2019

Foundations of Data Science

CAMBRIDGE COMPUTER SCIENCE TRIPOS PART IB, PAPER 6

ASHWIN AHUJA

Table of Contents

Probabilistic Modelling

Likelihood

Likelihood is the probability of the observed outcome, viewed as a function of the unknown parameter. Important to note that likelihood is not a probability density – does not integrate to 1. **The likelihood function** measures how much evidence there is for a particular parameter value. **The maximum likelihood estimator** is the parameter value with the maximum likelihood.

Example: take a biased coin. Likelihood function is how much evidence there is for a particular value of p. The higher lik(p) is, the more likely that value of p is.

$$lik(p \mid x) = \binom{n}{x} p^x (1-p)^{n-x}$$

In order to get the value which maximise the likelihood, we solve:

$$\frac{d}{dp} lik(p) = \binom{n}{x} (xp^{x-1}(1-p)^{n-x} - (n-x)p^x(1-p)^{n-x-1}) = 0$$

Often, easier to maximise log(lik(.)) rather than lik(.) and it necessarily has the same solution:

$$\log lik(p) = k + x log\ p + (n-x)\log(1-p) \ \ where\ k\ is\ a\ constant$$

$$\frac{d}{dp}\log lik(p) = \frac{x}{p} - \frac{n-x}{1-p} = 0 => p = \frac{x}{n}$$

Random Variables

Random Variable is a function that can give different answers. It takes values in Ω means that the return value of the function is an element of the set Ω. Every random variable X has a probability distribution:

$$\mathbb{P}(x \in A)$$

which specifies the probability that the return value lies in a subset A ⊆Ω – if Ω is countable, the random variable is said to be discrete and then:

$$\mathbb{P}(x \in A) = \sum_{x \in A} \mathbb{P}(X = x)$$

For many applications, work with real-valued random variables, and define the cumulative distribution function, to be:

$$F(x) = \mathbb{P}(X \leq x)$$

If the function is differentiable, then X is said to be a continuous random variable with density function *f(x) = F'(x)* and:

$$\mathbb{P}(X \leq x) = \int_{-\infty}^{x} f(x)\, dx$$

For a continuous random variable, ℙ(X = x) = 0 for every x, and so:

$$\mathbb{P}(a \leq X \leq b) = \mathbb{P}(a < X \leq b) = \mathbb{P}(a \leq X < b) = \mathbb{P}(a < X < b)$$

$$\Pr_X(x) = \begin{cases} \mathbb{P}(X = x) & \text{when } X \text{ is a discrete random variable} \\ & (\Pr_X \text{ is called the } \textit{probability mass function}) \\ f(x) & \text{when } X \text{ is a continuous random variable with density } f \\ & (\Pr_X \text{ is called the } \textit{probability density function}) \end{cases}$$

A conditional random variable (X | C) is just a random variable whose distribution is conditional: if Y = (X | C) then ℙ(Y ∈ A) = ℙ(X ∈ A | C)

$$\Pr_X(x \mid C) = \begin{cases} \mathbb{P}(X = x \mid C) & \text{when } X \text{ is discrete} \\ G'(x) \text{ where } G(x) = \mathbb{P}(X \leq x \mid C) & \text{when } X \text{ is continuous.} \end{cases}$$

Independence and joint distributions

Any pair of random variables (X, Y) has a joint distribution which specifies the probability of any joint event C. For a pair of continuous random variables, the joint distribution can be specified by a joint probability density $Pr_{X,Y}(x, y)$

$$\mathbb{P}((x,y) \in C) = \int_{(x,y) \in C} Pr_{X,Y}(x,y)\, dx\, dy$$

The marginal density of X is:

$$Pr_X(x) = \int_y Pr_{X,Y}(x,y)\, dy$$

When random variables are discrete, you can just replace integrals by sums.

Independence: Random variables are independent iff:

$$\mathbb{P}(X \in A, Y \in B) = \mathbb{P}(X \in A)\ \mathbb{P}(Y \in B)\ for\ all\ A, B$$
$$OR\ Pr_{X,Y}(x,y) = Pr_X(x)\ Pr_Y(y)\ for\ all\ x, y$$
$$OR\ \mathbb{P}(X \in A \mid Y \in B) = \mathbb{P}(X \in A)\ for\ all\ A\ and\ B\ with\ \mathbb{P}(Y \in B) > 0$$

Rules for Conditional Probability

A and B are events:

1. $\mathbb{P}(\Omega) = 1$ where Ω is the entire sample space
2. Conditional probability: $\mathbb{P}(A \mid B) = \mathbb{P}(A \cap B) / \mathbb{P}(B)$, when $\mathbb{P}(B) > 0$
3. Sum Rule: If {B1, B2, …} partition Ω then $\mathbb{P}(A) = \sum_i \mathbb{P}(A \cap B_i)$
    a. Law of total probability: $\mathbb{P}(A) = \sum_i \mathbb{P}(Bi)\mathbb{P}(A \mid B_i)$
4. A and B are independent iff $\mathbb{P}(A, B) = \mathbb{P}(A)\ \mathbb{P}(B)$
5. **Bayes Rule**
    a. $$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A)\ \mathbb{P}(B \mid A)}{\mathbb{P}(B)} \quad \text{if } \mathbb{P}(B) > 0$$

Rules for Joint Distributions of Random Variables

For continuous random variables, replace sums by integrals:

1'. Densities sum to one: $\sum_x \Pr_X(x) = 1$
2'. Conditional density: $\Pr_X(x \mid Y = y) = \Pr_{X,Y}(x,y) / \Pr_Y(y)$, when $\Pr_Y(y) > 0$
3'. Marginal density: $\Pr_X(x) = \sum_y \Pr_{X,Y}(x,y)$
Law of total probability: $\mathbb{P}(A) = \sum_x \Pr_X(x)\, \mathbb{P}(A \mid X = x)$
4'. X and Y are said to be *independent* if $\Pr_{X,Y}(x,y) = \Pr_X(x) \Pr_Y(y)$
5'. Bayes' rule:

$$\Pr_X(x \mid Y = y) = \frac{\Pr_X(x) \Pr_Y(y \mid X = x)}{\Pr_Y(y)} \quad \text{if } \Pr_Y(y) > 0$$

