Supervision Assignments in MLBI
Lent Term 2018
Set 1

Supervisor : Dionysis Manousakas
dm754@cam.ac.uk

May 22, 2019

1. (*Overfitting / Model evaluation*)

   i. Is a classifier trained on less training data less likely to overfit?

   answer

   More likely to overfit, since less data increase the probability that the classifier will start learning noise

   ii. Given m data points, does the training error converge to the true error as $m \to \infty$?

   answer

   Yes, if they are i.i.d.

   iii. You are a reviewer for an international Machine Learning conference. Would you accept or reject each paper? Provide a one sentence justification.

   - My algorithm is better than yours. Look at the training error rates!

   answer

   Reject - the training error is optimistically biased.

   - My algorithm is better than yours. Look at the test error rates! (Footnote: reported results for parameter $\lambda = 0.4724332783241324108 3478$.)

   answer

   Reject - A λ with 15 decimal places suggests a highly tuned solution, probably looking at the test data.

   - My algorithm is better than yours. Look at the test error rates! (Footnote: reported results for best value of λ.)

   answer

   Reject - Choosing λ based on the test data?

   - My algorithm is better than yours. Look at the test error rates! (Footnote: reported results for best value of λ, chosen with 10-fold cross validation.)

   answer

   Accept - Cross validation is the appropriate method for selecting parameters.

2. (*Bayes rule for medical diagnosis*)

After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a serious disease, and that the test is 99% accurate (i.e., the probability of testing positive given that you have the disease is 0.99, as is the probability of testing negative given that you don't have the disease). The good news is that this is a rare disease, striking only one in 10,000 people. What are the chances that you actually have the disease? Show your calculations as well as giving the final result. (Murphy's book Exercise 2.4)

answer

We apply directly the Bayes rule

$$\begin{aligned} &P(disease|test) = P(test|disease)P(disease)/P(test) \\ &= \frac{P(test|disease)P(disease)}{P(test|disease)P(disease) + P(test|not\,disease)P(not\,disease)} \\ &= 0.99 * 0.0001/(0.99 * 0.0001 + 0.01 * 0.9999) \approx 0.009804 \end{aligned} \tag{1}$$

3. (*Bayes rule)*)

i. Consider a classification problem with two classes and n binary attributes. How many parameters would you need to learn with a Naive Bayes classifier?

answer:

NB has $1 + 2n$ parameters :

- prior $P(y = T)$, and
- for every attribute x_i,we have $p(x_i = T|y_i = T)$ and $p(x_i = T|y_i = F)$.

ii. Suppose we have a set of k hypotheses (or models)

$$h = [h_1, h_2, ..., h_k]$$

for an observed dataset $\mathcal{D}$. According to the Bayes rule

$$p(h_i|\mathcal{D}) \propto p(\mathcal{D}|h_i)p(h_i)$$

for $i \in 1, 2, ...k$.

Write the most uninformed and most informed prior distributions we can assume over the given hypotheses' set. How can we make the most informed prior less rigid?

answer:

The least informed is assigning a uniform prior over the classes (we do not favor a priori any of the hypotheses and let the data speak for them), while the most rigid is using a delta (or dirac) prior, which assigns the entire weight of the prior to only one class (and zero to the rest of the classes, which implies zero posterior regardless of the likelihoods). This can become less rigid by smoothing, which will result to giving a minimum nonzero threshold to the prior probabilities of each one of the classes.

4. (*The Monty Hall problem*)

On a game show, a contestant is told the rules as follows: There are three doors, labelled 1, 2, 3. A single prize has been hidden behind one of them. You get to select one door. Initially your chosen door will not be opened. Instead, the gameshow host will open one of the other two doors, and he will do

so in such a way as not to reveal the prize. For example, if you first choose door 1, he will then open one of doors 2 and 3, and it is guaranteed that he will choose which one to open so that the prize will not be revealed. At this point, you will be given a fresh choice of door: you can either stick with your first choice, or you can switch to the other closed door. All the doors will then be opened and you will receive whatever is behind your final choice of door. Imagine that the contestant chooses door 1 first; then the gameshow host opens door 3, revealing nothing behind the door, as promised. Should the contestant (a) stick with door 1, or (b) switch to door 2, or (c) does it make no difference? You may assume that initially, the prize is equally likely to be behind any of the 3 doors. Hint: use Bayes rule. (Murphy's book Exercise 2.5)

answer

Let's solve it using Bayes' rule, although there are simpler ways to approach it...

