In this note, I review the book of David Mackay, and solve some challenging exercises in the book. To review and summarize the book, I quote or copy and paste the sentences of the book without approval of the author of the book. I do not use this note with the purpose of financial profit. You can find the materials for this note from **my github**, such as Mathematica notes, their pdf copies and some relevant python codes.

1 Introduction

1.1 Some formula in the chapter 1 of the book

Let's derive Stirling's approximation by an unconventional route. We start from the Poisson distribution with mean λ,

$$P(r\,|\,\lambda) = e^{-\lambda}\frac{\lambda^r}{r!} \qquad r \in \{0,1,2,\ldots\}. \tag{1.8}$$

For large λ, this distribution is well approximated – at least in the vicinity of $r \simeq \lambda$ – by a Gaussian distribution with mean λ and variance λ:

$$e^{-\lambda}\frac{\lambda^r}{r!} \simeq \frac{1}{\sqrt{2\pi\lambda}}\,e^{-\frac{(r-\lambda)^2}{2\lambda}}. \tag{1.9}$$

Let's plug $r = \lambda$ into this formula, then rearrange it.

$$e^{-\lambda}\frac{\lambda^\lambda}{\lambda!} \simeq \frac{1}{\sqrt{2\pi\lambda}} \tag{1.10}$$

$$\Rightarrow \quad \lambda! \simeq \lambda^\lambda e^{-\lambda}\sqrt{2\pi\lambda}. \tag{1.11}$$

This is Stirling's approximation for the factorial function.

$$x! \simeq x^x e^{-x}\sqrt{2\pi x} \quad \Leftrightarrow \quad \ln x! \simeq x\ln x - x + \tfrac{1}{2}\ln 2\pi x. \tag{1.12}$$

We have derived not only the leading order behaviour, $x! \simeq x^x e^{-x}$, but also, at no cost, the next-order correction term $\sqrt{2\pi x}$. We now apply Stirling's approximation to $\ln\binom{N}{r}$:

$$\ln\binom{N}{r} \equiv \ln\frac{N!}{(N-r)!\,r!} \simeq (N-r)\ln\frac{N}{N-r} + r\ln\frac{N}{r}. \tag{1.13}$$

Since all the terms in this equation are logarithms, this result can be rewritten in any base. We will denote natural logarithms ($\log_e$) by 'ln', and logarithms to base 2 ($\log_2$) by 'log'.

If we introduce the *binary entropy function*,

$$H_2(x) \equiv x\log\frac{1}{x} + (1-x)\log\frac{1}{(1-x)}, \tag{1.14}$$

then we can rewrite the approximation (1.13) as

$$\log\binom{N}{r} \simeq NH_2(r/N), \tag{1.15}$$

or, equivalently,

$$\binom{N}{r} \simeq 2^{NH_2(r/N)}. \tag{1.16}$$

If we need a more accurate approximation, we can include terms of the next order from Stirling's approximation (1.12):

$$\log\binom{N}{r} \simeq NH_2(r/N) - \tfrac{1}{2}\log\left[2\pi N\frac{N-r}{N}\frac{r}{N}\right]. \tag{1.17}$$

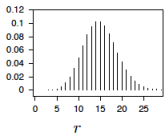

Figure 1.2. The Poisson distribution $P(r\,|\,\lambda = 15)$.

Recall that $\log_2 x = \dfrac{\log_e x}{\log_e 2}$.

Note that $\dfrac{\partial \log_2 x}{\partial x} = \dfrac{1}{\log_e 2}\dfrac{1}{x}$.

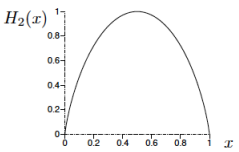

Figure 1.3. The binary entropy function.

Figure 1: Some formula

1.2 Assumptions in inference

First, once assumptions are made, the inferences are objective and unique, reproducible with complete agreement by anyone who has the same information and makes the same assumptions. For example, given the assumptions listed above, $\mathcal{H}$, and the data D, everyone will agree about the posterior probability of the decay length

$$P(\lambda|D,\mathcal{H}) = \frac{P(D|\lambda,\mathcal{H})P(\lambda|\mathcal{H})}{P(D|\mathcal{H})}$$

Second, when the assumptions are explicit, they are easier to criticize, and easier to modify indeed, we can quantify the sensitivity of our inferences to the details of the assumptions.

Third, when we are not sure which of various alternative assumptions is the most appropriate for a problem, we can treat this question as another inference task. Thus, given data D, we can compare alternative assumptions $\mathcal{H}$ using Bayes' theorem

$$P(\mathcal{H}|D,I) = \frac{P(D|\mathcal{H},I)P(\mathcal{H}|I)}{P(D|I)},$$

where I denotes the highest assumptions, which we are not questioning.

Fourth, we can take into account our uncertainty regarding such assumptions when we make subsequent predictions. Rather than choosing one particular assumption $\mathcal{H}^*$, and working out our predictions about some quantity t, $P(t|D,\mathcal{H},I)$, we obtain predictions that take into account our uncertainty about H by using the sum rule

$$P(\mathbf{t}|D,I) = \sum_{\mathcal{H}} P(\mathbf{t}|D,\mathcal{H},I)P(\mathcal{H}|D,I)$$

This is another contrast with orthodox statistics, in which it is conventional to 'test' a default model, and then, if the test 'accepts the model' at some 'significance level', to use exclusively that model to make predictions. *probability theory reaches parts that ad hoc methods cannot reach.*

Model comparison as inference. Assume we have two hypotheses. In order to perform model comparison, We wish to know how probable $P(\mathcal{H}_1)$ is given the data. By Bayes' theorem,

$$P(\mathcal{H}_1|\mathbf{s},F) = \frac{P(s|F,\mathcal{H}_1)P(\mathcal{H}_1)}{P(\mathbf{s}|F)},$$

and

$$P(\mathcal{H}_0|\mathbf{s},F) = \frac{P(\mathbf{s}|F,\mathcal{H}_0)P(\mathcal{H}_0)}{P(\mathbf{s}|F)}$$

