Applied Bayesian Data Analysis — Exercise 3 B

Kim Albertsson

CERN and Luleå University of Technology

kim.albertsson@ltu.se

October 15, 2019

Question B.2

Plot the outcome probabilities given $\theta \in 0.50, 0.25$ using a bar or a stem plot.

Equation for Bernoulli distribution (pdf):

$$p(x|\theta) = \theta^x(1-\theta)^{1-x},\ x \in \{0,1\}$$

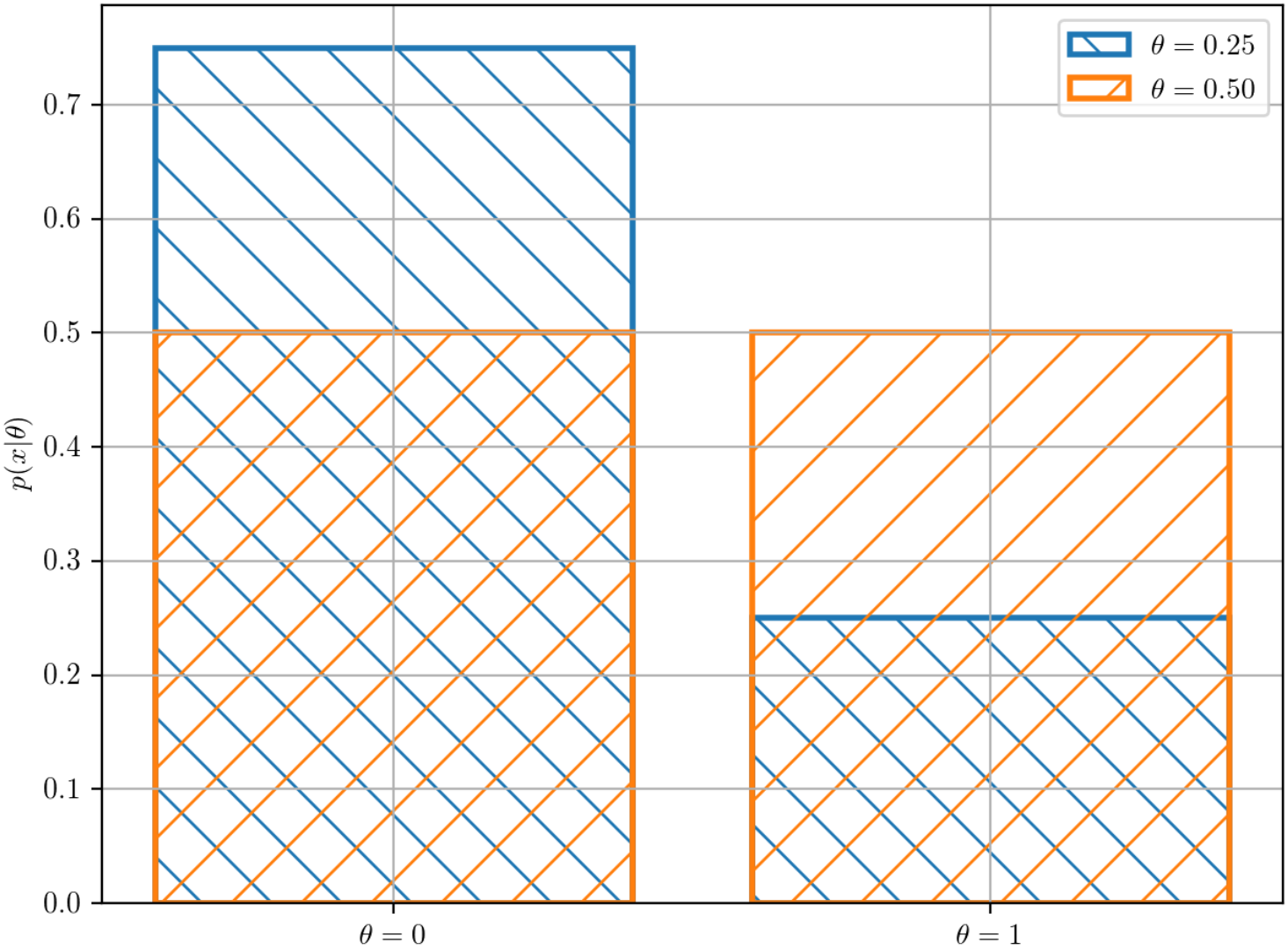
Code for Bernoulli distribution (pdf):

```
class Bernoulli(object):
    @staticmethod
    def pdf(x, theta=0.5):
        return theta**x * (1-theta)**(1-x)
    @staticmethod
    def likelihood(xs, theta=0.5):
        return np.prod([Bernoulli.pdf(x, theta)
                        for x in xs])
```

