Bayesian Machine Learning

Homework 2
Due: Thursday, October 7, 11:59 pm ET

Model Comparison, Occam's Razor, and the Laplace Approximation
(49 marks)

The *evidence* $p(\mathcal{D}|\mathcal{M})$, also known as the *marginal likelihood*, is the probability that if we were to randomly sample parameters $\boldsymbol{\theta}$ from $\mathcal{M}$ that we would create dataset $\mathcal{D}$:

$$p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\mathcal{M}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{M}) d\boldsymbol{\theta} \tag{1}$$

Simple models $\mathcal{M}$ can only generate a small number of datasets, but because the marginal likelihood must normalise, it will generate these datasets with high probability. Complex models can generate a wide range of datasets, but each with typically low probability. For a given dataset, the marginal likelihood will favour a model of more appropriate complexity, as illustrated in Figure 1.

1. (12 marks): Consider the Bayesian linear regression model,

$$y = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{z}) + \epsilon(\mathbf{x}) \tag{2}$$
$$\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma^2) \tag{3}$$
$$p(\mathbf{w}) = \mathcal{N}(0, \alpha^2 I) \tag{4}$$

   where the data $\mathcal{D}$ consist of N input-output pairs, $\{\mathbf{x}_i, y_i\}_{i=1}^N$, $\mathbf{w}$ is a set of linear weights which have a Gaussian prior with zero mean and covariance $\alpha^2 I$, $\mathbf{z}$ is a set of deterministic parameters of the basis functions $\boldsymbol{\phi}$, and σ^2 is the variance of additive Gaussian noise. Let $\mathbf{y} = (y_1, \ldots, y_N)^\top$ and $X = \{\mathbf{x}_i\}_{i=1}^N$.

   (a) (2 marks): Draw the directed graphical model corresponding to the joint distribution over all parameters.

   (b) (2 marks): Derive an expression for the log marginal likelihood $\log p(\mathbf{y}|\mathbf{z}, X, \alpha^2, \sigma^2)$ showing all relevant steps.

   (c) (8 marks): Derive expressions for the derivatives of this log marginal likelihood with respect to *hyperparameters* $\mathbf{z}$, α^2, and σ^2. You can make reference to matrix derivative identities.

   (a) Refer to figure 3.

   (b)

$$p(\mathbf{y}, \mathbf{z}, X, \alpha^2, \sigma^2) = p(\mathbf{y}|\alpha^2, \sigma^2, X, \mathbf{z}) p(\alpha^2) p(\sigma^2) p(X) \tag{5}$$
$$= \sum_{i=1}^{n} p(y|\alpha^2, \sigma^2, x_i, z) p(\alpha^2) p(\sigma^2) \tag{6}$$

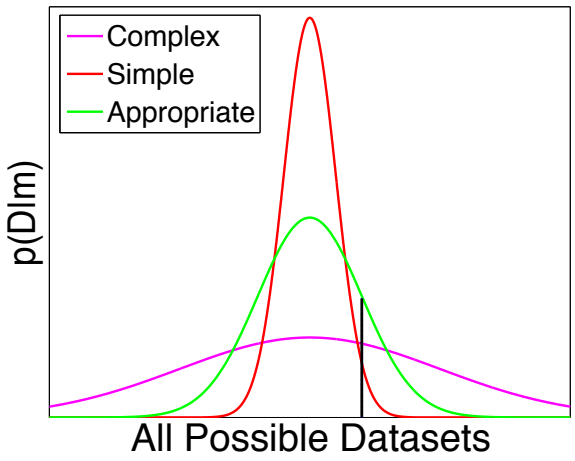


Figure 1: Bayesian Occam's Razor. The marginal likelihood (evidence) vs. all possible datasets $\mathcal{D}$. The vertical black line corresponds to an example dataset $\mathcal{D}$.

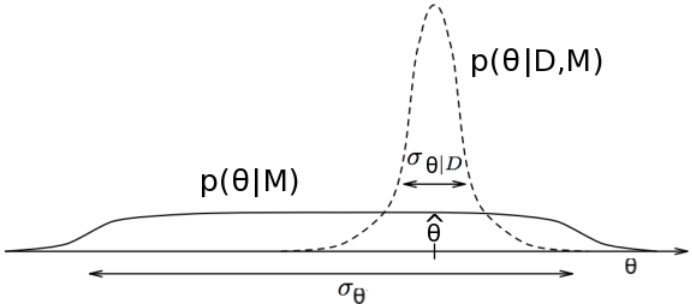


Figure 2: The posterior $p(\theta|\mathcal{D},\mathcal{M})$ and prior $p(\theta|\mathcal{M})$ over parameters θ under model $\mathcal{M}$. $\hat{\theta} = \text{argmax}_\theta p(\theta|\mathcal{D},\mathcal{M})$.

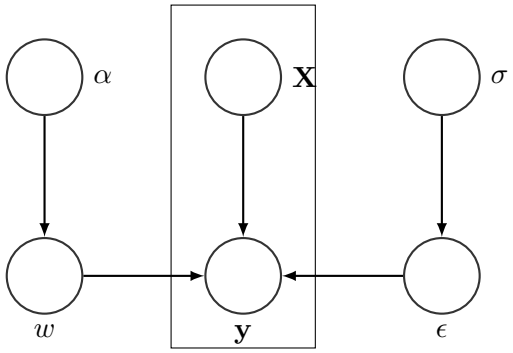


Figure 3: 1.(a)

We have,

$$y = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{z}) + \epsilon(\mathbf{x}) \tag{7}$$

w is the weight vector follow the gaussian distribution, $\phi(x, z)$ is a deterministic matrix, thus

$w^T\phi(x,z)$ is still gaussian, and $\epsilon(x)$ is a gaussian noise. y follows the gaussian distribution plus a gaussian distribution, thus y is a gaussian distribution. Trivially, the mean of $\mathbf{y}$ is 0 ($\mu = 0$).

i is the ith attribute of y.

