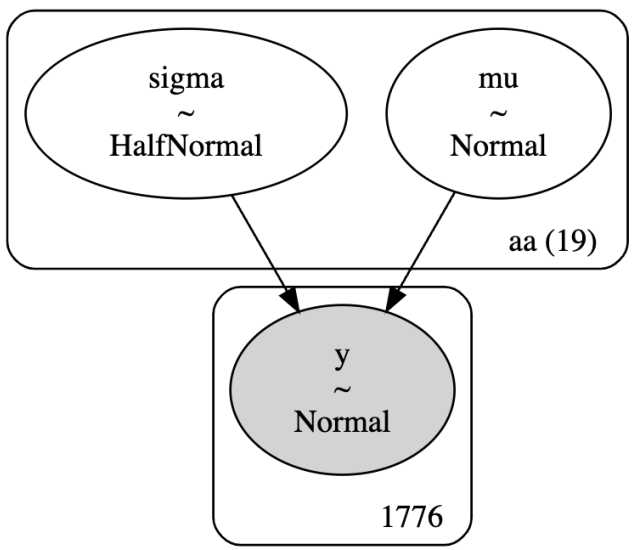
In [9]:
```
# Non-hierarchical model.
with pm.Model(coords=coords) as cs_nh:
    # One separate prior for each group.
    mu = pm.Normal('mu', mu=0, sigma=10, dims="aa")
    sigma = pm.HalfNormal("sigma", sigma=10, dims="aa")
    # Likelihood.
    y = pm.Normal("y", mu=mu[idx], sigma=sigma[idx], observed=diff)
    idata_cs_nh = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mu, sigma]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_
```

In [10]:
```
pm.model_to_graphviz(cs_nh)
```

Out[10]:



- We model each group independently
- Same structure of model to compare different groups

Chemical shift: hierarchical model

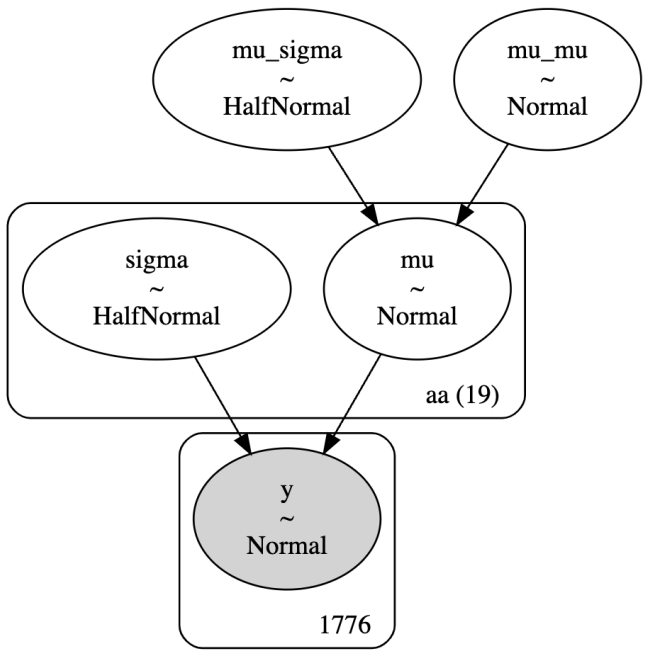
In [12]:
```
with pm.Model(coords=coords) as cs_h:
    # Hyper-priors.
    mu_mu = pm.Normal("mu_mu", mu=0, sigma=10)
    mu_sigma = pm.HalfNormal("mu_sigma", sigma=10)

    # Priors.
    mu = pm.Normal("mu", mu=mu_mu, sigma=mu_sigma, dims="aa")
    sigma = pm.HalfNormal("sigma", sigma=10, dims="aa")

    # Likelihood (same as before).
    y = pm.Normal("y", mu=mu[idx], sigma=sigma[idx], observed=diff)
    idata_cs_h = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mu_mu, mu_sigma, mu, sigma]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_
```

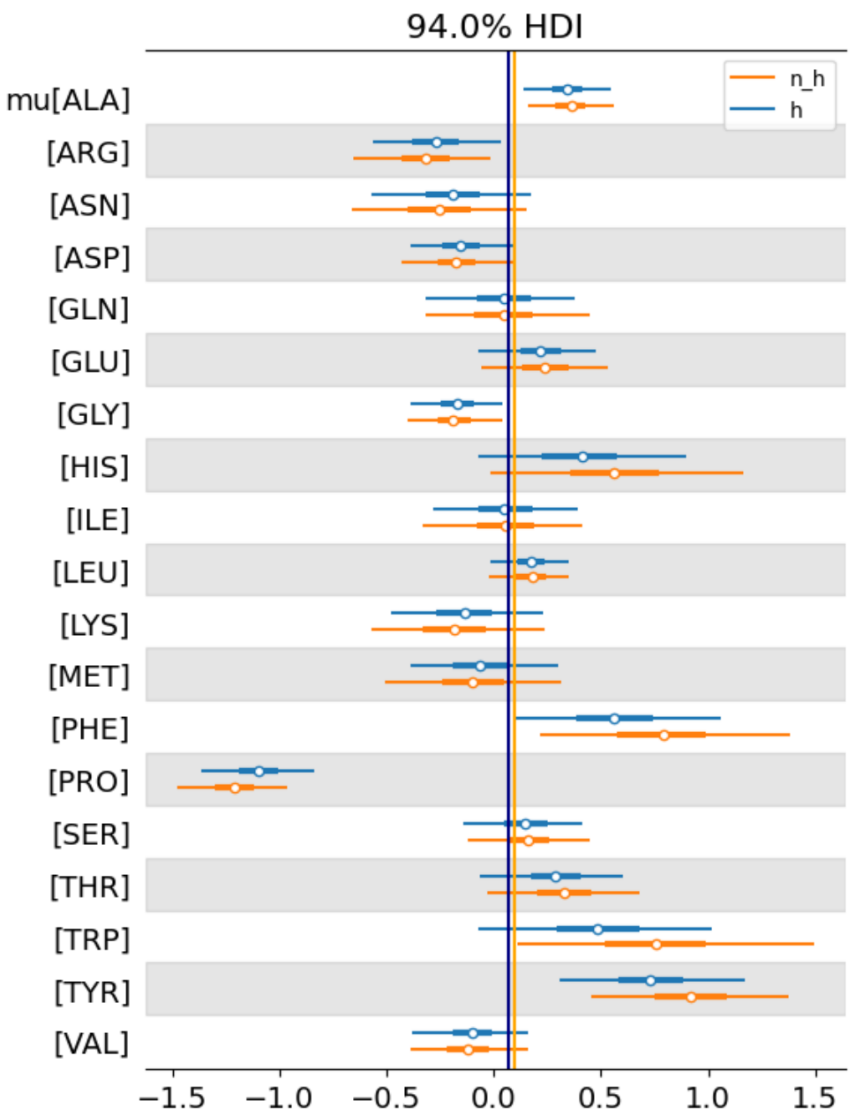
In [13]:
```
pm.model_to_graphviz(cs_h)
```

