

VE414 Lecture 7

Jing Liu

UM-SJTU Joint Institute

September 24, 2019

- You should have noticed that the following is indispensable

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$



- However, to obtain any posterior, in addition to identifying a likelihood to model the data generating process, and choosing a prior for the unknown, we need to “normalise” the product of the two to have the posterior.
- So far, the only way we can obtain the posterior is to use a conjugate prior, any other prior in general will very likely lead us to an expression

$$\text{Likelihood} \times \text{Prior}$$

that cannot be integrated analytically, e.g

$$f_{Y|X} \propto \exp\left(-\frac{(x-y)^2}{2}\right) \times \frac{1}{1+y^2}$$

- Often only the posterior up to an multiplicative constant is readily available.

- In terms of point estimates, having only the posterior

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

up to an multiplicative constant means we will not have the mean in general.

Q: What should we do if a point estimate is required?

- If a point estimate is required, we could use the mode instead of the mean.

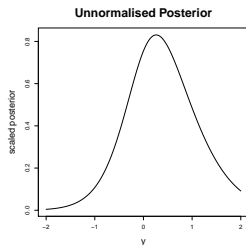
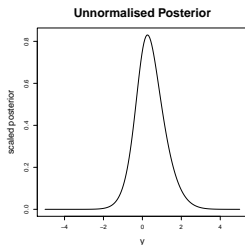
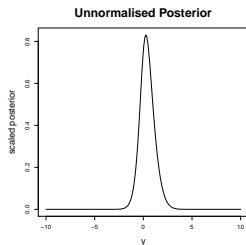
$$\arg \max_y f_{Y|X}(y | x)$$

Q: Why will we be able to avoid the integral in this case?

$$\hat{y}_{MAP} = \arg \max_y f_{Y|X}(y | x) = \arg \max_y f_{X|Y}(x | y) f_Y(y)$$

- The optimisation is easier than the integration, especially in high dimension.

- For univariate optimisation, it is always a good idea to plot the function.
- The posterior is not available, but the unnormalised posterior can provide



information on where the posterior peaks and how quickly it drops towards 0.

- In this case, we could simply start with a small interval, say

$$[-1, 1]$$

and use a reliable optimisation method to find where the peak is.

- Julia has a large number of optimisation algorithms, the following package

```
julia> using Pkg  
julia> Pkg.add("Optim")
```

is sufficient for us.

- Load the package **Optim** into the current session,

```
julia> using Optim
```

- Let us solve for $x = 0.75$, and set the lower and upper bound,

```
julia> x = 0.75; lower = -1; upper = 1;
```

- Define the objective function, which is simply the negative of our integrand,

```
julia> un_posterior(y) = -exp(-(x-y)^2/2)/(1+y^2)
```

```
un_posterior (generic function with 1 method)
```

```
julia> ymap = optimize(un_posterior,  
                      lower, upper, GoldenSection())
```

Results of Optimization Algorithm

```
* Algorithm: Golden Section Search  
* Search Interval: [-1.000000, 1.000000]  
* Minimizer: 2.611106e-01  
* Minimum: -8.307208e-01  
* Iterations: 40  
* Convergence:  
max(|x-x_upper|, |x-x_lower|) <=  
2*(1.5e-08*|x|+2.2e-16) :true
```

- Numerical optimisation will not be covered in this course, but you should know it is easier than numerical integration, especially so in high dimension.

Q: Does that mean we can always avoid the integral?

Q: How should we deal with the integral when the MAP estimate is not ideal?