The normalizing constant in both cases is the total probability of geting the observed data. and

$$P(s|F) = P(\mathbf{s}|F, \mathcal{H}_1)P(\mathcal{H}_1) + P(\mathbf{s}|F, \mathcal{H}_0)P(\mathcal{H}_0)$$

To evaluate the posterior probabilities of the hypotheses we need to assign values to the prior probabilities $P(\mathcal{H}_1)$ and $P(\mathcal{H}_0)$; in this case, we might set these to 1/2 each. And we need to evaluate the data-dependent terms $P(s|F, \mathcal{H}_1)$ and $P(s|F, \mathcal{H}_0)$. We can give names to these quantities. The quantity $P(s|F, \mathcal{H}_1)$ is a measure of how much the data favour $\mathcal{H}_1$), and we call it the evidence for model $\mathcal{H}_1$). *How model comparison works : The evidence for a model is usually the normalizing constant of an earlier Bayesian inference.*

2 Clustering and Maximum Likelihood

2.1 Motivation of clustering

First, a good clustering has predictive power. Second, clusters can be a useful aid to communication because they allow lossy compression. A third reason for making a cluster model is that failures of the cluster model may highlight interesting objects that deserve special attention. A fourth reason for liking clustering algorithms is that they may serve as models of learning processes in neural systems.

2.2 K-means

The K-means algorithm is an algorithm for putting N data points in an M-dimensional space into K clusters. Each cluster is parameterized by a vector $\mathbf{m}^{(k)}$ called its mean.

First of all, set K means $\mathbf{m}^{(k)}$ to random values. In the assignment step, each data point n is assigned to the nearest mean.

$$\hat{k}^{(n)} = argmin_k d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})$$

An alternative, equivalent representation of this assignment of points to clusters is given by 'responsibilities', which are indicator variables $r_k^{(n)}$. In the assignment step, we set $r_k^{(n)}$ to one if mean k is the closest mean to datapoint $\mathbf{x}^{(n)}$; otherwise, $r_k^{(n)}$ is zero.

$$r_k^{(n)} = \{ \begin{array}{ll} 1 & if\ \hat{k}^{(n)} = k \\ 0 & if\ \hat{k}^{(n)} \neq k \end{array}$$

In the update step, the means are adjusted to match the sample means of the data points that they are responsible for. The update step is very similar

to how to find the center of the mass in physics.

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

where $R^{(k)}$ is the total responsibility of mean k

$$R^{(k)} = \sum_n r_k^{(n)}$$

2.3 Exercise 22.5

$$P(k_n = 1|x_n, \boldsymbol{\theta}) = \frac{P(x_n|k_n = 1, \boldsymbol{\theta})P(k_n = 1, \boldsymbol{\theta})}{P(x_n, \boldsymbol{\theta})}$$

$$= \frac{P(x_n|k_n = 1, \boldsymbol{\theta})P(k_n = 1, \boldsymbol{\theta})}{\sum_{k_n} P(x_n|k_n = 1, \boldsymbol{\theta})P(k_n = 1, \boldsymbol{\theta}) + P(x_n|k_n = 2, \boldsymbol{\theta})P(k_n = 2, \boldsymbol{\theta})}$$

where $\boldsymbol{\theta} = (\mu_k,\ \sigma_k)$.

$$P(k_n = 1, \boldsymbol{\theta}) \equiv p_1, \quad P(k_n = 2, \boldsymbol{\theta}) \equiv p_2$$

Then,

$$P(k_n = 1|x_n, \boldsymbol{\theta}) = \frac{p_1}{p_1 + p_2 \exp[-(w_1 x_n + w0)]}$$

$$P(k_n = 2|x_n, \boldsymbol{\theta}) = \frac{p_2}{p_2 + p_1 \exp[-(w_1 x_n + w0)]}$$

where $w_1 = 2(\mu_1 - \mu_2)$, $w_0 = -(\mu_1 - \mu_2)(\mu_1 + \mu_2)$

$$P(k_n = k|x_n, \boldsymbol{\theta}) \equiv p_{k|n}$$

By assumption, the prior probability $p_1 = p_2 = 1/2$ then, (22.17) of the book is satisfied.

$$L \equiv \log \Pi_n P(x_n|\{\mu_k\},\ \sigma)$$

then trivially,

$$\frac{\partial}{\partial \mu_k} L = \sum_n \frac{p_{k|n}(x_n - \mu_k)}{\sigma^2}$$

$$\frac{\partial^2}{\partial {\mu_k}^2} L = -\sum_n \frac{p_{k|n}}{\sigma^2}$$

The new updated $\boldsymbol{\mu}'$ should maximize the likelihood. Then,

$$\frac{\partial}{\partial \mu'_k} L = \sum_n \frac{p_{k|n}(x_n - \mu'_k)}{\sigma^2} = 0$$

$$\sum_n p_{k|n}\ x_n - \sum_n p_{k|n}\ \mu'_k = \sum_n p_{k|n}\ x_n - \mu'_k \sum_n p_{k|n} = 0$$

Therefore,

$$\mu'_k = \frac{\sum_n p_{k|n}\ x_n}{\sum_n p_{k|n}}$$

Note that this equation is exactly the same as the updated means from the responsibilities and data points in soft K-means clustering. $p_{k|n}$ is the responsibility $r_n^{(k)}$.

$$\frac{\frac{\partial}{\partial \mu_k} L}{\frac{\partial^2}{\partial {\mu_k}^2} L} = \frac{\sum_n p_{k|n}\ x_n - \mu_k \sum_n p_{k|n}}{-\sum_n p_{k|n}} = -\mu'_k + \mu_k$$

Thus,

$$\mu'_k = \mu_k - \frac{\frac{\partial}{\partial \mu_k} L}{\frac{\partial^2}{\partial {\mu_k}^2} L}$$

2.4 Exercise 22.15

$$N = 7$$

$$x_n = (-27.02, 3.57, 8.191, 9.898, 9.603, 9.945, 10.056)$$

$$\frac{\sum_n x_n}{N} = 3.46329$$

It must not be the correct mean.