$$
\begin{aligned}
cov(y_i, y_i) &= \mathbf{E}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z}) + \epsilon_i)(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_j}, \mathbf{z}) + \epsilon_i) && (8)\\
&= \mathbf{E}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z})\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z}) + \epsilon_i\epsilon_i) && (9)\\
&= \mathbf{E}(\boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z})^\top \mathbf{w}\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z}) + \epsilon_i\epsilon_i) && (10)\\
&= \boldsymbol{\phi_i}^\top \mathbf{E}(\mathbf{w}\mathbf{w}^\top)\boldsymbol{\phi_i} + \sigma^2 && (11)\\
&= \boldsymbol{\phi_i}^\top \alpha^2 I \boldsymbol{\phi_i} + \sigma^2 && (12)
\end{aligned}
$$

$$
\begin{aligned}
cov(y_i, y_j) &= \mathbf{E}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z}) + \epsilon_i)(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_j}, \mathbf{z}) + \epsilon_j) && (13)\\
&= \mathbf{E}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z})\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_j}, \mathbf{z}) + \epsilon_i\epsilon_j) && (14)\\
&= \mathbf{E}(\boldsymbol{\phi}(\mathbf{x_i}, \mathbf{z})^\top \mathbf{w}\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x_j}, \mathbf{z}) + \epsilon_i\epsilon_j) && (15)\\
&= \boldsymbol{\phi_i}^\top \mathbf{E}(\mathbf{w}\mathbf{w}^\top)\boldsymbol{\phi_j} && (16)\\
&= \boldsymbol{\phi_i}^\top \alpha^2 I \boldsymbol{\phi_j} && (17)
\end{aligned}
$$

We have every item in the covariance matrix of y.

$$\Sigma_{ij} = cov(y_i, y_j)$$

$$K(\mathbf{X}, \mathbf{X}) = cov(\phi(x,z), \phi(x,z)) \tag{18}$$

$$\boldsymbol{\Sigma} = \alpha^2 K(\mathbf{X}, \mathbf{X}) + \sigma^2 \boldsymbol{I} \tag{19}$$

$$
\begin{aligned}
p(\mathbf{y}|\alpha^2, \sigma^2, X, \mathbf{z}) &= \frac{\exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^N|\boldsymbol{\Sigma}|}} && (20)\\
&= \frac{\exp\left(-\frac{1}{2}(\mathbf{y})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{y})\right)}{\sqrt{(2\pi)^N|\boldsymbol{\Sigma}|}} && (21)
\end{aligned}
$$

$$
\begin{aligned}
\ln p(\mathbf{y}|\alpha^2, \sigma^2, X, \mathbf{z}) &= \left(-\frac{1}{2}(\mathbf{y})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{y})\right) - \ln\sqrt{(2\pi)^N|\boldsymbol{\Sigma}|} && (22)\\
&= \left(-\frac{1}{2}(\mathbf{y})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{y})\right) - \frac{1}{2}\ln(2\pi)^N|\boldsymbol{\Sigma}| && (23)\\
&= \left(-\frac{1}{2}(\mathbf{y})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{y})\right) - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| && (24)
\end{aligned}
$$

Here we use ln for convenience, the only difference with log is the normalization const.

(c)

$$K(x,x) = \frac{\partial \mathbf{\Sigma}}{\partial \alpha^2} = cov(\phi(x,z), \phi(x,z)) \tag{25}$$

$$\begin{aligned}
\frac{\partial \ln p(\mathbf{y}|z, \mathbf{X}, \alpha^2, \sigma^2)}{\partial \alpha^2} &= -\frac{1}{2}\frac{\partial \mathbf{y}^{\mathrm{T}} \mathbf{\Sigma}^{-1} \mathbf{y}}{\partial \alpha^2} - \frac{1}{2}\frac{\partial \ln |\Sigma|}{\partial \alpha^2} && (26)\\
&= -\frac{1}{2}\mathbf{y}^{\top}\frac{\partial \Sigma^{-1}}{\partial \alpha^2}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial \alpha^2}\right) && (27)\\
&= \frac{1}{2}\mathbf{y}^{\top}\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial \alpha^2}\mathbf{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial \alpha^2}\right) && (28)\\
&= \frac{1}{2}\mathbf{y}^{\top}\mathbf{\Sigma}^{-1}K(X,X)\mathbf{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}K(X,X)\right) && (29)
\end{aligned}$$

similarly, we have

$$\frac{\partial \mathbf{\Sigma}}{\partial \sigma^2} = I \tag{30}$$

$$\begin{aligned}
\frac{\partial \ln p(\mathbf{y}|z, \mathbf{x}, \sigma^2, \sigma^2)}{\partial \sigma^2} &= -\frac{1}{2}\frac{\partial \mathbf{y}^{\mathrm{T}} \mathbf{\Sigma}^{-1} \mathbf{y}}{\partial \sigma^2} - \frac{1}{2}\frac{\partial \ln |\Sigma|}{\partial \sigma^2} && (31)\\
&= -\frac{1}{2}\mathbf{y}^{\top}\frac{\partial \Sigma^{-1}}{\partial \sigma^2}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial \sigma^2}\right) && (32)\\
&= \frac{1}{2}\mathbf{y}^{\top}\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial \sigma^2}\mathbf{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial \sigma^2}\right) && (33)\\
&= \frac{1}{2}\mathbf{y}^{\top}\mathbf{\Sigma}^{-1}I\mathbf{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}I\right) && (34)
\end{aligned}$$

In HW(1), If we define,

$$\frac{1}{S^2} = \frac{K(\mathbf{x},\mathbf{x})}{\sigma^2} + \frac{1}{\alpha^2} \tag{35}$$

$$\begin{aligned}
&\Sigma^{-1} \cdot \Sigma'\\
&= \Sigma_Y^{-1} \cdot \Sigma_Y'\\
&= \left(-\frac{S^2}{\sigma^4}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \frac{1}{\sigma^2}\mathbf{I}_n\right)\left(\alpha^2\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \sigma^2\mathbf{I}_n\right)\\
&= -\frac{S^2\alpha^2}{\sigma^4}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \frac{\alpha^2}{\sigma^2}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} - \frac{S^2}{\sigma^2}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \mathbf{I}_n\\
&= -\frac{S^2\alpha^2}{\sigma^4}\phi(\mathbf{x})\left(\frac{\sigma^2}{S^2} - \frac{\sigma^2}{\alpha^2}\right)\phi(\mathbf{x})^{\mathrm{T}} + \frac{\alpha^2}{\sigma^2}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} - \frac{S^2}{\sigma^2}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \mathbf{I}_n\\
&= -\left(\frac{\alpha^2}{\sigma^2} - \frac{S^2}{\sigma^2}\right)\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \frac{\alpha^2}{\sigma^2}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} - \frac{S^2}{\sigma^2}\phi(\mathbf{x})\phi(\mathbf{x})^{\mathrm{T}} + \mathbf{I}_n\\
&= \mathbf{I}_n
\end{aligned} \tag{36}$$

Thus, we have a method to get the inverse of Σ.

