Bayesian Methods

Peter Komar

April 6, 2019

Contents

1 Foundations

1.1 Definitions, identities

Notation

- Lower-case letters (a, b, c, x, y) stand for real numbers.

- Upper-case letters (A, B, C, X, Y) stand for random variables, and $A = a$ is an event.

- We write the probability of X taking the value x as $P(X = x) =: P(x)$.

- Or, if X is a continuous variable, $P(x \leq X \leq x + \delta) =: P(x)\,\delta$ (for small, positive δ).

- The comma between events stands for "and", i.e. $P(X = x \text{ and } Y = y) =: P(x, y)$

- The vertical bar separates the events from conditions, i.e. $P(A = a, \text{ given } B = b) =: P(a \mid b)$

- Both integration and summation are denoted as $\int_{\infty}^{+\infty}[\ldots]da =: \sum_{a \in \mathbb{R}}[\ldots] =: \sum_a[\ldots]$

Conditional probability identities (for every a, b, c)

- $P(a, b)$ refers to the joint of a and b, which may be dependent, i.e. $P(a, b) \neq P(a)P(b)$, in general.

- Definition of conditional ("a, given b"): $P(a \mid b) = \frac{P(a,b)}{P(b)}$

- After rearranging, we can write the joint ("a and b") as $P(a, b) = P(a \mid b)\,P(b)$.

- The same holds under a common condition c, i.e. ("a and b, given c") $P(a, b \mid c) = P(a \mid b, c)\,P(b \mid c)$

- Normalization is required on the left argument (the "event"), i.e. $\sum_a P(a \mid b) = 1$.

- But summing the right argument (the "condition") does not yield 1, i.e. $\sum_b P(a \mid b) \neq 1$, in general.

Marginal

- Summing over all but one variable of a joint yields the marginal, $P(a) = \sum_b P(a, b) = \sum_b P(a \mid b)\,P(b)$.

Bayes theorem

- By expressing the joint $P(x, b) = P(b, x)$ in two different ways, one can show: $P(b \mid x) = \frac{1}{P(x)} P(x \mid b)\,P(b)$.

joint

$P(x, b)$

conditionals

$$P(x \mid b) = \frac{P(x, b)}{P(b)}$$

$$P(b \mid x) = \frac{P(x, b)}{P(x)}$$

marginals

$$P(x) = \sum_b P(x, b)$$

$$P(b) = \sum_x P(x, b)$$
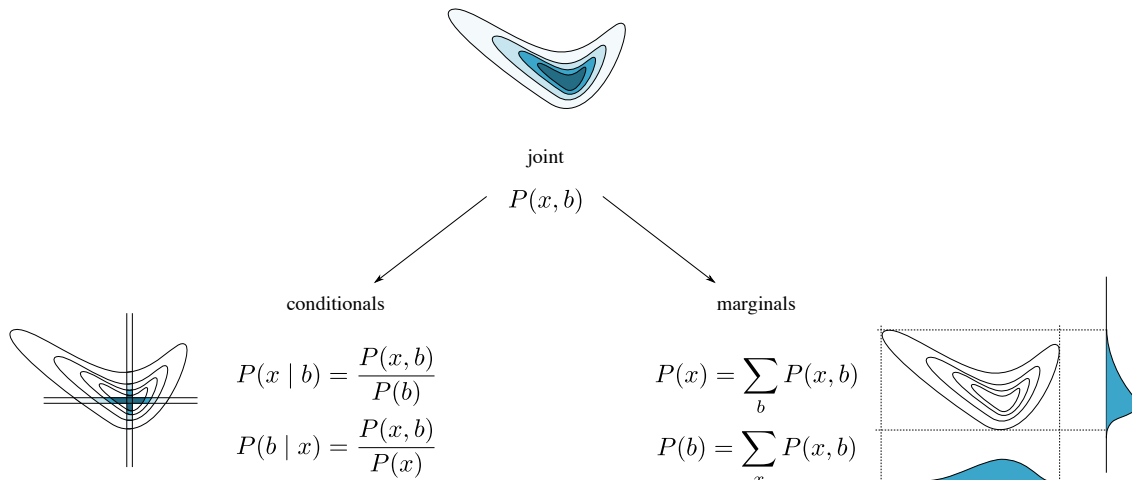
1.2   Bayesian inference

Prior, likelihood, posterior

- We collect data: $D = \{x_1, x_2, \ldots x_n\}$, where each x_i is a sample from the same process.

- We describe a model by specifying three components:

  1. its parameter θ (dimension, range, etc.), and
  2. the prior distribution of θ, $P(\theta)$, and
  3. the generative (or "forward") probability $P(x_i \mid \theta)$

- We assume that the samples were generated independently, which allows us to write the likelihood, $P(D \mid \theta)$, as the product $P(D \mid \theta) = P(x_1 \mid \theta) \, P(x_2 \mid \theta) \, \ldots \, P(x_n \mid \theta) = \prod_{i=1}^{n} P(x_i \mid \theta)$

- Calculating the unnormalized posterior, $P^*(\theta \mid D)$, is easy: $P^*(\theta \mid D) := P(D \mid \theta) \, P(\theta)$

- We need to normalize it though. The normalization constant is $Z = \sum_\theta P^*(\theta \mid D)$.

- Once calculated, Z can be used to obtain the posterior, $P(\theta \mid D) = \frac{1}{Z} P^*(\theta \mid D)$

Example

Three light bulbs of the same kind lasted for 1, 2 and 5 months under continuous use. Let us estimate the lifetime of this kind of light bulb.

- The data consists of the three times: $D = \{t_1, t_2, t_3\} = \{1, 2, 5\}$

- We model this process with

  1. a single "typical lifetime" variable $T > 0$,
  2. realistically $T < 1000$ months, but otherwise we don't know, so we use a flat prior: $P(T) = \frac{1}{1000}$, uniform on [0,1000].
  3. we assume no aging, which means the actual length of their life is exponential distributed with a mean of T: $P(t \mid T) = \frac{1}{T} \exp\left(-\frac{t}{T}\right)$

- We write the likelihood as $P(D \mid T) = \prod_i P(t_i \mid T) = \prod_{i=1}^{3} \frac{1}{T} \exp\left(-\frac{t_i}{T}\right) = \frac{1}{T^3} \exp\left(-\frac{1+2+5}{T}\right)$,

- and the unnormalized posterior as $P^*(T \mid D) = \frac{1}{T^3} \exp\left(-\frac{8}{T}\right) \frac{1}{1000}$.

- We carry out the normalization (finding Z and $P(T \mid D)$) numerically:

```
1  import numpy as np
2
3  T_arr = np.linspace(1e-6, 1000, 10_000)
4  Pstar_arr = 1.0/T_arr**3 * np.exp(-8/T_arr) / 1000.0
5  Z = np.sum(Pstar_arr)
6  P_arr = Pstar_arr / Z
```

  yielding $Z = 1.562 \times 10^{-4}$

- We calculate the expected lifetime (given the observed data), $\mathbb{E}(T \mid D) = \sum_T T \, P(T \mid D)$, and

- the standard deviation, $\text{std}(T \mid D) = \sqrt{\sum_T (T - \mathbb{E}(T))^2 \, P(T \mid D)}$, using the regular formulas.

```
1  T_ev = np.sum(T_arr * P_arr)
2  T_std = np.sqrt(np.sum((T_arr - T_ev)**2 * P_arr))
```

  yielding $\mathbb{E}(T \mid D) = 7.937, \quad \text{std}(T \mid D) = 14.48$.

1.3 Model comparison

New definition: Evidence

- We observe some data D,

- specify one model, M_A with its parameter α, prior $P(\alpha \mid M_A)$, and likelihood $P(D \mid \alpha, M_A)$,

- specify another model, M_B with its parameter β, prior $P(\beta \mid M_B)$, and likelihood $P(D \mid \beta, M_B)$,

- and declare prior probabilities for the models themselves: $P(M_A), P(M_B)$, so that $P(M_A) + P(M_B) = 1$.

- We calculate the model evidence (or "model likelihood") by marginalizing over the parameters

  1. $P(D \mid M_A) = \sum_\alpha P(D \mid \alpha, M_A) \, P(\alpha \mid M_A)$,
  2. $P(D \mid M_B) = \sum_\beta P(D \mid \beta, M_B) \, P(\beta \mid M_B)$,

- and we calculate the unnormalized posteriors: $P^*(M \mid D) = P(D \mid M) \, P(M)$, for both models.

- Finally we obtain the normalization constant: $Z = P^*(M_A \mid D) + P^*(M_B \mid D)$,

- allowing us to write the posterior probabilities of the models $P(M|D) = P^*(M|D) \, / Z$, for both models.

Example

While waiting for the checked bag at the airport carousel, one can consider two possibilities: 1) The bag could have missed the flight, and will never come, or 2) it was on the plane and it has a flat chance of arriving within 0 to, let's say, 20 minutes. What is the posterior probability of model 2, if 14 minutes have already passed and the bag has not arrived?

- The only observation we have is $D = \{$Bag has not arrived after $t_{\mathrm{wait}} = 14$ minutes$\}$

- The first model, M_1, says the bag missed the plane. This means, no matter how much we waited it was pre-destined to not come out on the carousel, i.e. $P(D \mid M_1) = 1$. This model has no parameters.

- The second model, M_2, assumes an equal chance for the bag to arrive any minute within the 20-minute window, which can be written as $P(t_{\mathrm{bag}} \mid M_2) = 1/20$ for $t_{\mathrm{bag}} \in [0, 20]$. Since every waiting time until the bag actually arrives is pre-destined to occur, we can write the likelihood as

$$P(D \mid t_{\mathrm{bag}}, M_2) = [t_{\mathrm{wait}} < t_{\mathrm{bag}}] \begin{cases} 1 & , \quad \text{if } t_{\mathrm{wait}} < t_{\mathrm{bag}} \\ 0 & , \quad \text{otherwise.} \end{cases}$$

- Let's say we assume an initial 10% chance for the bag to have missed the flight, i.e. $P(M_1) = 0.1$, and therefore $P(M_2) = 1 - P(M_1) = 0.9$.

- Since model 1 has no parameters, its likelihood (or evidence) is easy to look up from the model specification: $P(D \mid M_1) = 1$.

- Model 2 has one parameter, t_{wait}, over which we need to sum to obtain its evidence:

$$P(D \mid M_2) = \sum_{t_{\mathrm{bag}}} P(D \mid t_{\mathrm{bag}}, M_2) \, P(t_{\mathrm{bag}} \mid M_2) = \sum_{t_{\mathrm{bag}}} [14 < t_{\mathrm{bag}}] \times \frac{1}{20} = \sum_{t_{\mathrm{bag}} > 14} \frac{1}{20} = \frac{20 - 14}{20}$$

- Unnormalized posteriors we get by multiplying: $P^*(M_1 \mid D) = 1 \times 0.1$, $\quad P^*(M_2 \mid D) = \frac{20-14}{20} \times 0.9$,

- the sum of which is the normalization constant, $Z = 0.1 + \frac{3}{10} \times 0.9 = 0.37$.

- Now we calculate the posterior probability of model 2 as $P(M_2 \mid D) = P^*(M_2 \mid D)/Z = 0.7297$.

Additionally, we can calculate $P(M_2 \mid t_{\mathrm{wait}})$ for all waiting times between 0 and 20 minutes. This is shown below.
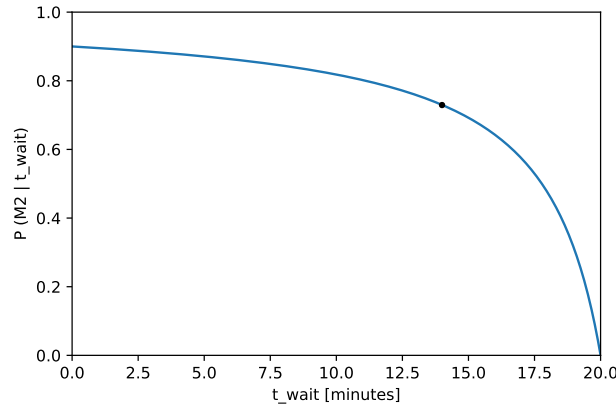
1.4   Prediction

New definition: Predictive distribution

- We measure some data, $D = \{x_1, x_2, \ldots x_n\}$, where each sample is assumed to be independently generated from the same process.

- We model the process with a parameter θ, its prior $P(\theta)$ and a generative distribution $P(x \mid \theta)$.

- We calculate the posterior of the parameter $P(\theta \mid D) = P^*(\theta \mid D)/Z = \ldots$, following the steps in section 1.2.

- The predictive distribution, $P(X_{n+1} = x \mid D)$, describes what we can expect of an unobserved $n + 1$th data point X_{n+1}, given the observations x_1 to x_n. It is the average of the generating distribution over all conceivable values of the parameter weighted by its posterior, i.e.

$$P(X_{n+1} = x \mid D) = \sum_{\theta} P(x \mid \theta) \, P(\theta \mid D)$$

- Sometime, what we are interested is not the distribution of a new sample, but some interesting function the model parameter, $f(\theta)$. The distribution of such a custom metric can be calculated with a similar sum:

$$P(f(\theta) \mid D) = \sum_{\theta} f(\theta) \, P(\theta \mid D)$$

Note: The fact that $P(x_{n+1} \mid D) \neq P(x_{n+1})$ may feel surprising. Did we not assume that the data points were generated independently? Indeed we did. What is usually meant by "independence" is $P(x_i, x_j \mid \theta) = P(x_i \mid \theta)P(x_j \mid \theta)$, which is exactly what we spelled out in section 1.2. Although this *conditional* independence holds for every fixed value of θ, the $\{x_i\}$ variables become dependent after we marginalize out θ. This is the mathematical equivalent of the fact that if the parameter is unknown, than every observation provides a piece of information about it, and that information, in turn, affects what we can expect of other observations. All this is because conditional and unconditional independence are separate properties, i.e.

$$P(a, b) = P(a)P(b) \quad \not\Leftrightarrow \quad \forall c : \; P(a, b \mid c) = P(a \mid c)P(b \mid c)$$

Example

   Two players, A and B are playing a game of luck, where at the beginning of the game a ball is rolled on a pool table to divide the table in two un-equal halves: A's side and B's side. In each subsequent round, a ball is rolled. A point is given to the player on whose side the ball stops. A and B are playing this game until one of them reaches 6 points. The current score is 5 to 3 in favor of A. What is the chance that A will win this game?

- The only observation we have is the current score, $D = \{n_A = 5, n_B = 3\}$.

- The story describes the model accurately:

   1. The unknown parameter is the position of the first ball, $0 \leq b \leq 1$.
   2. Based on the text, we assume a uniform prior, $P(b) = 1$, density for $b \in [0, 1]$.
   3. The probability that B scores a point is $P(\text{B scores} \mid b) = b$ for every following roll.

- Since we do not know the order in which they scored the points, the likelihood is a binomial distribution with 3 successes, 5+3 attempts, and probability b, i.e. $P(D \mid b) = \text{Binomial}(3 \mid 5 + 3, \; b)$

- The unnormalized posterior is simply $P^*(b \mid D) = \text{Binomial}(5 \mid 8, b) \times 1$.

- We evaluate the normalization constant $Z = \sum_b \text{Binomial}(5 \mid 8, b)$ and the posterior $P(b \mid D) = P^*(b \mid D)/Z$ numerically

```
1  import numpy as np
2  from scipy.stats import binom
3
4  b_arr = np.linspace(0, 1, 1000)
5  Pstar_arr = binom.pmf(3, 8, b_arr)
6  Z = np.sum(Pstar_arr)
7  P_arr = Pstar_arr / Z
```

- Now, let us determine the probability of A winning the game, as a function of b. Player B is in an unfortunate position, he needs to score three times in a row, to win. Any other outcome means player A wins. With this in mind, we can write $P(\text{A wins} \mid b, D) = 1 - P(\text{B wins} \mid b, D) = 1 - P(\text{B scores 3 times} \mid b) = 1 - b^3 =: f(b)$.

- Finally, the probability A winning, considering all values of b is $P(\text{A wins} \mid D) = \sum_b f(b) \, P(b \mid D)$

```
1  P_Awins = np.sum((1 - b_arr**3) * P_arr)
```

   yielding $P(\text{A wins} \mid D) = 0.909$

2   Maximum Likelihood Estimate and Exact inference

The one-size-fits-all method of inferring unknown model parameters is called "Maximum Likelihood Estimate". It is calculated by finding the parameter values that maximize the generative probability of the observed data.

2.1   Maximum likelihood estimate

MLE-method

- We record a series of measurements, $D = \{x_1, x_2, \ldots x_N\}$.

- We construct a model that specifies the probability of each data point x_i, $P(x_i \mid \theta)$, as a function of model parameter(s) θ, the value(s) of which we wish to infer.

- The sum of the log probability term yields the log likelihood *of the parameter*,

$$L(\theta) := \log P(D \mid \theta) = \log \prod_{i=1}^{N} P(D \mid \theta) = \sum_{i=1}^{N} \log P(x_i \mid \theta)$$

- Optimizing the value of θ to get to the maximum of L yields the MLE of θ:

$$\theta_{\text{MLE}} = \text{argmax}_\theta L(\theta),$$

This can be done either

  - **numerically**, by gradient descent or EM methods (see section 5), or
  - **analytically**, by setting the first derivatives to 0, and solving the resulting system of equations.

Example 1: Normal model (analytical MLE)

- We observe a collection of real values $D = \{x_i \in \mathbb{R}\}_{i=1}^{N}$

- The normal model has two parameters, $\mu \in \mathbb{R}$, and $\sigma^2 > 0$, which define the probability density of each data point as
$$P(x_i \mid \mu, \sigma^2) = \text{Normal}(x_i \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right].$$

- The log likelihood of the model parameters is

$$L(\mu, \sigma^2) = \sum_{i=1}^{N} \log\left(\text{Normal}(x_i \mid \mu, \sigma^2)\right) = -\frac{N}{2}\log(\sigma^2) - \sum_{i=1}^{N}\frac{(x_i - \mu)^2}{2\sigma^2} + \text{const.}$$

- To calculate the formulas for the analytical MLE of μ and σ^2, we calculate the first order partial derivatives of L, and set them to zero. (With $[\ldots]_{\text{MLE}}$ with denote that the enclosed formula is evaluated at the MLE point, $(\mu_{\text{MLE}}, (\sigma^2)_{\text{MLE}})$.)

$$0 = \left[\frac{\partial L}{\partial \mu}\right]_{\text{MLE}} = \left[\sum_{i=1}^{N}\frac{\mu - x_i}{\sigma^2}\right]_{\text{MLE}} \qquad \Rightarrow \qquad \mu_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

$$0 = \left[\frac{\partial L}{\partial(\sigma^2)}\right]_{\text{MLE}} = \left[-\frac{N}{2\sigma^2} + \sum_{i=1}^{N}\frac{(x_i - \mu)^2}{2(\sigma^2)^2}\right]_{\text{MLE}} \qquad \Rightarrow \qquad (\sigma^2)_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\text{MLE}})^2$$

This result shows that the empirical mean and empirical variance (the right hand sides of the equations above) are exactly the MLE estimates of μ and σ^2, respectively.