- The easiest way to obtain a rough idea $f_{Y|X}$ without analytically computing

$$\int_{-\infty}^{\infty} f_{X|Y}(x | y) f_Y(y) dy$$

is to use a **grid approximation**, in which the integral is approximated by

$$\int_{-\infty}^{\infty} f_{X|Y}(x | y) f_Y(y) dy \approx \frac{b-a}{n} \sum_{i=1}^n f_{X|Y}(x | y_i) f_Y(y_i)$$

where y_i s are equally spaced values over the support $[a, b]$

- Consider the following model with $X_3 = 2$ again,

$$\begin{aligned} P &\sim \text{Uniform}(0, 1) \\ X_k | P &\sim \text{Binomial}(k, p) \\ \implies P | X_3 = 2 &\sim \text{Beta}(3, 2) \end{aligned}$$

- If you are Bayes, the integral that you have to deal with is

$$\int_0^1 \binom{3}{2} p^2 (1-p)^1 \cdot 1 dp$$

- In this case, we actually can work out the exact normalising constant easily

$$A = \binom{3}{2} \int_0^1 p^2 (1-p)^1 \cdot 1 dp = \binom{3}{2} \frac{1}{12}$$

- So the exact posterior is given by

$$f_{P|X_3=2}(p \mid 2) = 12p^2(1-P) \quad \text{for } p \in [0, 1]$$

- Let us see how well a grid approximation performs

```
julia> range(0, length=6, stop=1)
```

```
0.0:0.2:1.0
```


- Put the last answer, which is a range object of the grid values, into an array

```
julia> p_grid = collect(ans[2:6])
```

```
5-element Array{Float64,1}:
```

```
0.2
```

```
0.4
```

```
0.6
```

```
0.8
```

```
1.0
```

- Assume uniform prior

```
julia> prior = fill(1, 5)
```

```
5-element Array{Int64,1}:
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

- Initialise an array for likelihood

```
julia> likelihood = Array{Float64, 1}(undef, 5);
```

- You need the next two lines only if you haven't installed **Distributions**

```
julia> using Pkg  
julia> Pkg.add("Distributions")
```

- Load the package **Distributions**

```
julia> using Distributions
```

- Compute the likelihood at each point in the grid

```
julia> k = 3; x = 2;  
julia> for i in 1:1:5  
        p = p_grid[i];  
        d = Binomial(k, p);  
        likelihood[i] = pdf(d, x);  
    end
```

- Compute the product of likelihood and prior

```
julia> unnormalised_posterior = likelihood .* prior
```

```
5-element Array{Float64,1}:  
 0.096000000000000002  
 0.288000000000000014  
 0.43199999999999994  
 0.384  
 0.0
```

- Normalising constant

```
julia> A = (1-0) * sum(unnormalised_posterior) / 5
```

```
0.240000000000000005
```

- Posterior Values

```
julia> posterior = unnormalised_posterior / A;
```

- We expect our approximation on the normalising constant to improve

```
julia> A
```

```
0.240000000000000005
```

when the number of points in the grid increases, we have 5 points currently.

- With 10 points, we have

```
julia> A
```

```
0.2475
```

- With 100 points, we have

```
julia> A
```

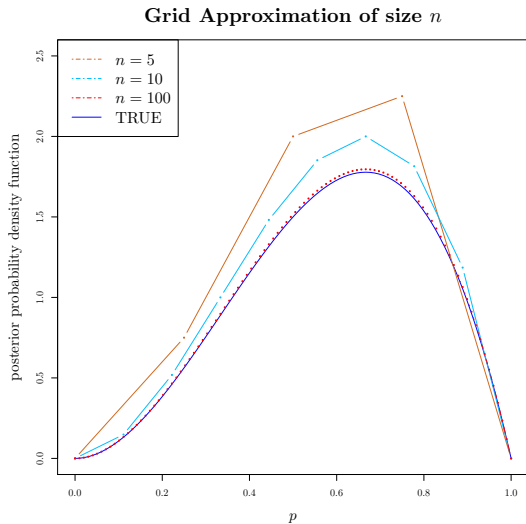
```
0.24997500000000003
```

- With 10000 points, we have