For z, we calculate each component of z_i, and later we concat those result into the one final vector.

$$zp = \frac{\partial \ln p(\mathbf{y}|z, \mathbf{X}, z_i, z_i)}{\partial z} \tag{37}$$

$$\frac{\partial \mathbf{\Sigma}}{\partial z_i} = \frac{\partial K(X, X)}{\partial z_i} \tag{38}$$

$$\frac{\partial \ln p(\mathbf{y}|z, \mathbf{x}, z_i, z_i)}{\partial z_i} = -\frac{1}{2}\frac{\partial \mathbf{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{y}}{\partial z_i} - \frac{1}{2}\frac{\partial \ln|\Sigma|}{\partial z_i} \tag{39}$$

$$= -\frac{1}{2}\mathbf{y}^{\top}\frac{\partial \Sigma^{-1}}{\partial z_i}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial z_i}\right) \tag{40}$$

$$= \frac{1}{2}\mathbf{y}^{\top}\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial z_i}\mathbf{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial \mathbf{\Sigma}}{\partial z_i}\right) \tag{41}$$

$$= \frac{1}{2}\mathbf{y}^{\top}\mathbf{\Sigma}^{-1}\frac{\partial K(\mathbf{x}, \mathbf{x})}{\partial z_i}\mathbf{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^{-1}\frac{\partial K(\mathbf{x}, \mathbf{x})}{\partial z_i}\right) \tag{42}$$

$\frac{\partial K(\mathbf{x},\mathbf{x})}{\partial z_i}$ equals to each item in $K(\mathbf{x}, \mathbf{x})$ derivative to each component z_i in z, and form the new matrix.

2. (14 marks): The posterior $p(\boldsymbol{\theta}|\mathcal{M}, \mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}, \boldsymbol{\theta})p(\boldsymbol{\theta})$ will often be sharply peaked around its maximum value, as in Figure 2. The evidence in Eq. (1) can thus be approximated by its height times its width $\sigma_{\theta|\mathcal{D}}$:

$$\overbrace{p(\mathcal{D}|\mathcal{M})}^{\text{evidence}} \approx \overbrace{p(\mathcal{D}|\hat{\boldsymbol{\theta}}, \mathcal{M})}^{\text{data fit}}\overbrace{p(\hat{\boldsymbol{\theta}}|\mathcal{M})\sigma_{\theta|\mathcal{D}}}^{\text{Occam factor}} \tag{43}$$

The evidence thus naturally compartmentalizes into data fit and Occam factor terms. Suppose for simplicity that the prior is uniform on a large interval such that $p(\hat{\boldsymbol{\theta}}|\mathcal{M}) = 1/\sigma_\theta$. The Occam's factor then becomes $\frac{\sigma_{\theta|\mathcal{D}}}{\sigma_\theta}$.

(a) (2 marks): Provide an interpretation of the Occam's factor $\frac{\sigma_{\theta|\mathcal{D}}}{\sigma_\theta}$, wrt Figure 1.

(b) (4 marks): Show that if we use Laplace's method to approximate the posterior $p(\boldsymbol{\theta}|\mathcal{M}, \mathcal{D})$ as a Gaussian, then the Occam's factor becomes $p(\hat{\boldsymbol{\theta}}|\mathcal{M})\det(\frac{A}{2pi})^{-1/2}$ where $A = -\nabla\nabla \log p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$. Use this expression to interpret each of the terms in the log marginal likelihood you derived for question 1(b).

(c) (6 marks): Derive an approximation for the log evidence $\log p(D|\mathcal{M})$ assuming a broad Gaussian prior distribution and iid observations, strictly in terms of the number of datapoints N, the number of parameters m (dimensionality of $\boldsymbol{\theta}$), and $\log p(D|\hat{\boldsymbol{\theta}})$. Show all of your work.

(d) (2 marks): Relate the Hessian A to the covariance matrix of a Gaussian prior over parameters.

(a) Occam factor is equal to the ratio of the posterior accessible volume of models' parameter space to the prior accessible volume. More specifically, it can represent the probability of some specific parameters times the probability of generate dataset given that parameters.

For the uniform simplification in the figure, the Occam becomes the ratio of posterior model parameter space over prior model parameter space.

Simple model will get a small shrink of parameter space, thus it will have a large occam factor.

Complex model will get a large shrink, thus it will have a small occam factor.

(b)

$$P\left(\theta \mid M, D\right) \propto P\left(D \mid \theta, M\right) P\left(\theta \mid M\right) \tag{44}$$

We name $Z(\theta) = P\left(\theta \mid M, D\right)$, which follows the gaussian format. We Taylor-expand the logarithm of $Z(\theta)$ around the peak:

$$\ln Z^*(\theta) \simeq \ln Z^*(\hat{\theta}) - \frac{1}{2}\left(\theta - \hat{\theta}\right)^\top \mathbf{A} \left(\theta - \hat{\theta}\right) + \cdots \tag{45}$$