Example 2: Cauchy distribution (numerical MLE)

- Let's say we observe the following five data point $D = \{-10, 1, 2, 5, 20\}$, and

- we wish to fit a Cauchy distribution to this data, parameterized by $m \in \mathbb{R}$ and $s > 0$.

$$P(x_i \mid m, s) = \text{Cauchy}(x_i \mid m, s) = \frac{1}{s\pi} \frac{1}{1 + [(x_i - m)/s]^2}$$

- The log likelihood of the model parameters is

$$L(m, s) = \sum_{i=1}^{N} \log \text{Cauchy}(x_i \mid m, s) = -N \log(s) - \sum_{i=1}^{N} \log\left(1 + \left[\frac{x_i - m}{s}\right]^2\right) + \text{const.}$$

- Since there is no closed-form solution, we us numerical maximization to find m_{MLE} and s_{MLE}. The following python code uses `scipy.optimize.minimize()` to find the local maximum near the initial starting point, $m_0 = 0$, $s_0 = 10$.

```python
1  import numpy as np
2  from scipy.optimize import minimize
3
4  def cauchy_total_log_likelihood(X, m, s):
5      X = np.array(X)
6
7      L = 0
8      L += -len(X)/2 * np.log(s**2)
9      L += -np.sum( np.log(1 + (X - m)**2 / s**2 ) )
10
11     return L
12
13 X = [-10, 1, 2, 5, 20]
14 def func_to_minimize(theta):
15     m = theta[0]
16     s = theta[1]
17     return - cauchy_total_log_likelihood(X, m, s)
18
19 m0 = 0
20 s0 = 10
21 result = minimize(func_to_minimize, [m0, s0])
22 m_MLE, s_MLE = result.x
```

- This yields $m_{\text{MLE}} = 2.251$, $s_{\text{MLE}} = 3.090$. Here is the corresponding Cauchy probability density:
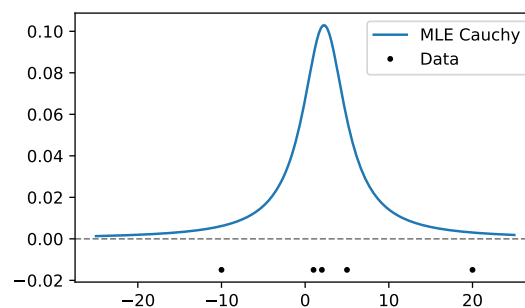
2.2   Exact inference examples

A very small number of models can be inferred exactly. This means, we can derive closed form solutions for the posterior distribution of their parameters. Consequently the posterior function and various summary statistics (e.g. mean and standard deviation) of the model parameter can be calculated exactly and efficiently.

Binomial model

- We record the number of successes k_i in each of the $i = 1, \ldots N$ experimental runs ($N = 1$ is also possible), each of which contains n_i attempts. The data we collect is $D = \{(k_1, n_1), (k_2, n_2), \ldots, (k_N, n_N)\}$, where $0 \leq k_i \leq n_i$, positive integers.

- (Note: The resulting formulas will show that it is enough to know the total number of successes k_{tot} and total number of attempts n_{tot}, if the same binomial model is responsible for all experimental runs.)

- If each attempt is independent of the others, then the binomial model is justified, and the corresponding probability for a single experiment can be written as

$$P(k_i \mid n_i, p) = \text{Binomial}(k_i \mid n_i, p) = \binom{n_i}{k_i} p^{k_i} (1-p)^{n_i - k_i}$$

which is a function of the success probability p, $0 \leq p \leq$, for which we assume a flat prior density: $P(p) = 1$, on the $[0, 1]$ interval.

- The posterior of p, after considering all experimental runs, can be recognized as a beta distribution:

$$
\begin{aligned}
P(p \mid D) &= \frac{1}{Z} \prod_{i=1}^{N} \left[p^{k_i} (1-p)^{n_i - k_i} \right] = \frac{1}{Z} p^{k_{\text{tot}}} (1-p)^{n_{\text{tot}} - k_{\text{tot}}} \\
&= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha - 1} (1-p)^{\beta - 1} = \text{Beta}(p \mid \alpha = k_{\text{tot}} + 1, \beta = n_{\text{tot}} - k_{\text{tot}} + 1),
\end{aligned}
$$

where $k_{\text{tot}} = \sum_i k_i$ and $n_{\text{tot}} = \sum_i n_i$. Using the known formulas of the beta distribution, we can write the mean, mode and standard deviation of p as

$$\mathbb{E}(p) = \frac{\alpha}{\alpha + \beta} = \frac{k_{\text{tot}} + 1}{n_{\text{tot}} + 2}, \quad \text{mode}(p) = \frac{\alpha - 1}{\alpha + \beta - 2} = \frac{k_{\text{tot}}}{n_{\text{tot}}},$$

$$\text{std}(p) = \frac{\sqrt{\alpha\beta}}{(\alpha + \beta)\sqrt{\alpha + \beta + 1}} = \frac{\sqrt{(k_{\text{tot}} + 1)(n_{\text{tot}} - k_{\text{tot}} + 1)}}{(n_{\text{tot}} + 2)\sqrt{n_{\text{tot}} + 3}}$$

- Note: The mode of the posterior (calculated under flat prior) coincides with the maximum likelihood estimate, $p_{\text{MLE}} = k_{\text{tot}}/n_{\text{tot}}$.

Poisson model

- Within each of the $i = 1, \ldots N$ measurement windows, we observe k_i number of events. The collected data is $D = \{k_1, k_2, \ldots k_N\}$, where $k_i \geq 0$, positive integers.

- (Note: The formula of the posterior will show that knowing the total number of events k_{tot} and the number of windows N is enough.)

- The Poisson model for a single measurement window prescribes the probability

$$P(k_i \mid \lambda) = \text{Poisson}(k \mid \lambda) = e^{-\lambda} \frac{\lambda^{k_i}}{k_i!},$$

where $\lambda > 0$ is the "typical number of observations", for which we assume a flat (and improper) prior: $P(\lambda) = \text{const}$.

- The posterior of λ turns out to be a gamma distribution:

$$P(\lambda \mid D) \;\; = \;\; \frac{1}{Z} \prod_{i=1}^{N} \left[e^{-\lambda} \lambda^{k_i} \right] = \frac{1}{Z} e^{-N\lambda} \lambda^{k_{\text{tot}}} = \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} = \text{Gamma}(\lambda \mid \alpha = k_{\text{tot}} + 1, \beta = N),$$

  where $k_{\text{tot}} = \sum_i k_i$. Using the known formulas for the gamma distribution, the mean, mode and standard deviation of λ can be written as

$$\mathbb{E}(\lambda) = \frac{\alpha}{\beta} = \frac{k_{\text{tot}}+1}{N}, \quad \text{mode}(\lambda) = \frac{\alpha-1}{\beta} = \frac{k_{\text{tot}}}{N}, \quad \text{std}(\lambda) = \frac{\sqrt{\alpha}}{\beta} = \frac{\sqrt{k_{\text{tot}}+1}}{N}$$

- Note: The mode of the posterior (calculated under flat prior) coincides with the maximum likelihood estimate, $\lambda_{\text{MLE}} = k_{\text{tot}}/N$.

Multinomial model

- Similarly to the binomial model, we perform a series of experimental runs, $i = 1, \ldots N$, each consisting some number of attempts, each of which can result in M different outcomes. For the ith experimental run the collected data is $D_i = (k_{i,1}, k_{i,2}, \ldots k_{i,M})$, where $k_{i,j} \geq 0$ is the number of times outcome j was found in experiment i. The complete observed data is $D = \{D_1, D_2, \ldots D_N\}$.

- (Note: The formulas for the posterior will show that it is enough to know the number of each outcomes aggregated across all runs, $k_{\text{tot},j}$, if the same multinomial process is responsible for all runs.)

- The multinomial model (which is justified if the attempts are independent from each other) prescribes the probability for the outcome vector $\{k_{i,j}\}_{j=1}^{M}$ of one experiment

$$P(\{k_{i,j}\}_{j=1}^{M} \mid p) = \text{Multinomial}(\{k_{i,j}\}_{j=1}^{M} \mid p) = k_{i,\text{tot}}! \prod_{j} \frac{p_j^{k_{i,j}}}{k_{i,j}!},$$

  where each element of the probability vector $p = (p_1, p_2, \ldots p_M)$, $0 \leq p_j \leq 1$ is the probability of outcome j. Since one of the outcomes is certain to happen, $\sum_j p_j = 1$. We assume a flat prior for the probability vector $P(p) = \text{const.}$ (on the $M-1$ dimensional unit simplex).

- The posterior of p, after considering all runs, can be written as a Dirichlet distribution:

$$P(p \mid D) \;\; = \;\; \frac{1}{Z} \prod_{i=1}^{N} \prod_{j=1}^{M} (p_j)^{k_{i,j}} = \frac{1}{Z} \prod_{j=1}^{M} (p_j)^{k_{\text{tot},j}} = \Gamma(\alpha_{\text{tot}}) \prod_{j=1}^{M} \frac{(p_j)^{\alpha_j-1}}{\Gamma(\alpha_j)} = \text{Dirichlet}(p \mid \alpha_j = k_{\text{tot},j} + 1),$$

  where $k_{\text{tot},j} = \sum_i k_{i,j}$, and $\alpha_{\text{tot}} = \sum_j \alpha_j = k_{\text{tot},\text{tot}} + M$. Mean, mode and marginal standard deviation are

$$\mathbb{E}(p_j) = \frac{\alpha_j}{\alpha_{\text{tot}}} = \frac{k_{\text{tot},j}+1}{k_{\text{tot},\text{tot}}+M}, \quad \text{mode}(p): \; p_j = \frac{\alpha_j-1}{\alpha_{\text{tot}}-M} = \frac{k_{\text{tot},j}}{k_{\text{tot},\text{tot}}}$$

$$\text{std}(p_j) = \frac{\sqrt{\alpha_j(\alpha_{\text{tot}}-\alpha_j)}}{\alpha_{\text{tot}}\sqrt{\alpha_{\text{tot}}+1}} = \frac{\sqrt{(k_{\text{tot},j}+1)(k_{\text{tot},\text{tot}}-k_{\text{tot},j}+M-1)}}{(k_{\text{tot},\text{tot}}+M)\sqrt{k_{\text{tot},\text{tot}}+M+1}}$$

- Note: The mode of the posterior (calculated under flat prior) coincides with the maximum likelihood estimate, $p_{\text{MLE},j} = k_{\text{tot},j}/k_{\text{tot},\text{tot}}$.

Exponential model

- We observe a sequence of events, $i = 1, \ldots N$. We record the waiting times t_i between event $i - 1$ and event i (t_1 is the waiting time from the start of observation until the first event). The data is $D = \{t_1, t_2, \ldots t_N\}$, where $t_i > 0$.

- (Note: The formula for the posterior will show that, if the events are generated by a Poisson process, then it is enough to know the total elapsed time t_{tot} and the number of events N.)

- If the events are generated by a Poisson process (i.e. they are independent from each other and the elapsed time), then the waiting times are exponentially distributed:

$$P(t_i \mid \gamma) = \text{Exponential}(t_i \mid \gamma) = \gamma e^{-\gamma t_i},$$

  where $\gamma > 0$ is the rate of events, for which we assume a flat (and improper) prior: $P(\gamma) = \text{const.}$

- The posterior of γ can be written as a gamma distribution:

$$P(\gamma \mid D) \;=\; \frac{1}{Z} \prod_{i=1}^{N} \left[\gamma e^{-\gamma t_i} \right] = \frac{1}{Z} \gamma^N e^{-\gamma t_{\text{tot}}} = \frac{\beta^\alpha}{\Gamma(\alpha)} \gamma^{\alpha-1} e^{-\beta \gamma} = \text{Gamma}(\gamma \mid \alpha = N+1, \beta = t_{\text{tot}}),$$

  where $t_{\text{tot}} = \sum_i t_i$. Using the known formulas for the gamma distribution, we can write the mean, mode, standard deviation of γ as

$$\mathbb{E}(\gamma) = \frac{\alpha}{\beta} = \frac{N+1}{t_{\text{tot}}}, \quad \text{mode}(\gamma) = \frac{\alpha-1}{\beta} = \frac{N}{t_{\text{tot}}}, \quad \text{std}(\gamma) = \frac{\sqrt{\alpha}}{\beta} = \frac{\sqrt{N+1}}{t_{\text{tot}}}$$

- Note: The mode of the posterior (calculated under flat prior) coincides with the maximum likelihood estimate, $\gamma_{\text{MLE}} = N/t_{\text{tot}}$.

- Note: The result is meaningful even for $N = 0$, which corresponds to a t_{tot}-long measurement session during which no event was observed. This results in a posterior which is identical to an exponential distribution: $P(\gamma \mid D) = \text{Gamma}(\gamma \mid \alpha = 1, \ \beta = t_{\text{tot}}) = t_{\text{tot}} e^{-t_{\text{tot}} \gamma} = \text{Exponential}(\gamma \mid t_{\text{tot}})$.

Normal

- We observe one-dimensional data points $D = \{x_1, x_2, \ldots x_N\}$, where $x \in \mathbb{R}$.

- The normal model prescribes the following probability for each data point:

$$P(x_i \mid \mu, \sigma^2) = \text{Normal}(x_i \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2} \right],$$

  as a function of μ (expected value) $\in \mathbb{R}$, and σ^2 (variance) > 0, for which we assume a flat (and improper) priors: $P(\mu, \sigma^2) = \text{const.}$

- With the notations, $m = \frac{1}{N} \sum_i x_i$ being the empirical mean, $s^2 = \frac{1}{N} \sum_i (x_i - m)^2$ being the empirical variance, the joint posterior of μ, σ^2 can be written as

$$
\begin{aligned}
P(\mu, \sigma^2 \mid D) \;=\;& \frac{1}{Z} \prod_{i=1}^{N} \left\{ \frac{1}{\sqrt{\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2} \right] \right\} = \frac{1}{Z} \left(\frac{1}{\sigma^2} \right)^{N/2} \exp\left[-\frac{Ns^2 + N(\mu - m)^2}{2\sigma^2} \right] \\
=\;& \frac{\sqrt{\lambda}}{\sqrt{2\pi\sigma^2}} \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2} \right)^{\alpha+1} \exp\left[-\frac{2\beta + \lambda(\mu - \mu_c)^2}{2\sigma^2} \right] \\
=\;& \text{Normal-Inverse-Gamma}\left(\mu, \sigma^2 \;\middle|\; \alpha = \frac{N-3}{2}, \ \beta = \frac{Ns^2}{2}, \ \mu_c = m, \ \lambda = N \right).
\end{aligned}
$$

  This is a two-dimensional joint distribution, the mode of which is at

$$\text{mode}(\mu, \sigma^2) = (m, s^2).$$

- Integrating out σ^2 results in the marginal distribution of μ:

$$
\begin{aligned}
P(\mu \mid D) &= \sum_{\sigma^2} P(\mu, \sigma^2 \mid D) = \frac{\Gamma\left(\frac{N-2}{2}\right)}{\Gamma\left(\frac{N-3}{2}\right)} \frac{1}{\sqrt{\pi s^2}} \left[1 + \frac{(\mu - m)^2}{s^2}\right]^{-(N-2)/2} \\
&= \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \left[1 + \frac{1}{\nu}\left(\frac{\mu - \text{loc}}{\text{scale}}\right)^2\right]^{-(\nu+1)/2} \frac{1}{\text{scale}} \\
&= \text{t-distr}\left(\mu \,\Big|\, \text{loc} = m,\ \text{scale} = \frac{s}{\sqrt{N-3}},\ \nu = N-3\right),
\end{aligned}
$$

where ν is the "degrees of freedom" of the Student's t distribution. Using the known formulas for the shifted and scaled Student's t distribution, we can write the mean, *marginal* mode and standard deviation of μ as

$$
\mathbb{E}(\mu) = m, \quad \text{mode}(\mu) = m, \quad \text{std}(\mu) = \frac{s}{\sqrt{N-3}}\sqrt{\frac{\nu}{\nu-2}} = \frac{s}{\sqrt{N-5}},
$$

- Integrating out μ results in the marginal distribution of σ^2:

$$
\begin{aligned}
P(\sigma^2 \mid D) &= \sum_{\mu} P(\mu, \sigma^2 \mid D) = \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left(-\frac{\beta}{\sigma^2}\right) \\
&= \text{Inverse-Gamma}\left(\sigma^2 \,\Big|\, \alpha = \frac{N-3}{2},\ \beta = \frac{Ns^2}{2}\right)
\end{aligned}
$$

Using the known formulas for the inverse gamma distribution, we can write the mean, *marginal* mode and standard deviation of σ^2 as

$$
\mathbb{E}(\sigma^2) = \frac{\beta}{\alpha-1} = s^2 \frac{N}{N-5}, \quad \text{mode}(\sigma^2) = \frac{\beta}{\alpha+1} = s^2 \frac{N}{N-1}
$$

$$
\text{std}(\sigma^2) = \frac{\beta}{(\alpha-1)\sqrt{\alpha-2}} = s^2 \frac{\sqrt{2}N}{(N-5)\sqrt{N-7}}
$$

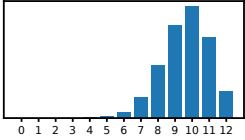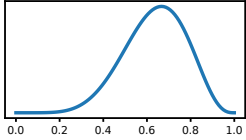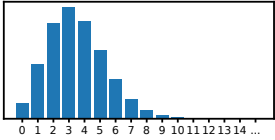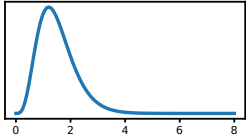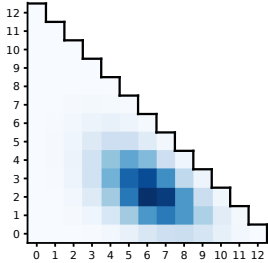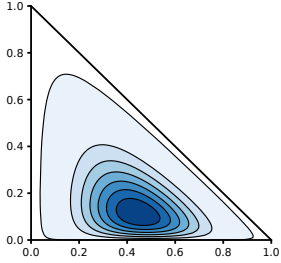- Note: While the mean and the standard deviation are unaffected by marginalization, the mode, in general, depends on whether we evaluate it on the joint distribution $P(\mu, \sigma^2 \mid D)$ or on the marginals $P(\mu \mid D)$ and $P(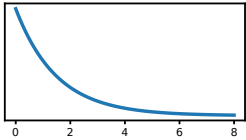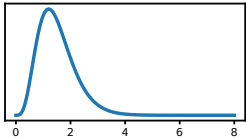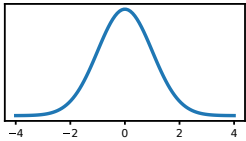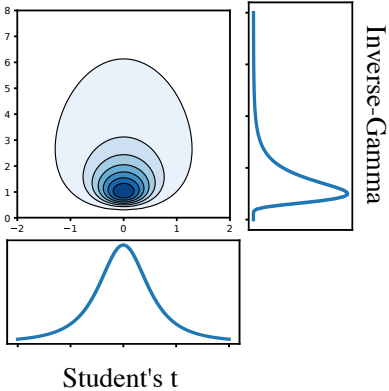\sigma^2 \mid D)$. With the assumption of the flat prior, the mode of μ happens to be the same both in the joint in and the marginal, and it coincides with its maximum likelihood estimate.

$$
[\text{mode}(\mu, \sigma^2)]_1 = \text{mode}(\mu) = m = \frac{1}{N}\sum_{i=1}^{N} x_i = \mu_{\text{MLE}}
$$

With the assumption of flat prior, the mode of σ^2 in the joint is identical to its maximum likelihood estimate, while the mode of the σ^2 marginal coincides with the widely-used unbiased estimator of σ^2.

$$
[\text{mode}(\mu, \sigma^2)]_2 = s^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2 = (\sigma^2)_{\text{MLE}} \neq \text{mode}(\sigma^2) = s^2 \frac{N}{N-1} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2
$$

The next page shows a graphical summary of the exactly-solvable models discussed above. The left column features a typical distribution the model prescribes for the data. The right column shows a typical posterior probability density for the model parameters.