```
julia> A
```

```
0.24999999997500003
```

- The approximation seems to be reasonably good when n reaches 100.



- Now consider the following case again,

$$f_{Y|X} \propto \exp\left(-\frac{(x-y)^2}{2}\right) \times \frac{1}{1+y^2}$$

which requires to compute the following integral to obtain the posterior

$$A = \int_{-\infty}^{\infty} \exp\left(-\frac{(x-y)^2}{2}\right) \times \frac{1}{1+y^2} dy$$

Q: Do you see an issue of using a grid approximation in this case?

- We could resolve it by splitting the integral and transforming y

$$A = \int_{-\infty}^c \exp\left(-\frac{(x-y)^2}{2}\right) \times \frac{1}{1+y^2} dy \quad \text{for some } c \in \mathbb{R}. \\ + \int_c^{\infty} \exp\left(-\frac{(x-y)^2}{2}\right) \times \frac{1}{1+y^2} dy$$

so that we only have to deal with integrals over a finite interval.

- That is, we have to choose some c value and find two transformations,

$$u = g(y) \implies y = g^{-1}(u) \quad \text{and} \quad v = h(y) \implies y = h^{-1}(v)$$

one for each of the followings

$$A_1 = \int_{-\infty}^c f(y) dy \quad \text{and} \quad A_2 = \int_c^{\infty} f(y) dy$$

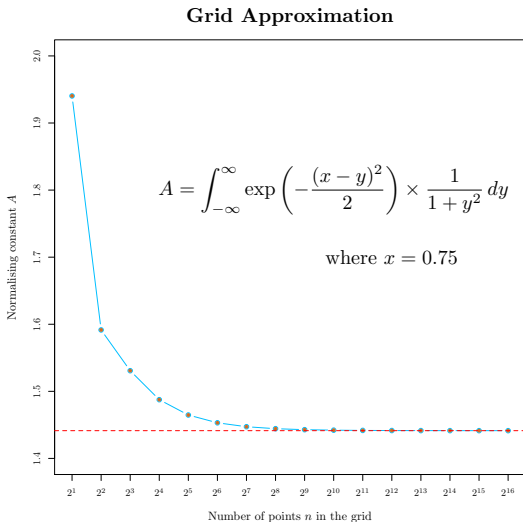
so that the two integrals are replaced by

$$A_1 = \int_a^b \left(\frac{f}{g'} \right) \circ g^{-1}(u) du \quad \text{and} \quad A_2 = \int_d^e \left(\frac{f}{h'} \right) \circ h^{-1}(v) dv$$

where a , b , c and d are finite

$$\begin{aligned} b &= g(c) & e &= \lim_{y \rightarrow \infty} h(y) \\ a &= \lim_{y \rightarrow -\infty} g(y) & d &= h(c) \end{aligned} \quad \text{and}$$

- The approximation seems to be reasonably good when n reaches $2^{10} = 1024$.



Q: Can we do better than a simple grid approximation?

- A simple grid approximation can be understood as a weighted sum

$$\int_a^b f(x) dx \approx \frac{b-a}{n} \sum_{i=1}^n f(x_i) = \sum_{i=1}^n w_i f(x_i)$$

where the weight is chosen to be the same

$$w_i = \frac{b-a}{n}$$

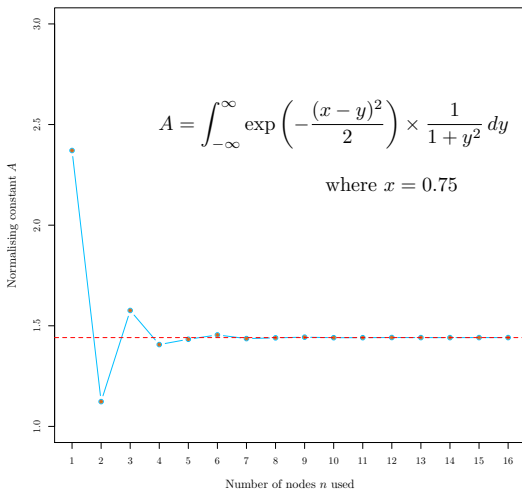
and the points x_i s are chosen to be equally spaced.

- **Gaussian quadrature** chooses the points for evaluation in an optimal, rather than equally spaced, way. The points, x_i , which are known as **nodes**, are chosen along with the weights w_i to minimise the error in general obtained in

$$\int_a^b f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

- Pay attention to the scale in x -axis, Gauss has a lot to offer as usual!

Gauss Legendre Quadrature



- Just like optimisation, quadrature in detail will not be covered in this course, but you need know Julia can find the nodes and weights for you easily.