For simplicity let:

- D3: The Event of Monty Hall opening door 3.
- C1: The Event of finding the prize behind door 1.
- C2: The Event of finding the prize behind door 2.
- C3: The Event of finding the prize behind door 3.

The prior probability of Monty Hall finding the prize behind any door is obviously $P(C1) = P(C2) = P(C3) = 1/3$.

Assume you chose the door No.1. So let's focus here on the 3rd Door!

The probability that Monty Hall opens door No.3. given the car is behind door No.1 is: $p(D3|C1) = 1/2$.

We know also that Monty Hall will never open the door which has the car so the probability that Monty Hall opens door No.3. given that the car is behind door No.3. is: $p(D3|C3) = 0$.

Contrary, the probability that Monty Hall opens door No.3. given the car is behind door No.2. is: $p(D3|C2) = 1$.

Now to the Bayes's part:

$$p(C1|D3) = p(D3|C1) * p(C1)/p(D3) = (1/2 * 1/3)/(1/2) = 1/3$$

$$p(C2|D3) = p(D3|C2) * p(C2)/p(D3) = (1 * 1/3)/(1/2) = 2/3$$

therefore, if you choose No.1. initially and Monty Hall opens door No.3. to reveal that it has no car, the probability of a car standing behind door No.2. is $2/3$. Therefore you should *always switch* your selection of your initial door.

5. (*Cross-validation*)

i. In n-fold cross-validation each data point belongs to exactly one test fold, so the test folds are independent. Are the error estimates of the separate folds also independent? So, given that the data in test folds i and j are independent, are e_i and e_j, the error estimates on test folds i and j, also independent? Is there an a priori good choice of n for n-fold cross-validation?

ii. Assume that we are using some classifier of fixed complexity. Draw a graph showing two curves: test error vs. the number of training examples and cross-validation error vs. the number of training examples.

iii. Assume that we are using classifiers of increasing complexity. Draw a graph showing three curves: test error vs. complexity, cross-validation error vs. complexity and training error vs. complexity.

answer

i. False. Since a data point appears in multiple folds the training sets are dependent and thus test fold error estimates are dependent.

There is no a priori good choice for the number of folds in cross-validation. We do not know the relation between sample size and the accuracy. High n increases correlation in training set and decreases variance of estimates. How much depends on the data and the learning method. However, in any case high n allows a larger training set which is desirable.

6. (*Discriminative vs Generative classifiers*)

i. Your billionaire friend needs your help. She needs to classify job applications into good/bad categories, and also to detect job applicants who lie in their applications using density estimation to detect outliers. To meet these needs, do you recommend using a discriminative or generative classifier? Why?

ii. Your billionaire friend also wants to classify software applications to detect bug-prone applications using features of the source code. This pilot project only has a few applications to be used as training data, though. To create the most accurate classifier, do you recommend using a discriminative or generative classifier? Why?

iii. Finally, your billionaire friend also wants to classify companies to decide which one to acquire. This project has lots of training data based on several decades of research. To create the most accurate classifier, do you recommend using a discriminative or generative classifier? Why?

answer

i. Generative, since for density estimation we need $p(x|y)$.

ii. Discriminative. We have only few datapoints, thus we'd rather not do any further assumptions on how the data are generated and use full expressive power of our model to distinguish the bug-prone applications.

iii. Generative. We have lots of training data, thus we can impose reasonable assumptions on how the data are generated and train a generative classifier. Placing some structure on the data will also help us prevent potential issues of overfitting.

7. (*Models for binary vectors*)

Consider a data set of binary (black and white) images. Each image is arranged into a vector of pixels by concatenating the columns of pixels in the image. The data set has N images $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\}$ and each image has D pixels, where D is (number of rows $\times$ number of columns) in the image. For example, image $\mathbf{x}^{(n)}$ is a vector $(x_1^{(n)}, ..., x_D^{(n)})$ where $x_d^{(n)} \in \{0, 1\}$ for all $n \in \{1, ..., N\}$ and $d \in \{1, ..., D\}$.

(a) Explain why a multivariate Gaussian would not be an appropriate model for this data set of images.