Model	Posterior
Binomial	Beta
Poisson	Gamma
Multinomial	Dirichlet
Exponential	Gamma
Normal	Normal-Inverse-Gamma

Inverse-Gamma

Student's t

3   Priors, Regularization, AIC, BIC, LRT

Every time we increase the complexity of our model, for a fixed data set, its Maximum Likelihood fitting accuracy increases. While this trend seems encouraging, it is desired only up to a certain point, after which the model starts adapting to the idiosyncrasies of the data while becoming insensitive to its features that are truly generalizable. The resulting a fitted model will work very well for the data at hand, but under-perform on similar but slightly different future data. This is "overfitting".

Since models, being a mathematical constructs, cannot tell the difference between generalizable features and noise in the data, we need to guide them by providing additional information. We need to come to terms with our own expectations (based on intuition or historical data) and turn them into mathematical requirements. Priors and regularization, and model selection via various information criteria are common ways.

3.1   Improper and proper priors

Improper priors are not normalizable, i.e. $\sum_\theta P(\theta) = \infty$. Examples are

- Flat priors: $P(\theta) = $ const. over any infinite domain.

- "Uninformative" priors, which are usually derived from transformation invariance or max-entropy principles:

  - Location parameter: $P(m) = $ const., on $m \in (-\infty, +\infty)$,
  - Scale parameter: $P(s) = \frac{\text{const.}}{s}$, on $s \in (0, \infty)$,
  - Probability parameter: $P(p) = \frac{\text{const.}}{p(1-p)}$, on $p \in (0, 1)$.

- "Reference priors" that maximize the sensitivity of the inference of one selected model parameter.

- Jeffrey's priors for certain models, which equalizes the Fisher information on the parameter space, e.g. $P(\sigma^2) = 1/\sigma^2$ for the Normal model.

Priors that are normalized are proper priors, i.e. $\sum_\theta P(\theta) = 1$. Examples are

- Distributions with carefully chosen parameter values, e.g. $\text{Beta}(\theta \mid \alpha = 1, \beta = 1)$

- Jeffrey's priors for certain models, e.g. $P(p) = \frac{1}{\pi\sqrt{p(1-p)}}$ for Bernoulli model (or Binomial model).

Note: A prior (proper or improper) $P(\theta)$ can be used in the Maximum Likelihood Estimation context by adding their logarithm to the log likelihood function $L(\theta)$ before maximizing,

$$\theta_{\text{MLE}} = \arg\max_\theta L(\theta) \qquad \rightarrow \qquad \theta_{\text{MAP}} = \arg\max_\theta \Big[L(\theta) + \log P(\theta)\Big],$$

yielding the "maximum a posteriori" estimate, in other words, the mode of the posterior distribution.

3.2   Regularization

Regularization is a general process during which we add a data-independent term to the (data-dependent) cost function before minimizing it.

- We collect data D,

- specify a model M with parameters θ, and log likelihood $L(\theta) = \log P(D \mid \theta)$.

- The cost function is usually derived directly from the log likelihood: $\text{cost}(\theta, D) = -L(\theta)$, but some models define the cost function directly without referencing statistical models.

- We introduce a penalty term $\text{penalty}(\theta)$ that is high for implausible θ values,

- and minimizes the regularized cost to obtain the regularized optimum $\theta_{\text{reg,opt}}$,

$$\theta_{\text{reg,opt}} = \arg\min_\theta (\text{cost}(\theta, D) + \text{penalty}(\theta))$$

3.3   Linear regression

Linear regression is the simplest model where number of model parameter can be systematically increased without bounds. (Once one runs out of raw features, $x_{.,k}$, interactions binary, and higher order, interactions features, e.g. $x_{.,k}x_{.,l}$.) Let's see how regularization is used to guide model fitting.

- The process start by recording K features and one "outcome" variable for each of the N observations. This results in the data set $D = \{ (\{x_{i,k}\}_{k=1}^K, \ y_i) \}_{i=1}^N$, where $x_i \in \mathbb{R}^K$ is a feature vector and $y_i \in \mathbb{R}$ is the outcome variable that we wish the model to predict.

- The linear model

  - aims to predict the outcome y by linearly combining the K components of the feature vector x. For this, the model needs K coefficients (or weights) $b = \{b_k\}_{k=1}^K \in \mathbb{R}^K$.

  - The noise is often modeled as one dimensional normal distribution along y with unknown variance.

  - These two consideration result in the following three (identical) formulation of the model:

$$y_i = \sum_{k=1}^K x_{i,k}b_k + \varepsilon_i, \quad \text{with} \quad P(\varepsilon_i) = \text{Normal}(\varepsilon_i \mid \mu = 0, \sigma^2 = \sigma^2)$$

$$\text{or, equivalently}$$

$$y = Xb + \varepsilon, \quad \text{with} \quad P(\varepsilon) = \text{Multi-variate-normal}(\varepsilon \mid \mu = 0, \Sigma = \mathbb{I}\sigma^2)$$

$$\text{or, equivalently}$$

$$P(y \mid X, b, \sigma^2) = \prod_{i=1}^N \text{Normal}\left(y_i \mid \mu = (Xb)_i, \ \sigma^2 = \sigma^2\right)$$

    where X is a matrix of features, also called the "design matrx", whose elements are $X_{i,k} = x_{i,k}$, and the product between X and b is understood to be matrix multiplication, i.e. $(Xb)_i = \sum_k x_{i,k}b_k$. The symbol $\mathbb{I}$ inside the argument of the multi-variate normal distribution is the K-dimensional identity matrix.

- Using the formula of the normal distribution, the log likelihood can be written as

$$L(b, \sigma^2) = \log P(y \mid X, b, \sigma^2) = -\frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^N \left[y_i - \sum_k x_{i,k}b_k\right]^2$$

Taking the partial derivatives with respect to b_k (and σ^2), setting them to zero and solving the system of equations yields the (non-regularized) maximum likelihood estimates

$$b_{\text{MLE}} = \arg\max_b L(b, \sigma^2) = (X^\top X)^{-1}X^\top y, \qquad (\sigma^2)_{\text{MLE}} = \frac{1}{N}||y - Xb||^2,$$

where $X^\top$ is the transpose of X, i.e. $(X^\top)_{k,i} = x_{i,k}$, and $(X^\top X)_{k,l} = \sum_i x_{i,k}x_{i,l}$, and $(X^\top y)_k = \sum_i x_{i,k}y_i$. The operator $||.||$ stands for the L_2 norm of a vector, in other words, its Euclidean length: $||y|| = \sqrt{\sum_i y_i^2}$.

- The three most commonly used regularization methods are

  - "L1 regularization", which tries to keep the total absolute sum of the coefficients low using a penalty term that is equivalent to specifying Laplace prior for each b_k,

$$\text{penalty}(b) = \alpha_1 \sum_k |b_k| \qquad \Leftrightarrow \qquad P(b_k) = \text{const.} \times e^{-\alpha_1 |b_k|} = \text{Laplace}(b_k \mid \text{loc} = 0, \text{scale} = 1/\alpha_1)$$

– "L2 regularization", which tries to keep the total Euclidean length of the coefficient vector low using a penalty term that is equivalent to specifying normal prior for each b_k,

$$\text{penalty}(b) = \frac{\alpha_2}{2} \sum_k (b_k)^2 \qquad \Leftrightarrow \qquad P(b_k) = \text{const.} \times e^{-\alpha_2 (b_k)^2/2} = \text{Normal}(b_k \mid \mu = 0, \sigma^2 = 1/\alpha_2)$$

– "Elastic net" regularization, which combines L1 and L2 regularization with adjustable weights α_1, α_2,

$$\text{penalty}(b) = \alpha_1 \sum_k |b_k| + \frac{\alpha_2}{2} \sum_k (b_k)^2$$

In all three cases the "hyperparameters" α_1 and α_2 are often optimized using Leave-one-out or M-fold cross-validation.

- Note: A fully Bayesian treatment of regularized models can be carried out by treating the hyperparameters as just another set of model parameters, and determine their posterior (usually numerically).

3.4   Model comparison with asymptotic metrics

In the maximum likelihood estimation context, when models with different set of parameters are fitted to the same data set, we often wish to determine which model has a better potential to generalize to future similar data sets. While some form of regularization (e.g. L1) is able to pick out important model parameters from unimportant ones, we need a different framework when comparing models with completely different structures.

Asymptotic metrics, such as AIC, BIC, LRT p-value are useful for this analysis. Model selection methods based on these metrics, as their name suggests, are reliable only if we are close to the asymptotic limit of infinite data. In practice this requires us to have significantly more observations than model parameters.

Consider the general maximum likelihood fits of two models:

- We focus on a fixed dataset $D = \{x_i\}_{i=1}^N$, and specify two models.

- One model, of lower complexity, is often called the "null model", M_0 with

  – parameters θ_0,
  – log likelihood $L_0(\theta_0) = \log P(D \mid \theta_0, M_0)$, and
  – MLE fit $\theta_{0,\text{MLE}} = \text{argmax}_{\theta_0} L_0(\theta_0)$.

- Another model, of higher complexity, is often called "alternate model", M_1 with

  – parameters θ_1,
  – log likelihood $L_1(\theta_1) = \log P(D \mid \theta_1, M_1)$, and
  – MLE fit $\theta_{1,\text{MLE}} = \text{argmax}_{\theta_1} L_1(\theta_1)$.

Akaike Information Criterion (AIC): This method corrects the log likelihood with the number of parameters $\dim(\theta)$ before comparison.

- $\text{AIC}(M_i) := -2\Big[L_i(\theta_{i,\text{MLE}}) - \dim(\theta_i)\Big]$ for both $i = 0, 1$ models.

- If $\text{AIC}(M_1) < \text{AIC}(M_0)$, then M_1 is more plausible.

Bayesian Information Criterion (BIC): This methods corrects the log likelihood with the number of parameter weighted by the logarithm of the total number of observations. It is considered more robust than AIC.

- $\text{BIC}(M_i) := -2\Big[L_i(\theta_{i,\text{MLE}}) - \frac{\ln(N)}{2}\dim(\theta_i)\Big]$ for both $i = 0, 1$ models.

- If $\mathrm{BIC}(M_1) < \mathrm{BIC}(M_0)$, then M_1 is more plausible.

Likelihood Ratio Test (LRT): If the null model is a restricted version of the alternate model ("nested models"), the likelihood ratio test determine if the extension from the null model to the alternate model increases the likelihood more than expected by random noise in the null model.

- First, we calculate the log likelihood ratio, $\mathrm{logLR} := \log \frac{P(D \mid M_1, \theta_{1,\mathrm{MLE}})}{P(D \mid M_0, \theta_{0,\mathrm{MLE}})} = L_1(\theta_{1,\mathrm{MLE}}) - L_0(\theta_{0,\mathrm{MLE}})$

- We expect $\mathrm{logLR} > 0$, otherwise the null model is unquestionably superior.

- We calculate LRT p-value $:= 1 - \mathrm{cdf}\, \chi^2 \Big(2\, \mathrm{logRL} \,\Big|\, \mathrm{dof} = \dim(\theta_1) - \dim(\theta_0) \Big)$, where $\mathrm{cdf}\, \chi^2(\ldots \mid \mathrm{dof} = d)$ is the cumulative distribution function of the χ^2 distribution with degrees of freedom d.

- A small enough p-value, the threshold of which depends on how big false positive rate we can stomach, indicates that M_1 is more plausible than M_0.

Model evidence: Using the either of the information criteria (IC), we can approximate the evidence for a model.

- The evidence for model M_i is $P(D \mid M_i) = \sum_{\theta_i} P(D \mid \theta_i, M_i) \approx \exp\left(-\frac{1}{2}\, \mathrm{IC}\right)$,

- where IC can be either AIC or BIC (or WAIC, WBIC)

- Under uniform prior on the models (i.e. $P(M_0) = P(M_1)$), the posterior probability of the alternate model being correct is

$$P(M_1 \mid D) \approx \frac{\exp\left(-\frac{1}{2}\, \mathrm{IC}_1\right)}{\exp\left(-\frac{1}{2}\, \mathrm{IC}_0\right) + \exp\left(-\frac{1}{2}\, \mathrm{IC}_1\right)}$$

3.5   Example: Linear regression

Let's see how the asymptotic metrics described above work for a simple liner regression problem.

- We generate data: $D = \{(x_i, y_i)\}_{i=1}^N$, from the distribution $y = 1 - 3x - x^2/2 + x^3 + \varepsilon$ with $\text{std}(\varepsilon) = 2$.

```
1  import numpy as np
2  from numpy.polynomial.polynomial import polyval
3  from scipy.stats import norm
4
5  c_true = [1, -3, -0.5, 1]
6  sigma_true = 2
7  x_data = np.linspace(-3, 3, 20)
8  y_data = [polyval(x, c_true) + norm.rvs(loc=0, scale=sigma_true)
9           for x in x_data]
```
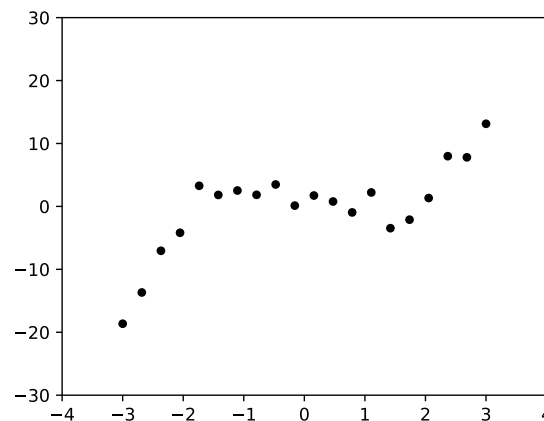

- Let's consider a series of models: M_K, where $K = 0, 1, 2, \ldots 11$ indicates the degree of the polynomial for the "main curve". For each model M_K the parameter vector is $c = (c_0, c_1, \ldots c_K) \in \mathbb{R}^{K+1}$.

- Linear features vector is $X_i := (1, x_i, (x_i)^2, (x_i)^3, \ldots (x_i)^K) \in \mathbb{R}^{K+1}$.

```
1  def generate_polynomial_features(x_data, degree):
2      K = degree
3      N = len(x_data)
4      X = np.zeros([N, K+1])
5      for i, x in enumerate(x_data):
6          for k in range(0, K+1, 1):
7              X[i,k] = x**k
8      return X
```

- The log likelihood for the model $y = \sum_{k=0}^K X_{i,k} c_k + \varepsilon$, with $P(\varepsilon) = \text{Normal}(\varepsilon \mid 0, \sigma^2)$, is $L(c) = \sum_i \log\left(\text{Normal}(y_i \mid \mu = (Xc)_i, \sigma^2 = \sigma^2)\right)$

```
1  def log_likelihood(X, y, c, sigma2):
2      N = len(y_data)
3      log_like = 0
4      log_like += - N/2.0 * np.log(sigma2)
5      log_like += - 1.0/(2 * sigma2) * vector_norm(y - X.dot(c))**2
6      return log_like
```

- We calculate the MLE solution using the formulas $c_{\mathrm{MLE}} = (X^\top X)^{-1} X^\top y$, $\quad (\sigma^2)_{\mathrm{MLE}} = \frac{1}{N}||y - Xc_{\mathrm{MLE}}||^2$,

```
1  def fit_MLE_ploynomial(X, y):
2      c_MLE = inv(X.T.dot(X)).dot(X.T).dot(y_data)
3      sigma2_MLE = 1.0/len(y) * vector_norm(y - X.dot(c_MLE))**2
4      return c_MLE, sigma2_MLE
```

We plot the fits on top of the data below, where the solid curve shows the mean of the prediction y as a function of x, and the shaded region around it indicates the 2σ vertical interval around it. Visualizing this way and checking that $\sim 97.5\%$ (19-20) of the data points is covered by the shaded region allows us to verify that all fits are consistent with the data.
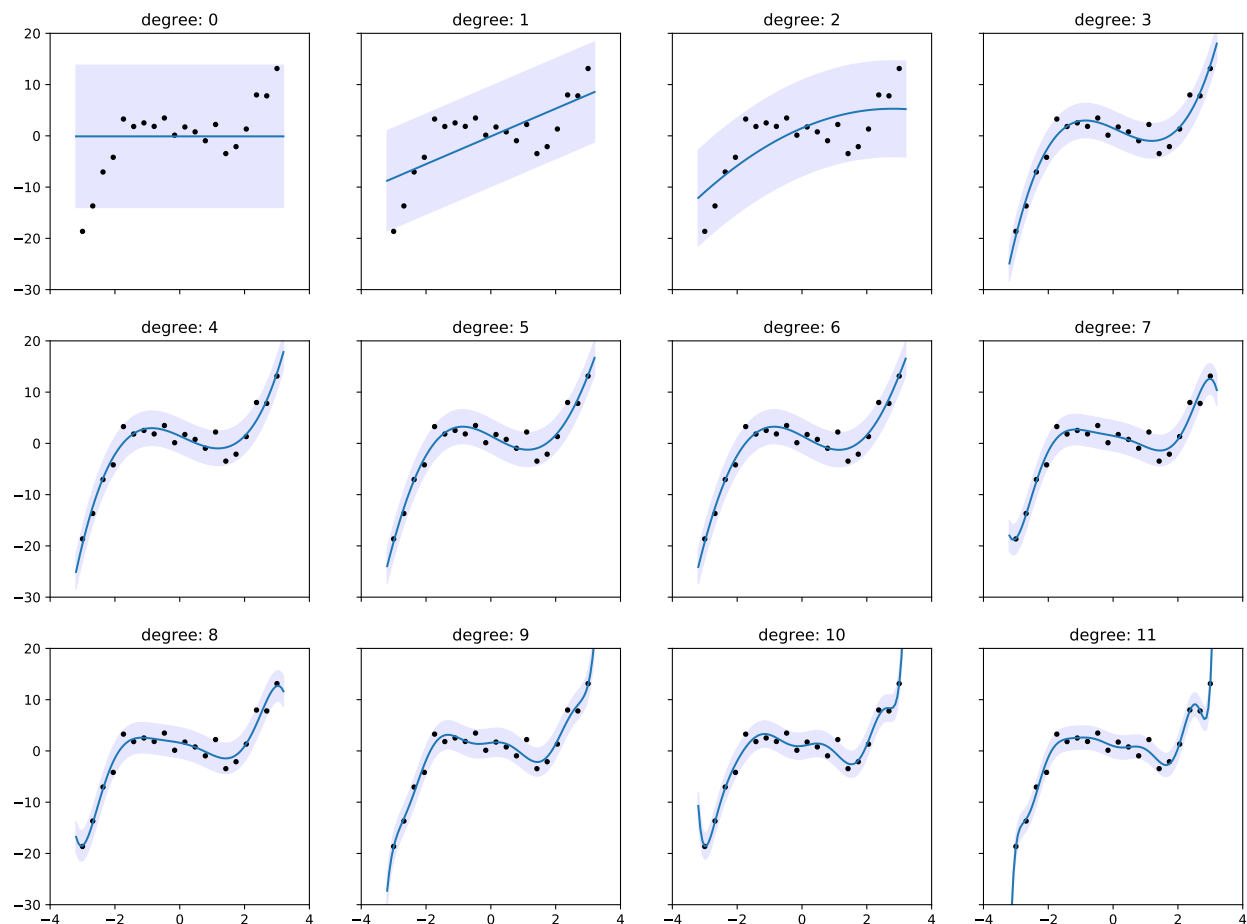
Let's compare the different models $M_0, M_1, M_2 \ldots M_{11}$ using AIC, BIC, likelihood ratio test.