The likelihood is as followings by the description of the problem.

$$P(\{x_n\}|\sigma_n, \mu) = (\frac{1}{2\pi})^{N/2} \Pi_n \frac{1}{\sigma_n} \exp(-\sum_n \frac{(x_n - \mu)^2}{2\sigma_n^2})$$

To find the maximum likelihood,

$$L \equiv \log P(\{x_n\}|\sigma_n, \mu) = -\sum_n \log\ \sigma_n - \sum_n \frac{(x_n - \mu)^2}{2\sigma_n^2} = 0$$

$$\frac{\partial L}{\partial x_n} = -\sum_n \left(\frac{x_n - \mu}{\sigma_n^2} \right) = -\sum_n \frac{x_n}{\sigma_n^2} + \mu \sum_n \frac{1}{\sigma_n^2}$$

$$\mu = \frac{\sum_n \frac{x_n}{\sigma_n^2}}{\frac{1}{\sigma_n^2}}$$

When $x_n = (-27.02, 3.57)$, the correspondent σ_n are so huge, then it contributes very tiny in μ. Then, the mean should be close to mean of $(8.191, 9.898, 9.603, 9.945, 10.056)$.

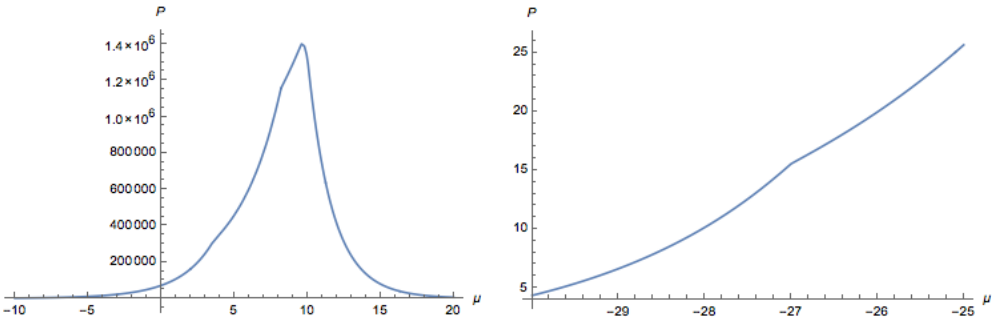

Figure 2: The distribution functions are not normalized yet. However, this plot already shows the maximum likelihood should be around $\mu = 10$. The small bumps around $x_n = (-27,\ 3.6,\ 8)$ are also seen in the graph.

2.5 Exercise 24.3

The setting is same as the exercise 22.15.

Bayesian for posterior probability of σ_n is

$$P(\sigma_n|\{x_n\},\mu) = \frac{P(\{x_n\}|\sigma_n,\mu)P(\sigma_n)}{P(\{x_n\}|\mu)}$$

Given

$$P(\{x_n\}|\sigma_n,\mu) = (\frac{1}{2\pi})^{N/2}\left(\Pi_n \frac{1}{\sigma_n}\right)\exp\left(-\sum_n \frac{(x_n-\mu)^2}{2\sigma_n^2}\right)$$

and

$$P(\sigma_n) = (\frac{1}{\Gamma(c)\ s^c})^N \Pi_n\ (\sigma_n)^{c-1}\ \exp(\frac{\sum_n \sigma_n}{s})$$

where $(s,\ c) = (10, 0.1)$.

and the normalizing constant is

$$P(\{x_n\}|\mu) = \int_0^\infty \Pi_n\ d\sigma_n\ P(\{x_n\}|\sigma_n,\mu)P(\sigma_n)$$

The posterior probability of μ is

$$P(\mu|x_n) = \frac{P(\{x_n\}|\mu)P(\mu)}{P(\{x_n\})}$$

and the prior is determined by some assumption. $P(\mu) = \frac{1}{\sigma_\mu} = const.$, then the normalizing constant is

$$P(\{x_n\}) = \int_{-\infty}^\infty d\mu\ P(\{x_n\}|\mu)\frac{1}{\sigma_\mu}$$

3 Gaussian approximation and model comparison

3.1 Laplace approximation

3.1.1 Exercise 27.1

r is a positive integer, and posterior over lambda

$$P(\lambda|r) = \frac{P(r|\lambda)P(\lambda)}{P(r)}$$

$$P(r|\lambda) = \exp(\lambda)\frac{\lambda^r}{r!}$$

By assumption

$$P(\lambda) = \frac{1}{\lambda}$$

Then, the normalizing constant is

$$P(r) = \int_0^\infty d\lambda \ \exp(\lambda)\frac{\lambda^r}{r!}\frac{1}{\lambda} = \frac{\Gamma(r)}{r!} = \frac{1}{r}$$

We need to find the *lambda* for maximum likelihood. By differentiate the posterior probability distribution function with respect to λ, it has a maximum at $\lambda = \lambda_0 = r - 1$, and

$$P(\lambda = \lambda_0 = r - 1|r) = \frac{(r-1)^{(r-1)}\exp(r-1)}{(r-1)!}$$

$$c = -\frac{\partial^2}{\partial\lambda^2}\log \ P(\lambda|r)|_{\lambda=\lambda_0=r-1} = \frac{1-r}{\lambda^2}|_{\lambda=\lambda_0=r-1} = \frac{1}{r-1}$$

Then, the Laplace approximation is

$$G(\lambda|r) = P(\lambda = \lambda_0 = r - 1|r)\exp\left(-\frac{c}{2}(\lambda - \lambda_0)^2\right)$$

3.1.2 Exercise 27.3

Bayesian for posterior of ω_0, ω_1. $y(x_n)$ is the mean of t_n data points.

$$P(\{\omega_i\}|t_n, x_n, \sigma_\nu) = \frac{P(t_n|\{\omega_i\}, x_n, \sigma_\nu)P(\omega_0)P(\omega_1)}{P(t_n|x_n, \sigma_\nu)}$$

$P(t_n|\{\omega_i\}, \sigma_\nu)$, $P(\omega_0)$, $P(\omega_1)$ are all Gaussian distributed. Prior $P(\omega_0)$, $P(\omega_1)$ are assumed to be Gaussian.

$$P(t_n|\sigma_\nu) = \int_{-\infty}^{\infty} d\omega_0 \int_{-\infty}^{\infty} d\omega_1 \ P(t_n|\{\omega_i\}, \sigma_\nu)P(\omega_0)P(\omega_1)$$