Out[13]:



- Add two hyperpriors, one for the mean of μ and one for the standard deviation of μ
- We assume that the variance is the same for all groups
  - This is a modeling choice
  - One could also add hyperpriors for σ
- It is an intermediate situation between a single group and 20 separate groups

Chemical shift: results



- We have two models and we want to compare the estimates
- There are 20 groups and each model has 4 estimates
  - We plot the 94% credible intervals
  - The black vertical line is the global mean according to the hierarchical model
  - The blue (hierarchical) means are pulled towards the mean with respect to the orange (non-hierarchical) ones
  - There is "shrinkage"

Shrinkage

- Hierarchical models shrink parameters towards a common mean, as groups share information through the hyper-prior
  - Groups are modeled neither as independent nor as a single group, but as something in between
  - The model is less responsive to extreme values in individual groups
  - A hierarchical model improves estimation for small groups using data from other groups
- The amount of shrinkage depends on the data:
  - Groups with more data influence the estimate more than those with fewer data points
  - Similar groups reinforce common estimation
- **Result**: inference is more stable

Football example

- We have data for different leagues over 4 years
- We want to compute goals-per-shot metric
  - We can estimate it with n shots and y goals
- θ is the success rate for each player
  - There are `n_players`
- The success rate $gs \sim Beta(\theta)$
- The hyperparams of the Beta distributions are 4-dim vectors μ_p and ν_p for each position
  - `DF` defender
  - `MF` midfielder
  - `FW` forward
  - `GK` goalkeeper

You need to know when to stop

- One can create hierarchical models with as many levels as we want
- The more levels:
  - Don't improve the quality of the inference
  - Make the interpretation difficult
- The goal is to take advantage of the structure of the data

Simple linear model

- Probabilistic programming
- **Simple linear model**
  - Logistic regression
  - Variable variance
  - Multiple linear regression
- Comparing models
- Inference engines

Linear model

- Many problems can be stated as in the following:
  - X and Y are uni-dimensional continuous RVs
  - Given a dataset of paired observations $\{(x_1, y_1), ..., (x_n, y_n)\}$
  - X can be considered an independent variable
  - We want to model/predict variable (dependent) Y
- E.g.,
  - Average temperature vs Number of Nobel laureates in that country
  - Daily sugar intake vs Blood glucose level
  - Years of education vs Annual income
  - Advertising spend vs Product sales
  - Number of hours studied vs Exam score
  - Coffee consumption vs Productivity rating

Linear Model: Frequentist Approach

- A linear model is described by:

$$y = \alpha + \beta x$$

- Frequentist approach:
  - Find the parameters α, β using least square fitting
    - I.e., pick the values that minimizes the average quadratic error between observed y and predicted $\hat{y}$
  - Point-estimate from least squares corresponds to a maximum a-posteriori with flat priors

Linear Model: Bayesian Approach

- A linear model is described by:

$$y = \alpha + \beta x$$

- The Bayesian approach assumes
  - The data is distributed as a Gaussian with mean $\alpha + \beta x$ and standard deviation σ: $y \sim N(mu = \alpha + \beta x, \sigma)$
  - α, β, σ are independent
  - The priors on the values are:

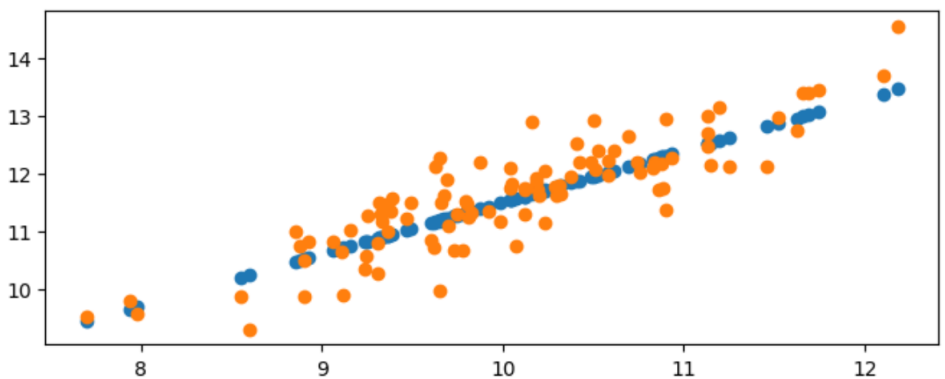
$$\alpha \sim Normal(\mu_\alpha, \sigma_\alpha)$$
$$\beta \sim Normal(\mu_\beta, \sigma_\beta)$$
$$\sigma \sim HalfNormal(0, \sigma_\epsilon)$$

- We can use **vague priors**:
  - The prior of the intercept α is often centered around 0
  - Sometimes we have info on the sign of β
  - Half-Cauchy / exponential distribution for σ is a good way to regularize
  - Uniform prior if the parameter is restricted to hard boundaries

Linear model: synthetic example

[122]:
```
np.random.seed(1)

# Number of samples.
N = 100

# Parameters.
alpha_real = 2.5
beta_real = 0.9
sigma_eps_real = 0.5

# Generate data.
x = np.random.normal(10, 1, N)
y_real = alpha_real + beta_real * x

# Add noise.
eps_real = np.random.normal(0, sigma_eps_real, size=N)
y = y_real + eps_real
```

[123]:
```
plt.scatter(x, y_real)
plt.scatter(x, y);
```



- Generate random data from the ground truth
  - $\alpha = 2.5$
  - $beta = 0.9$
  - Add some noise

Linear model: synthetic example

[125]:
```
with pm.Model() as model_g:
    alpha = pm.Normal("alpha", mu=0, sigma=10)
    beta = pm.Normal("beta", mu=0, sigma=1)
    sigma = pm.HalfCauchy("sigma", 5)
    #
    mu = pm.Deterministic("mu", alpha + beta * x)
    y_pred = pm.Normal("y_pred", mu=mu, sigma=sigma, observed=y)
    idata_g = pm.sample(2000, tune=1000)
```

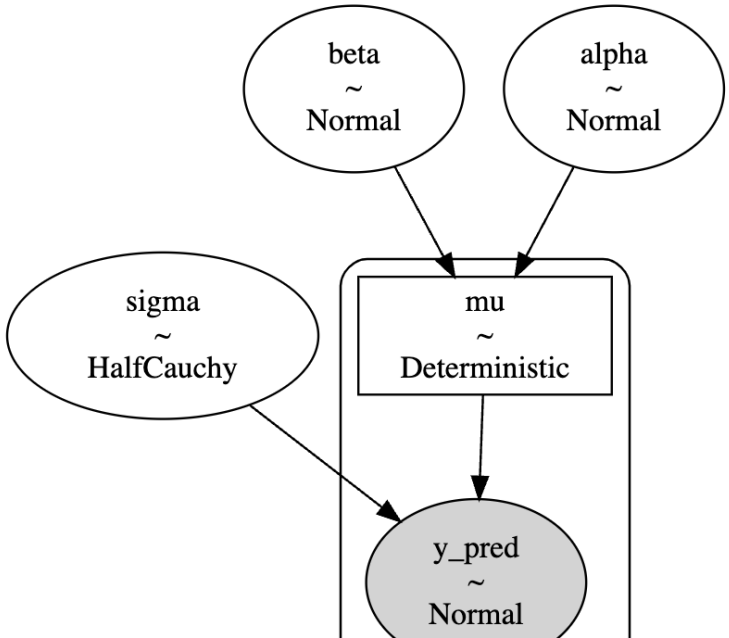
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, sigma]
```

