# TWEAG

Bayesian data analysis with TensorFlow Probability

www.tweag.io

# Your hosts

**Simeon will give the presentation**

- background in computational biology
- Data Scientist at Tweag since 2019

**Dorran will happily answer questions**

- previous positions in geophysics
- Data Scientist at Tweag since 2019

# Tweag I/O

**TODO**

Tweag I/O is a software innovation lab and consultancy based in Paris with employees all around the world.
We specialize in

- software engineering, with a focus on functional programming
- DevOps, with a focus on reproducible software systems and builds
- data science

# What you're in for

This tutorial consists of alternating blocks of

- theory / example slides
- practical examples on either external websites or Google Colab notebooks. Links are provided at `https://github.com/tweag/tutorial-dsc-2020/`
  **TODO**: provide new repo link

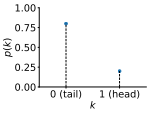
Requirements:

- a Google account (for the practical exercises)
- elementary knowledge in probability theory and statistics

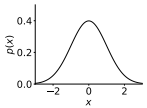
# Reminder: Probabilities

Probability distributions can be...

discrete: $\text{Ber}(\text{k}; \text{b}) = b^k(1-b)^{1-k}$




continuous: $\mathcal{N}(\text{x}; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$




Important concepts:

Conditional probability

$p(A|B)$: probability that $A$ is true, given $B$ is true

Joint probability

$p(A, B)$: probability that both $A$ and $B$ are true

Conditional joint probability

$p(A, B|C)$: probability that both $A$ and $B$ are true, given $C$ is true

# Bayesian vs frequentist probabilities

Example: fair coin flip with (bias $b = \frac{1}{2}$)

**Frequentist probability**

$p(\text{"head"}|b)$:

$\frac{\text{\# of heads}}{\text{\# total flips}}$ for $\infty$ many fair coin flips

**Bayesian probability**

$p(\text{"head"}|b)$:

measure of *belief* in the statement “flip results in head” given single fair coin flip

# Prior beliefs

Assume: unknown bias $b$

**Prior probability**

Encodes prior belief in $b$ *before* flipping the coin

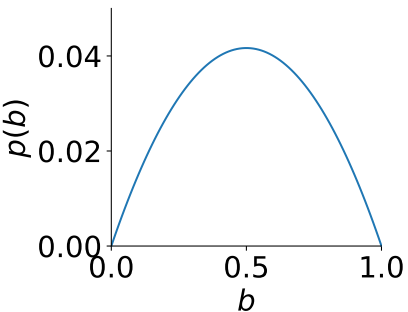
What is known about $b$?

- $b$ is a probability: $0 \leq b \leq 1$
- most coins are fair

$\rightarrow$ choose prior distribution defined between $0$ and $1$, with maximum at and symmetric around $b = \frac{1}{2}$

Example:

$$b \sim \text{Beta}(\alpha = 2, \beta = 2)$$

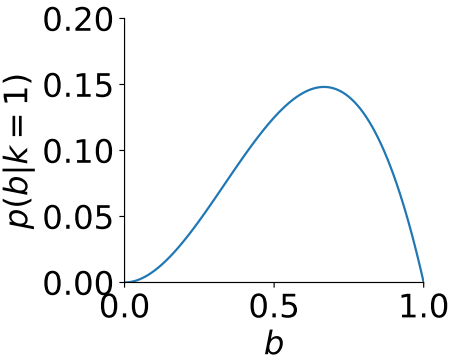


# Posterior belief

Now: flip coin one time, result: head

**Posterior belief**

$p(b|\text{head})$: updated prior belief after obtaining new data



# Update rule

**Bayes' theorem**

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B)}$$

(easily derived from rules for conditional probabilities)

In data analysis:

$$\underbrace{p(x|D,I)}_{\text{posterior}} = \underbrace{p(D|x,I)}_{\text{likelihood}} \times \underbrace{p(x|I)}_{\text{prior}} / \underbrace{p(D|I)}_{\text{evidence}}$$

- $x$: model parameter
- $D$: data
- $I$: prior information (often not made explicit)

# Likelihood

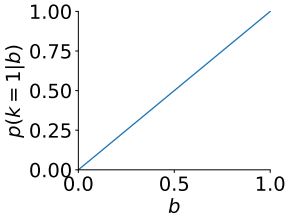
$p(D|x)$: probability of the data given fixed model parameters
$\rightarrow$ models data-generating process

In our case:

$$p(k|b) = \text{Ber}(k; b) \propto b^k (1-b)^{k-1}$$

with

$$k = \begin{cases} 0: & \text{tail} \\ 1: & \text{head} \end{cases}$$



# Evidence

$p(D) = \int \mathrm{d}x\, p(D|x)p(x)$:

normalization constant (long story...)