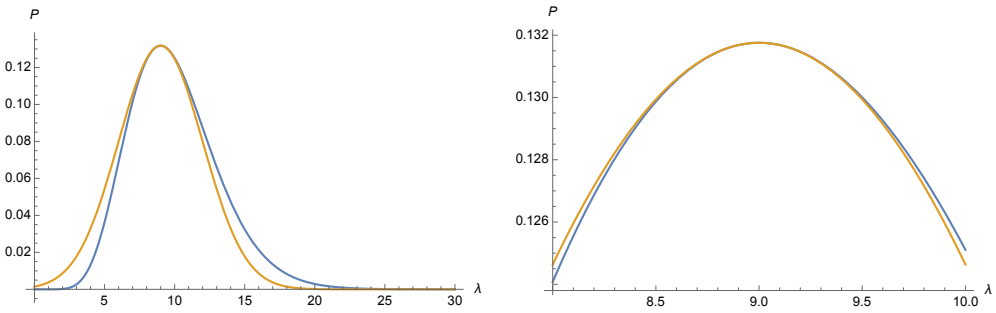

Figure 3: $r = 10$ is set. The blue plot is Poisson distribution and the orange one is the Laplace approximation.

$$P(\{t_n\}|\{\omega_i\}, x_n, \sigma) = (\frac{1}{2\pi\sigma_\nu^2})^{N/2} \exp\left(-\sum_n \frac{(t_n - y(x_n))^2}{2\sigma^2}\right)$$

$$y(x_n) = \omega_0 + \omega_1 x_n$$

$$P(\omega_i) = (\frac{1}{2\pi\sigma^2})^{1/2} \exp\left(-\frac{\omega_i^2}{2\sigma_i^2}\right)$$

Because the mean of ω_i should be zero obviously, and the variances of ω_0 and ω_1, i.e. σ_1, σ_2 could be same without loss of generality. The variance of t_n would have a very broad range of values dependent on x_n. I assumed $\sigma_i^2 = \sigma/3$ at Figure 4.

To use Laplace approximation, find $\{\omega_i\}$ satisfy $\frac{\partial}{\partial\omega_i} \log P(\{\omega_i\}|t_n, x_n, \sigma)$. [1]

$$\log P(\{\omega_i\}|t_n, x_n, \sigma) = -\left(\sum_n \frac{(t_n - \omega_0 - \omega_1 x_n)^2}{2\sigma^2}\right) - \frac{\omega_0^2}{2\sigma_0^2} - \frac{\omega_1^2}{2\sigma_1^2} \quad (3.1)$$

Some reader might notice that $-\sum_n \frac{(t_n-\omega_0-\omega_1 x_n)^2}{2\sigma^2}$ term is minimized for the linear regression. We have two extra terms from the priors. It is *Bayesian linear regression*.

By the way, ω_0^* and ω_1^* satisfy Eq. 3.1 is quite complicated. I tried to find the simplified form, but can't. I would just keep symbolic expressions of ω_0^*, ω_1^*.

Using the Eq. (27.6) of the book,

$$\mathbf{A} = \frac{1}{\sigma^2}\begin{pmatrix} N+1 & -\sum_n x_n \\ -\sum_n x_n & (\sum_n x_n)^2 + 1 \end{pmatrix}$$

[1] Do you have any better suggestion?

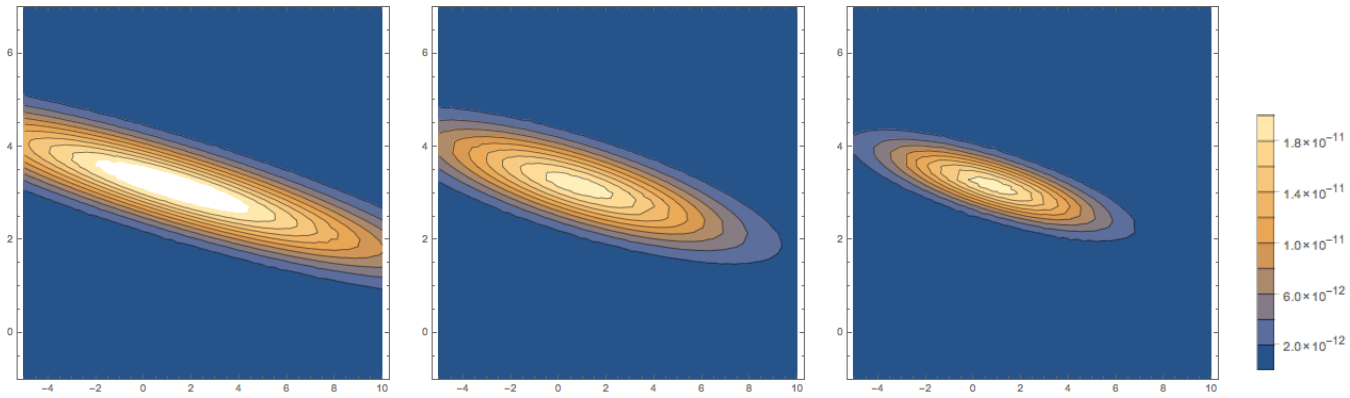

Figure 4: The likelihood and posterior probability distribution plot and its Laplace approximation. I have put 10 random data points x_n and t_n $3x_n + 2 \pm 2$. The left plot has the maximum likelihood around $(\omega_0, \omega_2) = (1.79794, 3.02217)$, and the central posterior is about $(\omega_0, \omega_2) = (0.812961, 3.16977)$. The discrepancy occurs by the assumed priors. Because the mean of ω_i should be zero obviously, and the variances of ω_0 and ω_1, i.e. σ_1, σ_2 could be same without loss of generality. The variance of t_n would have a very broad range of values dependent on x_n. I assumed $\sigma_i^2 = \sigma/3$. It is chosen from base of the mean value of x_n. The last one is obtained by Laplace approximation. I did not normalize them.