```
Sampling 4 chains, 0 divergences
```

```
Sampling 4 chains for 1_000 tune and 2_000 draw iterations (4_000
```

[127]:
```
pm.model_to_graphviz(model_g)
```

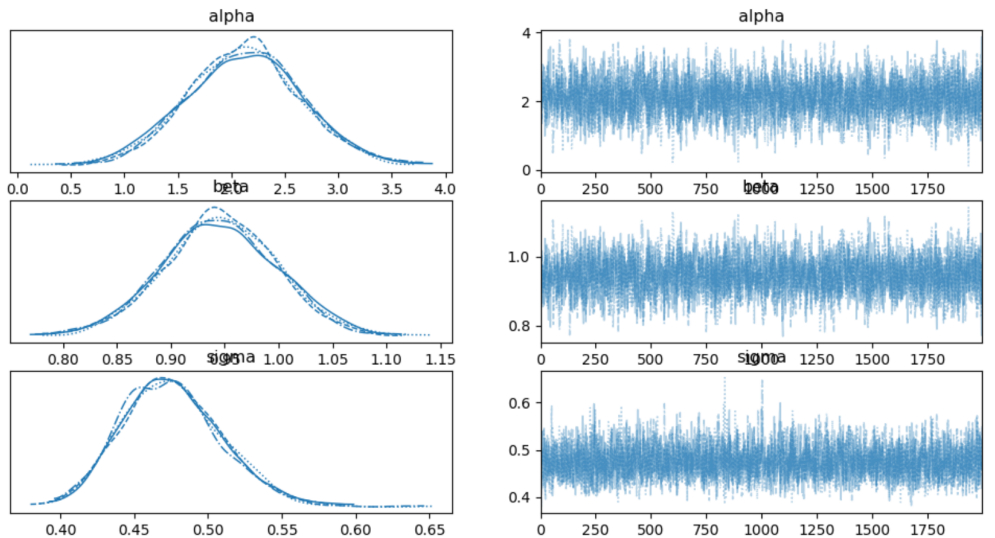
[127]:



- Create linear model in PyMC with some vague priors

Linear model: synthetic example

[129]:
```
az.plot_trace(idata_g, var_names=["alpha", "beta", "sigma"]);
```



[128]:
```
az.summary(idata_g, var_names="alpha beta sigma".split(), kind="stats")
```

[128]:

	mean	sd	hdi_3%	hdi_97%
alpha	2.118	0.538	1.099	3.136
beta	0.946	0.053	0.845	1.046
sigma	0.476	0.034	0.412	0.538

- Run the sampler
- Ground truth
  - $\alpha = 2.5$
  - $beta = 0.9$
- Recovered values
  - $\alpha = 2.12$
  - $beta = 0.95$

Linear model: bike rental example

- We want to model relationship between temperature X and number of bikes rented in a city Y
- Pics
- Use an intermediate variable $\mu = \alpha + \beta * X$, which is the mean number of bikes rented for a temperature X

$$Y \sim N(\mu, \sigma)$$

  - The fit model is $\mu = 69 + 7.9X$
- Interpreting the model
  - When the temperature is 0, the expected value of rented bikes is 69
  - For every degree of temperature increase, the expected value increases of 7.9
- All parameters have uncertainty
  - The posterior accounts for the combined uncertainty
  - There are bands representing the quantiles [0.25, 0.75] and [0.03, 0.97] of the prediction
- Do you see a problem? 

Linear model: bike rental example (criticism)

- **Problems**
  - The model can output a negative number of bikes which makes no sense
  - The model predict a real number of bikes, while the count is discrete
- It is not surprising since we are using a Normal as likelihood
  - Extends to negative numbers
  - Is continuous
- **Solutions**:
  - We could clip predictions lower than 0 and discretize the output (hack)
  - Use a better model that is defined only for positive numbers

Generalized Liner Model (GLM)

- It is a generalization of the linear model that uses different distributions for the likelihood

$$\alpha \sim \textit{prior}$$
$$\beta \sim \textit{prior}$$
$$\mu = \alpha + \beta X$$
$$\theta \sim \textit{prior}$$
$$Y \sim \phi(f(\mu), \theta)$$

where:

- ϕ is an arbitrary distribution
  - E.g., Normal, Student's t, negative binomial
- θ is an auxiliary parameter for the distribution ϕ
  - E.g., σ for a Normal
- $f()$ is the "inverse link function" which transforms the values of μ from the real line to the domain of ϕ

Count data

- "Count data" is when the RV is a discrete number and bounded at 0
  - E.g., the number of rented bikes
- If the number is large, we can use a continuous distribution (e.g., Gaussian)
- Other times, it's modeled as discrete (e.g., Poisson, negative binomial)

Poisson distribution

- **Poisson distribution**
  - Models the number of events in a fixed interval of time, space, etc
  - Assumes events happen independently and at a constant average rate
- **E.g.,**
  - Call centers: model the number of customers per hour
  - Natural events: model the number of earthquakes in a region over time
  - Traffic flow: model the number of cars passing through a toll booth in a day
  - Biology: count the number of mutations in a DNA segment over time
- **Cons**:
  - Assumes mean equal to variance (can't model "overdispersion")

Negative binomial distribution

- Negative binomial distribution:
  - Models the number of failures/trials to achieve a fixed number of successes in a sequence of IID Bernoulli trials
  - Generalizes the geometric distribution, which models the number of trials needed to achieve the first success
  - Models overdispersion (i.e., when mean is much smaller than variance)
- **E.g.,**
  - Customer service: model the number of unsuccessful customer interactions before achieving success
  - Sports: number of games a team needs to lose before winning a number of games

Overdispersion

- **Overdispersion** occurs when the variance of count data is significantly larger (e.g., 15-20x) than the mean
- **E.g.,**
  - Epidemiology: when modeling disease outbreak, the number of new infections on a given day might fluctuate widely
  - Customer service: some days there are few complaints, other days there are many complaints
- **How to model it?**
  - A Poisson distribution has mean and variance equal 🚫
  - A negative binomial has a second parameter that controls the variance ✅
  - When using a Normal model
    - Largest mismatch when the model predicts a negative number of rented bikes
    - Even on the positive side we see that the fit is not that good
  - When using Negative Binomial model seems to be a better fit, although not perfect
    - The right tail for the predictions than observations, but also the probability of this very high demand is low
    - Overall the Negative Binomial model is better than the Normal model

Robust regression

- "Outliers" pull a regression line away from the bulk of the data
- Student's t-distribution has heavier tails than Normal, gives less importance to "outliers"

Logistic regression

- Probabilistic programming
- Simple linear model
  - **Logistic regression**
  - Variable variance
  - Multiple linear regression
- Comparing models
- Inference engines

Logistic regression

- Logistic regression model
  - Is a generalized linear model
  - Models the response variable as binary
  - E.g., ham/spam, safe/unsafe, cloudy/sunny, hotdog/not hotdog
- The logistic model is:

$$\theta = \mathit{logit}(\alpha + \beta x)$$
$$y \sim \mathit{Bernoulli}(\theta)$$

- Logistic function (aka sigmoid)

$$\mathit{logit}(z) = \frac{1}{1 + \exp^{-z}}$$

  - Is the "inverse link function"
  - Converts real numbers from θ to [0, 1] for the Bernoulli distribution

Iris dataset

- Classical dataset of measurements from flowers from 3 closely related species of Iris setosa, virginica, and versicolor
  - E.g., we want to predict the probability of a flower being `setosa` from the `sepal_length`

Classification with logistic regression

- "Classification with logistic regression" might seem a misnomer
  - **Regression** predicts a continuous variable given one or more input variables
  - **Classification** predicts the output as a discrete variable given one or more input variables
- Logistic regression:
  - Is a "regression" because it computes the probability that the output is a certain value
  - Models $\theta = \Pr(Y = 1|X)$
  - Can classify the output using a "decision rule" for classification, e.g.,

$$\Pr(Y = \text{versicolor}|\text{sepal\_length}) > 0.5$$

Boundary decision for a classifier

- The **boundary decision** δ for a classifier is the values of the independent variables that makes the prob equal to 0.5
- For logistic regression δ is given by:

$$0.5 = \mathit{logit}(\alpha + \beta\delta)$$
$$0 = \alpha + \beta\delta$$
$$\delta = -\frac{\alpha}{\beta}$$

- The decision boundary has uncertainty because we have uncertainty around the values of α and β also
- Note that 0.5 is chosen since we are ok with misclassifying a data point in either direction
  - Some times the misclassification cost is not symmetrical