(b) Assume that the images were modelled as independently and identically distributed samples from a D-dimensional **multivariate Bernoulli distribution** with parameter vector $\mathbf{p} = (p_1, ..., p_D)$, which has the form

$$P(\mathbf{x}|\mathbf{p}) = \prod_{d=1}^{D} p_d^{x_d}(1 - p_d)^{(1-x_d)}$$

where both $\mathbf{x}$ and $\mathbf{p}$ are D-dimensional vectors.

i. What is the equation for the maximum likelihood (ML) estimate of $\mathbf{p}$? Note that you can solve for $\mathbf{p}$ directly.

ii. Assuming independent Beta priors on the parameters p_d

$$P(p_d) = \frac{1}{B(\alpha, \beta)} p_d^{\alpha-1}(1-p_d)^{\beta-1}$$

and $P(\mathbf{p}) = \prod_d P(p_d)$, What is the maximum a posteriori (MAP) estimate of $\mathbf{p}$? Hint: maximise the log posterior with respect to $\mathbf{p}$.

(c) Download the dataset `binarydigits.txt`, which contains $N = 100$ images with $D = 64$ pixels each, in an $N \times D$ matrix. These pixels can be displayed as 8×8 images by rearranging them. View them in Matlab using the script `bindigit.m` (almost no Matlab knowledge required to do this).

i. Write code to learn the ML parameters of a multivariate Bernoulli from this data set and display these parameters as an 8×8 image.

ii. Modify your code to learn MAP parameters with $\alpha = \beta = 3$. What is the new learned parameter vector for this data set? Explain why this might be better or worse than the ML estimate.

(a) The given data set of images consists of a concatenation of binary vectors. (Multivariate) Gaussian is an appropriate model for (multivariate) continuous random variables. However, for (multivariate) binary variables it is more appropriate to use a (multivariate) Bernoulli distribution. A Gaussian variable could model approximately the mean of the iid images, if N is sufficiently large, according to central limit theorem.

But let's say for the sake of argument, that the we manage to map each pixel value to the real line (say the pixels are greyscale, having values $x_d^{(n)} \in [0, 1]$ and we have some reasonable mapping $[0, 1] \mapsto \mathbb{R}$). Even in that case it does not seem intuitive to model this data with Gaussian, since Gaussians are not expressive enough. They are only capable of modelling the center of the data (mean) and the covariance among dimensions (pixels) of images. However, this is not the natural way that model images, instead, it seems more natural to model images over some distribution of edges, lines, loops, symmetry features etc.

Also, adjusting Gaussian would mean adjusting $\frac{D(D+1)}{2}$ parameters, which, for some larger images can mean having too much parameters. It is highly unlikely that the dataset will have enough data points to adjust them, therefore overfitting may occur which means bad generalisation/producing bad model for future/unseen/generated images. (We have fitted the model to be specific for observed dataset and not general for plausible input from input space. For images it is impossible to cover whole input space since most of the data points from $\mathbb{R}^D$ do not represent valid picture).

Another thing about the data is that it is inherently divided into 10 clusters, due to existance of 10 different digits. Images that represent same digit vary much less that the images representing different digits, therefore single Gaussian should perform much worse as a model than mixture of 10 Gaussians, for example.

(b) i. In order to find ML solution for $\mathbf{p}$, we can find the maximum of log-likelihood:

$$\mathbf{p}^{ML} = \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \mathcal{L}(\mathbf{p}) = \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \ln\mathrm{P}(\mathcal{D}\mid\mathbf{p}) = \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \ln\textstyle\prod_{i=1}^{N}\mathrm{P}(\mathbf{x}_i\mid\mathbf{p})$$
$$= \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \ln\prod_{i=1}^{N}\prod_{d=1}^{D} p_d^{x_d^{(i)}}(1-p_d)^{1-x_d^{(i)}} = \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \sum_{i=1}^{N}\sum_{d=1}^{D}\left[x_d^{(i)}\ln p_d + (1-x_d^{(i)})\ln(1-p_d)\right]$$