Then, the Laplace approximation is

$$G(\{\omega_i\}|t_n) = P(\{\omega_i^*\}|t_n, x_n, \sigma) exp\left(-\frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}_0)^T \mathbf{A}(\boldsymbol{\omega} - \boldsymbol{\omega}_0)\right)$$

where $\boldsymbol{\omega} = (\omega_0, \omega_1)$ and $\boldsymbol{\omega}^* = (\omega_0^*, \omega_1^*)$

3.2 Occam's razor

This section is very well summarized at the figures of the book. Occam's factor is defined in the process of model comparison. The Occam's factor $\frac{\sigma_{\omega|D}}{\sigma_\omega}$ is equal to the ratio of the posterior accessible volume of $\mathcal{H}_i$'s parameter space to the prior accessible volume, or the factor by which $\mathcal{H}_i$'s hypothesis space collapses when the data arrive. The factor is smaller as the model is simple. The logarithm of the Occam factor is a measure of the amount of information we gain about the model's parameters when the data arrive.

Occam factor for several parameters If the posterior is well approximated by a Gaussian, then the Occam factor is obtained from the determinant of the corresponding covariance matrix (cf. equation (28.8) and Chapter 27):

$$P(D|\mathcal{H}_i) \sim P(D|\mathbf{w}_{MP}, \mathcal{H}_i) \times P(\mathbf{w}_{MP}|\mathcal{H}_i) \det(\mathbf{A}/2\pi)^{-1/2}$$

$$\text{Evidence} \sim \text{Best fit likelihood} \times \text{Occam factor}$$

where $\mathbf{A} = -\Delta\Delta \log P(D|\mathbf{w}_{MP}, \mathcal{H}_i)$. Bayesian model comparison is a simple extension of maximum likelihood model selection: the evidence is obtained by multiplying the best-fit likelihood by the Occam factor. To evaluate the Occam factor we need only the Hessian $\mathbf{A}$, if the Gaussian approximation is good. Thus the Bayesian method of model comparison by evaluating the evidence is no more computationally demanding than the task of finding for each model the best-fit parameters and their error bars.

3.2.1 Exercise 28.1

For given $x = 0.3, 0.5, 0.7, 0.8, 0.9$, which hypothesis would be more probable? By intuition, $\mathcal{H}_0$, because the data looks rather uniformly distributed.

$P(D|\mathcal{H}_0)$ is $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = 0.03125$ since the probability distribution is uniform.

$P(D|\mathcal{H}_1)$ is a product of $\frac{1}{2}(1 - 0.4x)$ for $x = 0.3, 0.5, 0.7, 0.8, 0.9$. It is about 0.00689357. The ratio $\frac{P(D|\mathcal{H}_0)}{P(D|\mathcal{H}_1)} \sim 4.53321$. $P(D|\mathcal{H}_0)$ is 4.5 times more probable.

3.2.2 Exercise 28.2

Look at the data points plot.

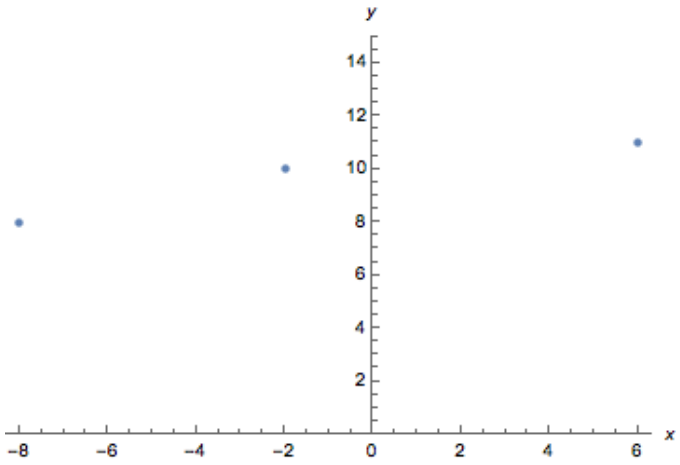

Figure 5:

The best fit would be almost flat but the slope is not zero, and the y-intersection would be around 10. Which hypothesis is more probable by your intuition?

The given data points,D= (x, t)= {(-8, 8), (-2, 10), (6, 11)}; The evidence is

$$P(D|\sigma_\nu, \sigma_{\omega_0}, \sigma_{\omega_1}, \mathcal{H}_i) = \int_{-\infty}^{\infty} d\omega_0 \int_{-\infty}^{\infty} d\omega_1 P(D|\omega_0, \omega_1, \sigma_\nu, \mathcal{H}_i) P(\omega_0|\sigma_{\omega_0}, \sigma_{\omega_1}, \mathcal{H}_i) P(\omega_1|\sigma_{\omega_0}, \sigma_{\omega_1}, \mathcal{H}_i)$$

For $\mathcal{H}_2$,

$$P(D|\sigma_\nu, \sigma_{\omega_0}, \sigma_{\omega_1}, \mathcal{H}_2) = \int_{-\infty}^{\infty} d\omega_0 \int_{-\infty}^{\infty} d\omega_1 \frac{1}{\sqrt{2\pi}\sigma_\nu} \exp\left(-\frac{(\omega_0 + \omega_1 x - t)^2}{2\sigma_\nu^2}\right) \frac{1}{\sqrt{2\pi}\sigma_{\omega_0}} \exp\left(-\frac{(\omega_0 - \mu_{\omega_0})^2}{2\sigma_{\omega_0}^2}\right) \frac{1}{\sqrt{2\pi}\sigma_{\omega_1}} \exp\left(-\frac{(\omega_1 - \mu_{\omega_1})^2}{2\sigma_{\omega_1}^2}\right)$$

Then, the evidence for $\mathcal{H}_2$ with respect to 3 data points are about 0.0302393, 0.0000391484, 0.0131697 for each. The total evidence is about 1.55905×10^{-8}.