  - E.g., minimize false negatives ("the patient has a disease, but we don't predict it") or false positives ("the patient doesn't have a disease, but we predict they do")

Odds

- The quantity, called the "odds of $y = 1$", is:

$$\text{odds} \stackrel{def}{=} \frac{\Pr(y = 1)}{1 - \Pr(y = 1)}$$

  i.e., probability of success over probability of failure

- Odds is the ratio between favorable vs unfavorable events
  - More intuitive than probability when "gambling"
  - E.g., the odds of getting a 2 rolling a fair die are $\frac{1/6}{5/6} = 1/5 = 0.2$ so it's one favorable event for 5 unfavorable events
  - There is a transformation between probability, odds, and log-odds
- Interpreting logistic regression in terms of odds
  - Since $\theta = \mathit{logit}(\alpha + \beta x)$ and $\theta = \Pr(y = 1)$ we get the expression:

$$\alpha + \beta x = \log(\frac{\Pr(y = 1)}{1 - \Pr(y = 1)})$$

  - Logistic regression models log-odds as linear regression
  - β is the increase of log-odds for a unit change of x

Variable variance

- Probabilistic programming
- Simple linear model
  - Logistic regression
  - **Variable variance**
  - Multiple linear regression
- Comparing models
- Inference engines

Heteroskedasticity

- **Heteroskedasticity** is when the variance of the errors is not constant in all the observations
  - E.g., the variance can be a linear function of the dependent variable
- E.g.,
  - The variance of height of babies as function of their age is heteroskedastic
  - σ is a linear function of the predictor variable
  - The mean μ is square root of a linear model

Multiple linear regression

- Probabilistic programming
- Simple linear model
  - Logistic regression
  - Variable variance
  - **Multiple linear regression**
- Comparing models
- Inference engines

Multiple linear regression

- It is not unusual to have several independent variables
  - E.g., student's grades = f(family income, mother's education, . . .)
- There are k independent variables and N observations

$$\mu = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

  - In practice we find an hyperplane of dimension k that explains the data
- The functional form of multiple linear regression is the same as polynomial regressions but the independent variables are powers of the same one

Multiple linear regression: synthetic example

Multiple linear regression: rented bike example

Comparing models

- Probabilistic programming
- Simple linear model
- **Comparing models**
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - Regularizing priors
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines

Models as maps of the real world

- Models are:
  - Approximations that help understand a particular problem
  - Not copies of the real world, but like a map of the real world
    - All models are wrong since they are not the actual territory
    - Not all models are equally wrong: some models are better than others at describing a given problem
- Models have a purpose in mind
  - A model can't reproduce all aspects of a problem equally well
  - Different models often capture different aspects of data

Posterior predictive checks

- The goal of PPC is:
  - Evaluate how well models explain the data used to fit the model
  - Understand what are the limitations of the model
  - Improve the model
- Given some data from a parabola + noise
  - Fit data with a linear
  - Fit data with a quadratic model
  - Compare various quantities of the predicted data from the model (i.e., posterior) vs the observed data
- Compare the KDE of the observed data and the predicted data from the models
  - The linear model KDE doesn't match the data
  - The quadratic model KDE is a better match
  - For both models there is a lot of uncertainty, especially at the tails of the distribution since we don't have many data points
- Compare the mean / the interquartile range for the data vs the model
  - E.g., plot the dispersion of mean and IQR for models vs data
  - The data set gives us a single point
  - The posterior gives us a distribution
- In general a statistics that is "orthogonal" to what the model is explicitly modeling is more informative

Bayesian p-value for a statistic

- A Bayesian p-value summarizes the comparison between simulated and observed data
- **Procedure**
  - Given the posterior predictive $\tilde{Y}$
  - Choose a summary statistic T (E.g., mean, median, standard deviation)
  - Compute T for:
    - The observed data T_{obs}
    - The simulated data T_{sim} from the posterior predictive
  - Compute the Bayesian p-value as the portion of simulated datasets where the test statistic is smaller than the observed data:

$$\text{Bayesian p-value} \stackrel{def}{=} \Pr(T_{sim} \geq T_{obs} | \tilde{Y})$$

  - If observed values agree with predicted ones, the value should be around 0.5

Bayesian p-value for entire distribution

- Instead of using a summary statistic, one can compute "the probability of predicting a lower or equal value for each observed value"
- If the model is well calibrated, it captures all observations equally well, the probability should be the same for all observed values
    - The output should be a uniform distribution

Bayesian p-value: example

- Study the height of people in a population
- **Fit the Bayesian model**
  - Assume a normal distribution with unknown mean and variance
  - Collect observed data of heights (e.g., 100 people)
  - Specify a prior distribution for mean and variance
  - Combine observed data with prior to obtain a posterior distribution of mean and variance of population height
- **Compute Bayesian p-value**
  - From posterior distribution:
    - Generate new simulated datasets
    - For each dataset, compute mean height
  - Use test statistic T, as the difference between the mean of the replicated dataset and the observed mean
  - Compute Bayesian p-value: the proportion of replicated datasets where the test statistic is $>=$ test statistic for observed data
    - A value close to 0.5 means the observed data is covered by the model
    - A value close to 0 or 1 indicates a poor fit

Bayesian vs frequentist p-value

- **Frequentist p-value** is the probability of getting observed data as or more extreme, assuming the null hypothesis is true
- **Bayesian p-value** is the probability that simulated data from the model (i.e., posterior predictive check) is as or more extreme than the observed data
- P-value measures inconsistency between observed data and:
  - a null hypothesis (frequentist approach)
  - model (Bayesian approach)
- Does p-value incorporate uncertainty?
  - (frequentist) No: use single point estimates
  - (Bayesian) Yes, incorporate uncertainty of parameter estimates

The balance between simplicity and accuracy

- Probabilistic programming
- Simple linear model
- Comparing models
  - **The balance between simplicity and accuracy**
  - Measures of predictive accuracy
  - Regularizing priors
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines

Occam's razor

- *"If you have **equivalent** explanations for the same phenomenon, you should choose the **simpler** one"*
  - Quality of explanation $\approx$ accuracy
  - Simpler $\approx$ number of model parameters