- AIC for model M_K is $\text{AIC}_K = -2[L_K - (K+2)]$, where $K + 2 = \dim(c, \sigma^2) = \dim(c) + \dim(\sigma^2) = (K+1) + 1$, and L_K is the maximal likelihood achievable with with model M_K.

```
1  def AIC(X, y, c, sigma2):
2          dim = len(c) + 1
3          loglike = log_likelihood(X, y, c, sigma2)
4          return -2 * (loglike - dim)
```

- BIC for model M_K is $\text{BIC}_K = -2[L_K - \frac{\ln N}{2}(K+2)]$

```
1  def BIC(X, y, c, sigma2):
2          N = len(y)
3          dim = len(c) + 1
4          loglike = log_likelihood(X, y, c, sigma2)
5          return -2 * (loglike - np.log(N)/2.0 * dim)
```

- Comparing model M_{K-1} (as null model) and M_K (as alternate model) yields the likelihood ratio test p-value $\text{LRT pvalue}_K = 1 - \text{cdf } \chi^2(2(L_K - L_{K-1}) \mid \text{dof} = 1)$

```
1  from scipy.stats import chi2
2
3  pvalues = [np.nan]
4  for deg in degrees[1:]:
5      L1 = loglikes[deg]
6      L0 = loglikes[deg-1]
7      logLR = L1 - L0
8      dof = 1
9      pvalue = chi2.sf(2*logLR, dof)
10     pvalues.append(pvalue)
```

We plot the log likelihood and the three asymptotic metrics (-2AIC_K, -2BIC_K and $-\log_{10}(\text{LRT p-value}_K)$) as a function of the polynomial degree K below. All three metrics show that model M_3 stands out from the rest, indicating that this is probably the correct one.
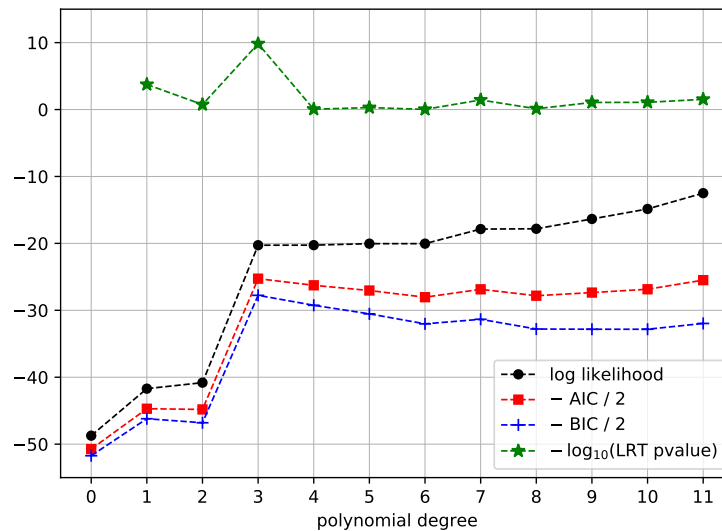
We can (approximately) calculate the probability that each model is correct.

- Using the BIC, the model evidence is $P(D \mid M_k) \approx e^{-\text{BIC}_k/2} / \sum_{k'=0}^{K} e^{-\text{BIC}_{k'}/2}$

```
1  def BIC_weigths(BICs):
2          BICs = np.array(BICs)
3          w = BICs − np.min(BICs)   # to avoid underflow in the next line
4          w = np.exp(−0.5*(w))
5          w /= np.sum(w)
6          return w
```

Comparing the probabilities visually tells us a lot about the plausibility of the different models. We can see that models M_0, M_1 and M_2 have no chance, M_3 has the highest chance of being correct ($\sim 73\%$), while M_4 and M_5 are still possible, albeit with lower probabilities, $\sim 17\%$ and $\sim 5\%$, respectively.
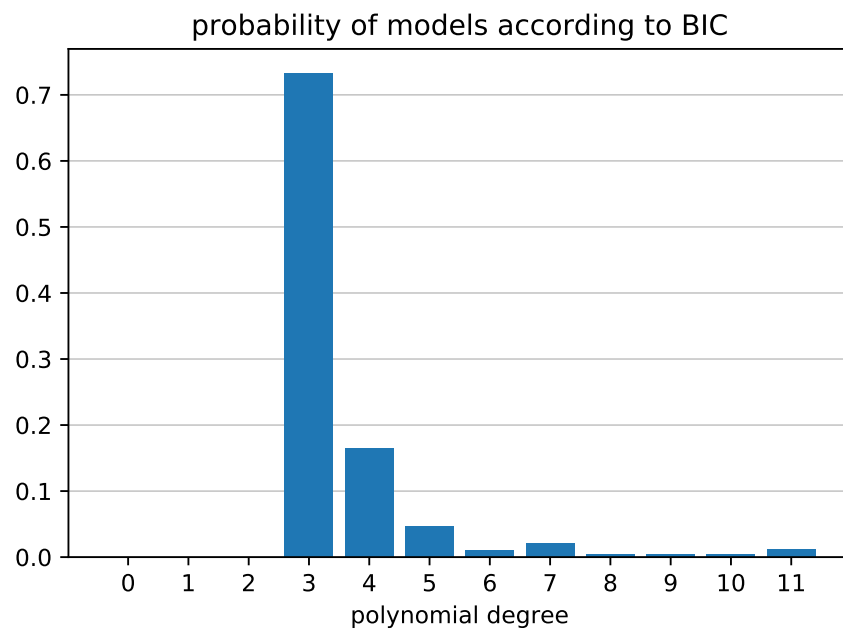
4   Graphical models

As the number of variables we wish to model increases the number of different models increases faster than exponentially. A common technique to get a handle on the dependencies of many real-world variables is to organize them into connected structures the components of which are simple and easily interpretable. This allows us to construct hierarchies, a powerful method for synthesizing knowledge.

Representing variables with the nodes and dependencies with the edges of a graph is a powerful method to gain a bird's eye view of the model and reason about statistical dependencies between the variables under different fixed conditions. This section introduces elements of a graphical representation technique using directed, acyclic graphs. (Other methods are undirected graphs and factor graphs.)

4.1   Elements

- A **directed connection** from variable x, to variable y, indicates that we intend to describe their joint distribution using the conditional probability $P(y \mid x)$, i.e. $P(x, y) = P(y \mid x)P(x)$. The graphical representation of this is

$$x \longrightarrow y$$

- A **chain** of connections, which is formed by two directed connections, one from x to z and another from z to y, indicates that we intend to describe the joint as $P(x, y, z) = P(y \mid z)P(z \mid x)P(x)$. This is represented by a the following graph.

$$x \longrightarrow z \longrightarrow y$$

  Note: This graph represents a restriction to the set of distributions for the joint, because the variable z separates x and y, indicating that x and y become independent once we fix the value of z (to any value), i.e. $P(x, y \mid z) = P(x \mid z)P(y \mid z)$. This, we write as $y \perp\!\!\!\perp x \mid z$. On the other hand, if z is unknown, x and y may be dependent, which we write as $y \not\perp\!\!\!\perp x \mid \emptyset$. The z variable in the middle in such situations is called the "mediator" between x and y.

- A **fork** is formed when two connections originate from one variable x and connect to two different variables y_1 and y_2, indicating that we intend to describe their joint as $P(x, y_1, y_2) = P(y_1 \mid x)P(y_2 \mid x)P(x)$. This is represented by a the following graph.

  Note: This dependency structure implied by this graph (on its own) is identical to the dependency structure of the chain, i.e. $y_1 \perp\!\!\!\perp y_2 \mid x$, and $y_1 \not\perp\!\!\!\perp y_2 \mid \emptyset$, but the interpretation is different. Variables x is interpreted as a "common cause" for y_1 and y_2.

- The **collider**, which is formed by directed connections from two variables x_1 and x_2 to one variable y, is the most interesting graphical element. It indicates that we believe that x_1 and x_2 are independent variables, i.e. the we expect to write the joint as $P(x_1, x_2, y) = P(y \mid x_1, x_2)P(x_1)P(x_2)$. This is represented by a the following graph.

  Note: By definition, this dependency structure implies $x_1 \perp\!\!\!\perp x_2 \mid \emptyset$, however the same does not hold if the value of y is fixed, $x_1 \not\perp\!\!\!\perp x_2 \mid y$. This effect is called "explaining away".

4.2 General rules

Converting between graph and formula

The general rules for converting between a graphical representation and the formula of the joint distribution are the following

- Each variable x (node of the graph) contributes a factor $P(x \mid \ldots)$ to the formula. The only question is what should go in place of $\ldots$ in the condition part.

- If the node has no incoming edges, the condition part should be left empty, i.e. the factor is $P(x \mid \emptyset) = P(x)$.

- If the node has incoming connections, the variables from which the connections originate ("parents") should be listed in the condition part, i.e. the factor is $P(x \mid \text{parents of } x)$.
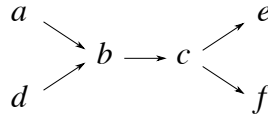
d-separation

Two variables are said to be "d-separated" if their independence (under specific conditions) is implied by the graph structure. For the general case of trying to determine independence between x and y under the condition of keeping variables $z_1, z_2, \ldots z_c$ fixed, the steps are the following.

1. List all possible paths between x and y through the graph disregarding the directions of the connections (i.e. moving in the opposite direction of the connections is also allowed). Each node is allowed to be visited at most once in every path.

2. For each node on each path, mark the way it is traversed with respect to the direction of the connections: mediator ($\leftarrow\leftarrow$ or $\rightarrow\rightarrow$), fork ($\leftarrow\rightarrow$) or collider ($\rightarrow\leftarrow$). (Note: It is possible for a node with three or more connections to be traversed differently by different paths.)

3. Disregard all paths of which any of the mediator or fork nodes is among the $\{z_1, z_2, \ldots z_c\}$ variables, on which we are conditioning. These paths are "blocked".

4. On the remaining paths, investigate the colliders. Going against our intuition, a collider blocks the path if **neither** itself **nor** any of its descendents is among $\{z_1, z_2, \ldots z_c\}$. (E.g. if we do not condition on any variable, then all paths that traverse at least one node in collider fashion are blocked.) Disregard these paths.

5. If all paths are blocked, in one way or another, x and y are conditionally independent, conditioned on the variables $z_1, z_2, \ldots z_c$, which we write as $x \perp\!\!\!\perp y \mid z_1, z_2, \ldots z_c$.
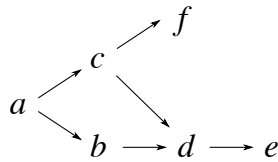
Examples

- $P(a, b, c, d, e) = P(a)P(d)P(b \mid a, d)P(c \mid b)P(e \mid c)P(f \mid c)$ is represented by

  Some examples of d-separation are $e \perp\!\!\!\perp f \mid c, \quad e \perp\!\!\!\perp a \mid b$ and $a \perp\!\!\!\perp d \mid \emptyset$, (but $a \not\perp\!\!\!\perp d \mid b$ and $a \not\perp\!\!\!\perp d \mid f$).

- $P(a, b, c, d, e, f) = P(a)P(c \mid a)P(b \mid a)P(f \mid c)P(d \mid b, c)P(e \mid d)$ is represented by

  Some examples of d-separation are $b \perp\!\!\!\perp f \mid a$ and $b \perp\!\!\!\perp f \mid a, c, e$ (but $b \not\perp\!\!\!\perp f \mid a, d$ and $b \not\perp\!\!\!\perp f \mid a, e$).

4.3   Real-life examples

- Fire causes Smoke, Smoke causes Alarm to set off, but given Smoke, there's no correlation between Fire and Alarm, i.e. Fire $\perp\!\!\!\perp$ Alarm | Smoke. This is represented by a chain

$$\text{Fire} \longrightarrow \text{Smoke} \longrightarrow \text{Alarm}$$

- Both rain and the Sprinkler can cause the formation of a Puddle, they are however independent (until we observe the Puddle), i.e. Rain $\perp\!\!\!\perp$ Sprinkler | $\emptyset$. This is represented by a collider

$$\begin{array}{c} \text{Rain} \\ \searrow \\ \text{Puddle} \\ \nearrow \\ \text{Sprinkler} \end{array}$$
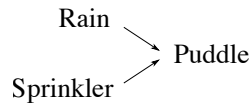
- Heat causes both Ice Cream sales and Crime to increase, but once we know there was a heatwave, they become independent, i.e. Crime $\perp\!\!\!\perp$ Ice Cream | Heat. This is represented by a fork

$$\begin{array}{c} \text{Crime} \\ \nearrow \\ \text{Heat} \\ \searrow \\ \text{Ice Cream sales} \end{array}$$
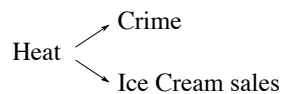
- Education affects Political View, which affects both Party membership and Voting behavior. This can be represented as

$$\text{Education} \longrightarrow \text{Political View} \longrightarrow \text{Vote}$$
$$\downarrow$$
$$\text{Party membership}$$
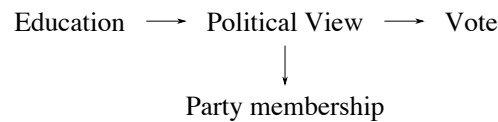
- Education and Experience both affect Salary, but Education also affects Experience. This can be represented as

$$\begin{array}{c} \text{Education} \\ \downarrow \quad \searrow \\ \quad \quad \text{Salary} \\ \downarrow \quad \nearrow \\ \text{Experience} \end{array}$$
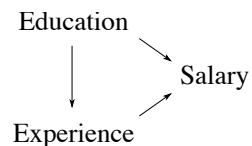
4.4   Plate notation

When a sequence of variables are connected to the rest of the graph in identical fashion, we simplify our notation by placing a representative of the variables inside a box ("on a plate") in place of the series, and label the box with the length of the sequence.
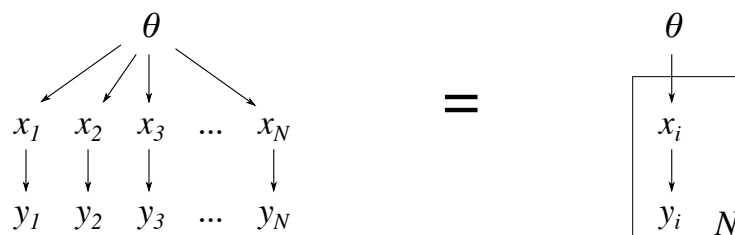
4.5   Hierarchical models

Using the graphical representation, we can define and understand models of multiple levels of hierarchy with ease.

The usual difficulty that comes with multi-level models is the rapid increase of the number of parameters. If not kept in check, this leads to overfitting (in MLE context) or inefficient sampling (in MCMC context). Once in a while, variables on intermediate levels of a hierarchical model can be marginalized out, giving rise to a log likelihood function with fewer parameters that enable a more efficient and more robust solutions.

Beta-Binomial model

Often different experimental runs, each of which contain observations with binary outcomes, have similar but not identical success rates. E.g. the win rate of athletes in a specific league is expected to be similar (since they are in the same league), but we certainly can't rule our individual differences. In such a case, the Beta-Binomial model can robustly estimate the distribution of win rates in the league.

- We observe N experimental runs, where in the ith experiment we find k_i successes out of n_i attempts. The data is $D = \{(k_i, n_i)\}_{i=1}^N$, where $0 \leq k_i \leq n_i$, with $k_i, n_i \in \mathbb{N}$.

- Model:

  - One level 1, we model number of successes with individual success rates, $p = \{p_i\}_{i=1}^N$, where $p_i \in [0,1]$, and assume that measurements are independent (given p). This leads to k_i being distributed as $P(k_i \mid n_i, p_i) = \text{Binomial}(k_i \mid n_i, p_i)$.

  - On level 2, we assume that the success rates come from a Beta distribution with parameters, α, β (both > 0), i.e. define $P(p_i \mid \alpha, \beta) = \text{Beta}(p_i \mid \alpha, \beta)$.

  This hierarchy can be represented as
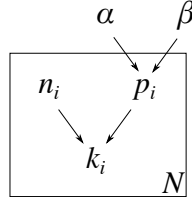
- If we treat the success rates as hidden data, we can write the likelihood as

$$P(D, p \mid \alpha, \beta) = \prod_{i=1}^N \left[\text{Binom}(k_i \mid n_i, p_i) \, \text{Beta}(p_i \mid \alpha, \beta) \right]$$

- For this model, we can marginalize out p analytically, yielding a more useful form of the likelihood:

$$P(D \mid \alpha, \beta) \;\; = \;\; \prod_{i=1}^N \left[\int dp_i \, \text{Binom}(k_i \mid n_i, p_i) \, \text{Beta}(p_i \mid \alpha, \beta) \right] = \prod_{i=1}^N \left[\text{Beta-Binom}(k_i \mid n_i, \alpha, \beta) \right]$$

$$\text{where} \qquad \text{Beta-Binom}(k \mid n, \alpha, \beta) = \frac{\Gamma(n+1)\Gamma(\alpha+\beta)}{\Gamma(n+\alpha+\beta)} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)} \frac{\Gamma(n-k+\beta)}{\Gamma(n-k+1)\Gamma(\beta)}$$

Using the definition of the beta function, $B(x,y) = \Gamma(x)\Gamma(y)/\Gamma(x+y)$, we can write the log likelihood as

$$L(\alpha, \beta) = \log(n) + \log B(n, \alpha+\beta) - \Big[\log(k) + \log B(k, \alpha) \Big]_{\text{if } k \neq 0} - \Big[\log(n-k) + \log B(n-k, \beta) \Big]_{\text{if } k \neq n}$$

which is numerically more robust to evaluate (because $\log B$ is often implemented directly in numerical packages, e.g. by `scipy.special.betaln` in python), and statistically more robust to optimize.

Gamma-Poisson model (aka. Negative Binomial model)

If we observe multiple sequences of events, each of which is generated by a Poisson process, but we think that the Poisson parameters are not identical, then the Gamma-Poisson model can provide robust estimates of the distribution of Poisson parameters. This is the simplest model for an inhomogeneous population of counts.

- We observe a sequence of counts (e.g. number of events), one for each experiment. The data is $D = \{k_i\}_{i=1}^N$, where k_i is the count observed in the ith experiment.

- Model:

  - On level 1, we model the counts from each experiment with a Poisson distribution $P(k_i \mid \lambda) = \text{Poisson}(k_i \mid \lambda_i)$. The Poisson parameters $\lambda = \{\lambda_i\}_{i=1}^N$, $\lambda_i > 0$, represent the typical number of events in each experiment.

  - On level 2, we assume that the Poisson parameters come from a Gamma distribution with parameter α, β (both > 0), i.e. $P(\lambda_i \mid \alpha, \beta) = \text{Gamma}(\lambda_i \mid \alpha, \beta)$.