Code for generating plot:

```
plt.figure()
x = [0, 1]
plt.bar(x, Bernoulli.pdf(x, theta=0.25),
        label='$\theta=0.25$', hatch='\\',
        facecolor='none', edgecolor='tab:orange',
        linewidth=2)
plt.bar(x, Bernoulli.pdf(x, theta=0.50),
        label='$\theta=0.25$', hatch='/',
        facecolor='none', edgecolor='tab:orange',
        linewidth=2)
plt.xticks([0, 1], ['$\\theta=0$', '$\\theta=1$'])
plt.ylabel('$p(x|\\theta)$')
plt.legend()
plt.grid()
plt.tight_layout()
```

Probability mass for two cases of the Bernoulli distribution:



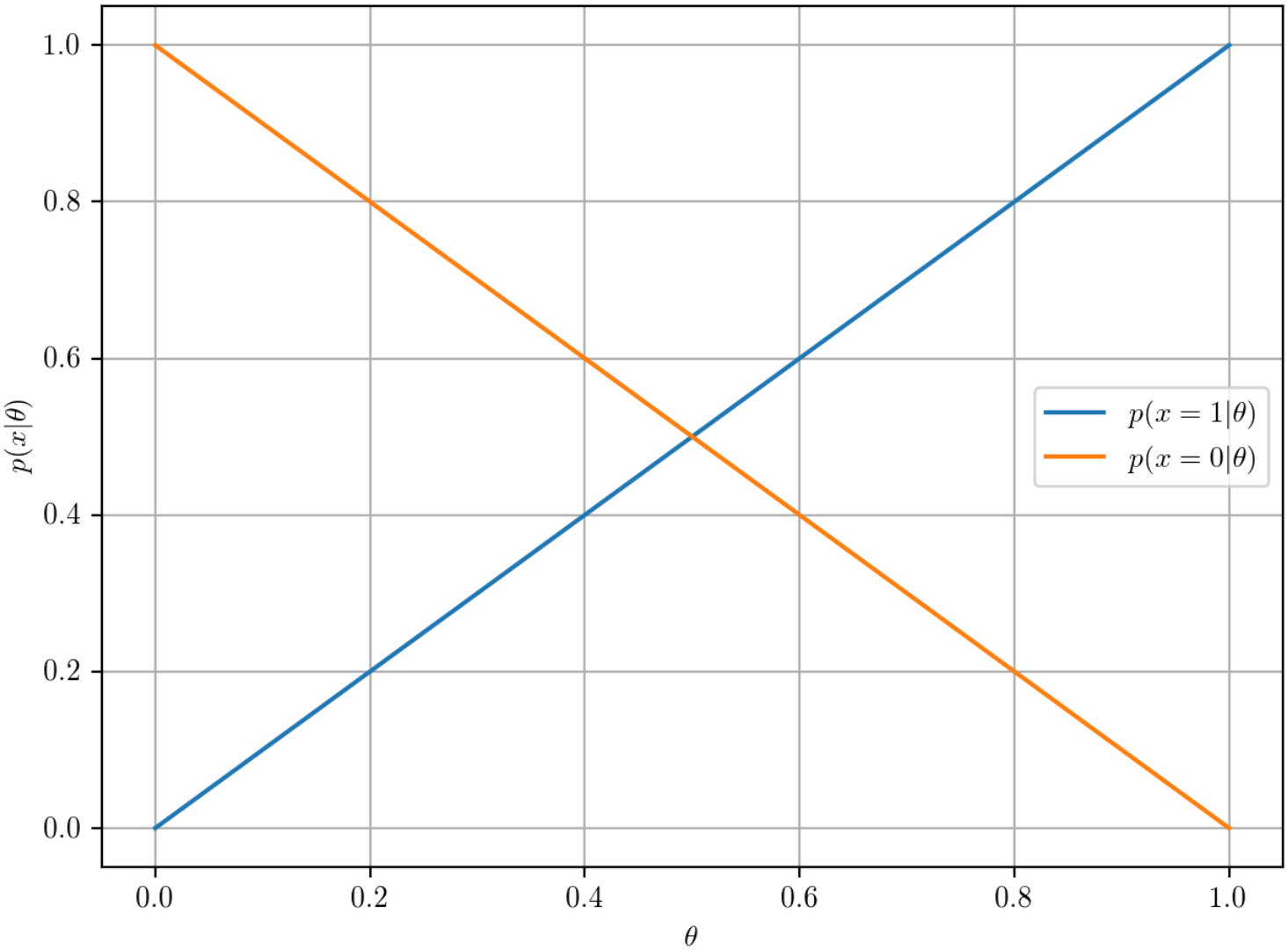
Question B.3

Plot the likelihood of a given parameter value θ given a particular outcome.

Code for generating plot:

```
plt.figure()
theta = np.linspace(0, 1, 100)
plt.plot(theta, Bernoulli.pdf(1, theta),
         label='$p(x=1|\\theta)$')
plt.plot(theta, Bernoulli.pdf(0, theta),
         label='$p(x=0|\\theta)$')
plt.xlabel('$\\theta$')
plt.ylabel('$p(\\theta|x)$')
plt.legend()
plt.grid()
plt.tight_layout()
```

Likelihoods for two cases of the Bernoulli distribution:



Question B.4.a

Evaluate the likelihood function for $\theta = 0.5$ for $n \in 10, 1000, 100000$, where n is the number of coin flips in the experiment. What happens for large values of n?

Reminder: Code for Bernoulli distribution

```
class Bernoulli(object):
    @staticmethod
    def pdf(x, theta=0.5):
        return theta**x * (1-theta)**(1-x)
    @staticmethod
    def likelihood(xs, theta=0.5):
        return np.prod([Bernoulli.pdf(x, theta)
                        for x in xs])
```

Note: Given $\theta = 0.5$, each sequence is equally likely. No need for sampling.