Now, in order to find the maximum, we should take $\dfrac{\mathrm{d}}{\mathrm{d}\mathbf{p}}\mathcal{L}(\mathbf{p})$ and set it to $\mathbf{0}$. We will do it element-wise, finding $\dfrac{\partial}{\partial p_j}\mathcal{L}(\mathbf{p})$:

$$\frac{\partial}{\partial p_j}\mathcal{L}(\mathbf{p}) = \sum_{i=1}^{N}\sum_{d=1}^{D}\frac{\partial}{\partial p_j}\left[x_d^{(i)}\ln p_d + (1-x_d^{(i)})\ln(1-p_d)\right]$$
$$= \sum_{i=1}^{N}\left[\frac{x_j^{(i)}}{p_j} + \frac{1-x_j^{(i)}}{p_j-1}\right] = 0 \implies p_j^{ML} = \frac{1}{N}\sum_{i=1}^{N}x_j^{(i)}$$

$$\mathbf{p}^{ML} = \frac{1}{N}\begin{bmatrix}\sum_{i=1}^{N}x_1^{(i)}\\ \sum_{i=1}^{N}x_2^{(i)}\\ \vdots\\ \sum_{i=1}^{N}x_D^{(i)}\end{bmatrix}$$

ii. Assuming Beta priors on the parameters, the MAP solution can be found as follows:

$$\mathbf{p}^{MAP} = \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \mathrm{P}(\mathcal{D}\mid\mathbf{p})\mathrm{P}(\mathbf{p}) = \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \ln\mathrm{P}(\mathcal{D}\mid\mathbf{p})\mathrm{P}(\mathbf{p})$$
$$= \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\ \left[\ln\mathrm{P}(\mathcal{D}\mid\mathbf{p}) + \ln\mathrm{P}(\mathbf{p})\right.$$
$$= \underset{\mathbf{p}\in[0,1]^D}{\arg\max}\left\{\sum_{i=1}^{N}\sum_{d=1}^{D}\left[x_d^{(i)}\ln p_d + (1-x_d^{(i)})\ln(1-p_d)\right]\right.$$
$$\left.\left. + \text{const.} + \sum_{d=1}^{D}\left[(\alpha-1)\ln p_d + (\beta-1)\ln(1-p_d)\right]\right\}$$

Now, similarly, we are taking derivative with respect to $\mathbf{p}$ of the log-posterior and equating it to $\mathbf{0}$:

$$\frac{\partial}{\partial p_j}\left\{\sum_{i=1}^{N}\sum_{d=1}^{D}\left[x_d^{(i)}\ln p_d + (1-x_d^{(i)})\ln(1-p_d)\right] + \text{const.} + \sum_{d=1}^{D}\left[(\alpha-1)\ln p_d + (\beta-1)\ln(1-p_d)\right]\right\}$$
$$= \sum_{i=1}^{N}\left[\frac{x_j^{(i)}}{p_j} + \frac{1-x_j^{(i)}}{p_j-1}\right] + \frac{\alpha-1}{p_j} + \frac{\beta-1}{p_j-1} = 0$$

Solving this equation for p_j, we get:

$$p_j^{MAP} = \frac{\alpha - 1 + \sum_{i=1}^{N} x_j^{(i)}}{\alpha + \beta - 2 + N}$$

$$\implies \mathbf{p}^{MAP} = \frac{\alpha - 1}{\alpha + \beta - 2 + N} + \frac{1}{\alpha + \beta - 2 + N} \begin{bmatrix} \sum_{i=1}^{N} x_1^{(i)} \\ \sum_{i=1}^{N} x_2^{(i)} \\ \vdots \\ \sum_{i=1}^{N} x_D^{(i)} \end{bmatrix}$$

$$= \frac{\alpha - 1}{\alpha + \beta - 2 + N} + \frac{N}{\alpha + \beta - 2 + N} \mathbf{p}^{ML}$$

(c) Here is code for learning ML parameters and MAP parameters with beta prior and params $\alpha = \beta = 3$:

Listing 1: paramlearning.m - main script

```
%% Initialising
clc;
clear all;
close all;

%% Loading data
X = load('binarydigits.txt');
N = size(X, 1);
D = size(X, 2);

%% ML estimate
pML = multiBernoulliML(X);

%% MAP estimate
alpha = 3;
beta  = 3;
pMAP  = multiBernoulliMAP(X, alpha, beta);

%% Visualising parameters as an image
figure(1);

subplot(1, 2, 1)
colormap(gray);
imshow(reshape(pML,8,8)');
axis off;
title('pML');

subplot(1, 2, 2)
colormap(gray);
imshow(reshape(pMAP,8,8)');
axis off;
title('pMAP');

saveas(1,'multiBernoulliParams.png')

%% Output parameters in a file
fid = fopen('multiBernoulliParams.txt', 'w');
```

```
fprintf(fid, ' #    pML      pMAP\n')
fprintf(fid, '%2.f %.6f  %.6f\n',[1:D; pML'; pMAP'])
fclose(fid);
```