Fitting Distributions

Random sample is a collection of random variables all drawn from the same distribution, and all independent of each other. If $Y = (X_1, \ldots, X_n)$ is the random sample and $y = (x_1, \ldots, x_n)$ is a collection of values (dataset) then:

$$Pr_Y(y) = Pr_X(x_1) \times \ldots \times Pr_X(x_n)$$

Where X is the common distribution. We also say X1, …, Xn are independent and identically distributed. If distribution of X depends on some parameter Θ which we'd like to estimate given a dataset. The likelihood given a single observation is:

$$lik(\theta \mid x) = Pr_X(x \mid \theta)$$

And the likelihood given a dataset is:

$$lik(\theta \mid x_1, \ldots, x_n) = Pr_X(x_1 \mid \theta) \times \ldots \times Pr_X(x_n \mid \theta)$$

Fitting the distribution means finding the maximum likelihood estimator for Θ, solving:

$$\hat{\theta} = argmax_\theta \log lik(\theta \mid x_1, \ldots, x_n)$$

Using Python:

```
# Numerical solution (using a scaled loglik, for numerical stability)
def loglik(p, x):
    return numpy.log(p) + (numpy.mean(x) - 1) * numpy.log(1-p)
initial_guess = numpy. array ([0.5])
mle = scipy.optimize.fmin(lambda p: -loglik(p,x), initial_guess)
(p,) = mle #unpack mle, which is a list of length 1
```

Multiple parameters: Differentiate with respect to each in turn and then find where derivative is zero and then solve the simultaneous equations

Custom Distributions

In order to design a distribution function is to plot the empirical cumulative distribution function:

$$\hat{F}(x) = \frac{1}{n}(how\ many\ items\ there\ are \leq x)$$

From there, if the data moves too fast, take logs, then alter it – and split it into a thing where you can find straight lines from the distribution and then doing a MLE for all the parameters.

Inversion Method

(1) Generate a simple random variable U ~ Uniform[0, 1]
(2) Solve F(X) = U for X
(3) Now X has cumulative distribution function F

Why it works: ensures for ever x the event {X ≤ x} is precisely the event {U ≤ F(x)} which has the probability F(x). In regions where density of Pr_X is high then F will be steep, and so U is more likely to hit those regions.

This requires us to solve F(X) = U, which is easy to do algebraically for continuous functions like the straight lines fit. Method also correct for discrete random variables.

Fitting a model

Multivariate dataset: contains records each consisting of a tuple of values. We want to understand how one item in the tuple depends on the others. The item we want to understand is called the **response variable** and the others are called **covariates or predictors**

Invent a probabilistic model in which we treat the response as a random variable, whose distribution depends on both the covariates and on unknown parameters. Use the maximum likelihood estimation to estimate the unknown parameters – this is fitting the model.

Exercise 1.11 (Binomial regression model).
The UK Home Office makes available several datasets of police records, at `data.police.uk`. The stop-and-search dataset has been preprocessed to list the number of stops and the number of those that led to the police finding something suspicious, for each police force and each year.

police_force	year	stops	find
bedfordshire	2017	786	231
cambridgeshire	2016	1691	621
cambridgeshire	2017	581	264

Fit the model $Y_i \sim \text{Binom}(x_i, p)$ where Y_i is the number of 'find' incidents in a given police force and year, x_i is the number of stops, and p is the parameter to estimate.

The binomial distribution is a discrete random variable commonly used for counting the number of successes in a sequence of yes-no trials. If $X \sim \text{Binom}(n, p)$ then n is the number of trials, $0 \le p \le 1$ is the success probability, and the probability mass function is

$$\Pr_X(r \mid n, p) = \binom{n}{r} p^r (1-p)^{n-r}, \quad r \in \{0, \dots, n\}.$$

We'll assume the records in the dataset are independent, since we're not told otherwise. In maths notation,

$$\Pr(y_1, \dots, y_n \mid p) = \prod_{i=1}^{n} \binom{x_i}{y_i} p^{y_i} (1-p)^{x_i - y_i}.$$

(Covariates are fixed and known so we're treating them as constants in this equation, not as parameters.) The log likelihood is

$$\begin{aligned} \log \text{lik}(p \mid y_1, \dots, y_n) &= \sum_i \left(\log \binom{x_i}{y_i} + y_i \log p + (x_i - y_i) \log(1-p) \right) \\ &= \kappa + \left(\sum_i y_i \right) \log p + \left(\sum_i x_i - \sum_i y_i \right) \log(1-p) \end{aligned}$$

where κ is a constant i.e. doesn't depend on p. The maximum likelihood estimator for p solves

$$\frac{d}{dp} \log \text{lik}(p \mid y_1, \dots, y_n) = 0$$

and the solution is

$$\hat{p} = \frac{\sum_i y_i}{\sum_i x_i}.$$

Random Variables

Mean and Variance

$$\mathbb{E}\,X = \begin{cases} \sum_x x \Pr_X(x) & \text{for a discrete random variable} \\ \int_x x \Pr_X(x)\,dx & \text{for a continuous random variable.} \end{cases}$$

The *variance* and *standard deviation* are

$$\operatorname{Var} X = \mathbb{E}\big((X - \mathbb{E}\,X)^2\big), \quad \text{std. dev}(X) = \sqrt{\operatorname{Var} X}.$$

The *conditional expectation* given either an event A or a random variable Y is[4]

$$\mathbb{E}(X \mid A) = \sum_x x \Pr_X(x \mid A) \quad \text{when } \mathbb{P}(A) > 0$$

$$\mathbb{E}(X \mid Y = y) = \sum_x x \Pr_X(x \mid Y = y) \quad \text{when } \Pr_Y(y) > 0$$

(and if X is a continuous random variables, just replace the sum by an integral).

$$\mathbb{E}\,1_{X\in A} = 1\times\mathbb{P}(1_{X\in A}=1) + 0\times\mathbb{P}(1_{X\in A}=0) = \mathbb{P}(X \in A). \tag{3}$$

