# COMP90051 Statistical Machine Learning

Workshop Week 2

**Xudong Han** 

https://github.com/HanXudong/COMP90051\_Workshops

#### Your tutor

- Xudong Han
- xudong.han@unimelb.edu.au
- Slides https://github.com/HanXudong/COMP90051\_Workshops

## Learning Outcomes

At the end of this workshop you should:

- Be familiar with the Python ecosystem for ML
- Develop intuition about the role of the prior and posterior in Bayesian inference. See "worksheet02.ipynb".

### The SciPy stack



- Library for working with large multidimensional arrays
- · High-level functions for arrays

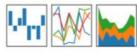


- · Scientific computing library
- Functionality includes: statistics/random number generation, linear algebra, optimisation, special functions



- · Machine learning library
- Includes implementations of most models covered in this course (exception: neural nets)





- Library for analysis and manipulation of tabular data
- Provides similar functionality to DataFrames and dplyr in R



- 2D plotting library
- Provides similar interface to MATLAB

## Other libraries in the ecosystem



- Library for numerical computations using data flow graphs
- · Often used for neural nets
- · Supports GPU, TPU acceleration



- A probabilistic programming language written in C++
- Great for smaller-scale statistical modelling
- Has a Python API



- · Cluster computing framework
- Supports scalable machine learning through Spark MLlib
- Has a Python API



- High-level neural net library written in Python
- Supports various backends: TensorFlow, CNTK and Theano



- A probabilistic programming language written in Python and built on top of Theano
- · More "Pythonic" than Stan

#### Bayesian statistics

- Probabilities correspond to beliefs
- Parameters
  - Modeled as r.v.'s having distributions
  - Prior belief in  $\theta$  encoded by prior distribution  $P(\theta)$ 
    - Parameters are modeled like r.v.'s (even if not really random)
    - Thus: data likelihood  $P_{\theta}(X)$  written as conditional  $P(X|\theta)$
  - \* Rather than point estimate  $\hat{\theta}$ , Bayesians update belief  $P(\theta)$  with observed data to  $P(\theta|X)$  the posterior distribution)



Laplace

## Running Jupyter Notebook

- 1. Download worksheet02.ipynb from the LMS
- 2. Move the downloaded file to a working directory %WORKDIR%
- 3. Start  $\rightarrow$  Anaconda3 (64-bit)  $\rightarrow$  Anaconda Prompt
- 4. Type the following command at the prompt:

```
jupyter notebook --notebook-
dir=%WORKDIR%
```

- 5. The Jupyter UI should open in a web browser.
- 6. Click on worksheet2.ipynb to get started.

## 1. A lucky find



You're interested in determining whether the coin is biased.

#### Bernoulli Distribution

• Let  $x_n$  denote the result of n<sup>th</sup> flip.

• 
$$x_n = \begin{cases} 1, if heads \\ 0, if tails \end{cases}$$

- $x_n \sim Bernoulli(\theta)$
- $f(x_n|\theta) = \theta^{x_n}(1-\theta)^{(1-x_n)}$
- For a fair coin,  $\theta = 0.5$ .

```
def toss_coin():
    if bernoulli.rvs(p = (int.from_bytes("coin".encode(), 'little') % 10000)/10000):
        return 1
    return 0
```

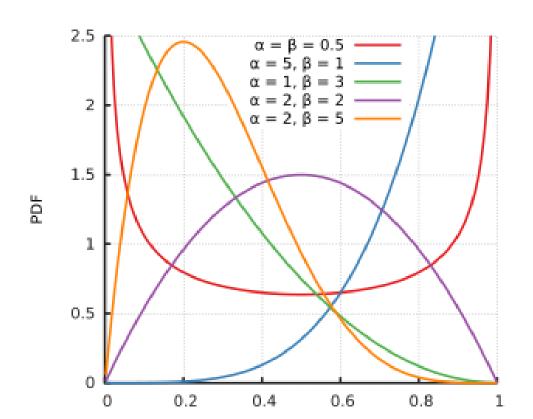
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bernoulli.html #scipy.stats.bernoulli