This hierarchy can be represented as
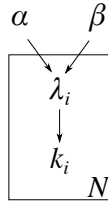
- If we treat λ parameters as hidden data, we can express the likelihood as

$$P(D, \lambda \mid \alpha, \beta) = \prod_{i=1}^N \left[\text{Poisson}(k_i \mid \lambda_i) \, \text{Gamma}(\lambda_i \mid \alpha, \beta) \right]$$

- For this model, we can integrate over λ analytically, which produces the marginal likelihood

$$P(D \mid \alpha, \beta) = \prod_{i=1}^N \left[\int d\lambda_i \, \text{Poisson}(k_i \mid \lambda_i) \, \text{Gamma}(\lambda_i \mid \alpha, \beta) \right] = \prod_{i=1}^N \left[\text{Gamma-Poisson}(k_i \mid \alpha, \beta) \right]$$

where

$$\text{Gamma-Poisson}(k \mid \alpha, \beta) = \frac{\Gamma(k + \alpha)}{\Gamma(k+1)\Gamma(\alpha)} \left(\frac{1}{\beta + 1} \right)^k \left(\frac{\beta}{\beta + 1} \right)^\alpha =$$

$$= \text{NegativeBinom}(k \mid r, p) = \binom{k + r - 1}{k} p^k (1 - p)^r, \quad \text{with } r = \alpha, \ p = \frac{1}{\beta + 1}$$

The log likelihood can be written as

$$L(\alpha, \beta) = \sum_{i=1}^N \left(\alpha \log(\beta) - (\alpha + k_i) \log(\beta + 1) - \Big[\log(k_i) + \log B(k_i, \alpha) \Big]_{\text{if } k_i \neq 0} \right)$$

which is numerically more robust to evaluate (because $\log B$ is often implemented directly in numerical packages, e.g. by `scipy.special.betaln` in python), and statistically more robust to optimize.

Dirichlet-Multinomial model

In a sequence of experimental runs, each of which contains experiments with more than two possible outcomes, we may believe that the chances of the possible outcomes is similar between runs, but not exactly the same. In such a setting, the Dirichlet-Multinomial model can robustly estimate the population of outcome probabilities.

- We perform N experimental runs, in each of which the individual experiments have M possible outcomes. (E.g. a classroom of N children visits an ice cream shop that sells M flavors of ice cream every week.) The ith run contains n_i attempts, out of which the different outcomes are present $(k_{i,1}, k_{i,2}, \ldots k_{i,M})$ number of times. The data is $D = \{k_i \in \mathbb{N}^M\}_{i=1}^N = \{(k_{i,1}, k_{i,2}, \ldots k_{i,M})\}_{i=1}^N$, where $k_{i,j} \in \mathbb{N}$ is the number of times outcome j was observed in run i, $0 \le k_{i,j} \le n_i$, and $\sum_{j=1}^M k_{i,j} = n_i$.

- Model:

  - On level 1, we model each run i with a probability vector $p_i = (p_{i,1}, p_{i,2}, \ldots p_{i,M})$, which indicates the probability of each outcome in this run. The full set of such vectors is $p = \{p_i \in \mathbb{R}^M\}_{i=1}^N = \{(p_{i,1}, p_{i,2}, \ldots p_{i,M})\}_{i=1}^N$, where $p_{i,j} > 0$, and $\sum_{j=1}^M p_{i,j} = 1$, $\forall i$. Assuming that the experiments are independent (given p) leads to $P(k_i \mid n_i, p_i) = \text{Multinomial}(k_i \mid n_i, p_i)$.

  - On level 2 we assume that each probability vector p_i comes from a common Dirichlet distribution with parameters $\alpha = (\alpha_1, \alpha_2, \ldots \alpha_M)$, where each $\alpha_j > 0$. This defines $P(p_i \mid \alpha) = \text{Dirichlet}(p_i \mid \alpha)$.

This hierarchy can be represented as
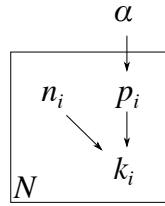
- Treating the success probability vectors as hidden data, we can write the likelihood as

$$P(D, p \mid \alpha) = \prod_{i=1}^N \Big[\text{Multinomial}(k_i \mid n_i, p_i) \, \text{Dirichlet}(p_i \mid \alpha) \Big]$$

- For this model, we can marginalize out p, and obtain the likelihood for α as

$$P(D \mid \alpha) = \prod_{i=1}^N \left[\int dp_i \, \text{Multinomial}(k_i \mid n_i, p_i) \, \text{Dirichlet}(p_i \mid \alpha) \right] = \prod_{i=1}^N \Big[\text{Dirichlet-Multinomial}(k_i \mid n_i, \alpha) \Big]$$

where
$$\text{Dirichlet-Multinomial}(k \mid n, \{\alpha_j\}_{j=1}^M) = \frac{\Gamma(n+1)\Gamma(\alpha_{\text{tot}})}{\Gamma(n + \alpha_{\text{tot}})} \prod_{j=1}^M \frac{\Gamma(k_j + \alpha_j)}{\Gamma(k_j + 1)\Gamma(\alpha_j)},$$

where $\alpha_{\text{tot}} = \sum_{j=1}^M \alpha_j$. The log likelihood can be written as

$$L(\alpha) = \log(n) + \log B(n, \alpha_{\text{tot}}) - \Big[\log(k_j) + \log B(k_j, \alpha_j) \Big]_{\text{if } k_j \ne 0}$$

which is numerically more robust to evaluate (because $\log B$ is often implemented directly in numerical packages, e.g. by `scipy.special.betaln` in python), and statistically more robust to optimize.

Random Effect Model

We often group observations by a feature that we suspect to be relevant for determining the measured quantity (e.g. grouping height measurements by gender). Random Effect Model can be used to robustly quantify how much of the populations variance is explained by the feature we used for grouping, i.e. how much smaller is the within-group variance compared to the total variance.

- We observe data, which we group into G non-overlapping groups: $D = \{x_g\}_{g=1}^G = \{(x_{g,1}, x_{g,2}, \ldots x_{g,N_g})\}_{g=1}^G$, where $x_{g,i} \in \mathbb{R}$ is measurement i in group g, and groups can be of different sizes N_g.

- We construct the following hierarchical model:

  - On level 1, we assume that each observation comes from its group's distribution $P(x_{g,i} \mid \mu_g, \sigma) = \text{Normal}(x_{g,i} \mid \mu_g, \sigma^2)$, where $\mu = \{\mu_g\}_{g=1}^G \in \mathbb{R}^G$ are the centers for each group and $\sigma^2 > 0$ is the **within-group variance**, shared by the groups.

  - On level 2, we assume that each group center μ_g comes from a global distribution, $P(\mu_g \mid \mu_0, \sigma_0) = \text{Normal}(\mu_g \mid \mu_0, \sigma_0^2)$, where $\mu_0 \in \mathbb{R}$ is the global center, and $\sigma_0^2 > 0$ is the **between-groups variance**. (The total variance is $\sigma^2 + \sigma_0^2$.)

This hierarchy can be represented by the following graph
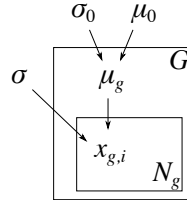
- We regard the group centers as hidden data, and write the likelihood (of μ_0, σ_0, σ) as

$$P(D, \mu \mid \mu_0, \sigma_0, \sigma) = \prod_{g=1}^G \left[\text{Normal}(\mu_g \mid \mu_0, \sigma_0^2) \times \prod_{i=1}^{N_g} \text{Normal}(x_{g,i} \mid \mu_g, \sigma^2) \right]$$

- Integrating over the group centers yields the marginal likelihood

$$
\begin{aligned}
P(D \mid \mu_0, \sigma_0, \sigma) &= \prod_{g=1}^G \int_{-\infty}^{+\infty} d\mu_g \left[\text{Normal}(\mu_g \mid \mu_0, \sigma_0^2) \times \prod_{i=1}^{N_g} \text{Normal}(x_{g,i} \mid \mu_g, \sigma^2) \right] \\
&= \prod_{g=1}^G \left[(2\pi\sigma^2)^{-\frac{N_g}{2}} \sqrt{\frac{\sigma^2}{\sigma^2 + N_g \sigma_0^2}} \ \exp\left(-\frac{N_g}{2} \frac{(\mu_0 - m_g)^2}{\sigma^2 + N_g \sigma_0^2} - \frac{N_g s_g^2}{2\sigma^2} \right) \right],
\end{aligned}
$$

where $m_g = \frac{1}{N_g} \sum_{i=1}^{N_g} x_{g,i}$ and $s_g^2 = \frac{1}{N_g} \sum_{i=1}^{N_g} x_{g,i}^2 - m_g^2$ are the observed average and variance of the observations in group g. The log likelihood is

$$L(\mu_0, \sigma_0^2, \sigma^2) = \sum_{g=1}^G \left[-\frac{N_g}{2} \log(2\pi\sigma^2) - \frac{1}{2} \log\left(\frac{\sigma^2}{\sigma^2 + N_g \sigma_0^2} \right) - \frac{N_g}{2} \frac{(\mu_0 - m_g)^2}{\sigma^2 + N_g \sigma_0^2} - \frac{N_g s_g^2}{2\sigma^2} \right].$$

Maximizing this function with respect to $(\mu_0, \sigma_0, \sigma)$ is a more efficient than maximizing $P(D, \mu \mid \mu_0, \sigma_0, \sigma)$ with respect to $(\mu, \mu_0, \sigma_0, \sigma)$ and yields more robust estimates.

4.6   Example: Beta-Binomial

The life-time performance of 10 different boxers are collected. Wins (k_i) and losses $(n_i - k_i)$ are tallied, and the observed win rate $p_{i,\text{obs}} = k_i/n_i$ is calculated. This is shown below
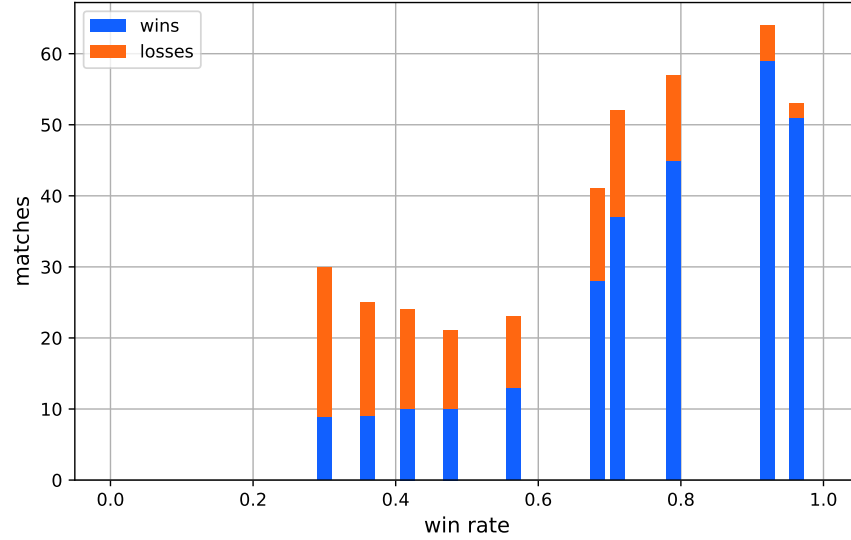
We would like to determine the win rate of an "typical boxer", i.e. the distribution of the win rate p. While the observed values p_{obs} are good estimates of the individual win rates, 5 boxers played less than 30 matches, while 5 played more than 40, which means the second group provides more information, and their observed win rates need to be taken with more certainty. The Beta-Binomial hierarchical model accounts for this inhomogeneity.

- Data: $\{n_i\} = [24, 23, 30, 21, 25, 53, 41, 52, 64, 57]$, $\{k_i\} = [10, 13, 9, 10, 9, 51, 28, 37, 59, 45]$, with $N = 10$.

- Parameters: $\alpha, \beta > 0$

- Model:

$$\log P(D \mid \alpha, \beta) = \sum_{i=1}^{N} \log \Big(\text{Beta-Binom}(k_i \mid n_i, \alpha, \beta) \Big)$$

$$\text{where} \quad \log \Big(\text{Beta-Binom}(k \mid n, \alpha, \beta) \Big) = -f(n, \alpha + \beta) + f(k, \alpha) + f(n - k, \beta),$$

$$\text{where} \quad f(k, \alpha) = \log \Gamma(k + \alpha) - \log \Gamma(k + 1) - \log \Gamma(\alpha)$$

which can be implemented as

```
from scipy.special import gammaln

def log_three_gamma_term(k, a):
    return gammaln(k+a) - gammaln(k+1) - gammaln(a)

def log_beta_binom(k, n, a, b):
    target = 0
    target += - log_three_gamma_term(n, a+b)
    target += log_three_gamma_term(k, a)
    target += log_three_gamma_term(n-k, b)
    return target
```

```
12
13  def log_likelihood(k, n, a, b):
14      target = 0
15      for ki, ni in zip(k, n):
16          target += log_beta_binom(ki, ni, a, b)
17      return target
```

- While assuming flat priors for α and β, we can numerically calculate their joint posterior and means.

```
1 import numpy as np
2 import pandas as pd
3
4 a_arr, b_arr = np.meshgrid(np.linspace(0.1, 20, 100),
5   np.linspace(0.1, 20, 100))
6 a_arr = a_arr.flatten()
7 b_arr = b_arr.flatten()
8
9 loglike = []
10 for a, b in zip(a_arr, b_arr):
11   loglike.append(log_likelihood(k, n, a, b))
12
13 df = pd.DataFrame({
14   'a': a_arr,
15   'b': b_arr,
16   'loglike': loglike
17 })
18
19 df['pstar'] = np.exp(df['loglike'] - df['loglike'].max())
20 Z = df['pstar'].sum()
21 df['posterior'] = df['pstar'] / Z
22
23 post_a = df.groupby(by='a')['posterior'].sum().reset_index()
24 post_b = df.groupby(by='b')['posterior'].sum().reset_index()
25
26 a_mean = (post_a['posterior'] * post_a['a']).sum()
27 b_mean = (post_b['posterior'] * post_b['b']).sum()
```

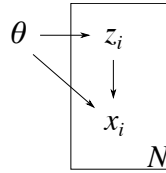giving $\mathbb{E}(\alpha \mid D) = 4.142$, $\mathbb{E}(\beta \mid D) = 2.289$.

Their (marginal) posteriors and the distribution of the win rate (for the mean $\alpha$ and $\beta$ values) are below.
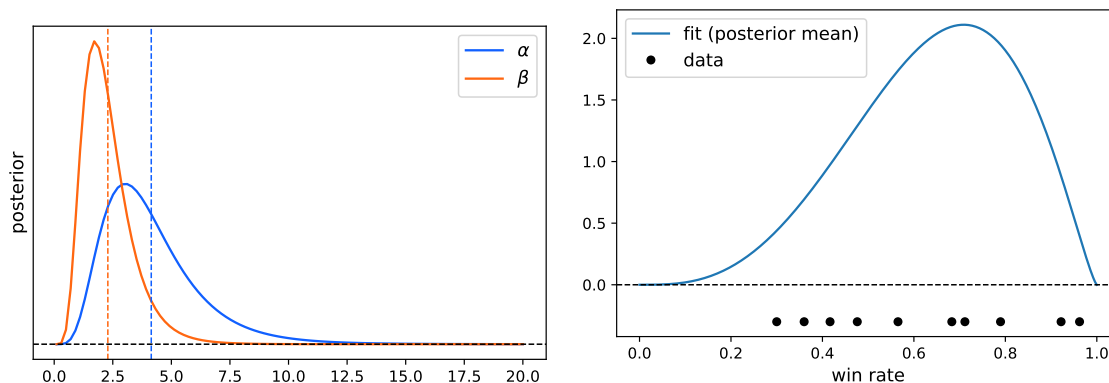
# 5    Hidden variables, EM, Mixture models

## 5.1    Definitions

- Known values:

    - Observations (or data), $D = \{x_i\}_{i=1}^N$

- Unknown values:

    - Parameters: $\theta = \{\theta_k\}_{k=1}^K$
    - Hidden variables (or hidden data): $Z = \{z_i\}_{i=1}^N$ (i.e. $Z$ is as numerous as $D$)



## 5.2    Expectation Maximization

Goal

- Given the likelihood $P(D \mid Z, \theta)$, and prior on hidden variables $P(Z \mid \theta)$,

- The joint is $P(D, Z \mid \theta) = P(D \mid Z, \theta)\, P(Z \mid \theta)$

- The marginal is $P(D \mid \theta) = \sum_Z P(D, Z \mid \theta)$

- We wish to find $\theta$ that maximizes the marginal, i.e.

$$\theta^{\text{MLE}} = \arg\max_\theta \left[ \log P(D \mid \theta) \right] = \arg\max_\theta \left[ \log \left( \sum_Z P(D, Z \mid \theta) \right) \right]$$

- Direct numerical optimization is usually feasible, but the EM method is often faster.

Expectation Maximization (EM) algorithm:

1. Start with realistic $\theta = \theta^{\text{old}}$

2. E-step: Calculate $P(Z \mid D, \theta^{\text{old}}) = \frac{P(D, Z \mid \theta^{\text{old}})}{\sum_{Z'} P(D, Z' \mid \theta^{\text{old}})}$

3. M-step: Find the optimal $\theta = \theta^{\text{new}}$ that maximizes $\sum_Z P(Z \mid D, \theta^{\text{old}}) \log P(D, Z \mid \theta)$

4. Set $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$, check convergence, and return to E-step if needed.

- The one-liner iteration formula is based on the joint $P(D, Z \mid \theta)$:

$$\theta^{\text{new}} = \arg\max_\theta \left[ \sum_Z \frac{P(D, Z \mid \theta^{\text{old}})}{\sum_{Z'} P(D, Z' \mid \theta^{\text{old}})} \log P(D, Z \mid \theta) \right]$$

## 5.3   Mixture models

- Data: $D = \{x_i\}_{i=1}^N$

- Parameters: $\theta, w$

    - Components: $c \in \{1, 2, \ldots C\}$
    - Parameters for each component $\theta = \{\theta_c\}_{c=1}^C$
    - Mixing proportions: $w = \{w_c \in [0, 1]\}_{c=1}^C$, such that $\sum_c w_c = 1$.

- Hidden variables: labels $L = \{l_i\}_{i=1}^N$, where $l_i \in \{1, 2, \ldots C\}$

- Model: mixture of distinct distributions

    - Generative distribution for each component: $P(x_i \mid l_i = c, \theta) = P(x_i \mid \theta_c)$
    - Probability of an observation coming from a component: $P(l_i = c) = w_c$
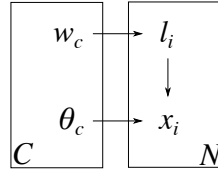


- Joint

$$P(D, L \mid \theta, w) = P(D \mid L, \theta)\, P(L \mid w) = \prod_{i=1}^N P(x_i \mid \theta_{l_i})\, w_{l_i}$$

- Marginal

$$P(D \mid \theta, w) = \sum_L P(D, L \mid \theta, w) = \prod_{i=1}^N \left[ \sum_{c=1}^C w_c\, P(x_i \mid \theta_c) \right]$$

EM algorithm

1. Start with initial values: $\theta^{\mathrm{old}}, w^{\mathrm{old}}$

2. In E-step, calculate
$$P(l_i = c \mid x_i, \theta^{\mathrm{old}}) = \frac{w_c^{\mathrm{old}} P(x_i \mid \theta_c^{\mathrm{old}})}{\sum_{c'} w_{c'}^{\mathrm{old}} P(x_i \mid \theta_{c'}^{\mathrm{old}})} =: r_{i,c}$$

3. In M-step, calculate

$$
\begin{aligned}
w_c^{\mathrm{new}} &= \frac{1}{N} \sum_{i=1}^N r_{i,c} \\
\theta_c^{\mathrm{new}} &= \arg\max_{\theta_c} \left[ \sum_{i=1}^N r_{i,c} \log P(x_i \mid \theta_c) \right] \\
&= \text{MLE of } \theta \text{ with data weights } \{r_{i,c}\}_{i=1}^N
\end{aligned}
$$

## 5.4   Gaussian Mixture Model

(aka. GMM or "soft K-means clustering")

- Data: $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ (point in $d$ dimension)

- Parameters:

    - Clusters: $k \in \{1, 2, \ldots K\}$
    - Mixture proportions: $\{w_k \in [0,1]\}_{k=1}^K$, where $\sum_k w_k = 1$
    - Cluster means: $\mu = \{\mu_k \in \mathbb{R}^d\}_{k=1}^K$
    - Cluster covariances: $\Sigma = \{\Sigma_k \in \mathbb{R}^{d \times d}, \text{positive definite}\}_{k=1}^K$.