- Ideally we should balance complexity and accuracy in a quantitative way
- Complexity vs accuracy
  - Increasing model complexity (e.g., number of model parameters) is accompanied by increasing in-sample accuracy, but not necessarily out-of-sample accuracy
  - The complex model:
    - Did not "learn" from the data but just "memorize" it
    - Do a bad job generalizing to predict potentially observable data

Overfitting and underfitting

- A model is **overfit** when it has many parameters, fitting the training data well but unseen data poorly
  - Overfitting in terms of signal/noise:
    - Each dataset has "signal" and "noise"
    - We want the model to learn the signal
    - A model overfits when it learns the noise, obscuring the signal
- A model is **underfit** when it has few parameters, fitting the dataset poorly
  - An underfit model doesn't learn the signal well
  - E.g., a constant fits a dataset, only learning the mean

Bias-variance trade-off

- A model has **high bias** when it has low ability to accommodate the data (i.e., underfitting)
  - E.g., a polynomial of degree 0
- A model has **high variance** when it has high capacity and it is sensitive to details in the data, capturing noise (i.e., overfitting)
  - E.g., a polynomial of degree 5
- Trade-off between bias and variance
  - Goal: balance simplicity and goodness of fit
  - Aim for a model that "fits the data right," avoiding overfitting or underfitting

Measures of predictive accuracy

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - **Measures of predictive accuracy**
  - Regularizing priors
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines

Accuracy measures

- **In-sample accuracy** is measured on the data used to fit a model
- **Out-of-sample accuracy** is measured on data not used to fit a model
  - Aka "predictive accuracy"
- In-sample accuracy $>$ out-of-sample accuracy
- There is a trade-off between how much data is used for training and for evaluating true accuracy

Information criteria: intuition

- Information criteria are tools to compare models in terms of fitting the data taking into account their complexity through a penalization term
  - out-of-sample accuracy $\approx$ in-sample accuracy $+$ a term penalizing model complexity

Model parameters for Bayesian vs non-Bayesian set-up

- Model parameters θ are:
  - A point estimate (frequentist)
  - A distribution estimated from the posterior (Bayesian)

Squared error metric

- Squared error between the data and the predictions from the model

$$\frac{1}{N}\sum_i (y_i - \mathbb{E}[y_i|\theta])^2$$

  where $\mathbb{E}[y_i|\theta]$ is the predicted value given the parameters
- Squaring errors:
  - Ensures that errors do not cancel out
  - Emphasizes large errors (vs absolute values)

Log-likelihood

- In a probabilistic set-up, log-likelihood is more natural
- **Log-likelihood** is defined as:

$$\sum_i \log \Pr(y_i|\theta)$$

- Deviance is defined as:

$$-2 \sum_i \log \Pr(y_i|\theta)$$

Maximum likelihood estimation (MLE)

- MLE finds the parameter values that make the observed data most probable
  - Denoted by $\hat{\theta}_{MLE}$
  - It's a point not a distribution
- Procedure:
  - Given the data $x_1, x_2, ..., x_n$
  - Assume it comes from a distribution with an unknown parameter θ
  - Pick the value of θ that makes the data most likely given a likelihood function

$$\begin{cases} L(\theta) = \log \Pr(x_1, x_2, ..., x_n | \theta) \\ \hat{\theta}_{MLE} = \text{argmax}_\theta L(\theta) \end{cases}$$

- In Bayesian terms, MLE is equivalent to the mode of θ using flat priors
  - Aka MAP (maximum a posteriori)

Akaike information criterion (AIC)

- AIC is defined as

$$AIC = -2 \sum \log \Pr(y_i | \hat{\theta}_{MLE}) + 2\text{num}_{params}$$

  where:
  - $\hat{\theta}_{MLE}$ is the maximum likelihood estimation of θ
  - num_{params} is the number of parameters
- **Interpretation**:
  - The first term measures how well the model fits the data
  - The second term penalizes complex models
- **Cons**:
  - Discard information about uncertainty of posterior estimation
  - MLE assumes flat priors (vs informative and weakly informative priors)
  - Number of parameters is not always a good measure of complexity
    - E.g., in hierarchical models the effective number of params is smaller

Bayesian information criteria

- Widely Applicable Information Criteria (WAIC)
  - Bayesian version of AIC
  - It has two terms:
    - One that measures how good the fit is
    - One that penalizes complex models
  - WAIC uses the posterior distribution to estimate both terms ‘
- Bayesian Information Criteria (BIC)
  - It is not fully Bayesian
    - Like AIC, it assumes flat priors and uses MLE

Cross-validation

- Cross-validation (CV)
  - Simple and effective solution to use all data to compare models
  - **Procedure**
    - Partition data into K portions of equal size and similar statistics
    - Use $K - 1$ partitions to train the model and the remaining partition to test it
    - Repeat K times
    - Average the results
- Leave-one-out cross-validation (LOO-CV)
  - **Procedure**:
    - The model is fit for all data, excluding one observation
    - The model's predictive accuracy is tested on the left out observation
    - Repeat the process for all observations and average
  - **Cons**
    - It is very computationally expensive since one needs to refit the model
- How to adapt cross-validation to a Bayesian approach?
  - CV and LOO require multiple model fits and fitting a Bayesian model is very expensive
  - Yes! There is a way to approximate using a single fit to the data

ELPD with LOO-CV

- We want to compute $ELPD_{LOO-CV}$ where:
  - "Expected Log-Pointwise predictive Density" (ELPD)
    - It should be ELPPD and not ELPD!
  - We use "Leave-One-Out Cross-Validation" (LOO-CV) to compute it
- The definition of ELPD with LOO-CV is:

$$ELPD_{LOO-CV} = \sum_{i=1}^{n} \log \int p(y_i|\theta)(\theta|y_{-i})d\theta$$

  - where:
    - Fit a model using all the data without y_{-i}
    - Predict the unseen y_i with the model
    - Integrate on all the posterior values
    - Repeat for all the points
- How to compute it efficiently?
  - Use "Pareto smooth importance sampling leave-one-out cross-validation"

Pointwise predictive density (PPD)

- The **pointwise predictive density** (or probability) for a given data point y_i is defined as the posterior predictive probability, given the rest of the data

$$PPD \stackrel{def}{=} \Pr(y_i|data - \{i\}) = \int p(y_i|\theta)p(\theta|y_{-i})d\theta$$

  where:
  - y_i is the observed data point
  - $p(y_i|\theta)$ is the likelihood of the data point y_i given the model parameters θ
  - $p(\theta|y_{-i})$ is the posterior distribution of the model parameters given the rest of the observed data
  - The integral averages over the posterior distribution of θ, capturing the uncertainty about the model parameters
- **Interpretation**
  - PPD measures how well a model, training on the data excluding y_i, predicts y_i
  - It's like cross-validation but using the Bayesian concept of averaging over the distribution of the parameters