For $\mathcal{H}_1$, $\omega_1 = 0$ is fixed, so we do not have related term.

$$P(D|\sigma_\nu, \sigma_{\omega_0}, \sigma_{\omega_1}, \mathcal{H}_1) = \int_{-\infty}^{\infty} d\omega_0 \int_{-\infty}^{\infty} d\omega_1 \frac{1}{\sqrt{2\pi}\sigma_\nu} \exp\left(-\frac{(\omega_0 + \omega_1 x - t)^2}{2\sigma_\nu^2}\right) \frac{1}{\sqrt{2\pi}\sigma_{\omega_0}} \exp\left(-\frac{(\omega_0 - \mu_{\omega_0})^2}{2\sigma_{\omega_0}^2}\right)$$

The evidence for $\mathcal{H}_1$ with respect to 3 data points are about 3.17456×10^{-8}, 3.91772×10^{-12}, 2.05583×10^{-14} for each. The total evidence is about 2.55684×10^{-33}.

Therefore, the horizontal line has much smaller evidence, and it agrees with the intuition.

4 Monte Carlo method

Where the likelihood function is multimodal, and has nasty unboundedly-high spikes in certain locations in the parameter space; so maximizing the posterior probability and

fitting a Gaussian is not always going to work. This difficulty with Laplace's method is one motivation for being interested in Monte Carlo methods. In fact, Monte Carlo methods provide a general-purpose set of tools with applications in Bayesian data modelling and many other

fields.

The book uses the word *sample* in the following sense: a sample from a distribution $P(\mathbf{x})$ is a single realization $\mathbf{x}$ whose probability distribution is $P(\mathbf{x})$. This contrasts with the alternative usage in statistics, where *sample* refers to a collection of realizations $\{\mathbf{x}\}$.

The aims of Monte Carlo methods are to solve one or both of the following problems.

Problem 1: to generate samples $\{x^{(r)}\}_{r=1}^{R}$ from a given probability distribution $P(x)$.

Problem 2: to estimate expectations of functions under this distribution, for example

$$\Phi = \int d^N \mathbf{x} P(\mathbf{x}) \phi(\mathbf{x})$$

Estimator

$$\hat{\Phi} = \frac{1}{R} \sum_r \phi(\mathbf{x}^{(r)})$$

If the vectors $\{x^{(r)}\}_{r=1}^R$ are generated from $P(x)$ then the expectation of $\hat{\Phi}$ is Φ, and the variance of $\hat{\Phi}$ is the variance,σ^2, of ϕ divided by R. [2]

$$\sigma^2 = \int d^N \mathbf{x} P(\mathbf{x}) (\phi(\mathbf{x}) - \Phi)^2$$

We can evaluate $P^*(\mathbf{x})$.

$$P(\mathbf{x}) = P^*(\mathbf{x})/Z$$

We do not know Z. Secondly, even if we did know Z, the problem of drawing samples from $P(\mathbf{x})$ is still a challenging one, especially in high-dimensional spaces, because there is no obvious way to sample from P without enumerating most or all of the possible states. Correct samples from P will by definition tend to come from places in $\mathbf{x}$-space where $P(\mathbf{x})$ is big; how can we identify those places where $P(\mathbf{x})$ is big, without evaluating $P(\mathbf{x})$ everywhere? There are only a few high-dimensional densities from which it is easy to draw samples, for example the Gaussian distribution.

4.1 Exercise 29.3

Problem : Consider a one-dimensional random walk, on each step of which the state moves randomly to the left or to the right with equal probability. Show that after N steps of size a, the state is likely to have moved only a distance about $\sqrt{N}a$. (Compute the root mean square distance travelled.)

Answer : Consider only right steps. p = 1/2. Then the average distance is $N\ p\ a = N\ a/2$ and the deviation is $\sqrt{Np(1-p)}a = \sqrt{N}a/2$. While consider the left step as well, the average is 0, and the deviation is between $\sqrt{N}a$ and zero.

4.2 Rejection sampling

We assume that we have a simpler proposal density $Q(x)$ which we can evaluate (within a multiplicative factor Z_Q, as before), and from which we can generate samples. We further assume that we know the value of a constant c such that

$$cQ^*(x) > P^*(x), \ for\ all\ x.$$

[2] Look at **this** for a simple example.

We generate two random numbers. The first, x, is generated from the proposal density $Q(x)$. We then evaluate $cQ^*(x)$ and generate a uniformly distributed random variable u from the interval $[0, cQ^*(x)]$. These two random numbers can be viewed as selecting a point in the two-dimensional plane.

We now evaluate $P^*(x)$ and accept or reject the sample x by comparing the value of u with the value of $P^*(x)$. If $u > P^*(x)$ then x is rejected; otherwise it is accepted, which means that we add x to our set of samples $\{x^{(r)}\}$. The value of u is discarded.

The reject sampling ipython notebook present the python code of the sampling. The bald is linked to ipython notebook in the GitHub. In the example, $Q^*(x)$ is trivially given as a constant, and $P^*(x)$ is given as a specific function.

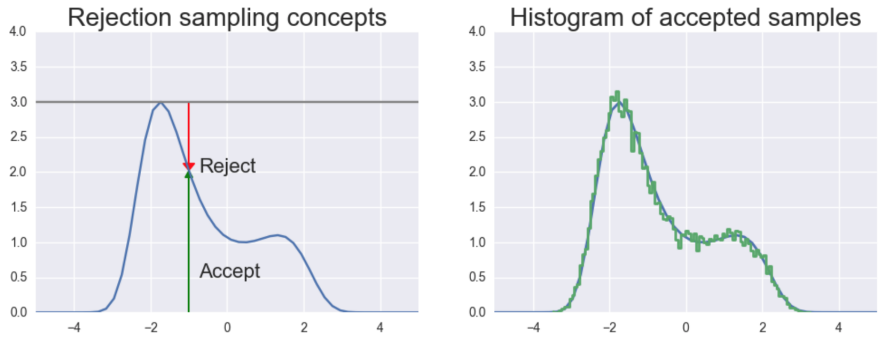

Figure 6: Reject sampling. $cQ^*(x)$ is a constant which is a maximum value of the function, $P^*(x)$.

The acceptance rate is the ratio of the volume under the curve $P(x)$ to the volume under $cQ(x)$, the fact that P and Q are both normalized here implies that the acceptance rate will be $1/c$. If it is N-dimensional problem, the acceptance rate becomes $(1/c)^N$

Rejection sampling, therefore, whilst a useful method for one-dimensional problems, is not expected to be a practical technique for generating samples from high-dimensional distributions $P(x)$.