- Hidden variables: cluster labels, $L = \{l_i\}_{i=1}^N$, where $l_i \in \{1, 2, \ldots K\}$

- Model

$$
\begin{aligned}
P(l_i = k) &= w_k \\
P(x_i \mid l_i = k, \mu, \Sigma) &= \text{Normal}(x_i \mid \mu_k, \Sigma_k) = \frac{1}{\sqrt{\det(2\pi\Sigma_k)}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^\top (\Sigma_k)^{-1}(x_i - \mu_k)\right)
\end{aligned}
$$

- Marginal:

$$
P(D \mid \mu, \Sigma, w) = \prod_{i=1}^N \left[\sum_{k=1}^K w_k \, \text{Normal}(x_i \mid \mu_k, \Sigma_k)\right]
$$

- EM algorithm

    1. Choose realistic $w^{\text{old}}, \mu^{\text{old}}, \Sigma^{\text{old}}$ initial values.

    2. In E-step, calculate

$$
r_{i,k} = \frac{w_k^{\text{old}} \, \text{Normal}(x_i \mid \mu_k^{\text{old}}, \Sigma_k^{\text{old}})}{\sum_{k'} w_{k'}^{\text{old}} \, \text{Normal}(x_i \mid \mu_{k'}^{\text{old}}, \Sigma_{k'}^{\text{old}})}
$$

    3. In M-step, calculate

$$
\begin{aligned}
w_k^{\text{new}} &= \frac{1}{N} \sum_{i=1}^N r_{i,k} \\
\mu_k^{\text{new}} &= \frac{1}{w_k^{\text{new}} N} \sum_{i=1}^N r_{i,k} \, x_i \\
\Sigma_k^{\text{new}} &= \frac{1}{w_k^{\text{new}} N} \sum_{i=1}^N r_{i,k}(x_i - \mu_k^{\text{new}})(x_i - \mu_k^{\text{new}})^\top
\end{aligned}
$$

The following python class implements the EM algorithm for fitting GMM.

```python
class GmmEm:
    def __init__(self, x):
        self.x = np.array(x)
        self.N, self.d = self.x.shape
        self.K = None
        self.weights = None
        self.means = None
        self.covs = None

    def initialize(self, K):
        self.K = K
        m0 = np.mean(x, axis=0)
        cov0 = np.cov(x.T)

        self.weights = [1.0/K] * K
        self.means = multivariate_normal.rvs(mean=m0, cov=cov0, size=K)
        cov_values, _ = np.linalg.eig(cov0)
        self.covs = np.array([np.eye(self.d) * 0.1 *cov_values.max()
                              for _ in range(K)])

    def e_step(self):
        r = []
        for k in range(K):
            r.append(self.weights[k] *
                    multivariate_normal.pdf(self.x,
                                            mean=self.means[k],
                                            cov=self.covs[k]))
        r = np.array(r).T
        r_sum = np.einsum('ik->i', r)
        r = np.einsum('ik,i->ik', r, 1.0/r_sum)
        return r

    def m_step(self, r):
        weights_new = 1.0/N * np.einsum('ik->k', r)
        means_new = 1.0/N * \
                    np.einsum('k,ik,id->kd',
                              1.0/weights_new,
                              r,
                              self.x)
        deviations = np.array([self.x - means_new[k] for k in range(self.K)])
        covs_new = 1.0/N * \
                    np.einsum('k,ik,kid,kiD->kdD',
                              1.0/weights_new,
                              r,
                              deviations,
                              deviations)

        self.weights = weights_new
        self.means = means_new
        self.covs = covs_new
```
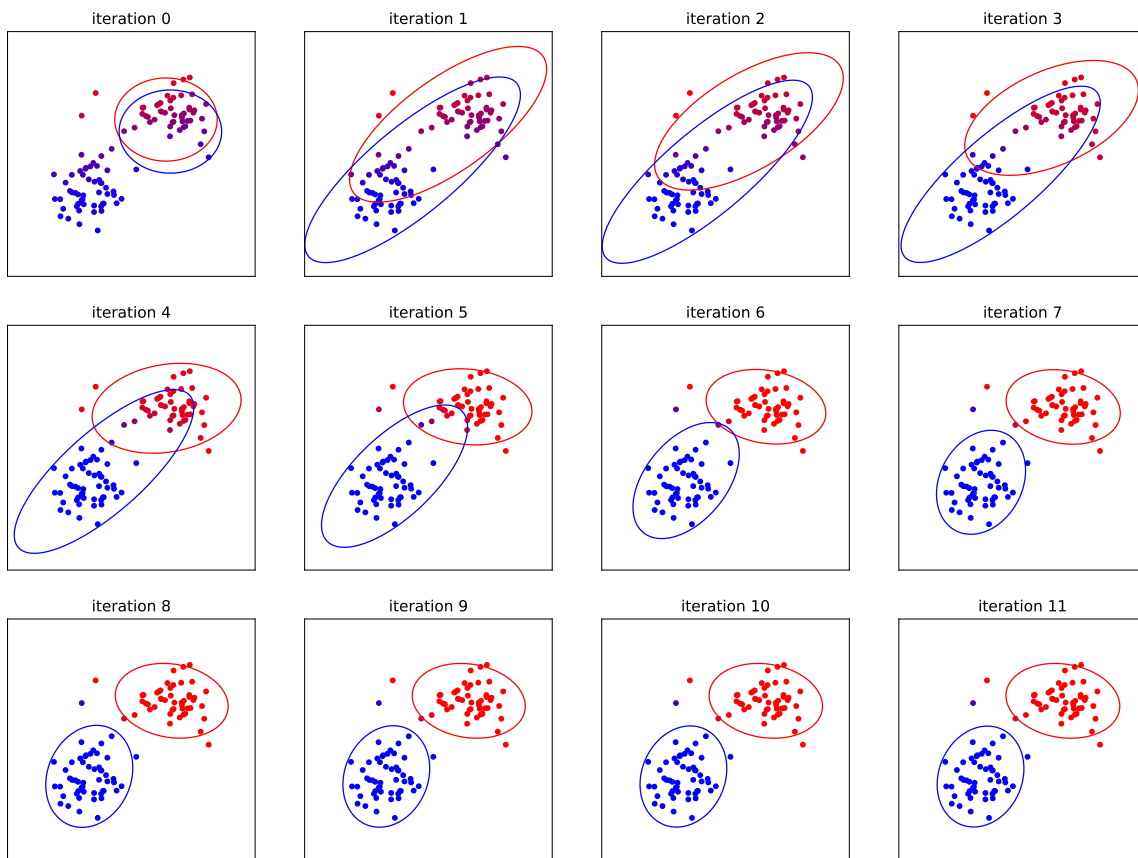
Example: GMM in 2D with $K = 2$
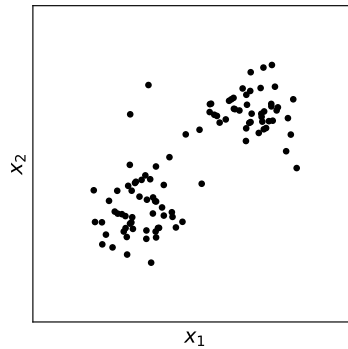
Iterating the E- and M-steps a couple of times, we arrive to the final set of r values.

```
1  gmm = GmmEm(x)
2
3  K = 2
4  gmm.initialize(K)
5  for it in range(12):
6      r = gmm.e_step()
7      gmm.m_step(r)
```

6 Curse of Dimensionality, Laplace approximation

6.1 High-dimensional example

- Data: $D = \{k_t\}_{t=1}^T$, where $k_t \in \mathbb{N}$ is the number of influenza cases at small clinic on each day of the year ($T = 365$).

- Parameters:

  - $\theta = \{\theta_t\}_{t=1}^T$, with $\theta_t = \log(\lambda_t)$ where $\lambda_t > 0$ is the intensity of influenza on a given day t.
  - $\sigma > 0$, the typical change $\theta_t - \theta_{t-1}$.

- Model:

  - Prior: $P(\theta_t \mid \theta_{t-1}) = \text{Normal}(\theta_t \mid \theta_{t-1}, \sigma^2)$, and $P(\sigma) = \text{const.}$
  - Data generation process: $P(k_t \mid \theta_t) = \text{Poisson}(k_t \mid \lambda = \exp(\theta_t))$

- Posterior:

$$P(\theta \mid D) = \frac{1}{Z} P^*(\theta \mid D) = \frac{1}{Z} \prod_{t=1}^T \left[P(\theta_t \mid \theta_{t-1}) \, P(k_t \mid \theta_t) \right]$$

with the understanding that "$P(\theta_1 \mid \theta_0)$" $= 1$. Here Z is the normalization constant.
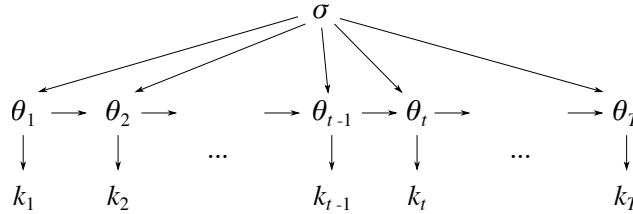
- Numerical solution would require evaluating P^* on a grid of different θ values. Even, at the very extreme, when we consider only 2 values for each θ_t, the number of evaluations becomes

$$2^{365} \approx 10^{109} > 10^{86} \text{ (the number of protons in the observable part of the universe),}$$

which makes it impossible to pursue this strategy.

6.2 Laplace approximation

- Goal: Determine posterior mean and variance of each parameter θ_t

- Challenge: The dimension of $\theta = \{\theta_t\}_{t=1}^T$, i.e. T is too high for direct numerical evaluation.

- Method: Approximate $P^*(\theta \mid D)$ near its maximum with a multi-variate normal distribution.

$$P^*(\theta \mid D) \quad \approx \quad \text{Normal}(\theta \mid \mu, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left[-\frac{1}{2}(\theta - \mu)^\top \Sigma^{-1} (\theta - \mu) \right]$$
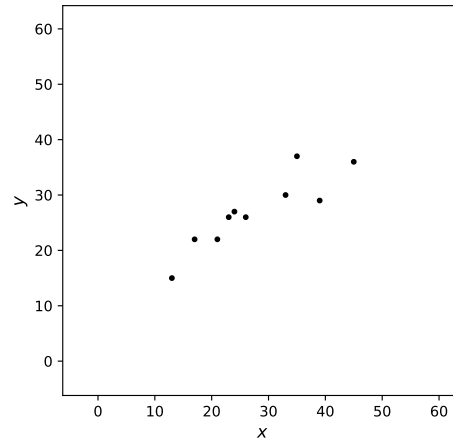
$$\text{where} \qquad \mu = \arg\max_\theta \left[\log P^* \right] \qquad \in \mathbb{R}^T$$

$$\Sigma = \left[-\frac{d}{d\theta}\frac{d}{d\theta} \log P^* \right]_{\theta=\mu}^{-1} \qquad \in \mathbb{R}^{T \times T}$$

where μ can be found with direct numerical or analytical minimization or an expectation maximization algorithm, and Σ can be evaluated analytically or approximated numerically.

6.3 Example: (x, y) linear regression

- Data: $D = \{D_x, D_y\}$,
  - where $D_x = \{x_i\}_{i=1}^N = [21, 24, 17, 39, 23, 45, 33, 26, 13, 35]$,
  - and $D_y = \{y_i\}_{i=1}^N = [22, 27, 22, 29, 26, 36, 30, 26, 15, 37]$

- Parameters: a (slope), b (intercept), σ^2 (strength of y-noise), using flat priors, i.e. $P(a, b, \sigma^2) = $ const.

- Model:

$$P(D_y \mid D_x, a, b, \sigma^2) = \prod_{i=1}^N \text{Normal}(y_i \mid \mu(x_i), \sigma^2), \qquad \text{where } \mu(x_i) = ax_i + b$$

- Unnormalized posterior:

$$\log P^*(a, b, \sigma^2 \mid D) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^N \left[y_i - (ax_i + b)\right]^2$$

- MLE estimate:

$$\left.\begin{array}{l} 0 = \frac{\partial}{\partial a}\log P^* = \frac{1}{\sigma^2}\left[\sum_i y_i x_i - a\sum_i x_i^2 - b\sum_i x_i\right] \\ 0 = \frac{\partial}{\partial b}\log P^* = \frac{1}{\sigma^2}\left[\sum_i y_i - a\sum_i x_i - bN\right] \\ 0 = \frac{\partial}{\partial (\sigma^2)}\log P^* = -\frac{N}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_i \left[y_i - (ax_i + b)\right]^2 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} a_{\text{MLE}} = \left(\overline{yx} - \overline{y}\,\overline{x}\right)/\left(\overline{x^2} - \overline{x}^2\right) \\ b_{\text{MLE}} = \overline{y} - a_{\text{MLE}}\,\overline{x} \\ (\sigma^2)_{\text{MLE}} = \frac{1}{N}\sum_i \left[y_i - (a_{\text{MLE}}\,x_i + b_{\text{MLE}})\right]^2 \end{array}\right.$$

where

$$\overline{x} = \frac{1}{N}\sum_i x_i = 27.6, \qquad \overline{y} = \frac{1}{N}\sum_i y_i = 27.0, \qquad \overline{x^2} = \frac{1}{N}\sum_i x_i^2 = 854.0, \qquad \overline{yx} = \frac{1}{N}\sum_i y_i x_i = 798.9,$$

```python
import numpy as np

def xy_linear_regression_MLE(x, y):
    N = len(x)
    ev_x = np.mean(x)
    ev_y = np.mean(y)
    ev_xx = np.mean(x * x)
    ev_yx = np.mean(y * x)
    ev_yy = np.mean(y * y)
```

```
10
11      a_MLE = (ev_yx − ev_y * ev_x) / (ev_xx − ev_x**2)
12      b_MLE = ev_y − a_MLE * ev_x
13      sigma2_MLE = 1.0/N * np.sum((y − (a_MLE * x + b_MLE))**2)
14
15      return a_MLE, b_MLE, sigma2_MLE
```

giving $\mu = (a_{\text{MLE}}, b_{\text{MLE}}, (\sigma^2)_{\text{MLE}})$, with $a_{\text{MLE}} = 0.5822$, $b_{\text{MLE}} = 10.93$, $(\sigma^2)_{\text{MLE}} = 7.737$.

- Laplace approximation:
  First, we calculate all second order derivatives at the MLE point:

$$\frac{\partial}{\partial a}\frac{\partial}{\partial a}\log P^* = -\frac{N}{\sigma^2}\overline{x^2}$$

$$\frac{\partial}{\partial b}\frac{\partial}{\partial a}\log P^* = \frac{\partial}{\partial a}\frac{\partial}{\partial b}\log P^* = -\frac{N}{\sigma^2}\overline{x}$$

$$\frac{\partial}{\partial b}\frac{\partial}{\partial b}\log P^* = -\frac{N}{\sigma^2}$$

$$\frac{\partial}{\partial(\sigma^2)}\frac{\partial}{\partial a}\log P^* = \frac{\partial}{\partial a}\frac{\partial}{\partial(\sigma^2)}\log P^* = 0$$

$$\frac{\partial}{\partial(\sigma^2)}\frac{\partial}{\partial b}\log P^* = \frac{\partial}{\partial b}\frac{\partial}{\partial(\sigma^2)}\log P^* = 0$$

$$\frac{\partial}{\partial(\sigma^2)}\frac{\partial}{\partial(\sigma^2)}\log P^* = -\frac{N}{2(\sigma^2)^2}$$

from which we construct the second derivative at the MLE point:

$$-\nabla\nabla\log P^*|_{\text{MLE}} = \frac{N}{(\sigma^2)_{\text{MLE}}}\left[\begin{array}{ccc} \overline{x^2} & \overline{x} & 0 \\ \overline{x} & 1 & 0 \\ 0 & 0 & \frac{1}{2(\sigma^2)_{\text{MLE}}} \end{array}\right]$$

```
1  minus_dd_logPstar = N / sigma2_MLE * \
2  np.array([
3      [ev_xx, ev_x, 0],
4      [ev_x, 1, 0],
5      [0, 0, 1/(2*sigma2_MLE)]
6  ])
7  Sigma = − np.linalg.inv(minus_dd_logPstar)
```
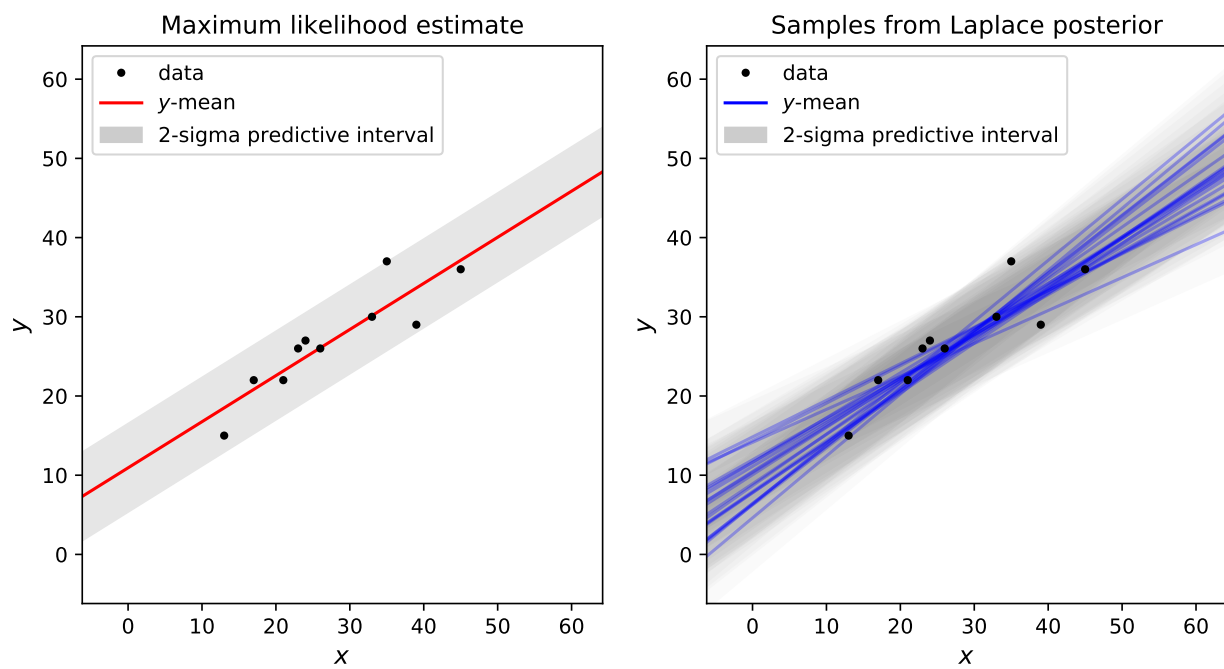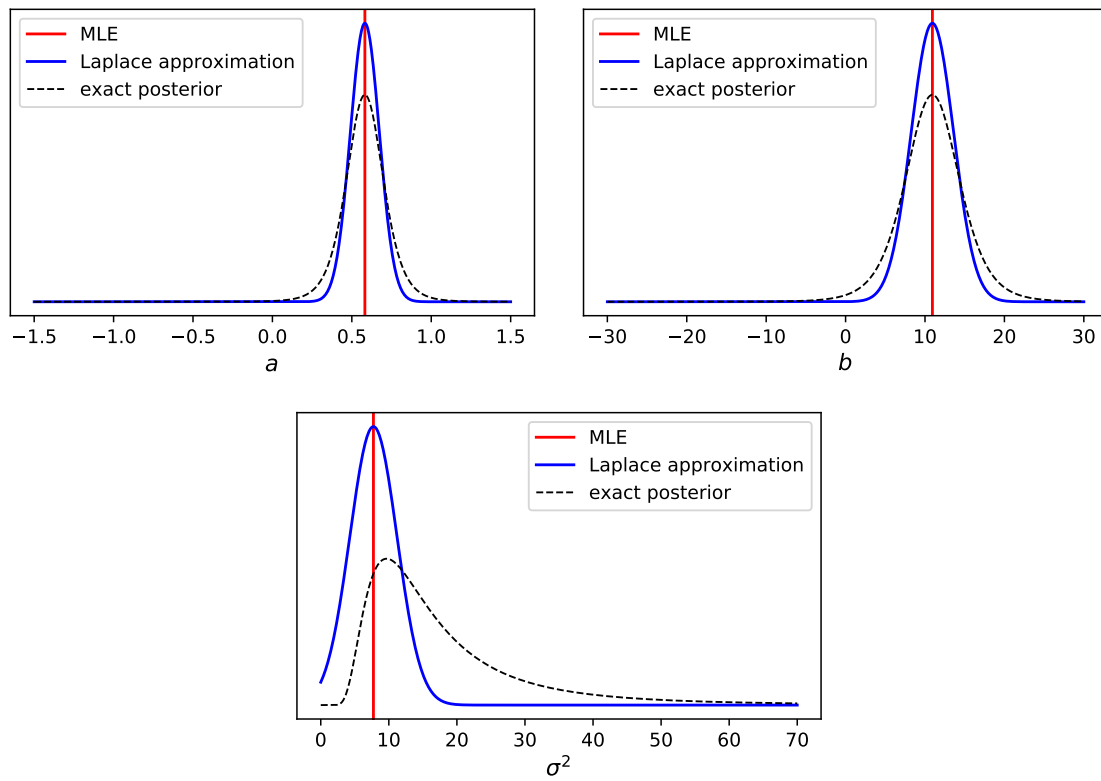
giving