```
>>> Bernoulli.likelihood([0]*10)
0.0009765625
>>> Bernoulli.likelihood([0]*1000)
9.332636185032189e-302
>>> Bernoulli.likelihood([0]*100000)
0.0
```

As n tends to infinity, the likelihood tends to zero $(\frac{1}{2}^{-n})$.

Question B.4.b

Can you evaluate the *log-likelihood* for larger n without problems of under- or overflow?

Q: Why log-likelihood?

Computational efficiency, use summation instead of multiplication.

Monotonic transformation, global minima/maxima coincide.

Q: Which log? — Does it matter? Use binary log for measure in “bits”?

Equation for Bernoulli distribution (log-pdf):

$$
\begin{aligned}
p(x|\theta) &= \theta^x(1-\theta)^{1-x},\ x \in \{0,1\} \\
\log p(x|\theta) &= \log(\theta^x(1-\theta)^{1-x}) \\
\log p(x|\theta) &= \log \theta^x + \log(1-\theta)^{1-x} \\
\log p(x|\theta) &= x \log \theta + (1-x)\log(1-\theta)
\end{aligned}
$$

Code for Bernoulli distribution (log-pdf)

```
class Bernoulli(object):
    @staticmethod
    def logpdf(x, theta=0.5):
        return (x*np.log(theta) +
                (1-x)*np.log(1-theta))
    @staticmethod
    def loglikelihood(seq, theta=0.5):
        return np.sum([Bernoulli.logpdf(x, theta)
                       for x in seq])
```

Note: Exact values depend on the base of the logarithm. Your mileage might vary.

```
>>> Bernoulli.loglikelihood([0]*10)
-6.9314718055994530
>>> Bernoulli.loglikelihood([0]*1000)
-693.14718055994530
>>> Bernoulli.loglikelihood([0]*100000)
-69314.718055994530
```

As n tends to infinity, the log-likelihood tends to negative infitnity ($log(\frac{1}{2})n$).