For all constants a and b, and for any two random variables X and Y,

$$\begin{aligned} &\mathbb{E}(aX + b) = a(\mathbb{E}\,X) + b \\ &\mathbb{E}(X + Y) = (\mathbb{E}\,X) + (\mathbb{E}\,Y). \end{aligned} \tag{4}$$

These two equations are known as "linearity of expectation". Also, for all constants a and b,

$$\begin{aligned} &\operatorname{Var}(aX + b) = a^2 \operatorname{Var} X \\ &\text{std. dev}(aX + b) = a\,\text{std. dev}(X). \end{aligned} \tag{5}$$

For any two independent random variables X and Y,

$$\begin{aligned} &\mathbb{E}(XY) = (\mathbb{E}\,X)(\mathbb{E}\,Y) \\ &\operatorname{Var}(X + Y) = \operatorname{Var} X + \operatorname{Var} Y \\ &\text{std. dev}(X + Y) = \sqrt{\text{std. dev}(X)^2 + \text{std. dev}(Y)^2}. \end{aligned} \tag{6}$$

When X and Y are not independent, it is sometimes useful to measure their dependence by *covariance* or *correlation*,

$$\operatorname{Cov}(X, Y) = \mathbb{E}\big((X - \mathbb{E}\,X)\,(Y - \mathbb{E}\,Y)\big), \quad \operatorname{corr}(X, Y) = \frac{\operatorname{Cov}(X, Y)}{\text{std. dev}(X)\,\text{std. dev}(Y)}.$$

For a function of a random variable $Y = h(X)$, there are two ways to work out its expectation:

$$\mathbb{E}\,Y = \sum_y y \Pr_Y(y) = \sum_x h(x) \Pr_X(x) \tag{7}$$

Confidence Intervals

A random variable can be approximated by:

$$X \approx \text{Normal}(\mu, \sigma^2) \quad \text{where} \quad \mu = \mathbb{E}\,X\,,\ \sigma^2 = \text{Var}\,X$$

and so $$\mathbb{P}\big(\mu - 1.96\sigma \le X \le \mu + 1.96\sigma\big) \approx 95\%\,.$$

(This is why variance is a useful measure of variability.)

If $X \sim \text{Normal}(\mu, \sigma^2)$ then the confidence interval (9) is exact; and also

$$aX + b \sim \text{Normal}(a\mu + b,\ a^2\sigma^2) \quad \text{for constants } a \text{ and } b$$
$$X + Y \sim \text{Normal}(\mu + \nu,\ \sigma^2 + \rho^2) \quad \text{if } Y \text{ is an independent Normal}(\nu, \rho^2).$$

Convergence Theorems

Let $X_1, X_2, \ldots$ be independent identically distributed random variables. Let $\bar{X}_n$ be the average of the first n of these,

$$\bar{X}_n = \frac{X_1 + \cdots + X_n}{n}.$$

Then

$$\bar{X}_n \to \mu \quad \text{as } n \to \infty \tag{10}$$

and furthermore

$$\bar{X}_n \approx \text{Normal}\Big(\mu, \frac{\sigma^2}{n}\Big) \tag{11}$$

where $\mu = \mathbb{E}\,X$ and $\sigma = \text{std. dev}(X)$, and X is the common distribution of the X_i. Equation (10) is called the *law of large numbers*, and (11) is called the *central limit theorem*.

Law of Large Numbers

We want to prove that $\bar{X}_n \to \mu$. Let's calculate the probability that they differ by more than some amount $\varepsilon > 0$:

$$\begin{aligned}
\mathbb{P}\big(|\bar{X}_n - \mu| > \varepsilon\big) &= \mathbb{P}\left(\frac{(\bar{X}_n - \mu)^2}{\varepsilon^2} > 1\right) \quad \text{by simple algebra} \\
&= \mathbb{E}\left(1\Big[\frac{(\bar{X}_n - \mu)^2}{\varepsilon^2} > 1\Big]\right) \quad \text{from page 17, } \mathbb{E}\,1_A = \mathbb{P}(A) \\
&\le \mathbb{E}\left(\frac{(\bar{X}_n - \mu)^2}{\varepsilon^2}\right) \quad \text{since } 1_{x>1} \le x \text{ for } x \ge 0 \\
&= \frac{1}{\varepsilon^2}\,\text{Var}\,\bar{X}_n \quad \text{by linearity of } \mathbb{E} \text{ and definition of Var} \\
&= \frac{1}{n^2\varepsilon^2}\,n\,\text{Var}\,X \quad \text{by linearity of Var, and independence} \\
&= \sigma^2\ /\ n\varepsilon^2 \\
&\to 0 \quad \text{as } n \to \infty.
\end{aligned}$$

Weak law of large numbers: $\mathbb{P}(|\overline{X_n} - \mu| > \varepsilon) \to 0$
Strong law of large numbers: $\mathbb{P}(\overline{X_n} \to \mu) = 1$ (this implies the weak law)

Therefore $\overline{X_n}$ approaches μ as n -> ∞ and the smaller σ is, the smaller the error is likely to be. The central limit theorem additionally lets us find a confidence interval for the error.

Central Limit Theorem

Consider $\sqrt{n}(\bar{X}_n - \mu)$. It's easy to use the rules for mean and variance to check that

$$\mathbb{E}\big(\sqrt{n}(\bar{X}_n - \mu)\big) = 0\,, \quad \mathrm{Var}\big(\sqrt{n}(\bar{X}_n - \mu)\big) = \sigma^2.$$

It can be proven (assuming $\sigma < \infty$) that

$$\mathbb{P}\big(\sqrt{n}(\bar{X}_n - \mu) \le x\big) \to \mathbb{P}\big(\mathrm{Normal}(0, \sigma^2) \le x\big) \quad \text{as } n \to \infty \text{ for all } x.$$