4.3 The Metropolis-Hastings method

The Metropolis-Hastings algorithm instead makes use of a proposal density Q which depends on the current state $x^{(t)}$. The density $Q(x', x^{(t)})$ might be a simple distribution such as a Gaussian centred on the current $x^{(t)}$. The proposal density $Q(x', x^{(t)})$ can be any fixed density from which we can draw samples. In contrast to importance sampling and rejection sampling, it is not necessary that $Q(x', x^{(t)})$ look at all similar to $P(x)$ in order for the algorithm to be practically useful.

We assume that we can evaluate $P^*(x)$ for any x. A tentative new state x' is generated from the proposal density $Q(x', x^{(t)})$. To decide whether to accept the new state, we compute the quantity

$$a = \frac{P^*(x)\ Q(x^{(t)}, x')}{P^*(x^{(t)})\ Q(x', x^{(t)})}$$

The algorithm is

- If $a \leq 1$ then the new state is accepted.
- Otherwise, the new state is accepted with probability a.
- If the step is accetpted, we set $x^{(t+1)} = x'$.
- If the step is rejected, then we set $x^{(t+1)} = x^{(t)}$ (The rejected points are not discarded.)

If the proposal density is a simple symmetrical density such as a Gaussian centred on the current point, then the factor $\frac{Q(x^{(t)},x')}{Q(x',x^{(t)})}$ is unity, and the Metropolis-Hastings method simply involves comparing the value of the target density at the two points. *i.e.* $a = \frac{P^*(x)}{P^*(x^{(t)})}$.

Rule of thumb: lower bound on number of iterations of a Metropolis method. If the largest length scale of the space of probable states is L, a Metropolis method whose proposal distribution generates a random walk with step size ϵ must be run for at least $T \sim (L/\epsilon)^2$ iterations to obtain an independent sample. This rule of thumb gives only a lower bound; the situation may be much worse, if, for example, the probability distribution consists of several islands of high probability separated by regions of low probability.

It is good news because, unlike the cases of rejection sampling and importance sampling, there is no catastrophic dependence on the dimensionality N in MH. Roughly, $T \sim (\sigma_{Max}/\sigma_{min})^2$ from the rule of thumb. Because this quadratic dependence on the lengthscale-ratio may still force us to make very lengthy simulations. We will discuss how to suppress the random works in Monte Carlo simulations later.

4.4 Gibbs sampling

In the general case of a system with K variables, a single iteration involves sampling one parameter at a time.

$$x_1^{(t+1)} \sim P(x_1 | x_2^{(t)}, x_3^{(t)}, x_4^{(t)}, ..., x_K^{(t)}),$$

$$x_2^{(t+1)} \sim P(x_2 | x_1^{(t)}, x_3^{(t)}, x_4^{(t)}, ..., x_K^{(t)}),$$

$$\cdots$$

Gibbs sampling suffers from the same defect as simple Metropolis algorithms - the state space is explored by a slow random walk, unless a fortuitous parameterization has been chosen that makes the probability distribution $P(x)$ separable. It slowly progresses made by Gibbs sampling when $L \gg \epsilon$. However Gibbs sampling involves no adjustable parameters, so it is an attractive strategy when one wants to get a model running quickly.

4.5 Markov chain

We denote by $p^{(t)}(\mathbf{x})$ the probability distribution of the state of a Markov chain simulator. Our aim is to find a Markov chain such that as $t \to \infty$, $p^{(t)}(\mathbf{x})$ tends to the desired distribution $P(x)$.

A Markov chain can be specified by an initial probability distribution $p^{(0)}(\mathbf{x})$ and a transition probability $T(\mathbf{x}'; \mathbf{x})$.

Then,

$$p^{(t+1)}(\mathbf{x}) = \int d^N\mathbf{x}\; T(\mathbf{x}'; \mathbf{x}) p^{(t)}(\mathbf{x})$$

Look at the link for a simple example of Markov chain and Metropolis demonstration of section 29.4 of the book of David Mackay.

The Markov chain must have the following properties.

1. The desired distribution P(x) is an invariant distribution of the chain's transition. An invariant distribution is an eigenvector of the transition probability matrix that has eigenvalue 1.

2. The chain must also be ergodic, that is, $p^{(t)}(\mathbf{x}) \to \pi(\mathbf{x})$ as $t \to \infty$, for any initial $p^{(0)}(\mathbf{x})$.

A chain might not be ergodic when the matrix are reducible, or the chain is a periodic set.

4.6 Exercise 29.7

Detailed balance condition.

$$T(\mathbf{x}_a; \mathbf{x}_b) P(\mathbf{x}_b) = T(\mathbf{x}_b; \mathbf{x}_a) P(\mathbf{x}_a),$$

for all $\mathbf{x}_a$ and $\mathbf{x}_b$. Then,

$$\int d^N x_b\; T(\mathbf{x}_a; \mathbf{x}_b) P(\mathbf{x}_b) = \int d^N x_b\; T(\mathbf{x}_b; \mathbf{x}_a) P(\mathbf{x}_a),$$

for all $\mathbf{x}_a$ and $\mathbf{x}_b$. Then, using (29.45) of the book,

$$\int d^N x_b\, d^N x_c\; B(\mathbf{x}_a; \mathbf{x}_c) B(\mathbf{x}_c; \mathbf{x}_b) P(\mathbf{x}_b) = \int d^N x_b\, d^N x_c\; B(\mathbf{x}_a; \mathbf{x}_c) B(\mathbf{x}_c; \mathbf{x}_a) P(\mathbf{x}_a),$$

for all $\mathbf{x}_a$.

Both the left hand and the right hand side are $P(x_a)$ for all $\mathbf{x}_a$.

4.7 Slice sampling

A single transition $(x, u) \rightarrow (x', u')$ of a one-dimensional slice sampling algorithm has the following steps, of which steps 3 and 8 will require further elaboration.