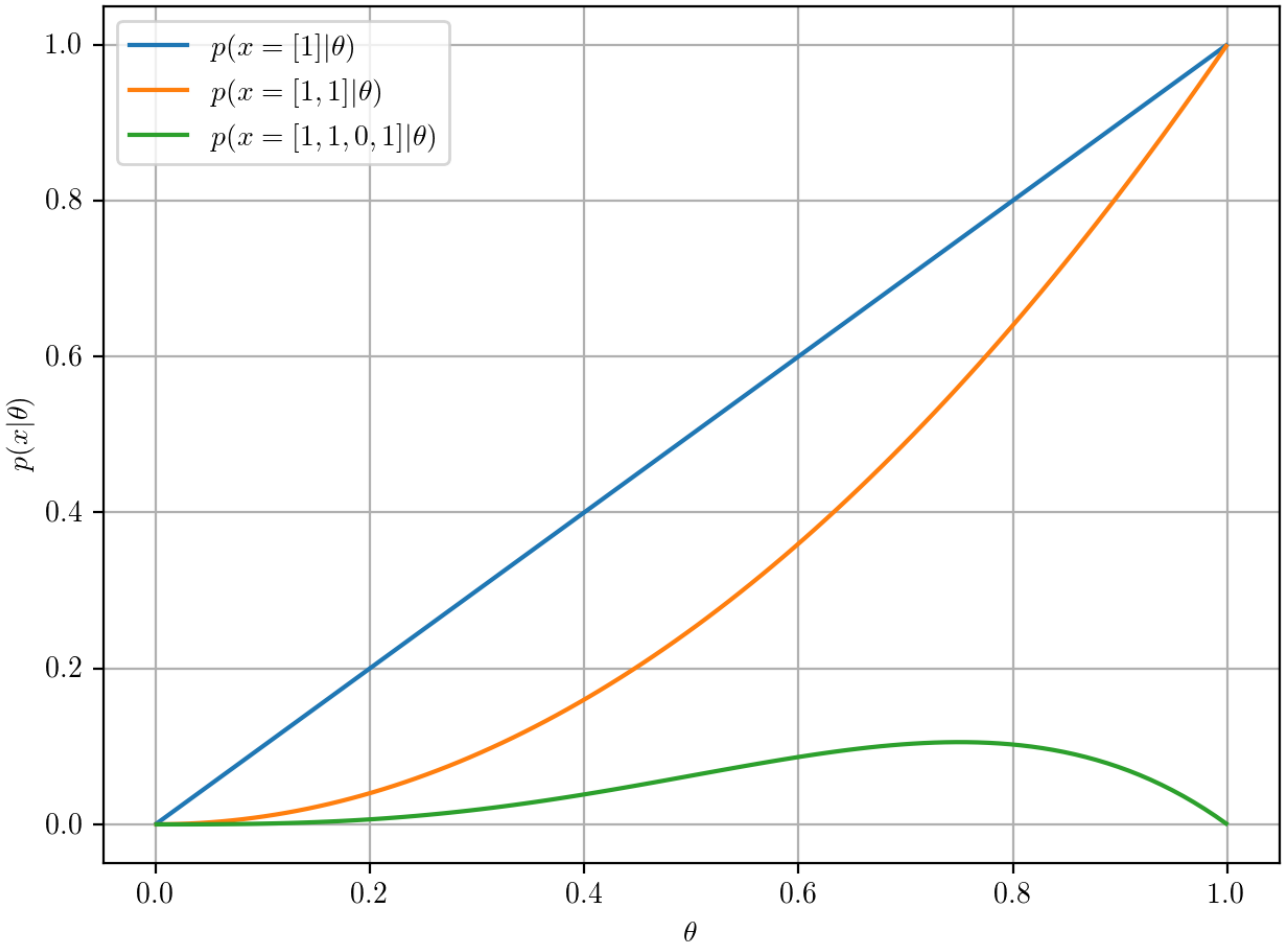
Question B.4.d-e

Plot the likelihood functionwith respect to θ given $x_0 = [1]; x_0 = [1, 1]; x_2 = [1, 1, 0, 1];$.

Explain the behaviour you see in terms of the likelihood of θ for the particular data sequences.