Expected Log Pointwise Predictive Density

- The ELPD is the average over unseen points of the log PPD

$$ELPD = \sum_{i=1}^{n} \log \int p(y_i|\theta_{-i}) p(\theta_{-i}|y_{-i}) d\theta$$

- **Interpretation**
  - It can be used to determine which model generalizes better to new data
  - ELPD measures the predictive accuracy of a Bayesian model on unseen data
  - Train on y_{-i}, i.e., all data excluding y_i
  - Test on y_i

Approximating PPD

- Calculating analytically the pointwise posterior density integral

$$PPD = \int p(y_i|\theta)p(\theta|y_{-i})d\theta$$

  is difficult
  - The posterior $p(\theta|y_{-i})$ rarely has a closed form
  - The integral on θ is on a high-dimensional space
- It can be approximated numerically given posterior samples s of the model parameters $\theta^{(s)}$

$$PPD \approx \frac{1}{S}\sum_s p(y_i|\theta_{-i}^{(s)})$$

  - Suppose we already have posterior samples $\theta^{(s)} \sim p(\theta|y)$ from the full dataset

PSIS-LOO-CV

- Compute the Expected Log Pointwise Predictive Density (ELPD) using Leave-One-Out Cross-Validation (LOO-CV):

$$ELPD_{LOO-CV} \stackrel{def}{=} \sum_i \log \int p(y_i|\theta)p(\theta|y_{-i})d\theta$$

- **Problem**: Train once per point
- **Solution**:
  - Pareto-Smoothed Importance Sampling (PSIS) Leave-One-Out Cross-Validation (LOO-CV) estimates the formula without refitting the model for every point
  - **Importance sampling**:
    - Use the full dataset to approximate the posterior distribution when a single observation is left out
    - Re-weight posterior samples based on importance
  - **Pareto-smoothing**:
    - Stabilize importance weights, reducing the impact of extreme weights
    - E.g., if an observation left out has a large influence on the posterior distribution
    - Provide diagnostics to assess the reliability of importance weights

Predictive accuracy with ArviZ

- If the inference data has the log-likelihood group

  `pm.sample(idata_kwargs="log_likelihood": True)`

  metrics such as WAIC and LOO (with / without ELPD) can be automatically computed
- In the first section
  - The first row is ELPD
  - The second row is the effective number of parameters
- In the second section, there is the Pareto k diagnostic
  - Since all the values are between 0 and 0.7, the approximation can be trusted

Comparing predictive accuracy with ArviZ

- In general the predictive accuracy metrics should be interpreted in relation to other models

Model averaging

- You have multiple models explaining the data: what do you do?
  1. Select a single model
     - Simple solution used in frequentist approach
     - "Model selection"
  2. Report all the models with their informations (e.g., standard errors, posterior predictive checks)
     - Express advantages and shortcomings of the models
  3. Average all the models
     - Build a meta-model using a weighted average of each model
     - Weight prediction by the difference between information criteria (e.g., WAIC, LOO) of the models
     - A hierarchical model is a continuous versions of multiple discrete models

Evidence of data given a model

- The Bayesian way to compare k models is to calculate the evidence of each model $\Pr(Y|M_k)$, i.e., the probability of observed data Y given each model M_k
  - Typically we ignore the evidence when we do parameter inference
- Consider the Bayes theorem for the parameters θ and the data Y, given a model M_k

$$\Pr(\theta|Y, M_k) = \frac{\Pr(Y|\theta, M_k)\Pr(\theta|M_k)}{\Pr(Y|M_k)}$$

- We find the parameters θ that maximizes the ratio, independently of the probability of the evidence

$$\text{argmax}_\theta \Pr(\theta|y, M_k) = \text{argmax}_\theta \Pr(y|\theta, M_k)\Pr(\theta|M_k)$$

- Even if we need to choose the best model among $M_1, ..., M_k$ we can pick the one that maximizes

$$\text{argmax}_k \Pr(M_k|y) \propto \Pr(y|M_k)\Pr(M_k)$$

Bayes factors

- The Bayes factors are defined as the ratio of the two marginal likelihoods under competing hypotheses

$$BF = \frac{\Pr(y|M_0)}{\Pr(y|M_1)}$$

where $BF > 1$ means that the model 0 explains the data better than model 1 | **Bayes factor** | **Support** | |———-|———-| | 1-3 | Anecdotal | | 3-10 | Moderate | | 10-30 | Strong | | 30-100 | Very strong | | >100 | Extreme |

- Intuition
  - Bayes factors are a quantitative tool that helps compare how likely two competing explanations (i.e., models) are, given the evidence you find
  - Bayes factors are like a scale that weigh how much evidence supports one theory over another

Assumption of Bayes factors

- The assumption of Bayes factor is that the models have the same prior probability
- Otherwise we need to compute the "posterior odds" as "Bayes factors" x "prior odds"

$$\frac{\Pr(M_0|y)}{\Pr(M_1|y)} = \frac{\Pr(y|M_0)}{\Pr(y|M_1)} \frac{\Pr(M_0)}{\Pr(M_1)} = \text{Bayes factors} \times \text{prior odds}$$

Bayes factors: pros and cons

- Looking at the definition of marginal likelihood (aka evidence):

$$p(y) = \int_{\theta} p(y|\theta)p(\theta)d\theta$$

- Making the dependency of the model M_k explicit

$$p(y|M_k) = \int_{\theta_k} p(y|\theta_k, M_k)p(\theta_k, M_k)d\theta_k$$

- Pros
  - Models with more parameters have a larger prior, so the Bayes factor has a built-in Occam's Razor
- Cons
  - The marginal likelihood needs to be computed numerically over a large dimensional space
  - The marginal likelihood depends on the value of the prior
    - Changing the prior might not affect the inference of θ but have a direct effect on the marginal likelihood

Hierarchical models: candies in a jar examples

- Each classroom has a jar filled with candies, each different but coming from the same candy shop
- Kids in each classroom need to guess the number of candies in each jar
- Individual guesses
  - Think of each jar as its own little puzzle
  - E.g., guess based on how big the jar is, how filled it is
  - Each jar has certain "parameters"
- Group learning
  - Consider what you learn from other jars since they come from the same candy shop
  - E.g., the shop prefers to use a certain type of candies, or fills the jar up to a certain level
  - The jars have certain "hyper-parameters"
- Sharing info
  - As you make more guesses, you start sharing what you have learned with your friends about each jar
  - The hierarchical model lets the info flow across models for individual jar

Computing Bayes factors as hierarchical models

- The computation of Bayes factors can be framed as a hierarchical model
  - The high-level parameter is an index assigned to each model and sampled from a categorical distribution
- We perform inference of the competing models at the same time, using a discrete variable jumping between models
  - The proportion we use to sample each model is proportional to $\Pr(M_k|y)$
- Then we compute the Bayes factors
- The models can be different in the prior, in the likelihood, or both

Common problems when computing Bayes factors

1. If one model is better than the other, then we will spend more time sampling from it
   - Cons: under-sample one of the models
2. Values of the parameters are updated, even when the parameters are not used to fit that model