Thus we have,

$$P\left(\theta \mid M, D\right) = Z(\theta) \tag{46}$$

$$\begin{aligned} P(D|M) = \text{normalization-term} &= Z^*\left(\hat{\theta}\right) \frac{1}{\sqrt{\det \frac{1}{2\pi}\mathbf{A}}} && (47)\\ &= Z^*\left(\hat{\theta}\right) \sqrt{\frac{(2\pi)^N}{\det \mathbf{A}}} && (48)\\ &= P\left(\hat{\theta} \mid M, D\right) \sqrt{\frac{(2\pi)^N}{\det \mathbf{A}}} && (49) \end{aligned}$$

$$\begin{aligned} \textit{evidence} = \text{normalization-term} &= P(\hat{\theta} \mid M, D)\det(\frac{A}{2\pi})^{-1/2} && (50)\\ &\propto P\left(D \mid \hat{\theta}, M\right) P\left(\hat{\theta} \mid M\right) \det(\frac{A}{2\pi})^{-1/2} && (51) \end{aligned}$$

Then occam' factor is $P\left(\hat{\theta} \mid M\right) \det(\frac{A}{2\pi})^{-1/2}$

(c) if the prior is gaussian, the posterior is gaussian.

$$\ln p(D|M) \approx \ln p(D|\hat{\theta}, M) + \ln P\left(\hat{\theta} \mid M\right) \det(\frac{A}{2\pi})^{-1/2} \tag{52}$$

$$\ln p(D \mid M) \approx \ln p(D \mid \hat{\theta}, \mathcal{M}) + \frac{m}{2} \ln 2\pi + \ln p(\hat{\theta} \mid \mathcal{M}) - \frac{1}{2} \ln \det(A) \tag{53}$$

We can use Fisher Information Matrix F to decompose A.

$$\ln p(D \mid M) \approx \ln p(D \mid \hat{\theta}, \mathcal{M}) + \frac{m}{2} \ln 2\pi + \ln p(\hat{\theta} \mid \mathcal{M}) - \frac{1}{2} \ln \det(NF) \tag{54}$$

$$\ln p(D \mid M) \approx \ln p(D \mid \hat{\theta}, \mathcal{M}) + \frac{m}{2} \ln 2\pi + \ln p(\hat{\theta} \mid \mathcal{M}) - \frac{1}{2} \ln \det(NF) \tag{55}$$

$$\ln p(D \mid M) \approx \ln p(D \mid \hat{\theta}, \mathcal{M}) + \frac{m}{2} \ln 2\pi + \ln p(\hat{\theta} \mid \mathcal{M}) - \frac{1}{2} \ln N^m \det(F) \tag{56}$$

$$\ln p(D \mid M) \approx \ln p(D \mid \hat{\theta}, \mathcal{M}) + \frac{m}{2} \ln 2\pi + \ln p(\hat{\theta} \mid \mathcal{M}) - \frac{m}{2} \ln N - \frac{1}{2} \ln \det(F) \tag{57}$$

With the increasing size N, $\ln p(\hat{\theta} \mid \mathcal{M})$ and other terms is relatively samll.

$$\ln p(D \mid M) \approx \ln p(D \mid \hat{\theta}, \mathcal{M}) - \frac{m}{2} \ln N \tag{58}$$

(d) We already know that the target function is gaussian, thus Hessian A is equal to the inverse of covariance matrix.

$$p(\boldsymbol{\theta}) = (2\pi)^{-\frac{N_{\theta}}{2}} |\boldsymbol{\Sigma}_{\boldsymbol{\theta}}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^{\star})^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}^{\star})\right] \tag{59}$$

$$J(\boldsymbol{\theta}) \equiv -\ln p(\boldsymbol{\theta}) = \frac{N_{\boldsymbol{\theta}}}{2} \ln 2\pi + \frac{1}{2} \ln |\boldsymbol{\Sigma}_{\boldsymbol{\theta}}| + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^{\star})^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}^{\star}) \tag{60}$$

$$\mathcal{H}^{(l,l')}(\boldsymbol{\theta}^{\star}) = \left.\frac{\partial^2 J(\boldsymbol{\theta})}{\partial \theta_l \partial \theta_{l'}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{\star}} = \left(\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}\right)^{(l,l')} \tag{61}$$

3. (33 marks): Load the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively from `occam1.mat` and `occam2.mat` in the assignment files `a2files.zip`. Suppose we are considering three models to explain the data:

   (i) $\mathcal{M}_1$: The Bayesian basis regression model of Eq.(2)-(4), but with

   $$\boldsymbol{\phi}(x, \mathbf{z}) = \boldsymbol{\phi}(x) = (1, x, x^2, x^3, x^4, x^5)^{\top} \tag{62}$$

   (ii) $\mathcal{M}_2$: The Bayesian basis regression model of Eq.(2)-(4), but with

   $$\boldsymbol{\phi}(x, \mathbf{z}) = \left(\exp[-\frac{(x-1)^2}{z_1^2}], \exp[-\frac{(x-5)^2}{z_2^2}]\right)^{\top}, \quad \mathbf{z} = (z_1, z_2)^{\top} \tag{63}$$

   (iii) $\mathcal{M}_3$: The Bayesian basis regression model of Eq.(2)-(4), but with

   $$\boldsymbol{\phi}(x, \mathbf{z}) = \boldsymbol{\phi}(x) = (x, \cos(2x))^{\top} \tag{64}$$

Parts of this question involve coding. Please hand in the Matlab, Octave, or Python code you used to solve this question, along with the plots of results generated by the code used to answer the questions. This code should be succinct and include comments. Your code should not exceed 5 pages in length. Answer all questions for both $\mathcal{D}_1$ and $\mathcal{D}_2$ unless the question explicitly states otherwise.