Monte Carlo integration

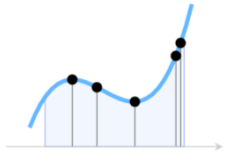
Let X be a random variable, and suppose we're interested in $\mathbb{E}\,h(X)$ for some function h. If it's difficult to calculate exactly, we can approximate it by

$$\mathbb{E}\,h(X) \approx \frac{1}{n}\sum_{i=1}^{n} h(X_i)$$

where $X_1, \ldots, X_n$ is a random sample drawn from distribution X. This is called *Monte Carlo integration*. As a special case,

$$\mathbb{P}(X \in A) = \mathbb{E}\,1_{X\in A} \approx \frac{1}{n}\sum_{i=1}^{n} 1_{X_i \in A}.$$

Why it works: where trapezium method, etc, split the integral into equally sized pieces, there is no real reason to do this, can just pick independent random variables and approximate:



$$\approx \frac{1}{n}\sum_{i=1}^{n} h(X_i).$$

Error of the Monte Carlo estimator: $O(\sigma/\sqrt{n})$ – however, need to be able to calculate σ, which is easier to estimate using the Monte Carlo method rather than calculating

$$\sigma^2 = \mathbb{E}(X-\mu)^2 \approx \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2$$

can plug in estimate for μ using Monte Carlo method – can keep a running estimate of σ and can stop when the error gets small enough.

Empirical Distribution

The *empirical distribution* of a dataset $x_1, \ldots, x_n$ is

$$\hat{\mathbb{P}}(A) = \frac{1}{n}(\text{how many items there are } \in A)$$

and the *empirical cumulative distribution function* for a numerical dataset is

$$\hat{F}(x) = \frac{1}{n}(\text{how many items there are } \leq x)$$

The zen of data science is in seeing datasets and random variables as two sides of the same coin. Empirical distributions *are* probability distributions.

- When the true distribution is unknown, we can use the dataset's empirical distribution instead. There's no need to approximate the dataset by fitting a standard random variable, when we can just *resample* from the dataset.
- When the true distribution is intractable, we can approximate it by the empirical distribution of a random sample. Instead of getting bogged down with integrals, we just use Monte Carlo integration.

Resampling: Sample from the empirical distribution function using the inversion method – the same as picking a value at random from the dataset, with each item being equally likely. For a parametric distribution, we only need to store a handful of parameters, rather than the full dataset. Can also have **high-dimensional modelling (active research topic)** – modelling with more parameters than there are samples in the dataset.

Standard Random Variables

Geometric

X ~ Geom(p)
Takes values in {1, 2, …, n}

$$\mathbb{P}(X = r) = (1-p)^{r-1}p, \quad \mathbb{P}(X \geq r) = (1-p)^{r-1}.$$

Mean: 1/p
Variance: (1-p)/p^2
Python: numpy.random.geometric(p)

Exponential

Used to model the time until an event – such as the time until a lump of radioactive matter emits its next particle.

X ~ Exp(λ) – λ is called the rate
Takes values in [0, ∞]

$$\Pr(x) = \lambda e^{-\lambda x}, \quad \mathbb{P}(X \geq x) = e^{-\lambda x}.$$

Chance of an event in a short interval of time [t, t + δ] is:

$$\mathbb{P}(X \leq t + \delta \mid X \geq t) = \frac{\mathbb{P}(X \in [t, t+\delta])}{\mathbb{P}(X \geq t)} = \frac{\int_t^{t+\delta} \lambda e^{-\lambda x}\, dx}{e^{-\lambda t}} \approx \delta\lambda.$$

Mean: 1/λ
Variance: 1/ λ^2
Python: numpy.random.exponential(scale = 1/λ)

Binomial

Toss a biased coin n times, and each coin has chance p of heads, total number of heads. When n is 1, it's called a Bernoulli random variable.

X ~ Binom(n, p)
Takes values in {0, 1, …, n}

$$\mathbb{P}(X = r) = \binom{n}{r} p^r (1-p)^{n-r}.$$

Mean: np
Variance: np(1-p)
Python: numpy.random.binomial(n, p)

Multinomial

N individuals each of whom falls into one of K categories and probability of falling into category k is p_k

X ~ Multinom(n, p)
Takes values in $\{0, 1, \ldots, n\}^K$

$$\mathbb{P}(X = x) = \frac{n!}{x_1!x_2!\cdots x_K!} p_1^{x_1} p_2^{x_2} \cdots p_K^{x_K}$$

Python: numpy.random.multinomial(n, p)

Poisson

If time between events is exp(λ) then total number of events in time t.

X ~ Poisson(λt)
Takes value in {0, 1, …}
Mean and Variance: λ
Python: numpy.random.poisson(lam = λ)

Normal / Gaussian

Good aggregate of lots of small pieces:

X ~ Normal(μ, σ^2)
Takes value in reals

$$\Pr(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad \mathbb{E}\,X = \mu, \quad \operatorname{Var} X = \sigma^2.$$

If you have two independent distributions X ~ Normal(μ, σ^2) and Y ~ Normal(ν, ρ^2) and a and b are real numbers

$$\mathbb{P}\big(\mu - 1.96\sigma \le X \le \mu + 1.96\sigma\big) = 95\%$$
$$aX + b \sim \text{Normal}(a\mu + b, a^2\sigma^2)$$
$$(X - \mu)/\sigma \sim \text{Normal}(0, 1)$$
$$X + Y \sim \text{Normal}\big(\mu + \nu, \sigma^2 + \rho^2\big)$$

Python: numpy.random.normal(loc=μ, scale=σ)

Pareto and Lognormal

Avoid reliance on a single black swan event.

X ~ Pareto(α)
Takes values in [0, ∞]

$$\Pr(x) = \alpha x^{-(\alpha+1)}, \quad \mathbb{P}(X \geq x) = x^{-\alpha}$$

$$\mathbb{E}\,X = \begin{cases} \infty \text{ if } \alpha \leq 1 \\ \alpha/(\alpha-1) \text{ otherwise,} \end{cases} \qquad \operatorname{Var} X = \begin{cases} \infty \text{ if } \alpha \leq 2 \\ \alpha \,/\, (\alpha-1)^2(\alpha-2)^2 \text{ otherwise.} \end{cases}$$

For α < 2, it tends to produce many small values and very occasional huge values

Zipf

Describes frequencies of words in texts – comparing rank and frequency.