Listing 2: multiBernoulliML.m

```
function pML = multiBernoulliML(X)
% Returns ML parameters of data assuming
% multivariate Bernoulli distribution

% Input:  N x D matrix - data, each data point row vector
% Output: D x 1 vector - pML parameters of multivar. Bernoulli

N = size(X, 1);

pML = 1/N * sum(X)';

end
```

Listing 3: multiBernoulliMAP.m

```
function pMAP = multiBernoulliMAP(X, alpha, beta)
% Returns ML parameters of data assuming
% multivariate Bernoulli distribution and Beta prior
% over params

% Input:  N x D matrix - data, each data point row vector
%         1 x 1 scalar - alpha - parameter of prior
%         1 x 1 scalar - beta  - parameter of prior
% Output: D x 1 vector - pML parameters of multivar. Bernoulli

N = size(X, 1);

pMAP = (alpha - 1)/(alpha + beta - 2 + N) + ...
                1/(alpha + beta - 2 + N) * sum(X)';

end
```

Listing 4: multiBernoulliParams.txt - output file

```
 #    pML      pMAP
 1 0.130000  0.144231
 2 0.210000  0.221154
 3 0.290000  0.298077
 4 0.430000  0.432692
 5 0.640000  0.634615
 6 0.770000  0.759615
 7 0.690000  0.682692
 8 0.500000  0.500000
 9 0.080000  0.096154
10 0.250000  0.259615
11 0.450000  0.451923
12 0.640000  0.634615
13 0.720000  0.711538
14 0.700000  0.692308
15 0.790000  0.778846
16 0.480000  0.480769
17 0.130000  0.144231
18 0.300000  0.307692
19 0.450000  0.451923
```

```
20 0.390000  0.394231
21 0.270000  0.278846
22 0.250000  0.259615
23 0.500000  0.500000
24 0.520000  0.519231
25 0.190000  0.201923
26 0.450000  0.451923
27 0.480000  0.480769
28 0.310000  0.317308
29 0.290000  0.298077
30 0.250000  0.259615
31 0.440000  0.442308
32 0.400000  0.403846
33 0.320000  0.326923
34 0.390000  0.394231
35 0.190000  0.201923
36 0.260000  0.269231
37 0.230000  0.240385
38 0.400000  0.403846
39 0.540000  0.538462
40 0.260000  0.269231
41 0.470000  0.471154
42 0.330000  0.336538
43 0.130000  0.144231
44 0.140000  0.153846
45 0.280000  0.288462
46 0.440000  0.442308
47 0.480000  0.480769
48 0.170000  0.182692
49 0.600000  0.596154
50 0.590000  0.586538
51 0.350000  0.355769
52 0.440000  0.442308
53 0.570000  0.567308
54 0.520000  0.519231
55 0.290000  0.298077
56 0.040000  0.057692
57 0.280000  0.288462
58 0.660000  0.653846
59 0.760000  0.750000
60 0.770000  0.759615
61 0.420000  0.423077
62 0.190000  0.201923
63 0.050000  0.067308
64 0.000000  0.019231
```

If we compare values of $\mathbf{p}^{ML}$ and $\mathbf{p}^{MAP}$, we see that $\mathbf{p}^{MAP}$ values vary somewhat less around the mean of 0.5. This was expected outcome since we incorporated prior belief about parameter values, claiming that we believe parameters are more likely to have value around 0.5 (beta distribution for $\alpha = 3$ and $\beta = 3$ shown in figure 2). This makes $\mathbf{p}^{MAP}$ estimate a bit more conservative, not trusting data as much as $\mathbf{p}^{ML}$.

The above can have a positive effect when we have small amount of data. In that case we don't want to astray too much listening to data since that specific dataset can lead us to wrong information. For example, the parameter $p_{64}^{ML} = 0$, meaning, the dataset told us it is impossible for 64th pixel to have value 1 (because it was never observed in $\mathcal{D}$). But just because we didn't observe 1 at that position in our sample, doesn't mean that all digit images in this world have

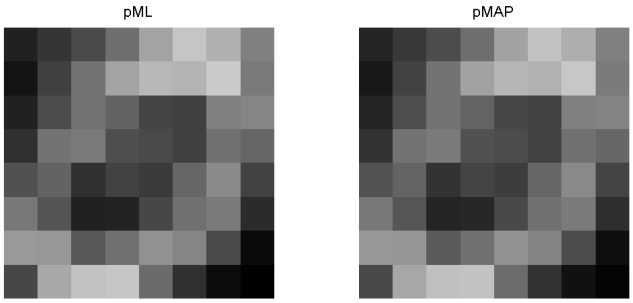

Figure 1: $\mathbf{p}^{ML}$ and $\mathbf{p}^{MAP}$ parameters displayed as image

value 0 at that position. MAP solution deals with that problem in some sense by applying prior belief.