(a) (20 marks): Using your work from question 1, write code to plot a histogram of the evidence for each of these three models, conditioned on the maximum marginal likelihood values of all the hyperparameters $\mathbf{z}$, α^2, and σ^2. To find these values, jointly optimize the log marginal likelihood with respect to these hyperparameters using a quasi-Newton method or non-linear conjugate gradients. Based on the histogram, which hypotheses do you believe generated the data?
Hint 1: If you compute the Cholesky decomposition of $A = R^\top R$, where R is an upper right triangular matrix, then $A^{-1}\mathbf{b} = R^{-1}(R^{-1})^\top \mathbf{b}$. You can use this decomposition in conjunction with the helper function `solve_chol.m` for numerically stable solutions to linear systems involving A. Note also that $\log\det(A) = 2\sum_i \log(R_{ii})$.
Hint 2: Use the function `checkgrad.m`, or compute a finite difference scheme, to numerically check the derivatives of the log marginal likelihood with respect to the model hyperparameters, as a means to debug and to check your answer for 1(c).
Hint 3: Some of the relevant matrices may be very poorly conditioned. It is often useful to add a constant small amount of *jitter* to the diagonal of these matrices, $A \to A + \epsilon I$, where ϵ is on the order of 10^{-6}, before performing operations such as Cholesky decompositions. This procedure is advocated by Radford Neal in his 1996 PhD thesis, *Bayesian Learning for Neural Networks*.
Hint 4: You may wish to constrain your hyperparameters to be positive, but perform an unconstrained optimization. To do so, you can optimize over the log hyperparameters in an unconstrained space. When computing derivatives of the log marginal likelihood with respect to the log parameters, you will find it helpful to use the chain rule.
Hint 5: I have included `minimize.m`, a robust implementation of non-linear conjugate gradients you can use to optimize the marginal likelihood with respect to hyperparameters. You are, however, free to use another gradient based optimizer if you wish.

(b) (8 marks): Explain why the evidence histogram might disagree with the maximum likelihood ranking of models (in general and with respect to $\mathcal{D}_1$ and $\mathcal{D}_2$).

(c) (2 marks): Give the posterior mean and variance over the parameters of the model with highest evidence on $\mathcal{D}_2$.

(d) (3 marks): What values of the *hyperparameters* maximized the marginal likelihood?

(e) Optional (1 bonus mark): Plot the posterior over each set of model weights $\mathbf{w}$ (using extra coding space if required) for each dataset.

(a)

In [3]:
```
from sol_3a1 import *
import matplotlib.pyplot as plt
```

dataset1 result

	loss = NLL	alpha_hat	sigma_hat	Z_hat
mode11	80.94710727457094	1e-06	2742.907868619646	None
model2	80.34049376533933	54.674196526383206	2542.715511207578	[302.39512784 299.98729093]
model3	80.40741481452056	1e-06	2651.8960431941096	None

dataset2 result

	loss = NLL	alpha_hat	sigma_hat	Z_hat
mode11	533435.0080889168	22.274354563080006	2515.1732607200665	None
mode12	533488.2228744307	298.5601754764213	2519.525066776425	[300.13923787 299.94610773]
mode13	533218.2923134756	11.81695054460237	2505.7351321647266	None

Dataset 1

In [13]:
```
log_evidence1, evidence1 = compute_evidence(loss=80.94710727457094, alpha_hat=1e
```

```
-348.1763260173209 6.150959695998708e-152
```

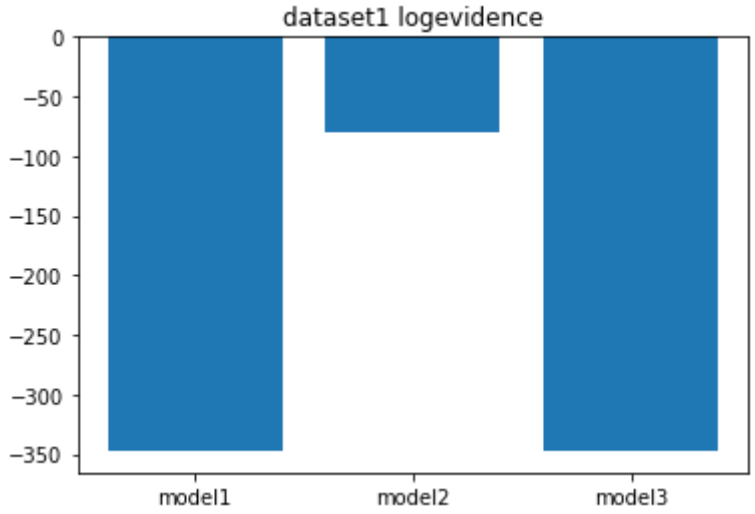
In [14]:
```
log_evidence2, evidence2 = compute_evidence(loss=80.34049376533933, alpha_hat=54
```

```
-79.37586775492639 3.368981589950336e-35
```

In [15]:
```
log_evidence3, evidence3 = compute_evidence(loss=80.40741481452056, alpha_hat=1e
```

```
-346.7627993877606 2.52830635668627e-151
```

In [21]:
```
data = [log_evidence1, log_evidence2, log_evidence3]
label = ["model1","model2","model3"]
plt.bar(range(len(data)), data, tick_label=label)
plt.title('dataset1 logevidence')
plt.show()
```



In [22]:
```
data = [evidence1, evidence2, evidence3]
label = ["model1","model2","model3"]
plt.bar(range(len(data)), data, tick_label=label)
plt.title('dataset1 evidence')
plt.show()
```

Dataset 2

In [6]:
```
x=np.array([float(xi) for xi in scio.loadmat('./hw2files/occam2.mat')['x']], dty
```

In [4]:
```
log_evidence1, evidence1 = compute_evidence(loss=533435.0080889168, alpha_hat=22
```

```
-614631.3936193986 0.0
```

In [13]:
```
log_evidence2, evidence2 = compute_evidence(loss=533488.2228744307, alpha_hat=29
```

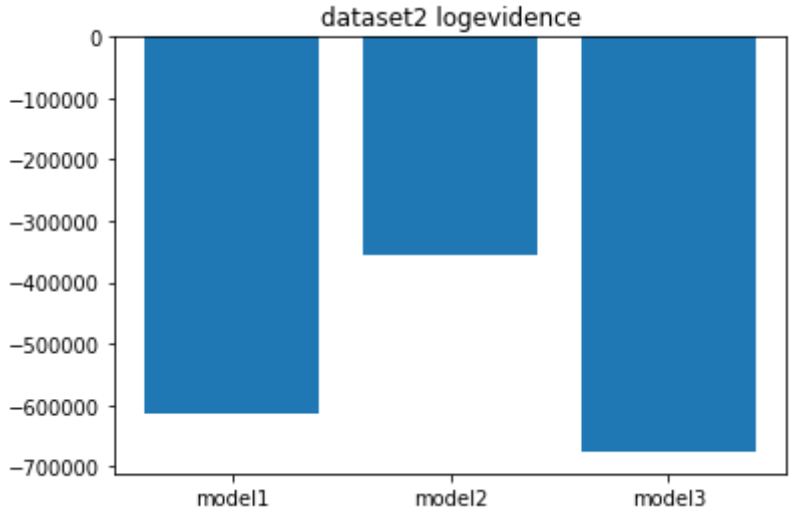
```
-355187.3938336382 0.0
```

In [14]:
```
# 533218.2923134756 11.81695054460237 2505.7351321647266 None
log_evidence3, evidence3 = compute_evidence(loss=533218.2923134756, alpha_hat=11
```

```
-677589.2313579913 0.0
```