   - E.g., when model 0 is chosen, the parameters in model 1 are updated, but they are only restricted by the prior
   - If the prior is too vague, the parameter values might be too far from previous accepted values and the step is rejected
   - TODO: ?

- Solutions to improve sampling
  - Force both models to be visited equally
  - Use "pseudo priors"

Using sequential Monte Carlo to compute Bayes factors

- TODO

Bayes factors and information criteria

-
- If we take the log of Bayes factors, we turn ratio of marginal likelihood into a difference, which is similar to comparing differences in information criteria
- We can interpret each marginal likelihood as having:
  - a fitting term (i.e., how well the model fits the data)
  - penalizing term (i.e., averaging over the prior)
    - more parameters $\rightarrow$ more diffused the prior $\rightarrow$ greater penalty
-
- TODO

Regularizing priors

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - **Regularizing priors**
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines

Priors and regularization

- Using weakly/informative priors is a way of pushing a model to prevent overfitting and generalize well
- This is similar to the idea of "regularization"
- Regularization
  - Reduce information that a model can represent and reduce chances to capture noise instead of signal
  - E.g., penalize large values for the parameters in a model
  - E.g., ridge and Lasso regression applies regularization to least square method

Popular regularization methods in Bayesian framework

- Ridge regression
  - Normal distribution for coefficients of linear model, pushing them toward zero
- Lasso regression
  - MAP of posterior using Laplace priors for coefficients
  - Because Laplace distribution looks like Gaussian with a sharp peak at zero, it provides "variable selection" since it induces sparsity of model

Mixture models

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - Regularizing priors
  - **Mixture models**
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines

Mixture models

- Mixture models are models that assume that data comes from a mixture of distributions or where the solution can be approximated as a mix of simpler distributions
- Mixtures can model data from sub-populations
  - E.g., distribution of heights in adult human population, made of male and female
  - E.g., clustering of handwritten digits (e.g., 10 sub-populations)
- Mixture models as statistical trick to handle complex distributions
  - E.g., mix of Gaussians to describe an arbitrary distribution
    - E.g., multimodal, skewed distributions
    - The number of distributions depends on the accuracy of the approximation and the details of the data
  - E.g., Kernel density estimation (KDE) is a (non-Bayesian) non-parametric estimation technique
    - place a Gaussian with a fixed variance on top of each data points
    - use the sums of all the individual Gaussians to approximate the empirical distribution of the data

Finite mixture models

- $=$ the PDF of the observed data is a weighted sum of the PDFs of K subgroups

$$p(y) = \sum_{i=1}^{K} w_i p(y|theta_i)$$

  where:
  - K is finite
  - $w_i \in [0, 1]$ (it's like the probability of component i)
  - $\sum_i w_i = 1$
  - $p(y|\theta_i)$ are simpler distributions (e.g., Gaussian or Poisson)

Conceptual solution of a mixture model

- The assumption of a mixture model is that the data is generated by K underlying distributions / components
  - In other words, each data point is assumed to come from one of the components, but we don't know which
- The goal is to determine which component of the model k the data point x most likely belongs to
  - For each point x, assign probabilities of belonging to one of the K components $\Pr(k|x)$
  - This is modeled as a random variable, which is called "latent variable" since it can't be observed
  - E.g., for two components ($K = 2$) we use a Bernoulli, using a Beta distribution as prior (since it is conjugate prior for Bernoulli)
  - E.g., for K outcomes we can use as prior
    - a Categorial distribution to generalize the Bernoulli
    - a Dirichlet distribution to generalize the Beta distribution
- One can estimate the likelihood of the observed data based on:
  - Parameters of the individual components
  - Probabilities that a given data point belongs to each component

Categorical distribution

- Discrete distribution representing "rolling a K-sided unfair die" or "choosing a random card from a deck of cards"
  - It generalizes a Bernoulli distribution $K = 2$
- It is parametrized with probabilities for each possible outcome $(p_1, p_2, ..., p_K)$, where p_i is the probability of outcome i

Dirichlet distribution

- Defined in a simplex which is a generalization of a triangle in K-dimensions, where K vectors elements are in $[0, 1]$ and sum to 1
  - A 1d simplex is an interval
  - A 2d simplex is a triangle
  - A 3d simplex is a tetrahedron
  - ...
- Dirichlet distribution generalizes the Beta distribution
  - Beta models the probability of a single proportion (e.g., the probability of success in a Bernoulli trial)
  - Beta models 2 outcomes, one with probability p and the other with $1 - p$
  - It uses two parameters to describe its shape
- The Dirichlet distribution models the probability of K outcomes
  - It represents the uncertainty over the proportions of different outcomes in a multinomial experiment

Dirichlet distribution: examples

- Bucket of colored balls
  - You have a bucket of colored balls, each ball comes in 3 colors
  - You want to figure out the probability of picking a ball of each color
  - You are uncertain about the probability and you model that uncertainty
- You want to divide a pie into 4 different slices
  - You want to model the size of the slices (making sure they split the entire pie), modeling the uncertainty

Re-parametrization using marginalization

- Consider a mixture model that include latent variables representing the component from which each observation is drawn
- Performing posterior sampling on these models is inefficient for several reasons:
  - Discrete variables introduce discontinuities
  - Latent variable introduces dependencies
- A solution is "marginalization"
  - Removing the latent variable by integrating out from the model
  - The observations are directly modeled from a mixture distribution
  - This makes sampling more efficient
  - The problem is that the model becomes more complex mathematically
  - `pymc` has distributions (e.g., `NormalMixture`) where the latent variable is already marginalized

Non-identifiability of parameters

- Parameters in a model are "not identifiable" when one or more parameters cannot be uniquely determined
- In practice multiple model fits give different parameters corresponding to the same model
  - E.g., given a bimodal distribution sum of two Gaussians, solving the same model can switch the value of the means (aka "label switching problem")
  - E.g., variables with high-correlation in linear models
- Solutions
  - Arrange the means of the components to be in increasing order