X ~ Zipf(n, s)
Takes values in {0, 1, …, n}

$$\mathbb{P}(X = r) = \frac{r^{-s}}{1 + 2^{-s} + \cdots + n^{-s}}$$

Beta

If we toss a biased coin n times, and each coin has chance p of heads, then number of heads takes Bin(n, p). A common prior distribution for p is Beta(α, β).

Takes values in (0, 1) and has parameters α > 0 and β > 0

$$\Pr(p) = \binom{\alpha+\beta-2}{\alpha-1} p^{\alpha-1}(1-p)^{\beta-1}$$

Mean: α/(α + β)
Python: numpy.random.beta(a=α, b=β)

Dirichlet

Generalisation of Beta – has k categories and α is a vector in $\mathbb{R}^K$
Takes value in

$$\Omega = \left\{ [x_1, \ldots, x_K] \in (0,1)^K \;:\; x_1 + \cdots + x_K = 1 \right\}$$

Generates probability distributions over the k categories – used in Bayesian inference to describe belief about a multinomial distribution – interpretation is "seen α_k items in category k"

$$\Pr([x_1, \ldots, x_K]) \propto x_1^{\alpha_1 - 1} x_2^{\alpha_2 - 1} \cdots x_K^{\alpha_K - 1}$$

Python: numpy.random.dirichlet(alpha=α)

Gamma

Sum of k independent exponential random variables – common choice of prior distribution for $1/\sigma^2$ in Bayesian calculations with Normal(μ, σ^2)

X ~ Γ(k, λ)
Takes values in [0, ∞]

$$\Pr(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$$

(k-1)! Is replaced by gamma function Γ(k) for non-integer k
Mean: kλ
Variance: k/λ^2
Python: numpy.random.gamma(shape=k, scale=1/λ)

Inference

Reaching conclusions on the basis of data and reasoning.

Bayesianism

Doctrine that one should represent uncertainty about unknown parameters by describing them as random variables. It insists that we set down a prior belief for all unknown parameters before we even think about incorporating data – doesn't really matter what the prior is as long as the dataset is large enough

Suppose we have a probabilistic model $\Pr_X(x|\theta)$ where X is a random variable representing possible outcomes of an experiment and θ is the unknown parameters in the model. Bayesianism says we should represent our uncertainty about unknown parameters by using a probability distribution. We start with a *prior distribution* with density $\Pr_\Theta(\theta)$ for θ, then observe data x, then calculate a *posterior distribution* with density $\Pr_\Theta(\theta \mid X = x)$, by using Bayes's rule:

$$\Pr_\Theta(\theta \mid X = x) = \frac{\Pr_\Theta(\theta)\Pr_X(x \mid \theta)}{\int_\phi \Pr_\Theta(\phi)\Pr_X(x \mid \phi)\, d\phi}$$

(with the integral replaced by a sum, if Θ is a discrete random variable). It's common to write this as

$$\Pr_\Theta(\theta \mid X = x) \propto \Pr_\Theta(\theta)\Pr_X(x \mid \theta)$$

where the constant of proportionality is whatever is needed to make the left hand side be a density function, i.e. to integrate to one.

Any conclusions we want to draw about the unknown parameters should simply be written out as probabilities about $(\Theta \mid X = x)$. Standard readouts are *posterior confidence intervals*, *posterior point estimates*, and *posterior predictive probabilities*.

Conjugate Priors: When posterior distribution ends up belonging to the same family as the prior distribution, just with different parameters

Sampling from posterior distribution: Can use the Gibbs sampler (based on Markov chains) to simple from a posterior distribution

Readouts from posterior distribution: For example, to see if a coin is biased, see if $\mathbb{P}(\Theta > ½ \mid X = 9)$ – can get this through integrating the probability from ½ to 1, or by resampling using the posterior distribution then using Monte Carlo integration

Confidence Interval: Can report confidence interval, therefore for Θ is a Uniform[0, 1]

$$\mathbb{P}(\Theta \in [\text{lo}, \text{hi}] \mid X = x) = 95\% .$$

An obvious way to do this is to pick lo and hi so that $\mathbb{P}(\Theta \leq \text{lo} \mid X = x) = 0.025$ and $\mathbb{P}(\Theta \leq \text{hi} \mid X = x) = 0.975$. In Python,

```
lo, hi = scipy.stats.beta.ppf([0.025, 0.975], a=x+1, b=n-x+1)  # (0.587, 0.977)
```

Or the Monte Carlo approach:

```
lo, hi = numpy.quantile(θsample, [.025, 0.975]) # (0.595, 0.977)
```

Posterior Point Estimates: When asked given the observed data what is the value of the parameter:

(1) Report value of parameter that maximises the mode of the posterior distribution. **This is the maximum a posteriori (MAP) estimate**
(2) Return mean or median of the posterior distribution
(3) Report a loss function, which measures the price you pay if you report the estimate and the true value. Report Θ' which minimises the expected posterior loss.

Posterior Predictive Probability: Probability that the next event will be something. Can use the law of total probability. For example, for the example of a coin flip – looking for probability next flip X' will be a head

$$\begin{aligned}
\mathbb{P}(X' = \text{heads} \mid X = x) &= \int_\theta \Pr_\Theta(\theta \mid X = x)\ \mathbb{P}(X' = \text{heads} \mid \Theta = \theta, X = x)\, dx \\
&= \int_\theta \Pr_\Theta(\theta \mid X = x)\, \theta\, dx \\
&= \mathbb{E}(\Theta \mid X = x) = \frac{x+1}{n+2} \approx 0.833 \quad \text{(the posterior mean).}
\end{aligned}$$

The Python for this is: numpy.mean(Θsample)

Nuisance Parameters

Example 3.3 (Nuisance parameters).
Given a random sample from distribution $X \sim \text{Uniform}[\theta, \theta+\phi]$, and using the prior distributions $\Theta \sim \text{Exp}(\lambda_0)$ and $\Phi \sim \text{Exp}(\mu_0)$, find the posterior density of ϕ.