$$\Sigma = \left[-\nabla\nabla\log P^*|_{\text{MLE}}\right]^{-1} = \left[\begin{array}{ccc} 0.0839 & -0.2315 & 0.0 \\ -0.2315 & 7.1634 & 0.0 \\ 0.0 & 0.0 & 11.197 \end{array}\right]$$

and

$$\begin{array}{rcl} \text{Var}(a \mid D) & \approx & \Sigma_{1,1} = 0.0839, \\ \text{Var}(b \mid D) & \approx & \Sigma_{2,2} = 7.1634, \\ \text{Var}(\sigma^2 \mid D) & \approx & \Sigma_{3,3} = 11.197 \end{array}$$

7 Monte Carlo methods

7.1 Monte Carlo simulation

- **Goal:** Analyze complicated distributions by drawing samples from them: $P(\theta) \mapsto \{\theta^{(s)}\}$.

- **Challenge:** From the vast majority of distributions, we don't know how to draw samples efficiently.

- **Method:**

    1. Draw samples from an easy-to-sample distribution.
    2. Transform the drawn values so they become samples from the distribution of question.
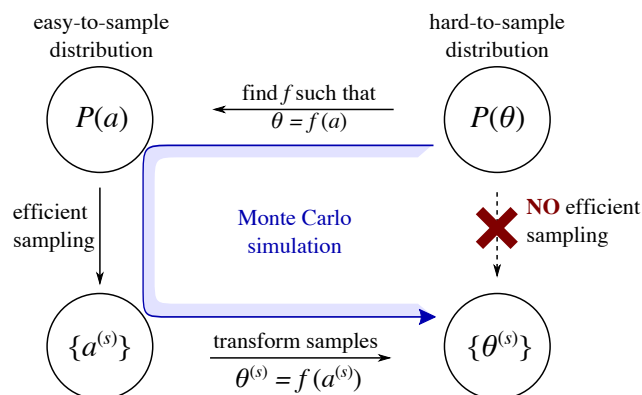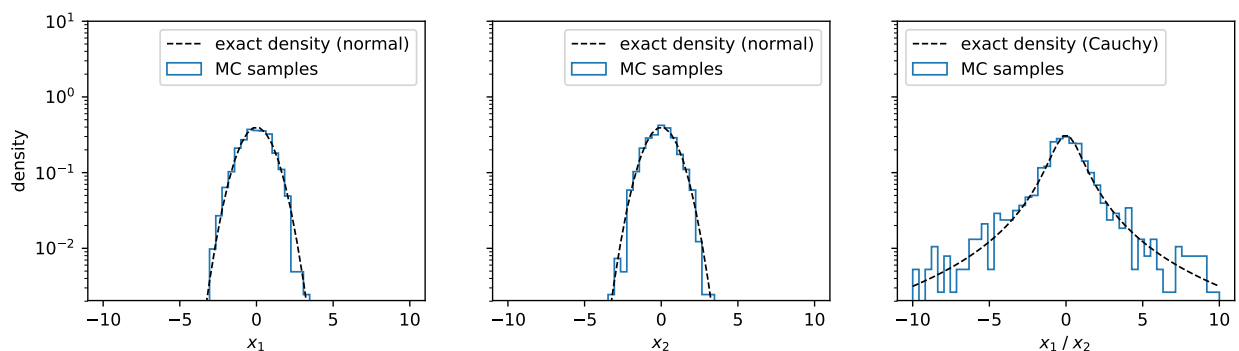

Example: Ratio of two normal variables

- Input: $P(x_1) = \text{Normal}(x_1 \mid 0, 1)$, $P(x_2) = \text{Normal}(x_2 \mid 0, 1)$

- Output: $y := x_1/x_2$, $P(y) = ?$

- MC method:

```
1 from scipy.stats import norm
2
3 samples = 1000
4 X1 = norm.rvs(loc=0, scale=1, size=samples)
5 X2 = norm.rvs(loc=0, scale=1, size=samples)
6 Y = X1 / X2
```

Example: Entropy of distributions from flat Dirichlet

- Input: $p = (p_1, p_2, p_3) \in [0,1]^{\times 3}$, such that $\sum_{k=1}^{3} p_k = 1$, where $P(p) = \text{Dirichlet}(p \mid \alpha = (1,1,1))$

- Output: $h := H(p) = -\sum_k p_k \log_2 p_k, \ \ P(h) = ?$

- MC method:

```python
import numpy as np
from scipy.stats import dirichlet

def entropy(p):
    h = 0
    for pk in p:
        if pk > 0:
            h += - pk * np.log2(pk)
    return h

alpha = (1,1,1)
sample_size = 10_000
p_samples = dirichlet.rvs(alpha, size=sample_size)
h_samples = []
for p in p_samples:
    h_samples.append(entropy(p))
```


Entropy of 3-categorical distributions from flat Dirichlet
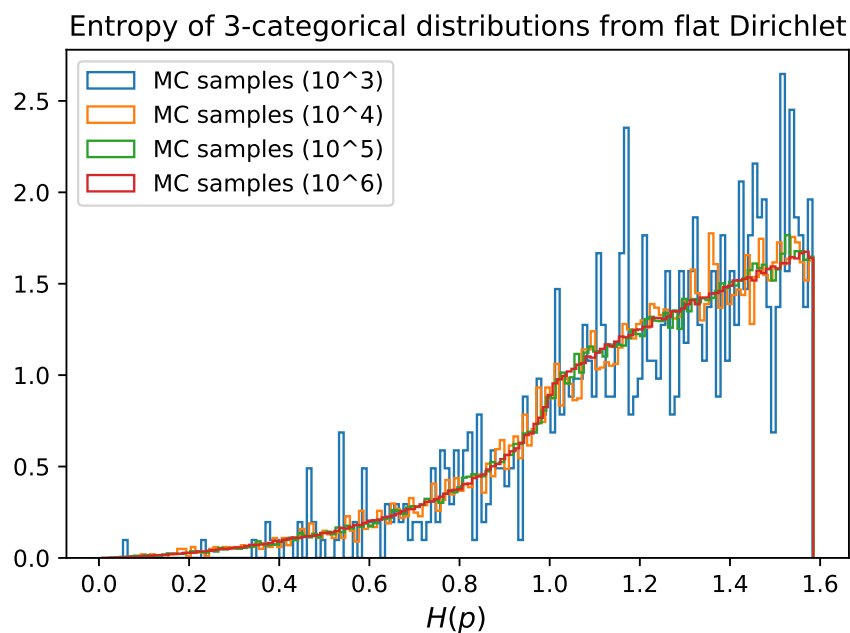
Example: Monty Hall problem

- Input:

  - In a game show, there a three doors $D = \{1, 2, 3\}$
  - A reward is placed behind door $r \in D$ uniformly randomly, i.e. $P(r) =$ uniform.
  - The player picks a door $p_1 \in D$ uniformly randomly, i.e. $P(p_1) =$ uniform (He doesn't know where the reward is.)
  - The game show master (who knows where the reward is), pick a door from the remaining two that does not have the reward, and opens it, $o \in D_{\text{can open}}$, where $D_{\text{can open}} = D \setminus (\{r\} \cup \{p_1\})$ randomly, i.e. $P(o) =$ uniform.
  - The game show master offers the player another chance to pick one of the remaining two doors $D_{\text{remaining}} = D \setminus \{o\}$. He can stick to his first choice, i.e. $p_2 = p_1$, or switch to the other door, i.e. $p_2 \in D_{\text{remaining}} \setminus \{p_1\}$
  - The game show master opens door p_2, and the player wins if $p_2 = r$, and loses otherwise.

- Output: What's the probability of winning with the two strategies, i.e.

$$P(\text{win} \mid \text{switch}) = P(r = p_2 \mid p_2 \neq p_1) = ?$$
$$P(\text{win} \mid \text{don't switch}) = P(r = p_2 \mid p_2 = p_1) = ?$$

- MC method:

```
from numpy.random import choice

doors = {1, 2, 3}

games = 1000
wins_with_switch = 0
wins_with_no_switch = 0
for g in range(games):
    reward = choice(list(doors))
    pick_1 = choice(list(doors))

    can_open = doors - set([pick_1]).union(set([reward]))
    openned = choice(list(can_open))
    remaining = doors - set([openned])

    pick_2 = list(remaining - set([pick_1]))[0]
    if pick_2 == reward:
        wins_with_switch += 1

    pick_2 = pick_1
    if pick_2 == reward:
        wins_with_no_switch += 1

P_win_with_switch = wins_with_switch / float(games)
P_win_with_no_switch = wins_with_no_switch / float(games)
```

Yielding something similar to $P(\text{win} \mid \text{switch}) \approx 0.653$, and $P(\text{win} \mid \text{don't switch}) \approx 0.347$.

7.2   Markov Chain Monte Carlo method

- **Goal:** Draw samples from an unnormalized posterior $P^*(\theta) \mapsto \{\theta^{(s)}\}$

- **Challenge:** We don't know how to do this directly.

- **Method:**

    1. Initialize $\theta^{(0)}$.
    2. Obtain a new $\theta^{(s+1)}$ value using the current value $\theta^{(s)}$ and the $P^*()$ function.
    3. Add the new value to the list of samples $\{\theta^{(s)}\}$. Return to step 2 with $s \leftarrow s + 1$.
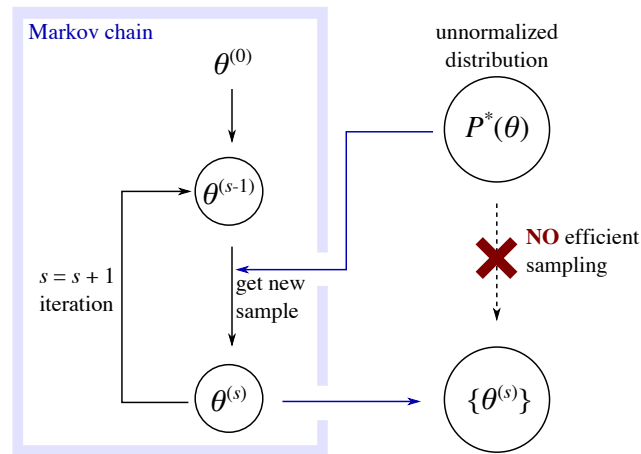
Various Markov chain-based methods exist: Metropolis-Hastings sampling, Gibbs sampling, Hamiltonian sampling.

7.3   Metropolis Hastings sampling

1. Start with $\theta^{(0)}$.

2. Propose a new value: $\theta^{\mathrm{new}} = \theta^{(s)} + \varepsilon$, where ε is drawn from $P(\varepsilon) = \mathrm{Normal}(\varepsilon \mid 0, s^2)$, where s is a fixed "step size".

3. Evaluate $\Delta L = \log P^*(\theta^{\mathrm{new}}) - \log P^*(\theta^{(s)})$, and depending on its value, we obtain $\theta^{(s+1)}$:

    (a) If $\Delta L >= 0$, then
    $$\theta^{(s+1)} = \theta^{\mathrm{new}},$$

    (b) if $\Delta L < 0$, then
    $$\theta^{(s+1)} = \left\{ \begin{array}{ll} \theta^{\mathrm{new}} & \text{with probability } \exp(\Delta L) \\ \theta^{(s)} & \text{with probability } 1 - \exp(\Delta L) \end{array} \right.$$

Example: Bimodal distribution

- Unnormalized distribution: $\log P^*(x) = x/2 - \left(1 - x^2\right)^2$, where $x \in \mathbb{R}$.

- 1-dimensional Metropolis-Hastings sampler:

```
1  def propose_MH(x, stepsize):
2      epsilon = norm.rvs(loc=0, scale=stepsize)
3      return x + epsilon
4
5  def new_sample_MH(x_current, x_proposed, log_Pstar):
6      delta_L = log_Pstar(x_proposed) - log_Pstar(x_current)
7      if delta_L >= 0:
8          return x_proposed
9      if np.random.random() < np.exp(delta_L):
10         return x_proposed
11     else:
12         return x_current
```

- Applying it to the $\log P^*$ in question:

```
1  def log_Pstar_camel(x):
2      return 0.5 * x -(1 - x**2)**2
3
4  x0 = 0
5  stepsize = 0.1
6  iterations = 20000
7  x_samples = []
8
9  x_curr = x0
10 for it in range(iterations):
11     x_proposed = propose_MH(x_curr, stepsize)
12     x = new_sample_MH(x_curr, x_proposed, log_Pstar_camel)
13     x_samples.append(x)
14     x_curr = x
```


unnormalized distribution

samples from Markov chain

histogram of samples
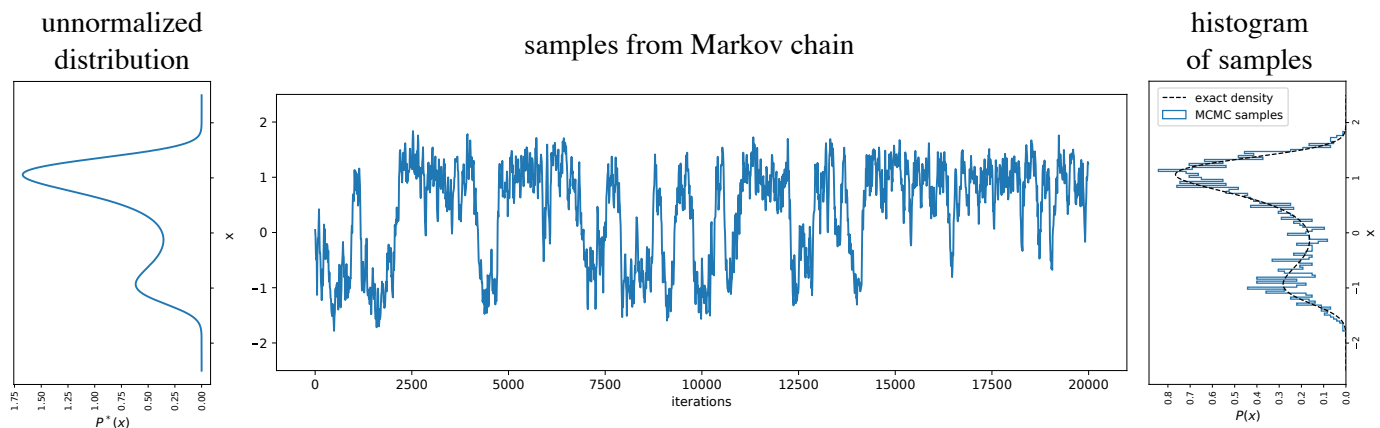
8   Gibbs sampling

Requirement: $P^*(\theta)$ is difficult to sample, but after partitioning θ into two (or more) sets of parameters, $\theta_1, \theta_2, \ldots$, their conditionals, i.e. $P(\theta_1 \mid \theta_2, \ldots)$ and $P(\theta_2 \mid \theta_1, \ldots)$, are easy to sample.

8.1   Gibbs sampling algorithm

1. Derive the conditional distributions $P(\theta_1 \mid \theta_2)$, $P(\theta_2 \mid \theta_1)$

2. Initialize $\theta_1^{(0)}$

3. Draw $\theta_2^{(s+1)}$ from $P(\theta_2 \mid \theta_1^{(s)})$

4. Draw $\theta_1^{(s+1)}$ from $P(\theta_1 \mid \theta_2^{(s+1)})$

5. Add $\theta^{(s+1)} = (\theta_1^{(s+1)}, \theta_2^{(s+1)})$ to the collection of samples $\{\theta^{(s)}\}$. Return to step 3.

8.2   Example: Triangle distribution in 2D

- Unnormalized distribution $P^*(\theta_1, \theta_2) = \max(0,\ 1 - |\theta_1| - |\theta_2|)$

- 2-dimensional Gibbs sampler: The conditionals are

$$
\begin{aligned}
P(\theta_1 \mid \theta_2) &= \text{Triangle}(\theta_1 \mid \text{loc} = 0, \text{scale} = 1 - |\theta_2|) \\
P(\theta_2 \mid \theta_1) &= \text{Triangle}(\theta_2 \mid \text{loc} = 0, \text{scale} = 1 - |\theta_1|),
\end{aligned}
$$

where $\text{Triangle}(x \mid \text{loc}, \text{scale}) = \frac{1}{2}\max(0,\ \text{scale - |x - loc|})$ is the symmetric 1-d triangle distribution.

```
1  from scipy.stats import triang
2  def triangle(loc=0, scale=1):
3      return triang(c=0.5, loc=-scale + loc, scale=2*scale)
4
5  iters = 10_000
6  theta1_samples = []
7  theta2_samples = []
8  theta_1 = 0
9  for it in range(iters):
10     theta_2 = triangle(scale=1-abs(theta_1)).rvs()
11     theta_1 = triangle(scale=1-abs(theta_2)).rvs()
12     theta1_samples.append(theta_1)
13     theta2_samples.append(theta_2)
```
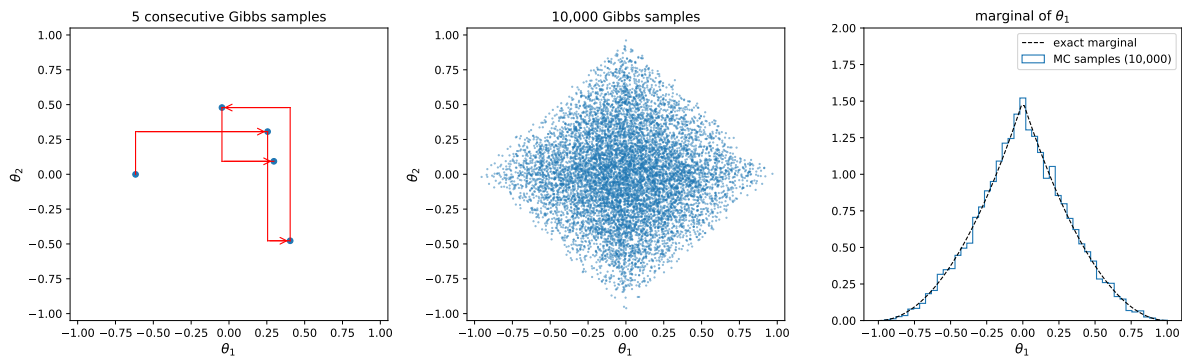
8.3   Example: Binomial clustering

- Data: $D = \{k_i\}_{i=1}^N$, where $k_i(\text{successes}) \in \{0, 1, 2, \dots n\}$, out of n trials.