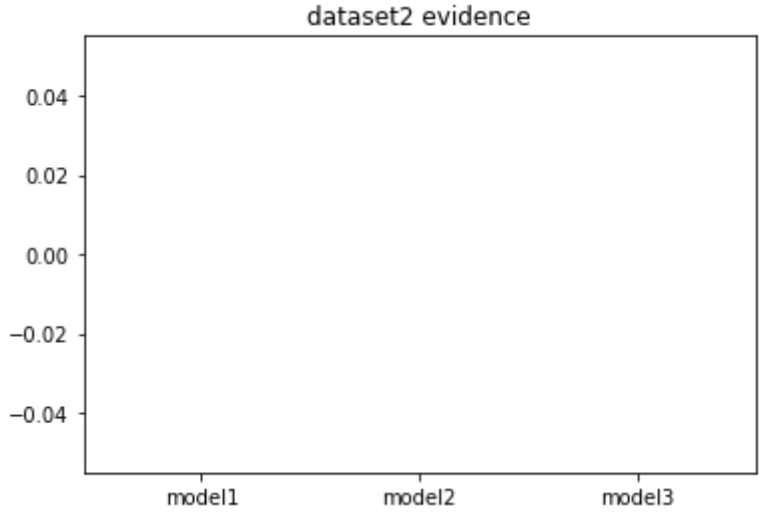
In [16]:

```
data = [log_evidence1, log_evidence2, log_evidence3]
label = ["model1","model2","model3"]
plt.bar(range(len(data)), data, tick_label=label)
plt.title('dataset2 logevidence')
plt.show()
```



In [17]:

```
data = [evidence1, evidence2, evidence3]
label = ["model1","model2","model3"]
plt.bar(range(len(data)), data, tick_label=label)
plt.title('dataset2 evidence')
plt.show()
```



Posterior

In [33]:

```
phi = phi2
alpha_hat = 298.5601754764213
sigma_hat = 2519.525066776425
z_hat = [300.13923787,299.94610773]
param_shape = 2
```

In [29]:
```
y=np.array([float(xi) for xi in scio.loadmat('./hw2files/occam2.mat')['y']], dty
```

In [28]:
```
phi_x = phi(x, z_hat)
```

In [26]:
```
m_0 = np.zeros(2)
print(m_0)
```

```
[0. 0.]
```

In [34]:
```
S_N = (1 / alpha_hat) * np.identity(param_shape) + (1 / sigma_hat) * phi_x.T @ p
m_N = np.linalg.inv(S_N) @ ((1 / alpha_hat) * np.identity(param_shape) @ m_0 + (
```

refer to bishop, we have the posterior update equation

we get posterior

mean = [-1.05609318 3.05200373]

variance = [[39.67203114 39.67567358] [39.67567358 39.68601867]]

In [35]:
```
print(m_N,S_N)
```

```
[-1.05609318  3.05200373] [[39.67203114 39.67567358]
 [39.67567358 39.68601867]]
```

In [ ]:
```

```

(b)

$$\text{Evidence} \simeq \text{Best fit likelihood} \times \text{Occam factor} \tag{65}$$

So the evidence rank maybe disagree with the maximum likelihood rank due to the effect of occam factor. A better fit model may have a small occam factor thus have a small evidence. A less fit model may have a larger occam factor which can lead to large evidence. Our target is to find a model that can balance between the best fit likelihood and occam factor, which is a "suitable" model in the statement above.

(c) Refer to the table above.

(d) Refer to the section posterior above.

Markov chain Monte Carlo
(15 marks)

Follow Iain Murray's MCMC practical at
`http://homepages.inf.ed.ac.uk/imurray2/teaching/09mlss/handout.pdf`

Complete section 4, and answer the MCMC questions in section 5.

Hand in (1) your code for section 4, and (2) your answers to the 5 MCMC questions. The code is worth 10 marks, and the questions are worth 1 mark each. There are thus 15 marks for this part of the assignment.

ENV Set

In [373…

```
import numpy as np
from scipy.io import loadmat
data = loadmat('astro_data.mat')
xx, vv = data['xx'], data['vv']

def norm(x, mean, std):
    return np.exp(-0.5 * ((x - mean) ** 2) / (std** 2)) / std

def log_pstar(state):
    log_omega, mm, pie, mu1, mu2, log_sigma1, log_sigma2 = state
    N = xx.shape[0]

    # exp process
    sigma1 = np.exp(log_sigma1)
    sigma2 = np.exp(log_sigma2)
    omega = np.exp(log_omega)

    x_mu = xx.mean()
    x_std = xx.std()
    ext = xx.max() - xx.min()

    log_ext = np.log(ext)

    # condition check
    forbidden_conditions = [pie < 0,
                            pie > 1,
                            np.abs(mm - x_mu) > 10 * x_std,
                            np.abs(mu1 - log_ext) > 20,
                            np.abs(mu2 - log_ext) > 20,
                            np.abs(log_sigma1) > np.log(20),
                            np.abs(log_sigma2) > np.log(20),
                            np.abs(log_omega) > 20,
                            ]

    if any(forbidden_conditions):
        return -np.inf

    log_A = 0.5 * np.log((xx - mm) ** 2 + (vv / omega) ** 2)

    log_prior = np.sum(np.log(pie * norm(log_A, mu1, sigma1) + (1 - pie) * norm(
    log_like = -2 * log_A.sum() - N * np.log(omega)
    logp = log_like + log_prior

    return logp


def dumb_metropolis(init, log_ptilde, iters, sigma, xx=xx, vv=vv):
    # init state
    state = init
    Logp_state = log_ptilde(state)

    # init all the samples
    param_shape = init.shape[0]
    samples = np.zeros((param_shape, iters))
```

```
    accept = 0
    for iter in range(0, iters):
        propose_state = state + sigma * np.random.randn(len(state))
        Logp_prop = log_ptilde(propose_state)
        if (np.log(np.random.rand()) < (Logp_prop - Logp_state)).all():
            state = propose_state        # accept propose param
            Logp_state = Logp_prop       # update state
            accept += 1
        samples[:, iter] = state.squeeze()
    accept_rate = accept / iters
    return (samples, accept_rate)
```