```
julia> using FastGaussQuadrature
julia> # You need to install it if you have not

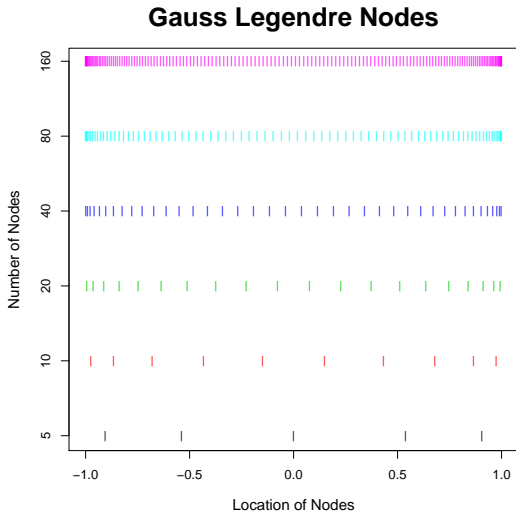
julia> x = 0.75; n = 5; # Number of nodes

julia> nodes, weights = gausslegendre(n);

julia> nodes
```

```
5-element Array{Float64,1}:
-0.906179845938664
-0.5384693101056831
 0.0
 0.5384693101056831
 0.906179845938664
```

- Note Gauss put more nodes near the ends, there is more than luck in play.



- Neither nodes nor weights depend on the integrand.

```
julia> weights
```

```
5-element Array{Float64,1}:  
 0.23692688505618908  
 0.47862867049936647  
 0.5688888888888889  
 0.47862867049936647  
 0.23692688505618908
```

```
julia> u_integrand = Array{Float64, 1}(undef, n);
```

```
julia> for i in 1:n  
    u = nodes[i];  
    ginv = (u-1)/(u+1);  
    logscale = log(1/2) + 2*log(1-ginv) -  
               (x-ginv)^2/2 - log(1+ginv^2);  
    u_integrand[i] = exp(logscale);  
end
```

```
julia> v_integrand = Array{Float64, 1}(undef, n);

julia> for i in 1:1:n
    v = nodes[i];
    hinv = (1-v)/(1+v);
    logscale = log(1/2) + 2*log(1+hinv) -
        (x-hinv)^2/2 - log(1+hinv^2);
    v_integrand[i] = exp(logscale);
end
```

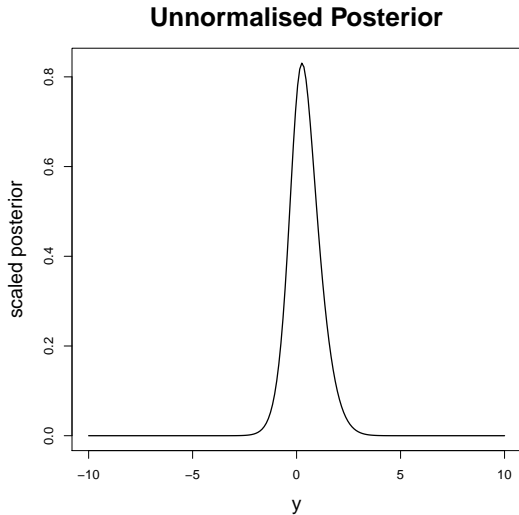
```
julia> A1 = sum(weights.*u_integrand);
```