In our case:

$$
\begin{aligned}
p(k=1) &= \int_0^1 \mathrm{d}b\, L(k=1|b)p(b) \\
&= \int_0^1 \mathrm{d}b\ \ \mathrm{Ber}(k;b) \times \mathrm{Beta}(k;\alpha=2,\beta=2)|_{k=1} \\
&= \int_0^1 \mathrm{d}b\ \ b^k(1-b)^{k-1}\frac{b(1-b)}{\frac{\Gamma(2)\Gamma(2)}{\Gamma(4)}}\Bigg|_{k=1} \\
&\vdots \\
&= \frac{1}{2}
\end{aligned}
$$

# Evidence

$p(D) = \int \mathrm{d}x\, p(D|x)p(x)$:

normalization constant (long story...)

In our case:

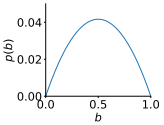
$$
\begin{aligned}
p(k=1) &= \int_0^1 \mathrm{d}b\, L(k=1|b)p(b) \\
&= \int_0^1 \mathrm{d}b\ \ \mathrm{Ber}(k;b) \cdot \mathrm{Beta}(b;\alpha=2,\beta=2)\big|_{k=1} \\
&= \int_0^1 \mathrm{d}b\ \ b^k(1-b) \cdot \frac{b(1-b)}{\frac{\Gamma(2)\Gamma(2)}{\Gamma(4)}}\Bigg|_{k=1} \\
&\ \ \vdots \\
&= \frac{1}{2}
\end{aligned}
$$

YIKES

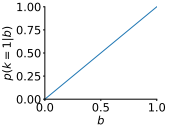
# Update rule

In our coin flip example:

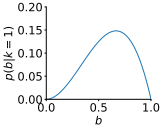
prior: $p(b) = \text{Beta}(b; \alpha = 2, \beta = 2)$
$\propto b(1-b)$




likelihood: $p(D|b) = \text{Ber}(k = 1; b)$
$= b$



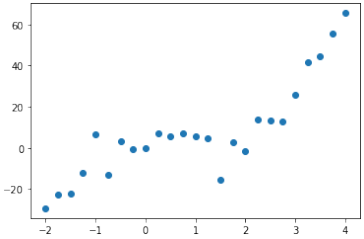

posterior: $p(b|D) \propto p(D|b) \times p(b)/p(D)$
$= \text{Beta}(b; \alpha = 3, \beta = 2)$
$\propto b^2(1-b)$



– no slides for interactive stuff –

# Real-world-ish application: regression

**Step 1:** postulate likelihood $p(D|x)$ or: think about and look at the data




Standard distributions generate certain types of data:

Poisson distribution: counts per interval

Exponential distribution: interarrival times

...

Often: measured data = idealized data + noise

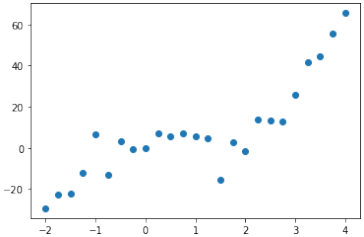
We guess:

idealized data: $\hat{f}(x;\vec{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$

noise: normally distributed

# Real-world-ish application: regression

**Step 2:** find sensible prior distributions




Incorporate all available information:

- noise $\sigma$ is positive quantitiy $\rightarrow$ choose $p(\sigma)$ with positive support
- eyeballing the data
- common sense
- ...

Warning: avoid introducing strong bias! Principled methods are available to find good prior distributions.

In our case:

$\sigma$: for example, $\sigma \sim \text{Lognormal}(\mu = 2, \sigma = \frac{1}{2})$

$\vec{\beta}$: broad, rather uninformative normal distributions, e.g., $\mathcal{N}(0, 5)$

# Real-world-ish application: regression

**Step 3:** specfiy model in probabilistic programming library

- programmatic formulation of statistical model
- powerful inference algorithms
- “debugging” functionality

Popular and easy-to-use choices are PyMC3, Stan, or TensorFlow Probability.