Code for plot:

```
x = np.linspace(0.001, 0.999, 100)
plt.figure()
plt.plot(x, np.exp([Bernoulli.loglikelihood([1], X)
                    for X in x]),
         label='$p(x=[1]|\\theta)$')
plt.plot(x, np.exp([Bernoulli.loglikelihood([1,1], X)
                    for X in x]),
         label='$p(x=[1,1]|\\theta)$')
plt.plot(x, np.exp([Bernoulli.loglikelihood([1,1,0,1],
                                            X)
                    for X in x]),
         label='$p(x=[1,1,0,1]|\\theta)$')
plt.xlabel('$\\theta$')
plt.ylabel('$p(\\theta|x)$')
plt.legend()
plt.grid()
plt.tight_layout()
```



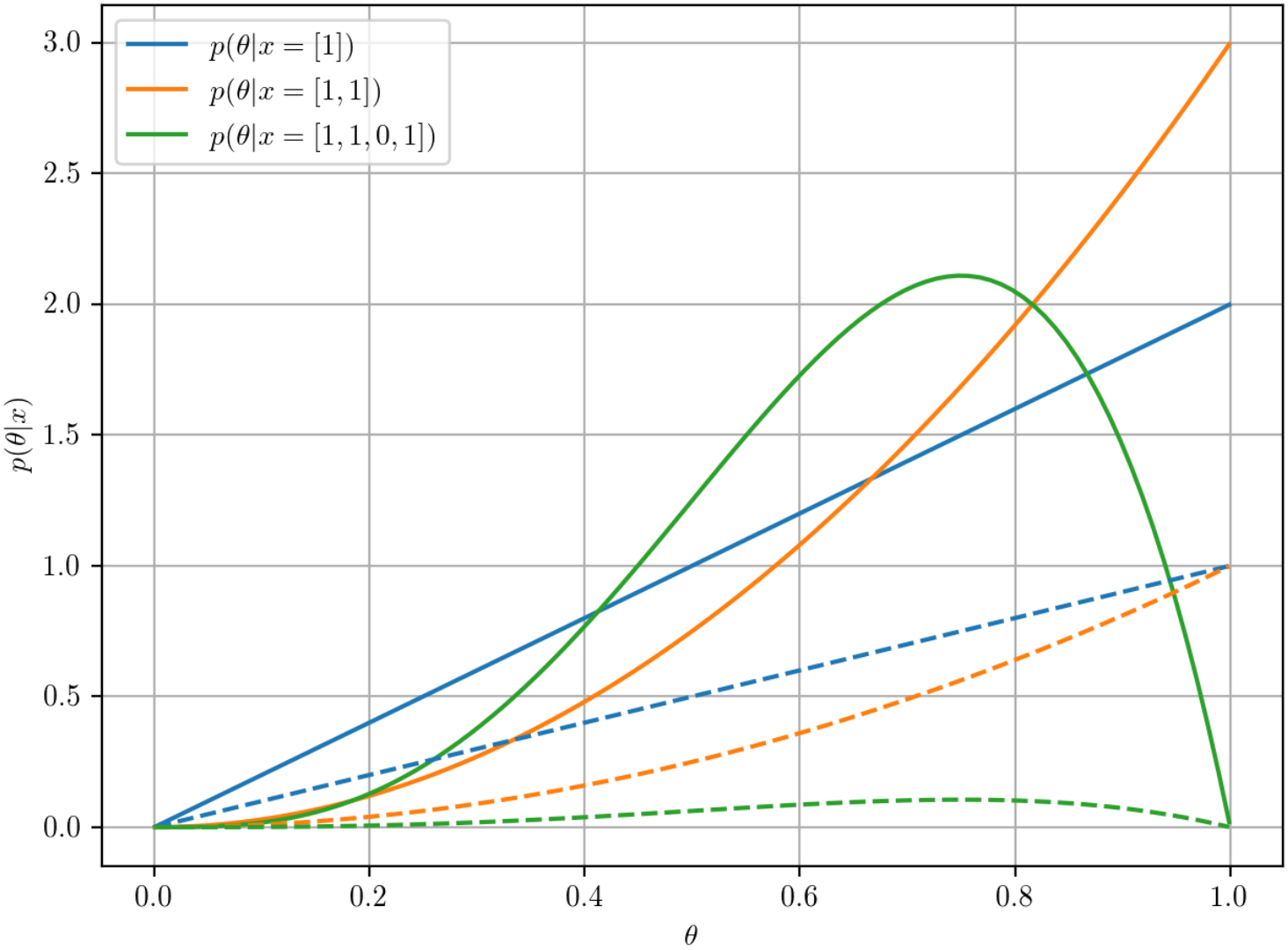
For $x_2 = [1, 1, 0, 1]$: Note that sequence contain both 1's and 0's. Thus the probability for $\theta = 0$ and $\theta = 1$ are 0. Since more 1's than 0's, maximum skewed towards $\theta = 1$.