In this question, θ is called a *nuisance parameter*. It's part of the probability model, but we're not interested in its value. In such cases we must first work out the posterior distribution for *all* the unknown parameters, because that's what Bayes' rule tells us to do. From this joint distribution we can extract the information we want, for example by marginalizing out the nuisance parameters.

First, write out the priors and the density from the model:

$$\begin{aligned}
\Pr_\Theta(\theta) &= \lambda_0 e^{-\lambda_0\theta} \\
\Pr_\Phi(\phi) &= \mu_0 e^{-\mu_0\phi} \\
\Pr(x_1,\dots,x_n \mid \theta,\phi) &= \prod_{i=1}^{n}\Big(\frac{1}{\phi}1_{x_i\in[\theta,\theta+\phi]}\Big) = \frac{1}{\phi^n}1_{m\geq\theta}1_{M\leq\theta+\phi}
\end{aligned}$$

where $m = \min_i x_i$ *and* $M = \max_i x_i$. *(In this problem the endpoints of the* X *distribution are themselves parameters, so it's a good idea to write out the density in full using indicator functions.) So the posterior is*

$$\Pr_{\Theta,\Phi}(\theta,\phi \mid \textit{data}) = \frac{1}{\kappa}\lambda_0 e^{-\lambda_0\theta}\mu_0 e^{-\mu_0\phi}\frac{1}{\phi^n}1_{\theta\leq m}1_{\theta+\phi\geq M}.$$

The posterior density for ϕ *can be found by marginalizing out the nuisance parameter:*

$$\begin{aligned}
\Pr_\Phi(\phi \mid \textit{data}) &= \int_\theta \Pr_{\Theta,\Phi}(\theta,\phi \mid \textit{data})\,d\theta \\
&= \frac{1}{\kappa}\frac{\mu_0 e^{-\mu_0\phi}}{\phi^n}\int_\theta \lambda_0 e^{-\lambda_0\theta}1_{\theta\leq m}1_{\theta\geq M-\phi}\,d\theta \\
&= \frac{1}{\kappa}\frac{\mu_0 e^{-\mu_0\phi}}{\phi^n}\Big(e^{-\lambda_0\max(M-\phi,0)} - e^{-\lambda_0 m}\Big)1_{\phi\geq M-m}.
\end{aligned}$$

Frequentism

Suppose we have a probabilistic model $\Pr_X(x|\theta)$ where X is a random variable representing possible outcomes of an experiment and θ is an unknown parameter. Frequentism says that θ is fixed but unknown, and we should base our thinking on all the ways that the experiment might have turned out but didn't.

The frequentist is interested in *output procedures* such as

```
def confint(x):
    lo = ...   # some function of x
    hi = ...   # some function of x
    print("θ is in [{}, {}]".format(lo, hi))
```

Running confint(X) will print a true statement or a false statement, depending on the random variable X, and the frequentist is interested in designing output procedures that prints a false statement with no more that 5% probability, whatever the value of θ. This is referred to as bounding the *error probability* of the procedure.

There is a general-purpose computational method called *bootstrap resampling* for estimating the error probability of an output procedure.

Resampling

If the trial were run again, what is a good way to use the data at hand to synthesize a result I could plausibly see

Parametric: (1) specify a probability model with unknown parameters, (2) fit the parameters using maximum likelihood (3) sample from the fitted probability

This relies on some sort of intuition – resample using intuition and parameters which have been found.

The Bootstrap: removes any need for maths or exhaustive optimization for bounding the error probability

(1) Write probability you're interested in
(2) Replace any unknown parameters by the MLE given the data – replace any random variables by their resampled versions
(3) Use the Monte Carlo method to estimate the probability of the expression in step 2

Non-Parametric: Sample from the empirical distribution, which is equivalent to picking a value at random from the observed dataset. The idea is that the best-fitting distribution is the dataset itself – can then do Monte Carlo integration or draw a histogram

> Example 3.6 (Non-parametric resampling).
> I collected a dataset $x_1, \ldots, x_n$ and I found the sample mean $\bar{x}$. If I repeat the exercise and collect further datasets, how much variability should I expect to see in the sample mean?

```
xs = [13, 5, 2, ...]    # the dataset
def sim_mean():
    n = len(xs)
    X = random.choices(xs, k=n)  # k = number of samples to draw
    return sum(X) / n

mc_samples = [sim_mean() for i in range(100000)]
matplotlib.pyplot.hist(mc_samples)
```

Model Selection

Model: explanation for the data – simpler models are likely to be closer to the truth and to generalise better to new scenarios. Can also be used to make predictions about new data.

Hypothesis Testing and P-Values

Come up with a null hypothesis and see if you reject it or not.

(1) Define a function test_statistic(y) – can be any function of the data. We aim for a function that is likely to be small if the null hypothesis is true and large if it is false.
(2) Assuming the null hypothesis is true, find the distribution of T = test_statistic(Y) – through resampling
(3) From the actual data – compute t = test_statistic(y). Mark t on histogram and measure p = $\mathbb{P}(T > t)$
(4) If p ≤ 5%, reject null hypothesis, otherwise don't reject it

p is the p-value or significance level, and it measures the probability of seeing results as extreme as we can actually saw, assuming null hypothesis is true.

Alternative Hypothesis

test_statistic(Y) = lik(model = A | Y = y) / lik(model = B | Y = y) where the likelihood of a model is found by maximising over all parameters in the model.

This is the likelihood ratio test and allows us to decide between two different models – if model B is false, then the denominator will be small, so test statistic will be large.

Creating a Model

Purposes of unknown parameters:

(1) Make the model expressive
(2) When we fit the model and inspect the estimated parameters, we learn which patterns are actually present in the dataset.