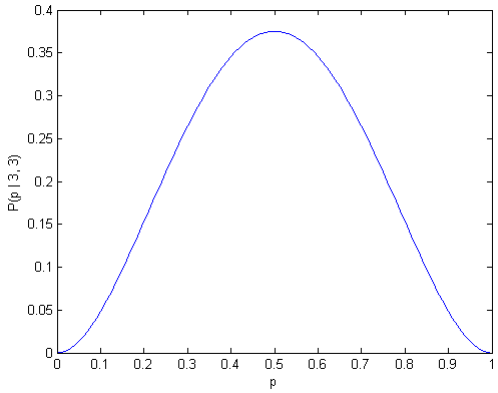

Figure 2: Beta distribution with parameters $\alpha = 3$ and $\beta = 3$

8. (*Model selection*)

   In the binary data model above, write down the expressions needed to calculate the (relative) probability of the three different models:

   (a) all D components are generated from a Bernoulli distribution with $p_d = 0.5$

   (b) all D components are generated from Bernoulli distributions with unknown, but identical, p_d

   (c) each component is Bernoulli distributed with separate, unknown p_d

   Assume the prior probability of all three models is the same. Which model is the most likely given the data in `binarydigits.txt`?

We have 3 possible models:

- $\mathcal{M}_A$ - multivariate Bernoulli with $p_d = 0.5$
- $\mathcal{M}_B$ - multivariate Bernoulli with unknown p_d, but same for all pixel values
- $\mathcal{M}_C$ - multivariate Bernoulli with unknown p_d, independent over dimensions

To compare the probability ratio between hypotheses $\mathcal{M}_i$ and $\mathcal{M}_j$ after observing data $\mathcal{D}$ we have to compute the following:

$$\frac{P(\mathcal{M}_i|\mathcal{D})}{P(\mathcal{M}_j|\mathcal{D})} = \frac{P(\mathcal{M}_i)}{P(\mathcal{M}_j)} \frac{P(\mathcal{D}|\mathcal{M}_i)}{P(\mathcal{D}|\mathcal{M}_j)}$$

In the absence of information that could bias our model towards one of the hypotheses, we assume prior plausibility of the three hypotheses is: $P(\mathcal{M}_A) = P(\mathcal{M}_B) = P(\mathcal{M}_C) = \frac{1}{3}$. Hence, to calculate which model, given this data, is the most likely, it suffices to compute the likelihood of the observations under the different models:

$$P(\mathcal{D}|\mathcal{M}_i) = \int_{\mathbf{p}} P(\mathcal{D}|\mathbf{p}, \mathcal{M}_i) P(\mathbf{p}|\mathcal{M}_i) d\mathbf{p},$$

where $P(\mathcal{D}|\mathbf{p}, \mathcal{M}_i) = \Pi_{d=1}^{D} \Pi_{i=1}^{N} p_d^{x_d^{(i)}} (1 - p_d)^{(1 - x_d^{(i)})}$.
For numerical reasons it is more convenient to find log-likelihood and then transform into linear space:

(a) For the first model, $P(\mathbf{p}|\mathcal{M}_i) = \delta(\mathbf{p} - \mathbf{0.5})$, i.e. prior over model parameters has all its mass placed on $p = [0.5, 0.5, ..., 0.5] = \mathbf{0.5}$. Hence, integrating out the parameters, gives for the loglikehood we have

$$\begin{aligned}
\ln P(\mathcal{D} \mid \mathcal{M}_A) &= \sum_{i=1}^{N} \ln P(x^{(i)}|\mathbf{p} = \mathbf{0.5}) \\
&= \sum_{i=1}^{N} \sum_{d=1}^{D} (x_d^{(i)} \ln 0.5 + (1 - x_d^{(i)}) \ln 0.5) \\
&= ND \ln 0.5 \\
&\approx -4436.14
\end{aligned}$$