**Step 4:** perform inference with appropriate algorithm

Most popular and powerful class: Markov chain Monte Carlo

Continuous parameters: Hamiltonian Monte Carlo (HMC) very efficient

Discrete parameters: e.g., Metropolis-Hastings

Combination of both: Gibbs sampling

Another popular option: variational inference (VI)

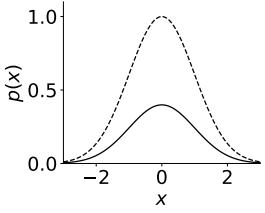
# Real-world issues

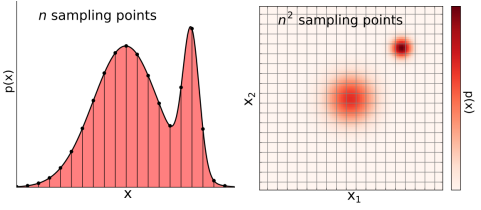
In reality, probability distributions often

- of non-standard form
- are multidimensional
- have highly correlated random variables
- are known only up to a normalization constant

Consequences:

- analytical evaluation of expectation values is impossible
- naïve sampling approaches are inefficient (curse of dimensionality)



# Approximation workhorse: Metropolis-Hastings

## Markov chain

Random process with

$$p(x_{i+1}|x_i, x_{i-1}, \ldots, x_1) = p(x_{i+1}|x_i)$$

$\rightarrow$ a Markov chain has no "memory"

In some conditions: converges to a unique invariant distribution $\pi(x)$

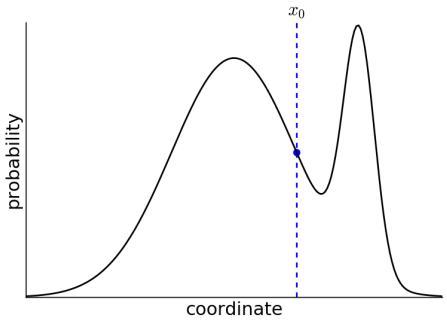
## Metropolis-Hastings algorithm

Construct Markov chain with invariant distribution $\pi(x) = p(x)$:

1. starting at state, $x_i$, propose a new state $x_i^*$ from $q(x_i^*|x_i)$
2. calculate acceptance probability $p_{\text{acc}}$
3. draw $u \sim \mathcal{U}(0, 1)$
4. if $u < p_{\text{acc}}$: $x_{i+1} = x_i^*$, else $x_{i+1} = x_i$

# Metropolis (-Hastings)

Initialize with any state $x_0$



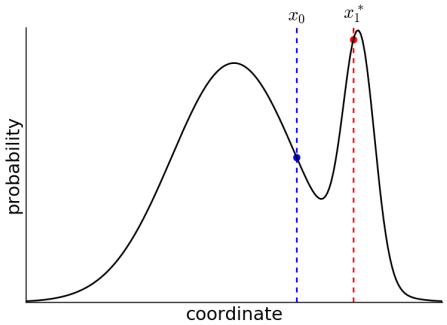

Sequence of states:
$(x_0)$

Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

# Metropolis (-Hastings)

Initial state: $x_0$

1. calculate a proposal state $x_0^*$ by randomly perturbing $x_0$



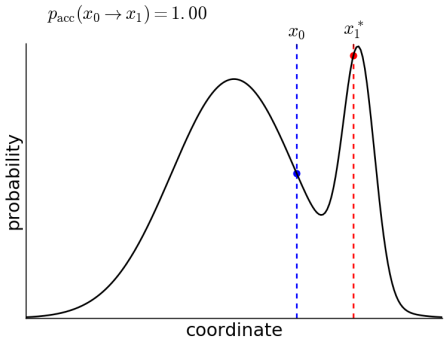

Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

# Metropolis (-Hastings)

Initial state: $x_0$

1. calculate a proposal state $x_0^*$ by randomly perturbing $x_0$
2. calculate acceptance probability