How to choose parameters:

(1) Use parameters that correspond to the questions we want to ask and the quantities we want to measure – **identifiability**
(2) Run maximum likelihood estimators, using a numerical optimisation library – **natural parameters**
    a. Should map the likelihood to between 0 and 1 through a/1+a – **logit and softwmax**

Logistic Regression

Logistic – uses the logic transform for parameters
Regression – has a response variable predicted by covariates

Feature Spaces

Feature: any measurable property of the objects being studied.
Linear Model: model with unknown parameters in which the parameters are weighted by features and combined linearly

Fitting a linear model

A *linear model* can be written as

$$y = \beta_1 e_1 + \cdots + \beta_K e_K + \varepsilon$$

where $y = (y_1, y_2, \dots)$ is the vector of responses with y_i the value for record i in the dataset, $e_1, \dots, e_K$ are feature vectors with $e_k = (e_{k,1}, e_{k,2}, \dots)$ where $e_{k,i}$ is the value of the kth feature for record i, β_k is the parameter that weights the kth feature, and $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots)$ is a vector of *residuals*, also called error or noise.

Least squares estimation means picking the parameters β to minimize the *mean square error* $\sum_i \varepsilon_i^2$. Use sklearn.linear_model.LinearRegression() to do this.

Important Note: Linear doesn't mean straight line – simply means linear algebra (adding vectors), but can have $\mathbf{x^2}$ as a vector

```
one, x, y = numpy.ones(len(iris)), iris['Sepal.Length'], iris['Petal.Length']
model = sklearn.linear_model.LinearRegression(fit_intercept=False)
# Specify the three feature vectors [one, x, x**2] and the response vector y
model.fit(numpy.column_stack([one, x, x**2]), y[:, numpy.newaxis])
((α,β,γ),) = model.coef_ # unpack a 1 × 3 array of parameters
```

sklearn model fitting function includes a one vector unless we explicitly tell it otherwise with fit_intercept=False. Can also write this as:

```
model2 = sklearn.linear_model.LinearRegression()
model2.fit(numpy.column_stack([x, x**2]), y[:, numpy.newaxis])
(α,),((β,γ),) = model2.intercept_, model2.coef_
```

Features

One-Hot Encoding: Turn an enum feature into a collection of binary features so it could be used in a linear model.

$$\text{Petal.Length} \approx \alpha_{\text{species}} + \beta_{\text{species}}\text{Sepal.Length}.$$

Here's the same equation, but written as a linear model:

$$\begin{aligned}\text{Petal.Length} \approx\ & \alpha_1 s_1 + \alpha_2 s_2 + \alpha_3 s_3 \\ & + \beta_1(s_1 \otimes \text{Sepal.Length}) + \beta_2(s_2 \otimes \text{Sepal.Length}) + \beta_3(s_3 \otimes \text{Sepal.Length})\end{aligned}$$

Each s_k is a binary vector marking out which rows belong to the kth species – **one-hot coding** of the vector. $\otimes$ means elementwise multiplication

As you add more features, the better the fit and the smaller the residual we can achieve. But a model with too many features tends to be bad at generalizing to new data. Therefore, we want to be careful about this:

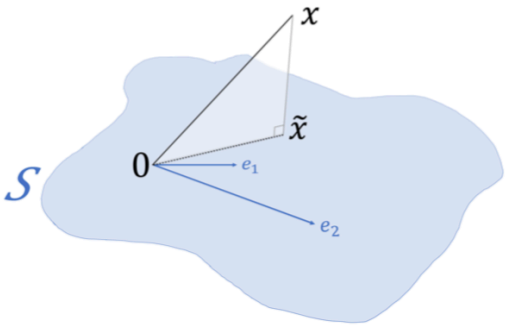
Dimension Reduction: General term for finding lower-dimensional representations

(1) Start with long list of possible features {e1, e2, …, ek}
(2) Construct two new features f1 and f2 each of which a linear combination of the raw features. The goal is to construct them to minimise the errors of:
    a. Y = a + B1f1 + B2f2
(3) Procedure gives us two features f1 and f2 that capture as much information as they can about Y – **two-dimensional embedding of the dataset's K dimensions**
(4) Can show data on scatter plot of f1 vs f2 – likely to reveal useful clusters

Feature Selection:

(1) Start with long list of possible features
(2) Pick m, a number of features to use, and find the best fitting model subject to the constraint that it's only allowed to use m of the possible features

Linear Mathematics



- Linearly independent basis vectors e_1 and e_2
- The set of linear combinations of those vectors, also known as the subspace spanned by those vectors, is $S = \{\lambda_1 e_1 + \lambda_2 e_2 \ : \ -\infty < \lambda_1, \lambda_2 < \infty\}$
- Another vector x can be projected onto the subspace, by finding the point $\tilde{x} = \hat{\lambda}_1 e_1 + \hat{\lambda}_2 e_2$ in S that is closest to x
- The residual $x - \tilde{x}$ is orthogonal to the basis vectors

For linear models in data science all we need is vectors in simple Euclidean space, $\mathbb{R}^n$ where n is the number of records in the dataset.

Other Definitions:

- V is the set whose elements are called vectors
- F is a field whose elements are called scalars – either reals or complex numbers
- Binary operation: V x V -> V is addition written: v + w
- Binary operation: F x V -> V is scalar multiplication, written: λv
- Binary operation: V x V -> F is inner product, written: v . w

Orthogonal Projection

The Projection Theorem. Let V be an inner product space, let $\{e_1, \ldots, e_n\}$ be a finite collection of vectors, and let S be the subspace spanned by these vectors. Given a vector $x \in V$, there is a unique vector $\tilde{x}$ that is closest to x, i.e. that solves[28]

$$\min_{x' \in S} \|x - x'\|^2.$$

Furthermore, $x - \tilde{x}$ is orthogonal to S, i.e.

$$(x - \tilde{x}) \cdot y = 0 \quad \text{for all } y \in S.$$

The vector $\tilde{x}$ is called the *orthogonal projection* of x onto S, and $x - \tilde{x}$ is called the *residual*.

If the e_i are linearly independent, i.e. if they form a basis for S, then we can find the coordinates of $\tilde{x}$ with respect to the e_i, and the coordinates are unique. If the e_i are linearly dependent, then there are multiple ways to write $\tilde{x}$ as a linear combination of the e_i.