In [371…

```
from scipy.io import loadmat
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [58]:

```
log_omega = 1
mm = 1
pie = 1
mu1 = 1
mu2 = 1
log_sigma1 = 1
log_sigma2 = 1
# logp = log_pstar(log_omega, mm, pie, mu1, mu2, log_sigma1, log_sigma2, xx, vv)
params = np.array([log_omega, mm, pie ,mu1, mu2, log_sigma1, log_sigma2,])
```

4.1

What is the effect of Metropolis's step-size parameter?

step_size = 1

In [49]:

```
def plot_history(samples, acceptance_rate):
    params_name = ['log_omega', 'mm', 'pie', 'mu1', 'mu2', 'log_sigma1', 'log_si

    df = pd.DataFrame(samples.T, columns=params_name)

    df['iter'] = df.index
    lines = df.plot.line(x='iter', y=params_name)
```

In [50]:

```
samples, acceptance_rate = dumb_metropolis(params, log_pstar, 1000, sigma=1)
```

In [51]:

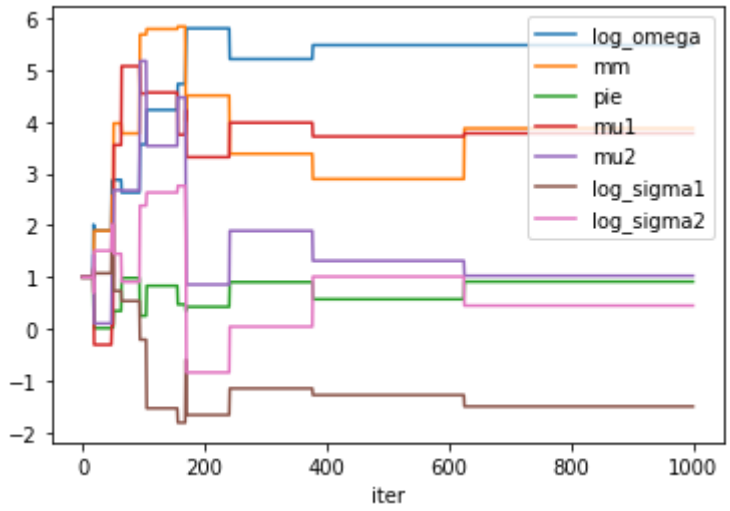
```
acceptance_rate
```

Out[51]:

```
0.013
```

In [52]:

```
plot_history(samples, acceptance_rate)
```



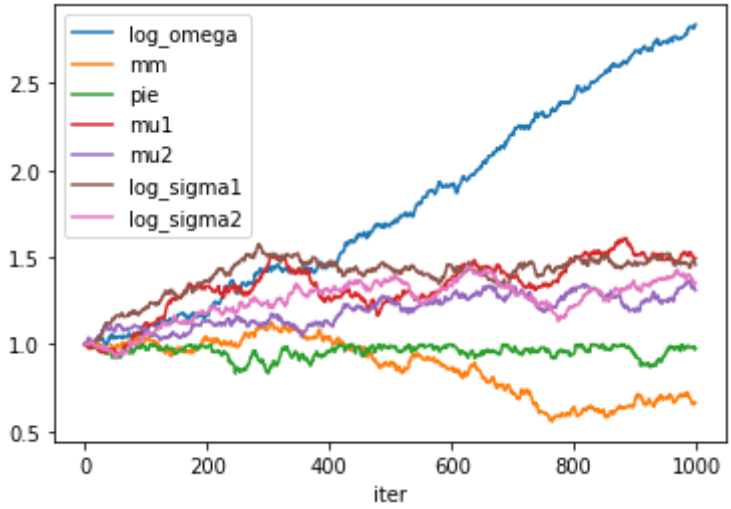
step_size = 0.01

In [53]:
```
samples, acceptance_rate = dumb_metropolis(params, log_pstar, 1000, sigma=0.01)
```

In [54]:
```
acceptance_rate
```

Out[54]: 0.777

In [55]:
```
plot_history(samples, acceptance_rate)
```



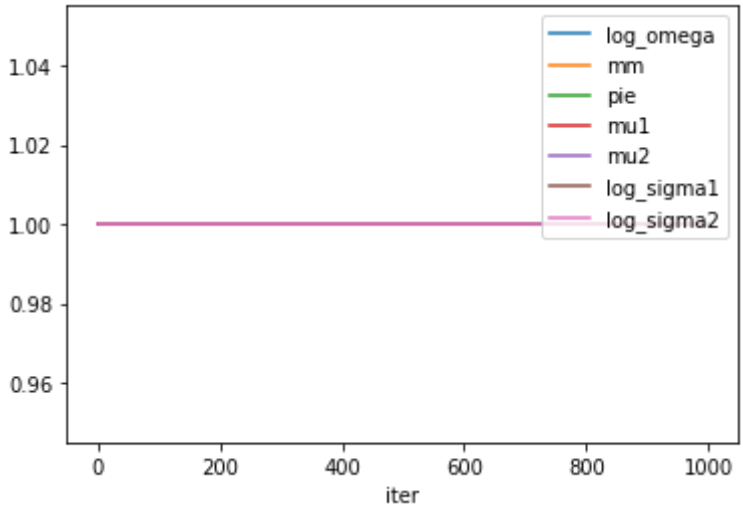
step_size = 10

In [46]:
```
samples, acceptance_rate = dumb_metropolis(params, log_pstar, 1000, sigma=10)
```

In [47]:
```
acceptance_rate
```

Out[47]: 0.0

In [48]:
```
plot_history(samples, acceptance_rate)
```



- Little step size will lead to small exploration range
- Too Large step size will crash the sampling, no sample accepted
- We need a suitable step size and burn-in time

4.2

Is the way you initialize the chain critical for Metropolis and/or slice sampling?

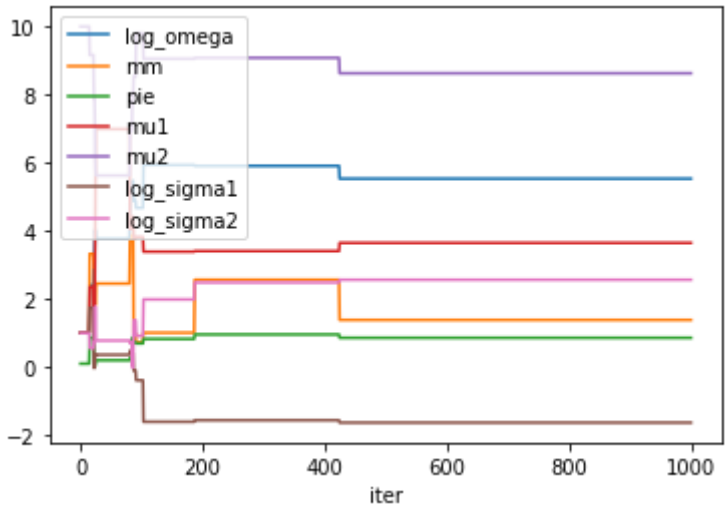
We just change our init

In [63]:
```
log_omega = 1
mm = 1
pie = 0.1
mu1 = 1
mu2 = 10
log_sigma1 = 1
log_sigma2 = 1
# logp = log_pstar(log_omega, mm, pie, mu1, mu2, log_sigma1, log_sigma2, xx, vv)
params = np.array([log_omega, mm, pie ,mu1, mu2, log_sigma1, log_sigma2,])
```

In [64]:
```
samples, acceptance_rate = dumb_metropolis(params, log_pstar, 1000, sigma=1)
acceptance_rate
```

Out[64]:
```
0.011
```

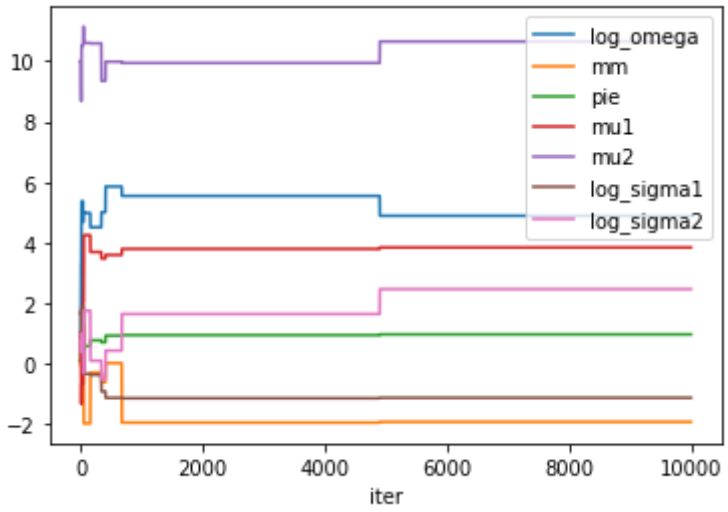
In [65]:
```
plot_history(samples, acceptance_rate)
```



In [66]:
```
samples, acceptance_rate = dumb_metropolis(params, log_pstar, 10000, sigma=1)
acceptance_rate
```

Out[66]: 0.0012

In [67]:
```
plot_history(samples, acceptance_rate)
```



We may need more burn-in time to make the sampling consistent.

4.3

What are the relative advantages of slice-sampling and Metropolis? Can you say good and bad things about both of them?

slice-sampling

pros: slice-sampling can be more efficient for not rejecting the samples, which is suitable for the case that we do not have much information about the distribution.

cons: slice-sampling may suffer from locality for those complicated distribuion. It is hard for slice-sampling to explore those far away high probability area across some low probability area.

Metropolis

pros: Metropolis is quite simple and can generally working in most cases.

cons: As we showed above, Metropolis may rely on its hyperparameters(step-size) and have burn-in period.

4.4

Why have I taken logs of quantities like ω and A? Need I have bothered? Does taking logs affect the model and/or the sampler?

Log can convert the multiply to addition, which in some sense can prevent the precision problem. The final result is not affected. (just like log likelihood)

4.5

The true values are ω = 875.2 and m= 31.79. Are your posterior beliefs consistent with this? If not, what do you think went wrong with the sampling, modelling or both?

In [316…
```
log_omega = 10
mm = 1
pie = 1
mu1 = 1
mu2 = 1
log_sigma1 = 0.1
log_sigma2 = 0.1
# logp = log_pstar(log_omega, mm, pie, mu1, mu2, log_sigma1, log_sigma2, xx, vv)
params = np.array([log_omega, mm, pie ,mu1, mu2, log_sigma1, log_sigma2,])
```

In [324…
```
samples, acceptance_rate = dumb_metropolis(params, log_pstar, 100000, sigma=0.08
acceptance_rate
```

Out[324… 0.12078

In [319…
```
ori_samples = samples
```

In [320…
```
samples = ori_samples
```

In [366…
```
# ori_samples = samples
print(samples.shape)
samples = samples.T
print(samples.shape)
# samples = samples.T
samples = samples[5000:]
```

```
samples = samples.T
print(samples.shape)
```

```
(7, 55000)
(55000, 7)
(7, 50000)
```

In [367…

```
params_name = ['log_omega', 'mm', 'pie', 'mu1', 'mu2', 'log_sigma1', 'log_sigma2
df = pd.DataFrame(samples.T, columns=params_name)
```

In [368…

```
df['mm'].mean()
```

Out[368…

```
31.874691194919823
```

In [369…

```
np.exp(df['log_omega'].mean())
```

Out[369…

```
873.151930476839
```

We have a burn-in = 50000, and total iter = 100000, we get similar posterior