    - E.g., in `pymc` add a "potential" to the likelihood (which doesn't depend on the data) so that a constraint is not violated
  - Use informative priors to guide a model towards a canonical representation

How to choose K

- How to decide the number of components K in a finite mixture model?
- Solution
  - Start with a small number of components K
  - Increase K to improve the model-fit evaluation
  - Use model selection techniques to find best trade-off
    - E.g., using posterior-predictive checks, WAIC, LOO, or domain expertise

7.5, Zero-inflated and hurdle models

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - Regularizing priors
  - Mixture models
  - **7.5, Zero-inflated and hurdle models**
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines
-

- When counting things, it is possible to get a count of zero
- Problem: some models (e.g., Poisson, Negative Binomial distribution) can generate fewer zeros compared to the data
- Solutions

Hurdle models

- A Bernoulli model determines if the count variable is zero or a different distribution truncated at 0, i.e., which doesn't assume the value 0
- E.g., hurdle Poisson, NegativeBinomial, Gamma, LogNormal

Zero-inflated vs Hurdle models

- Zero-inflated models are a mixture of zeros and a distribution of zeros and non-zeros
  - The probability $\Pr(x = 0)$ can only be larger than the base distribution, i.e., the probability of zero is "inflated"
- Hurdle models are a mixture of zeros and non-zeros
  - The probability $\Pr(x = 0)$ is independent of the base distribution

Hanging root-ograms

- Useful for treating issues such as over-dispersion and/or excess zeros in count data models

1. Plot the square root of observed and predicted values
   - It makes it simper to adjust for scale differences
2. Bars of observed data are hanging from the expected values
   - It makes it simple to see if it's a good fit or not by comparing the bottom of the values to a base

7.6, Mixture models and clustering

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - Regularizing priors
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - **7.6, Mixture models and clustering**
  - 7.7, Non-finite mixture models
  - 7.8, Continuous mixtures
- Inference engines

Clustering

- Goal: group objects so that the objects in a given group (aka cluster) are "closer" to each other than to those in the other groups
  - Degree of closeness is computed using a distance (e.g., Euclidean distance)
- Clustering is an unsupervised learning task
  - Similar to classification but without knowing the correct labels

Soft- vs hard- clustering

- Soft-clustering computes the probability of each data point belonging to each cluster
- Hard-clustering assigns each point to a single cluster
- One can turn soft-clustering into hard- by assigning each data point to the cluster with the highest probability
  - E.g., assigning points to the cluster with highest probability
  - E.g., using a threshold for probability of 0.5 (as in classification for logistic regression)

Probabilistic clustering

- Aka "model-based clustering"
- = clustering using probabilistic model, i.e., compute the probability of each point belonging to one of the customers
- E.g., we assume that data is generated by a mixture models

7.7, Non-finite mixture models

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - Regularizing priors
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - **7.7, Non-finite mixture models**
  - 7.8, Continuous mixtures
- Inference engines

How many mixture models to use?

- For some problems, it is easy to choose the number of mixtures to use, since we know how many groups there are in the data
  - E.g., when clustering handwritten digits we know there are 10 groups
- For other problems, we can't choose the number of groups a priori and we want it to estimate from the data
  - E.g., Dirichlet process

Parametric vs non-parametric models

- Parametric models have a fixed number of parameters
- Non-parametric models have (theoretically) an infinite number of parameters
  - Very flexible
  - The data decides how many parameters are needed
  - E.g., Gaussian process (GP), Bayesian Additive Regression Trees (BART), Dirichlet process (DP)

Dirichlet process

- Dirichlet distribution is the n-dimensional generalization of beta distribution
- Dirichlet process (DP) is the infinite-dimensional generalization of the Dirichlet distribution
  - A draw from a DP is actually a (discrete) Dirichlet distribution
- A DP has:
  - a base distribution $\mathcal{H}$ (e.g., Gaussian, Poisson, Laplace)
  - α a concentration parameter
    - Increasing α means less and less concentrated realization
    - For $\alpha \to \infty$ the realization of a DP are equal to the base distribution

Categorical distribution

- The categorical distribution (the most general discrete distribution) is parameterized specifying the probabilities of each possible outcome
- It is specified by:
  - Indicating the positions on the x axis and the height on the y axis
  - x positions are integers
  - y heights must sum to 1
- A generalization of this is to have x sampled from $\mathcal{H}$
  - $\mathcal{H}$ to be a Gaussian, so x are on the real line
  - $\mathcal{H}$ to be a Beta, so x are in [0, 1]
  - $\mathcal{H}$ to be a Poisson, so x are non-negative integers 0, 1, 2, ...

Stick-breaking process

- $=$ process to generate values on the y axis so that the sum is 1
- You start with a stick of length 1
- Break it into two parts (not necessarily equal)
  - Use the "concentration parameter" α to control the size of the break
- Save the first part
- Pick the second part and break it again
- Repeat until you get K values

Dirichlet process using stick-breaking

- The locations are sampled from the base distribution
- The weights are controlled by α
- As $\alpha \to \infty$ the DP distribution approximates the base distribution

Infinite mixture model

- We can place a Gaussian on each data point and weight from a Dirichlet process
- The same approach can be use for Laplace distribution

7.8, Continuous mixtures

- Probabilistic programming
- Simple linear model
- Comparing models
  - The balance between simplicity and accuracy
  - Measures of predictive accuracy
  - Regularizing priors
  - Mixture models
  - 7.5, Zero-inflated and hurdle models
  - 7.6, Mixture models and clustering
  - 7.7, Non-finite mixture models
  - **7.8, Continuous mixtures**
- Inference engines

Inference engines

- Probabilistic programming
- Simple linear model
- Comparing models
- **Inference engines**
  - 10.1, Inference engines

10.1, Inference engines

- Probabilistic programming
- Simple linear model
- Comparing models
- Inference engines
    - **10.1, Inference engines**

Inference engines as black

- Bayesian methods are numerically challenging
  - Intractable / computationally intractable integral to solve
- Probabilistic programming separates:
  1. Model building
     - Users should not care how sampling is carried out
  2. Inference process (can be a black box, leave PyMC to handle computation)
- Understanding how the posterior is sampled can help understand how different methods can fail
  - E.g., diagnose samples