Question B.5.a.ii.1-2

Plot, instead of the likelihoods, the posterior distribution assuming a $p(\theta) = \beta(1, 1)$ prior.

Compare the mathematical expressions in Eq. 6.1 and Eq. 6.2 with that of Eq. 6.8. Explain differences in plots using this.

Solid lines are posteriors, dashed are likelihoods. Notice the difference in shape.



Comparing equations for likelihood (6.2) and posterior (6.8) we see that the difference is a normalisation constant. Note: N is total number of coin flips. z is number of heads.

$$p(\theta) = \frac{\theta^{1-1}(1-\theta)^{1-1}}{B(1,1)} \tag{6.3}$$

$$p(x|\theta) = \theta^{z}(1-\theta)^{N-z} \tag{6.2}$$

$$p(\theta|x) = \frac{\theta^{z}(1-\theta)^{N-z}}{B(z+1, N-z+1)} \tag{6.8}$$

Question B.5.a.iii

Reproduce Figure 6.4 from the book.

Code for plot (1/4):

```
x = np.linspace(0.001, 0.999, 100)
h = 17 # N heads
t = 3  # N tails
y = [1]*h + [0]*t # Data sample

fig, axs = plt.subplots(nrows=3, ncols=3,
                        sharex=True)

axs[0,0].set_title('High belief in prior\n'
                   '$p(\\theta)=\\beta(100,100)$')
axs[0,1].set_title('Balanced\n'
                   '$p(\\theta)=\\beta(18.25,6.75)$')
axs[0,2].set_title('Low belief in prior\n'
                   '$p(\\theta)=\\beta(1,1)$')
```

Code for plot (2/4):

```
axs[0, 0].fill_between(x,
    np.exp([Beta.logpdf(X, a=100, b=100)
            for X in x]))
axs[0, 0].set_ylabel('$p(\\theta)$')
axs[1, 0].fill_between(x,
    np.exp([Bernoulli.loglikelihood(y, X)
            for X in x]))
axs[1, 0].set_ylabel('$p(x|\\theta)$')
axs[2, 0].fill_between(x,
    np.exp([Beta.logpdf(X, a=117, b=103)
            for X in x]))
axs[2, 0].set_ylabel('$p(\\theta|x)$')
```