- Parameters:

    - Cluster weights: $w = \{w_c\}_{c=1}^C$, where $w_c \in [0, 1]$ such that $\sum_c w_c = 1$, with flat prior, i.e. $P(w) = \text{const.}$
    - Probability of success in each cluster: $p = \{p_c\}_{c=1}^C$, where $p_c \in [0, 1]$, with a weak prior $P(p_c) = \text{Beta}(p_c \mid \alpha_c^{(0)}, \beta_c^{(0)})$, where $\alpha_c^{(0)}, \beta_c^{(0)}$ are small positive numbers.
    - Hidden labels: $L = \{l_i\}_{i=1}^N$, where $l_i \in \{1, 2, \dots C\}$.

- Model:

    - Level 1: $P(l_i = c \mid w) = w_c$
    - Level 2: $P(k_i \mid l_i = c, \ p) = \text{Binomial}(k_i \mid n, p_c)$
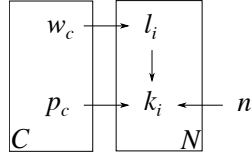
- Unnormalized posterior:

$$P^*(L, w, p \mid D) \propto \left[\prod_{i=1}^N P(k_i \mid p_{l_i}) \ P(l_i \mid w) \right] P(w)P(p) \quad \propto \quad \prod_{i=1}^N \text{Binomial}(k_i \mid n, p_{l_i}) \ w_{l_i}$$

- Partitioning:

$$\theta = (w, p, L) \qquad \rightarrow \qquad \theta_1 = (w, p), \quad \theta_2 = L$$

- $P(\theta_1 \mid \theta_2, D)$ conditionals:

$$P(w, p \mid L, D) \ = \ P(w \mid L) \ P(p \mid L, D)$$

where

$$P(w \mid L) \ \propto \ P(L \mid w)P(w) \propto \prod_{i=1}^N w_{l_i} = \prod_{c=1}^C (w_c)^{N_c}$$

$$= \ \text{Dirichlet}\left(w \ \Big| \ \alpha_c = 1 + N_c \right)$$

$$P(p \mid L, D) \ \propto \ P(D \mid L, \mu, \sigma)P(p) \propto \left[\prod_{i=1}^N \text{Binomial}(k_i \mid n, p_{l_i}) \right] \times \left[\prod_{c=1}^C \text{Beta}(p_c \mid \alpha_c^{(0)}, \beta_c^{(0)}) \right]$$

$$= \ \prod_{c=1}^C \text{Beta}(p_c \mid \alpha = \alpha_c^{(0)} + K_c, \ \beta = \beta_c^{(0)} + nN_c - K_c),$$

where $N_c = \sum_i \delta_{l_i, c}, \quad K_c = \sum_i \delta_{l_i, c} k_i$, where $\delta_{l_i, c} = 1$ if $l_i = c$ and 0 otherwise.

- $P(\theta_2 \mid \theta_1, D)$ conditionals:

$$P(L \mid w, p, D) \ \propto \ P(D \mid L, p) \ P(L \mid w) = \prod_{i=1}^N \text{Binomial}(k_i \mid p_{l_i}) \ w_{l_i}$$

$$= \ \prod_{i=1}^N \text{Categorical}\left(l_i \ \Big| \ f_c = \frac{w_c \text{Binomial}(k_i \mid n, p_c)}{\sum_{c'} w_{c'} \text{Binomial}(k_i \mid n, p_{c'})} \right)$$

- Gibbs sampling:

```
 1  import numpy as np
 2  from numpy.random import choice
 3  from scipy.stats import binom, beta, dirichlet
 4
 5  def sample_labels(k_data, n, w, p):
 6      labels = []
 7      for ki in k_data:
 8          f = w * binom.pmf(ki, n, p)
 9          f /= np.sum(f)
10          labels.append(choice(clusters, p=f))
11      return np.array(labels)
12
13  def sample_w(labels, clusters):
14      alpha = []
15      for c in clusters:
16          alpha.append(1 + np.sum(labels == c))
17      return dirichlet.rvs(alpha)[0]
18
19  def sample_p(labels, k_data, n, clusters, alpha0, beta0):
20      N = len(k_data)
21      p = []
22      for c in clusters:
23          Kc = np.sum(k_data[labels == c])
24          Nc = np.sum(labels == c)
25          a = alpha0[c] + Kc
26          b = beta0[c] + n*Nc - Kc
27          p.append(beta.rvs(a, b))
28      return np.array(p)
29
30  clusters = list(range(3))
31  alpha0 = [0, 0.5, 1]
32  beta0 = [1, 0.5, 0]
33
34  w = np.array([1./3] * 3 )
35  p = np.array([0.01, 0.5, 0.6])
36
37  labels_samples = []
38  w_samples = []
39  p_samples = []
40  iters = 100
41  for it in range(iters):
42      labels = sample_labels(k_data, n, w, p)
43      w = sample_w(labels, clusters)
44      p = sample_p(labels, k_data, n, clusters, alpha0, beta0)
45      labels_samples.append(labels)
46      w_samples.append(w)
47      p_samples.append(p)
```

Fitted binomial clusters
(using posterior means)
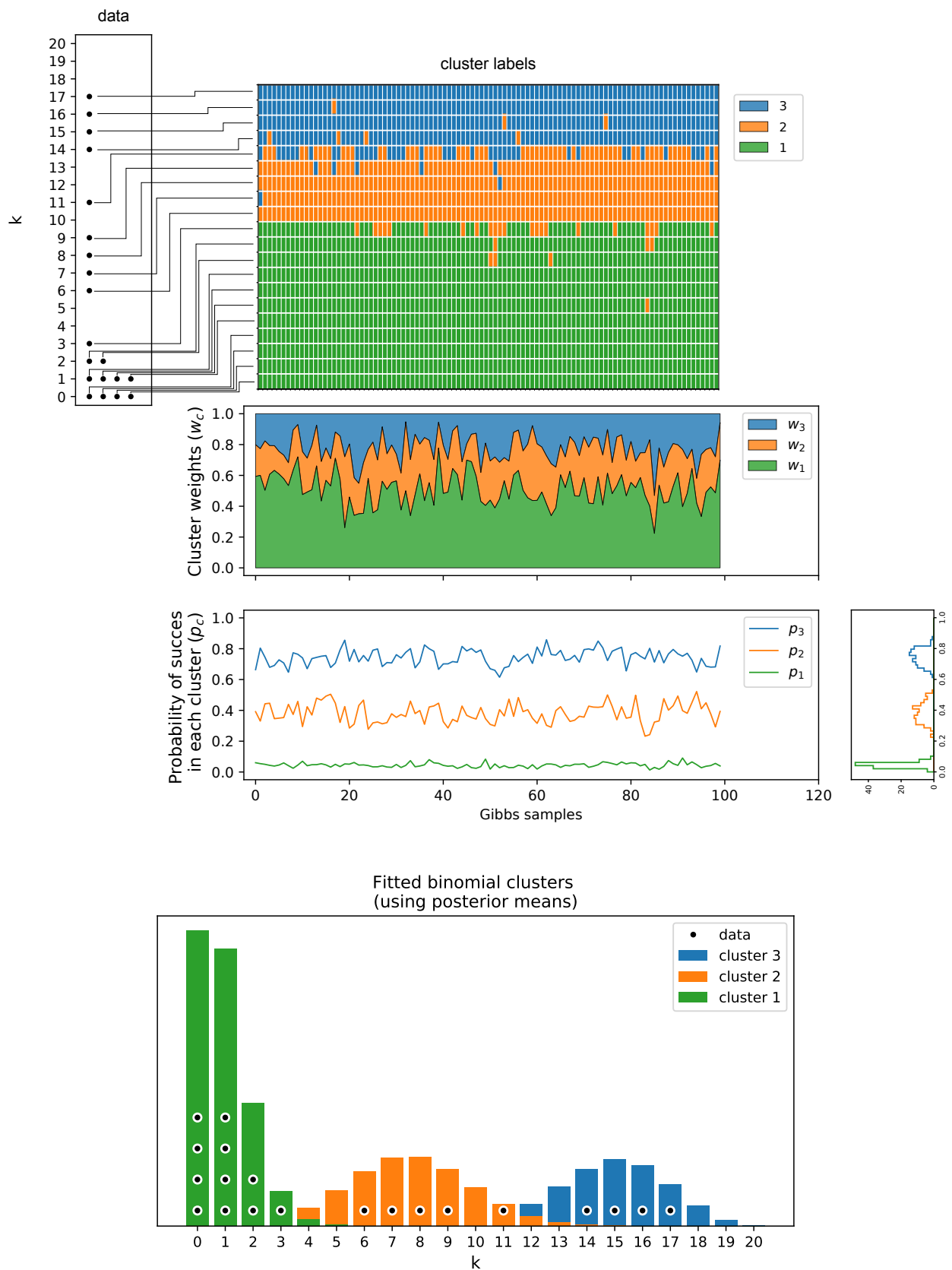
9   Viterbi algorithm, Belief propagation

9.1   Hidden Markov Model

- Data: $D = \{x_t\}_{t=1}^T$

- Parameters

  - Hidden states: $Z = \{z_t\}_{t=1}^T$, where $z_t \in S = \{1, 2, \ldots s_{\max}\}$
  - Prior probability: $w = \{w_s\}_{s \in S}$, where $w_s \in [0, 1]$, such that $\sum_s w_s = 1$.
  - Transition probabilities: $\{A_{s,r}\}_{s,r \in S}$, where $A_{s,r} \in [0, 1]$, such that $\sum_r A_{s,r} = 1$, for all s.
  - Emission probabilities: $\{B_s(x)\}_{s \in S}$, where $B_s(x)$ is a probability density of x, i.e. $x \mapsto \mathbb{R}$, and $\sum_x B_s(x) = 1$, for all s.

- Model:

  - level 1: $P(x_t \mid z_t, B) = B_{z_t}(x_t)$
  - level 2: $P(Z \mid w, A) = P(z_1 \mid w) \prod_{t=2}^T P(z_t \mid z_{t-1}, A) = w_{z_1} \prod_{t=2}^T A_{z_{t-1}, z_t}$
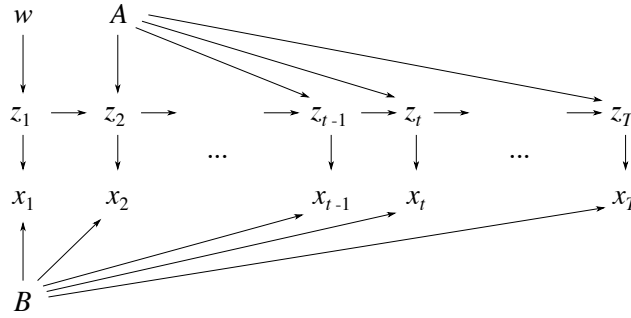
- Joint

$$P(D, Z \mid w, A, B) = P(D \mid Z, B)P(Z \mid w, A) = \left[\prod_{t=1}^T B_{z_t}(x_t) \right] \times \left[w_{z_1} \prod_{t=2}^T A_{z_{t-1}, z_t} \right]$$

- Most likely sequence of states:

$$Z_{\mathrm{MLE}} = \arg\max_Z P(D, Z \mid w, A, B) = \text{(see "Viterbi algorithm" below)}$$

- Marginal likelihood, marginals of z_t and joint marginal of (z_{t-1}, z_t):

$$\left.\begin{array}{l} P(D \mid w, A, B) = \sum_Z P(D, Z \mid w, A, B) \\ P(z_t \mid D, w, A, B) = \sum_{Z \setminus \{z_t\}} P(D, Z \mid w, A, B) \\ P(z_{t-1}, z_t \mid D, w, A, B) = \sum_{Z \setminus \{z_{t-1}, z_t\}} P(D, Z \mid w, A, B) \end{array}\right\} = \text{(see "Belief propagation" below)}$$

- Maximum-likelihood estimate of level 2 parameters

$$(w, A, B)_{\mathrm{MLE}} = \arg\max_{w, A, B} P(D \mid w, A, B) = \text{(see "Baum-Welch" algorithm below)}$$

9.2   Viterbi algorithm

Inputs:

- $f_1(s) := P(x_1, z_1 = s \mid w, B) = w_s B_s(x_1)$, for $s \in S$

- $f_t(r, s) := P(x_t, z_t = s \mid z_{t-1} = r, A, B) = A_{r,s} B_s(x_t)$, for $t = 2, 3, \ldots T$, and $r, s \in S$

- $F(Z) := f_1(z_1) \prod_{t=2}^{T} f_t(z_{t-1}, z_t)$

Outputs:

- $F_{\max} := \max_Z P(D, Z \mid w, A, B) = \max_Z F(Z)$

- $(z_1^*, z_2^*, \ldots z_T^*) := Z_{\mathrm{MLE}} = \arg\max_Z F(Z)$

Algorithm:

1. **"Discover"**
   For $s \in S$:

   $$\phi_1(s) \quad := \quad f_1(s)$$

   For $t \in [2, 3, \ldots T]$:
       For $s \in S$:

   $$\phi_t(s) \quad := \quad \max_{r \in S} \Big(\phi_{t-1}(r) \, f_t(r, s) \Big)$$
   $$a_t(s) \quad := \quad \arg\max_{r \in S} \Big(\phi_{t-1}(r) \, f_t(r, s) \Big)$$

2. **"Retrace"**

   $$F_{\max} \quad = \quad \max_{s \in S} \Big(\phi_T(s) \Big)$$
   $$z_T^* \quad = \quad \arg\max_{s \in S} \Big(\phi_T(s) \Big)$$

   For $t \in [T-1, T-2, \ldots, 2, 1]$:

   $$z_t^* \quad = \quad a_{t+1}(z_{t+1}^*)$$

9.3 Belief propagation on chain

Inputs:

- $f_1(s) := P(x_1, z_1 = s \mid w, B) = w_s B_s(x_1)$, for $s \in S$

- $f_t(r, s) := P(x_t, z_t = s \mid z_{t-1} = r, A, B) = A_{r,s} B_s(x_t)$, for $t = 2, 3, \ldots T$, and $r, s \in S$

- $F(Z) := f_1(z_1) \prod_{t=2}^{T} f_t(z_{t-1}, z_t)$

Outputs:

- $N := \sum_Z F(Z)$

- $M_t(s) := \sum_{Z \setminus \{z_t\}} F(z_1, z_2, \ldots z_{t-1}, \; s, \; z_{t+1}, \ldots z_T),$ $t \in \{1, 2, \ldots T\}, \; s \in S,$

- $Q_t(r, s) := \sum_{Z \setminus \{z_{t-1}, z_t\}} F(z_1, z_2, \ldots z_{t-2}, \; r, \; s, \; z_{t+1}, \ldots z_T),$ $t \in \;\; \{2, \ldots T\}, \; s \in S,$

Algorithm:

1. **"Forward"**
   For $s \in S$:

$$L_1(s) \;\; := \;\; f_1(s)$$

   For $t \in [2, 3, \ldots T]$:
        For $s \in S$:

$$L_t(s) \;\; := \;\; \sum_{r \in S} L_{t-1}(r) \, f_t(r, s)$$

2. **"Backward"**
   For $r \in S$:

$$R_T(r) \;\; := \;\; 1$$

   For $t \in [T-1, T-2, \ldots, 2, 1]$:
        For $s \in S$:

$$R_t(r) \;\; := \;\; \sum_{s \in S} f_{t+1}(r, s) \, R_{t+1}(s)$$

3. **"Combine"**

$$N \;\; = \;\; \sum_{s \in S} L_T(s)$$

   For $t \in [1, 2, \ldots T]$:
        For $s \in S$:

$$M_t(s) \;\; = \;\; L_t(s) \, R_t(s)$$

   For $t \in [2, \ldots T]$:
        For $(s, r) \in S \times S$:

$$Q_t(r, s) \;\; = \;\; L_{t-1}(r) \, f_t(r, s) \, R_t(s)$$

9.4   Baum-Welch algorithm

Inputs:

- Data: $\{x_t\}_{t=1}^T$

- Set of possible hidden states: S

- Parametrization of the emission probabilities: $\{B_s(x)\}_{s\in S}$. We consider two cases:

    1. $B_s(x) = B_{s,x}$ matrix for finite number of possible $x \in \{1, 2, \ldots K\}$ values.
    2. $B_s(x) = P(x \mid s, \beta_s)$, where β_s is the set of parameters for emission distribution of state s.

- Initial values: $w^{(0)}, A^{(0)}$, and $B = \{\{B_{s,x}^{(0)}\}_{x=1}^K\}_{s\in S}$ (for case 1) or $B = \{\beta_s^{(0)}\}_{s\in S}$ (for case 2).

Outputs:

- (w^*, A^*, B^*) that (locally) maximize $\sum_Z P(D, Z \mid w, A, B)$

Algorithm

1. Initialize, $(w, A, B) := (w^{(0)}, A^{(0)}, B^{(0)})$

2. E-step: Run "Belief propagation" with inputs

    - $f_1(s) := w_s B_s(x_1)$
    - $f_t(r, s) := A_{r,s} B_s(x_t)$

   which returns

    - $N = P(D \mid w, A, B)$
    - $M_t(s) = P(D, z_t = s \mid w, A, B)$
    - $Q_t(r, s) = P(D, z_{t-1} = r, z_t = s \mid w, A, B)$

3. M-step: Update parameters

$$
\begin{aligned}
w_s^{\text{new}} &= \frac{M_1(s)}{N} \\[2mm]
A_{r,s}^{\text{new}} &= \frac{\sum_{t=2}^T Q_t(r, s)}{\sum_{t=2}^T M_{t-1}(r)} \\[2mm]
\text{case 1}: B_{s,x}^{\text{new}} &= \frac{\sum_{t=1}^T \delta_{x,x_t} M_t(s)}{\sum_{t=1}^T M_t(s)} \\[2mm]
\text{case 2}: \beta_s^{\text{new}} &= \underset{\beta_s}{\arg\max} \sum_{t=1}^T M_t(s) \log P(x_t \mid s, \beta_s) = \text{MLE of } \beta_s \text{ with data weights } \{M_t(s)\}_{t=1}^T
\end{aligned}
$$

4. Check for convergence, and return to step 2 if needed.