$$p_{\text{acc}} = \min\left(1, \frac{p(x_0^*)}{p(x_0)}\right)$$




Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

# Metropolis (-Hastings)

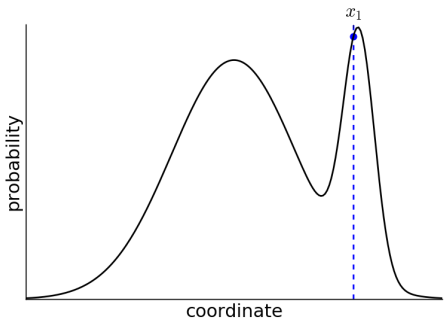
Initial state: $x_0$

1. calculate a proposal state $x_0^*$ by randomly perturbing $x_0$
2. calculate acceptance probability

$$p_{\text{acc}} = \min\left(1, \frac{p(x_0^*)}{p(x_0)}\right)$$

3. with probability $p_{\text{acc}}$, accept proposal state $x_0^*$ as the next state $x_1$, else copy $x_0$

Sequence of states:
$(x_0, x_1)$

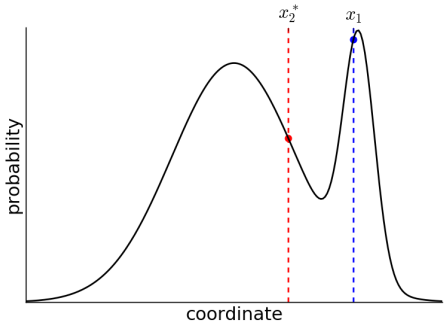



Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

# Metropolis (-Hastings)

Current state: $x_1$

1. calculate a proposal state $x_1^*$ by randomly perturbing $x_1$



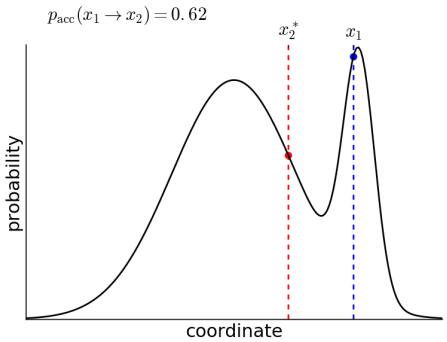

Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

# Metropolis (-Hastings)

Current state: $x_1$

1. calculate a proposal state $x_1^*$ by randomly perturbing $x_1$
2. calculate acceptance probability

$$p_{\text{acc}} = \min\left(1, \frac{p(x_1^*)}{p(x_1)}\right)$$




Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

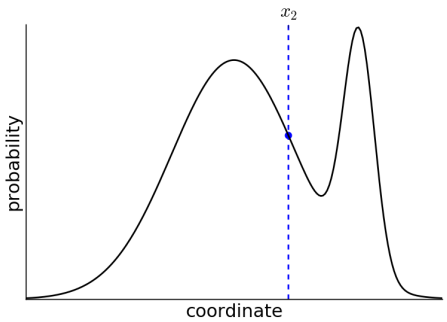
# Metropolis (-Hastings)

Current state: $x_1$

1. calculate a proposal state $x_1^*$ by randomly perturbing $x_1$
2. calculate acceptance probability

$$p_{\text{acc}} = \min\left(1, \frac{p(x_1^*)}{p(x_1)}\right)$$

3. with probability $p_{\text{acc}}$, accept proposal state $x_1^*$ as the next state $x_2$, else copy $x_1$




Sequence of states:
$(x_0, x_1, x_2)$

Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)

# Metropolis (-Hastings)

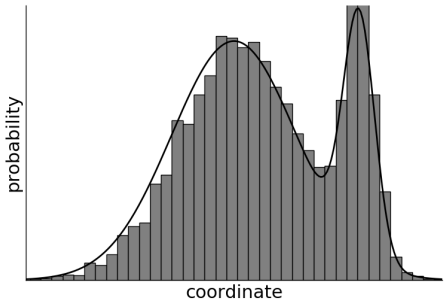
Current state: $x_1$

1. calculate a proposal state $x_1^*$ by randomly perturbing $x_1$
2. calculate acceptance probability

$$p_{\text{acc}} = \min\left(1, \frac{p(x_1^*)}{p(x_1)}\right)$$

3. with probability $p_{\text{acc}}$, accept proposal state $x_1^*$ as the next state $x_2$, else copy $x_1$

Sequence of states:
$(x_0, x_1, x_2, \ldots, x_n)$




Metropolis et al., J. Chem. Phys (1953); Hastings, Biometrika (1970)