#### 2. Prior belief

- How could we estimate the parameter  $\theta$ ?  $\theta$  = the number of heads / total number
- In a Bayesian way:
  - 1) Prior belief in  $\theta$  encoded by prior distribution  $P(\theta)$ .
  - 2) Beliefs (prior) + New observations (likelihood) = New beliefs (posterior)

#### Parameter heta and Beta distribution

- $\theta$  is a probability, thus  $0 \le \theta \le 1$
- https://en.wikipedia.org/wiki/Beta\_distribution



## Try

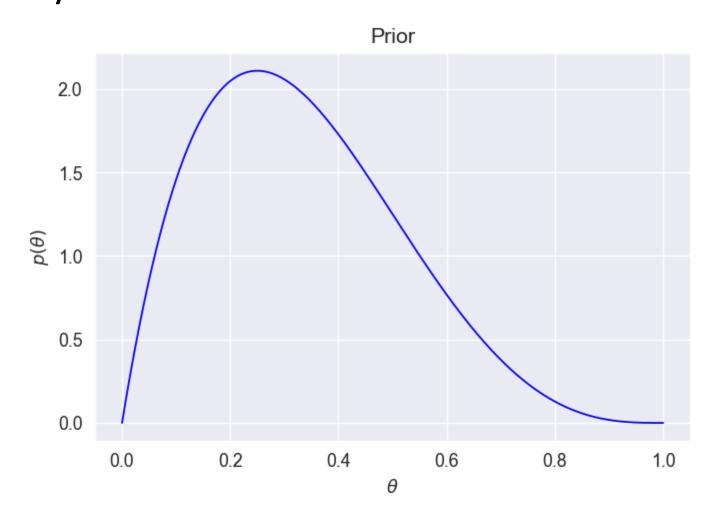
```
a = 2 b =4
a = 4 b = 2
a = 100 b = 200
```

```
a = ... # fill in
b = ... # fill in
theta = np.linspace(0, 1, 1000)

FIGURE_RESOLUTION = 128
plt.rcParams['figure.dpi'] = FIGURE_RESOLUTION

plt.plot(theta, beta.pdf(theta, a, b), 'b-', lw=1)
plt.title('Prior')
plt.xlabel(r'$\theta$')
plt.ylabel(r'$\theta$')
plt.ylabel(r'$p(\theta)$')
plt.show()
```

## Suppose we expect tails are more likely.



## 3. Posterior updates

- Now toss the coin once and denote the outcome by  $x_1$
- Update beliefs:

$$p(\theta|x_1) = \frac{p(\theta, x_1)}{p(x_1)} = \frac{p(x_1|\theta)p(\theta)}{p(x_1)} \propto p(x_1|\theta)p(\theta)$$

Now the likelihood:

$$p(x_1|\theta) = \theta^{x_1}(1-\theta)^{(1-x_1)}$$

•  $p(\theta)$  is our prior distribution. Beta(a, b)

$$p(\theta) = \frac{1}{B(a,b)} \theta^{a-1} (1-\theta)^{b-1}$$

**Exercise:** Show (on paper) that the posterior takes the form of a Beta distribution.

$$\theta | x_1 \sim \text{Beta}[x_1 + a, (1 - x_1) + b]$$

If the posterior and prior have the same function form, we say that the prior (in this case Beta) is *conjugate* to the likelihood function (in this case Bernoulli). This is convenient because the prior and posterior distributions have the same functional form, which admits a closed form expression for all posterior updates. This also allows us to ignore constant factors in intermediate calculations, as we may recover the normalizing factors by comparison with the standard form of the relevant density for the posterior.

[Hint: a similar calculation was performed in L2, slide 20 for the case of a normal prior and posterior.]