1. evaluate $P^*(x)$
2. draw a vertical coordinate $u' \sim \text{Uniform}(0, P^*(x))$
3. create a horizontal interval (x_l, x_r) enclosing x
   3a. draw $r \sim \text{Uniform}(0, 1)$
   3b. $x_l := x - rw$
   3c. $x_r := x + (1 - r)w$
   3d. while $(P^*(x_l) > u')$ $\{x_l := x - rw\}$
   3e. while $(P^*(x_r) > u')$ $\{x_r := x + w\}$
4. loop {
5. draw $x' \sim \text{Uniform}(x_l, x_r)$
6. evaluate $P^*(x')$
7. if $P^*(x') > u'$ break out of loop 4-9
8. else modify the interval (x_l, x_r)
   8a. if $(x' > x)\{x_r := x'\}$
   8b. else $\{x_l := x'\}$
9. }

Like a standard Metropolis method, slice sampling gets around by a random walk, but whereas in the Metropolis method, the choice of the step size is critical to the rate of progress, in slice sampling the step size is self-tuning. If the initial interval size w is too small by a factor f compared with the width of the probable region then the stepping-out procedure expands the interval size. The cost of this stepping-out is only linear in f, whereas in the Metropolis method the computer-time scales as the square of f if the step size is too small.

4.7.1 Exercise 29.10

Problem : Investigate the properties of slice sampling applied to the density function $P^*(x)$.

$$P^*(x) = \{ \begin{array}{ll} 10 & as\ 0 \leq x < 1 \\ 0 & as\ 1 \leq x \leq 11 \end{array}$$

x is a real variable between 0.0 and 11.0. How long does it take typically for slice sampling to get from an x in the peak region $x \in (0,\ 1)$ to an x in the tail region $x \in (1,\ 11)$, and vice versa? Confirm that the probabilities of these transitions do yield an asymptotic probability density that is correct.

Answer : I have explained the answer here(click it).

4.8 Practicalities

Can we predict how long a Markov chain Monte Carlo simulation will take to equilibrate?

By considering the random walks involved in a Markov chain Monte Carlo simulation we can obtain simple lower bounds on the time required for convergence. But predicting this time more precisely is a difficult problem, and most of the theoretical results giving upper bounds on the convergence time are of little practical use. The exact sampling methods will be discussed later and it offers a solution to this problem for certain Markov chains.

Can we diagnose or detect convergence in a running simulation?

This is also a difficult problem. There are a few practical tools available, but none of them is perfect (Cowles and Carlin, 1996).

Can we speed up the convergence time and time between independent samples of a Markov chain Monte Carlo method?

Here, there is good news, as described in the next chapter, which describes the Hamiltonian Monte Carlo method, overrelaxation, and simulated annealing.

Can the normalizing constant be evaluated?

If the target density $P(\mathbf{x})$ is given in the form of an unnormalized density $P(\mathbf{x})$ with $P(\mathbf{x}) = P^*(\mathbf{x})/Z$, the value of Z may well be of interest. Monte Carlo methods do not readily yield an estimate of this quantity, and it is an area of active research to find ways of evaluating it. Techniques for evaluating Z include:

1. Importance sampling (reviewed by Neal (1993b)) and annealed importance sampling (Neal, 1998).

2. 'Thermodynamic integration' during simulated annealing, the 'acceptance ratio' method, and 'umbrella sampling' (reviewed by Neal (1993b)).

3. 'Reversible jump Markov chain Monte Carlo' (Green, 1995).

One way of dealing with Z, however, may be to find a solution to one's task that does not require that Z be evaluated. In Bayesian data modelling one might be able to avoid the need to evaluate Z which would be important for model comparison by not having more than one model. Instead of using several models (differing in complexity, for example) and evaluating their relative posterior probabilities, one can make a single hierarchical model having, for example, various continuous hyperparameters which play a role similar to that played by the distinct models (Neal, 1996). In noting the possibility of not computing Z, I am not endorsing this approach. The normalizing constant Z is often the single most important number in the problem, and I think every effort should be devoted to calculating it.

5 Efficient Monte Carlo Methods

5.1 Summary of Monte Carlos methods

In high-dimensional problems the only satisfactory methods are those based on Markov chains, such as the Metropolis method, Gibbs sampling and slice sampling. Gibbs sampling is an attractive method because it has no adjustable parameters but its use is restricted to cases where samples can be generated from the conditional distributions. Slice sampling is attractive because, whilst it has step-length parameters, its performance is not very sensitive to their values.

Simple Metropolis algorithms and Gibbs sampling algorithms, although widely used, perform poorly because they explore the space by a slow random walk. The next chapter will discuss methods for speeding up Markov chain Monte Carlo simulations.

Slice sampling does not avoid random walk behaviour, but it automatically chooses the largest appropriate step size, thus reducing the bad effects of the random walk compared with, say, a Metropolis method with a tiny step size.

5.2 Hamiltonian Monte Carlo

The Hamiltonian Monte Carlo method is a Metropolis method, applicable to continuous state spaces, that makes use of gradient information to reduce random walk behaviour.

Using the Algorithm 30.1, figure 30.2 is reproduced in this ipython notebook(click it).

```
g = gradE ( x ) ;                  # set gradient using initial x
E = findE ( x ) ;                  # set objective function too

for l = 1:L                        # loop L times
   p = randn ( size(x) ) ;         # initial momentum is Normal(0,1)
   H = p' * p / 2 + E ;            # evaluate H(x,p)

   xnew = x ;  gnew = g ;
   for tau = 1:Tau                 # make Tau 'leapfrog' steps

      p = p - epsilon * gnew / 2 ; # make half-step in p
      xnew = xnew + epsilon * p ;  # make step in x
      gnew = gradE ( xnew ) ;      # find new gradient
      p = p - epsilon * gnew / 2 ; # make half-step in p

   endfor

   Enew = findE ( xnew ) ;         # find new value of H
   Hnew = p' * p / 2 + Enew ;
   dH = Hnew - H ;                 # Decide whether to accept

   if ( dH < 0 )                accept = 1 ;
   elseif ( rand() < exp(-dH) ) accept = 1 ;
   else                         accept = 0 ;
   endif

   if ( accept )
      g = gnew ;   x = xnew ;    E = Enew ;
   endif
endfor
```

Figure 7: Algorithm 30.1 of the book. Octave source code for the Hamiltonian Monte Carlo method.