Collinearity and matrix rank: Stacked collection of vectors to form a matrix then the rank of the matrix is the dimension of the space spanned by those vectors. Use: numpy.linalg.matrix_rank(numpy.column_stack([e1, e2]))

Linear Regression and Least Squares

A *linear regression* is a probabilistic model of the form

$$Y \sim \text{Normal}(\beta_1 e_1 + \cdots + \beta_K e_K, \sigma^2) \tag{25}$$

where $e_1, \ldots, e_K$ are covariates, Y is the random response, and σ and $\beta_1, \ldots, \beta_K$ are unknown parameters. Maximum likelihood estimation for this model is equivalent to least squares estimation for the corresponding linear model.

In a linear regression model, we can use all the inference tools from section 3: find confidence intervals for the parameters, test hypotheses, etc. (For inference based on resampling, use parametric resampling: first estimate the unknown parameters, then generate new observations from (25).)

Random Processes

Random (Stochastic) Process: Using probabilistic laws to describe how the system changes

Markov Chains

A *Markov chain* is a sequence $(X_0, X_1, X_2, \ldots)$ where each X_{n+1} is a discrete random variable, generated randomly based on only on X_n. The *state space* is the set of values from which the X_n are drawn.

If the probability distribution does not depend on n, i.e. if there is some matrix P such that

$$\mathbb{P}(X_{n+1} = y \mid X_n = x) = P_{xy}$$

we say it is a *time-homogeneous Markov chain* with *transition matrix* P.

Statement of Markov Assumption:

$$\mathbb{P}(X_{n+1} = x_{n+1} \mid X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n) = \mathbb{P}(X_{n+1} = x_{n+1} \mid X_n = x_n)$$
$$\text{for all } x_0, \ldots, x_{n+1}.$$

This is a statement that whatever the history, all that matters for the purposes of deciding the next state is the current state – the is memorylessness

Very many probabilistic models with dependencies (including Markov chains) can be represented by a *causal diagram*, a directed acyclic graph whose nodes are random variables and where the edges show which variables are used to generate which other variables. A Markov chain has a very simple causal diagram:

$$X_0 \to X_1 \to X_2 \to X_3 \to \ldots$$

Suppose we want to calculate a probability such as $\mathbb{P}(X_3 = x \mid X_0 = y)$. The general strategy for problems like this is:

1. Draw the causal diagram.
2. In the probability expression that we want to calculate, identify the random variables whose distributions we want to derive (in this case X_3). Pick out some useful ancestor variables in the causal diagram, somewhere between the variables we're given and the variables we want to calculate; these are called *latent variables* (in this case we might choose X_2 and X_1, and it's an art to pick out the most useful).
3. Incorporate the latent variables into the probability expression, typically by conditioning on them using the conditional form of the law of total probability.
4. Rewrite any probabilities to have the form 'probability of a child node, conditional on its parents'.
5. Solve!

Hitting Probability Theorem: Let A be a subset of a Markov chain's state space. The hitting probability from x is:

$$\mathbb{P}(ever\ hit\ A \mid start\ at\ x)$$

These hitting probabilities solve:

$$\pi_x = \sum_y P_{xy}\pi_y \ \ for\ \forall x \notin A, \qquad \pi_x = 1\ \forall x\ \in A$$

If the system of equations has multiple solution, the hitting probability from x is the minimum of π_x over all solutions $\pi \geq 0$

Limit theorems and equilibrium

Aim: Look at the long-run average behaviour of a time-homogeneous Markov chain with a finite state space

Stationary: Markov chain is stationary if its distribution does not change over time, ie if there is a vector π such that $\mathbb{P}(X_n = x) = \pi_x$ for all n. Conversely, if π is a probability distribution st:

$$\mathbb{P}(X_0 = x) = \pi_x \text{ for all } x \qquad \Longrightarrow \qquad \mathbb{P}(X_n = x) = \pi_x \text{ for all } x \text{ and } n$$

then π is called a stationary or equilibrium distribution. If π is a stationary distribution, and we pick the Markov chain's initial state X_0 randomly according to π, then X_1 will have distribution π and so on. If we pick the initial state in some other way, this is not true

Stationary distribution π must satisfy:

$$\pi_x = \sum_y \pi_y P_{yx} \qquad \text{for all } x$$

where P is the transition matrix

Theorem (Uniqueness of Stationary Distribution): Consider a Markov chain with transition matrix P and a finite state space. The Markov chain is irreducible if it is possible to get from any state to any other – this implies that the there is a unique stationary distribution and it is the unique solution to:

$$\pi = \pi P, \quad \pi^T . 1 = 1$$

Detailed Balance

Can sometimes use a trick to help us find stationary distribution for a Markov chain with very little algebra – using Gibbs sampling. It also can be used to generate random variables from Bayesian posterior distribution.

Theorem (detailed balance): Let X be a Markov chain with transition matrix P. If there is a vector π such that:

$$\pi_x P_{xy} = \pi_y P_{yx}\ for\ all\ states\ x\ and\ y$$

then π solves $\pi = \pi P$.

Theorem (ergodicity): Let X be an irreducible Markov chain with stationary distribution π. Then, the long-run average time spent in each state converges to π. Therefore:

$$\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} 1_{X_i = x}\right) \rightarrow \pi_x \quad as\ n \rightarrow \infty\ for\ all\ states\ x$$

If Markov chain's initial state X0 were chosen from π, then X_n would have π for every n, therefore $\mathbb{E}1_{X_i=x} = \mathbb{P}(X_i = x) = \pi_x$, for all I, and so the equation would be true exactly.

Limiting Behaviour and Aperiodicity

Theorem: Let X be a Markov chain – a state x is said to be aperiodic if there exists an n_0 st $\mathbb{P}(X_n = x \mid X_0 = x) > 0$ for all $n \geq n_0$. If the chain is irreducible and has an aperiodic state, then all its states are aperiodic and:

$$\mathbb{P}(X_n = x \mid X_0 = y) = [P^n]_{xy} \rightarrow \pi_x \ as\ n \rightarrow \infty\ for\ all\ states\ x\ and\ y$$

Can use this as a tool to generate a random variable from a distribution. Can create transition probabilities of a Markov chain to ensure it has stationary distribution as required. Then we can generate a random variable by starting the Markov chain in an arbitrary state and running it for a large number of steps and returning the state.