**Exercise:** Show that for N coin tosses, the posterior

$$p(\theta|x_1,\ldots,x_N) = \text{Beta}[N_H + a, N - N_H + b]$$
 where  $N_H = \sum_{n=1}^N x_n$  is the number of heads observed.

- We assume the tosses are independent.
- Likelihood now is the product of all the pdf of each x.

$$p(\theta|x_1, x_2, ..., x_n) = \prod f(x_i|\theta)$$

Remember

$$p(x_1|\theta) = \theta^{x_1}(1-\theta)^{(1-x_1)}$$

#### 4. MAP estimator and MLE estimator

The posterior  $\theta|x_1, \dots, x_N$  contains all the information we know about  $\theta$  after observing N coin tosses. One way of obtaining a point estimate of  $\theta$  from the posterior, is to take the value with the maximum a posteriori probability (MAP):

Mode

$$\hat{\theta}_{MAP} = \arg \max_{\theta} p(\theta|x_1, \dots, x_N)$$

$$= \frac{N_H + a - 1}{N + a + b - 2}$$

In general, the MAP estimator gives a different result to the maximum likelihood estimator (MLE) for  $\theta$ :

$$\hat{\theta}_{MLE} = \arg \max_{\theta} p(x_1, \dots, x_N | \theta)$$

$$= \frac{N_H}{N}$$

## 5. Convergence of the estimates

```
extra_tosses = 48
num tosses = 2 + extra tosses
num\ heads = 0
theta map = np.zeros(num tosses)
theta_mle = np.zeros(num_tosses)
for i in range(1, num tosses):
    if i == 1:
        num heads += x1
   elif i == 2:
        num heads += x2
    else:
        num heads += toss coin()
    theta_map[i] = ... # fill in
    theta_mle[i] = ... # fill in
theta map = theta map[1:]
theta mle = theta mle[1:]
```

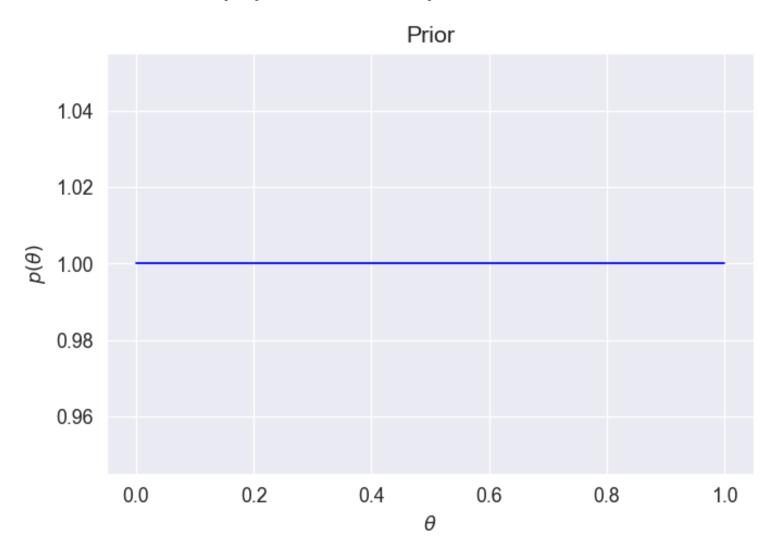
$$\hat{\theta}_{MAP} = \arg \max_{\theta} p(\theta|x_1, \dots, x_N)$$

$$= \frac{N_H + a - 1}{N + a + b - 2}$$

$$\hat{\theta}_{MLE} = \arg \max_{\theta} p(x_1, \dots, x_N | \theta)$$

$$= \frac{N_H}{N}$$

## What happens if you set a=b=1?



## What happens if you set a=b=1?

$$\hat{\theta}_{MAP} = \arg \max_{\theta} p(\theta|x_1, \dots, x_N) \quad \hat{\theta}_{MLE} = \arg \max_{\theta} p(x_1, \dots, x_N|\theta)$$

$$= \frac{N_H + a - 1}{N + a + b - 2} \qquad = \frac{N_H}{N}$$