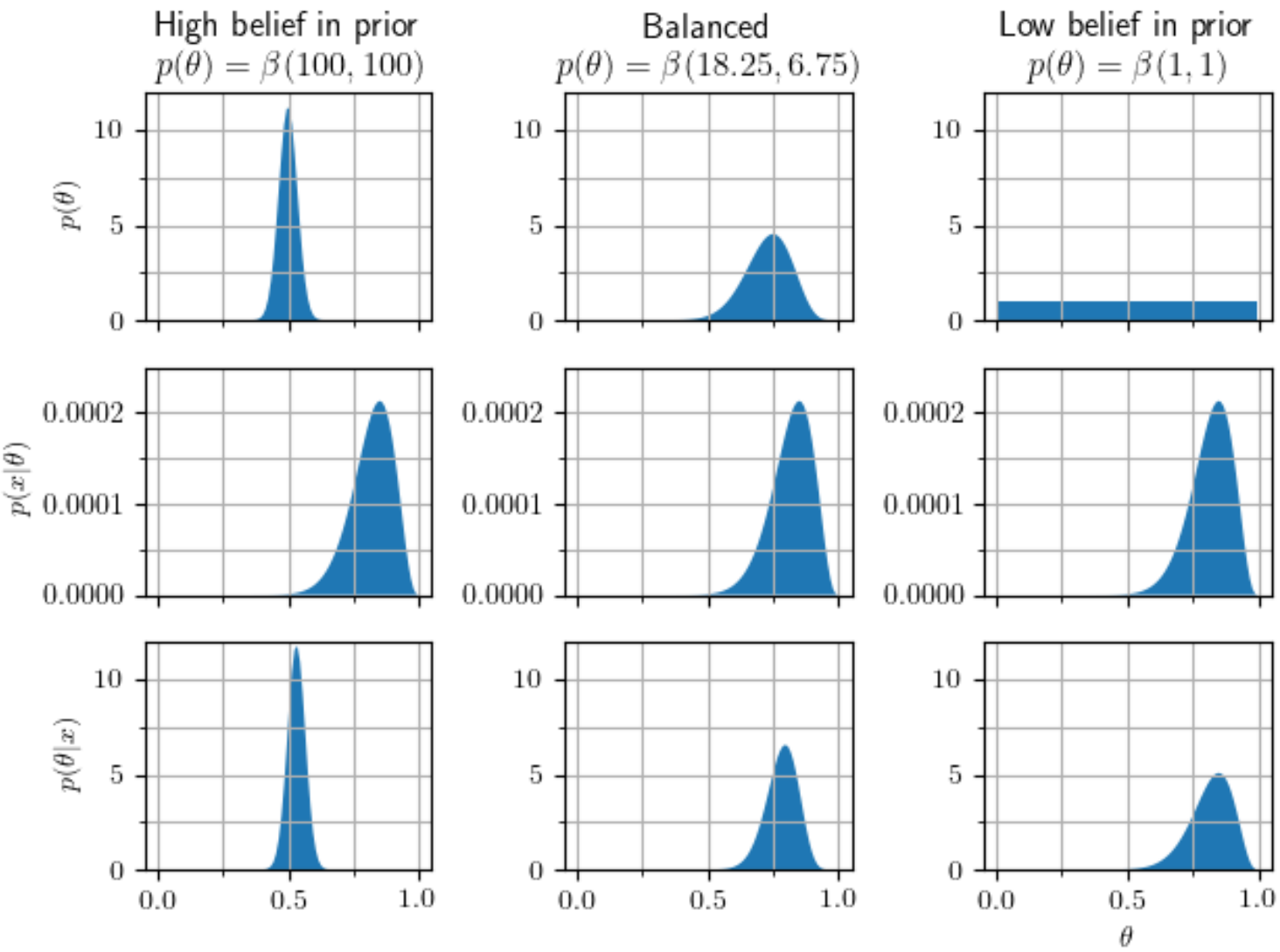
Code for plot (3/4):

```
axs[0, 1].fill_between(x,
    np.exp([Beta.logpdf(X, a=18.25, b=6.75)
            for X in x]))
axs[1, 1].fill_between(x,
    np.exp([Bernoulli.loglikelihood(y, X)
            for X in x]))
axs[2, 1].fill_between(x,
    np.exp([Beta.logpdf(X, a=35.25, b=9.75)
            for X in x]))
```

Code for plot (4/4):

```
axs[0, 2].fill_between(x,
    np.exp([Beta.logpdf(X, a=1, b=1)
            for X in x]))
axs[1, 2].fill_between(x,
    np.exp([Bernoulli.loglikelihood(y, X)
            for X in x]))
axs[2, 2].fill_between(x,
    np.exp([Beta.logpdf(X, a=1+h, b=1+t)
            for X in x]))
axs[2, 2].set_xlabel('$\\theta$')

# Plus some code for formatting
```


High belief in prior
$p(\theta) = \beta(100, 100)$
Balanced
$p(\theta) = \beta(18.25, 6.75)$
Low belief in prior
$p(\theta) = \beta(1, 1)$
$p(\theta)$
$p(x|\theta)$
$p(\theta|x)$
0.0000
0.0001
0.0002
0
5
10
0.0
0.5
1.0
θ

For complete code

See
`https://github.com/ashlaban/ltu-abda-2019/tree/master/ex3`