(b) For $\mathcal{M}_B$ we have $p_d = p_*$ for every d, hence we postulate a general Beta distribution over the model parameters

$$P(p_* = p|\mathcal{M}_B) = \text{Beta}(p, \alpha, \beta) = \frac{1}{B(\alpha, \beta)} p^{\alpha - 1} (1 - p)^{(\beta - 1)}.$$

Therefore we get the following for the loglikelihood of the data:

$$
\begin{aligned}
P(\mathcal{D}|\mathcal{M}_B) &= \int_p \prod_d \prod_i p^{x_d^{(i)}} (1-p)^{1-x_d^{(i)}} \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p^{\beta-1})\mathrm{d}p \\
&= \int_p p^{\sum_i \sum_d x_d^{(i)}+\alpha-1}(1-p)^{ND-\sum_i \sum_d (x_d^{(i)})+\beta-1} \frac{1}{B(\alpha,\beta)}\mathrm{d}p \\
&= \frac{B(\sum_n \sum_i x_d^{(i)}+\alpha, ND-\sum_i \sum_d x_d^{(i)}+\beta)}{B(\alpha,\beta)}
\end{aligned}
$$

(c) Here each p_d can be different. One postulates a general Beta prior over model parameters $P(p_* = p) = Beta(p,\alpha,\beta) = \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p)^{\beta-1}$. Therefore

$$
\begin{aligned}
P(\mathcal{D}|\mathcal{M}_C) &= \int_{p_d} \prod_d \prod_i p^{x_d^{(i)}} (1-p)^{1-x_d^{(i)}} \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p^{\beta-1})\mathrm{d}p_d \\
&= \prod_d \int_{p_d} p_d^{\sum_i x_d^{(i)}+\alpha-1}(1-p_d)^{N-\sum_i x_d^{(i)}+\beta-1} \frac{1}{B(\alpha,\beta)}\mathrm{d}p_d \\
&= \prod_d \frac{B(\sum_i x_d^{(i)}+\alpha, N-\sum_i x_d^{(i)}+\beta)}{B(\alpha,\beta)}
\end{aligned}
$$

9. (*Dependence of p-values on irrelevant information -or why we should become Bayesians :-) (from McKay's book)*

(a) In an expensive laboratory, Dr. Bloggs tosses a coin labelled a and b twelve times and the outcome is the string

$$aaabaaaabaab,$$

which contains three bs and nine as. What evidence do these data give that the coin is biased in favour of a? Is it significant at the level of 5% ?

(b) Dr. Bloggs pays careful attention to the calculation of 1.1, and responds *'no, no, the random variable in the experiment was not the number of bs: I decided before running the experiment that I would keep tossing the coin until I saw three bs; the random variable is thus the total number of tosses, n'*. A different calculation is required in order to assess the 'significance' of the result $n = 12$. Now, the probability distribution of n given $\mathcal{H}_0$ is the probability that the first $n-1$ tosses contain exactly $r-1$ bs and then the nth toss is a b. What evidence do these data give that the coin is biased in favour of a in this case? Is the evidence significant at the level of 5% ?

answer

(a) Dr. Bloggs consults his sampling theory friend who says 'let r be the number of bs and $n = 12$ be the total number of tosses; I view r as the random variable and find the probability of r taking on the value $r = 3$ or a more extreme value, assuming the null hypothesis $p_a = 0.5$ to be true'. She thus computes

$$P(r \leq 3|n = 12, \mathcal{H}_0) = \sum_{r=0}^{3} \binom{n}{r} \frac{1}{2^n} \approx 0.07$$

and reports 'at the significance level of 5%, there is not significant evidence of bias in favour of a'. Or, if the friend prefers to report p-values rather than simply compare p with 5%, she would report 'the p-value is 7%, which is not conventionally viewed as significantly small. If a two-tailed test seemed more appropriate, she might compute the two-tailed area, which is twice the above probability, and report the p-value is 15%, which is not significantly small'.

(b) Now the sampling theorist has to compute the following probability :

$$P(n|\mathcal{H}_0, r) = \binom{n-1}{r-1}\frac{1}{2^n}$$

For Dr.Bloggs' experiment she gets

$$P(n \geq 12|r = 3, \mathcal{H}_0) \approx 0.03$$

She reports back to Dr. Bloggs, the p-value is 3% – there is significant evidence of bias after all!

Conclusion: The p-values of sampling theory do depend on the stopping rule.