```
julia> A2 = sum(weights.*v_integrand);
```

```
julia> A = A1 + A2
```

```
1.4331167574845127
```

Q: What can we do without Julia? What would you do if you were Laplace?



- Laplace noticed three things that are common in a lot of the problems:
 1. Posterior is smooth and unimodal, or at least well separated modes.
 2. Has a strictly negative second derivative at its modal value \hat{y}_{MAP} .
 3. The integrand peaks sharply around \hat{y}_{MAP} when there is enough data.
- In general, suppose the integral we have to compute is in the following form

$$I(s) = \int_{-\infty}^{\infty} \exp(-sh(y)) dy$$

and $s \rightarrow \infty$ as the number of data $\rightarrow \infty$, for example,

$$\mathcal{L}(\mu; \bar{x}, \sigma^2) \propto \exp\left(-\frac{(\bar{x} - \mu)^2}{2(\sigma/\sqrt{n})^2}\right) = \exp\left(-\frac{1}{2(\sigma/\sqrt{n})^2}(\bar{x} - \mu)^2\right)$$

- That is, it is going to be a method when we have sufficiently large dataset.

- Suppose $h(y)$ is sufficiently smooth so that Taylor's theorem is applicable,

$$h(y) \approx h(\hat{y}) + \frac{1}{2}h''(\hat{y})(y - \hat{y})^2 + \frac{1}{6}h^{(3)}(\hat{y})(y - \hat{y})^3 + \frac{1}{24}h^{(4)}(\hat{y})(y - \hat{y})^4$$

where \hat{y} denotes the MAP estimate, so the 1st order derivative vanished.

- In terms of the integral, we have

$$\begin{aligned} I(s) &= \int_{-\infty}^{\infty} \exp(-sh(y)) dy \\ &\approx \int_{-\infty}^{\infty} \exp \left[-s \left(h(\hat{y}) + \frac{1}{2}h''(\hat{y})(y - \hat{y})^2 \right. \right. \\ &\quad \left. \left. + \frac{1}{6}h^{(3)}(\hat{y})(y - \hat{y})^3 + \frac{1}{24}h^{(4)}(\hat{y})(y - \hat{y})^4 \right) \right] dy \\ &= e^{-s\hat{h}} \int_{-\infty}^{\infty} e^{-s\hat{h}^{(2)}u^2/2} \exp \left(-\frac{s}{6}\hat{h}^{(3)}u^3 - \frac{s}{24}\hat{h}^{(4)}u^4 \right) du \end{aligned}$$

where the hat denotes the function evaluated at \hat{y} , and $u = y - \hat{y}$.

- Invoking Taylor's again, this time on the exponential function around zero,

$$\begin{aligned}
 J(u) &= \exp\left(-\frac{s}{6}\hat{h}^{(3)}u^3 - \frac{s}{24}\hat{h}^{(4)}u^4\right) \\
 &\approx 1 - \frac{s}{6}\hat{h}^{(3)}u^3 - \frac{s}{24}\hat{h}^{(4)}u^4 + \frac{1}{2}\left(\frac{s}{6}\hat{h}^{(3)}u^3 + \frac{s}{24}\hat{h}^{(4)}u^4\right)^2 \\
 &= 1 - \frac{s}{24}\hat{h}^{(4)}u^4 + \frac{s^2}{72}\left(\hat{h}^{(3)}\right)^2u^6 + \frac{s^2}{1052}\left(\hat{h}^{(4)}\right)^2u^8 + \text{odd powers}
 \end{aligned}$$

- Since the Gaussian function is an even function, the odd powers will vanish

$$I(s) \approx e^{-s\hat{h}} \int_{-\infty}^{\infty} J(u) \cdot e^{-s\hat{h}^{(2)}u^2/2} du$$

- The reason for using Taylor's again is to be able to use the moment formula

$$\int_{-\infty}^{\infty} x^{2m} e^{-\alpha x^2} dx = \frac{2m!}{m!2^{2m}} \sqrt{\pi} \alpha^{-(m+1)/2}$$

- Using this moment formula and setting $\sigma^2 = \frac{1}{\hat{h}^{(2)}}$, we have

$$I(s) \approx e^{-s\hat{h}} \sqrt{2\pi\sigma} s^{-1/2} \left(1 + \frac{5\sigma^6}{24s} \left(\hat{h}^{(3)} \right)^2 - \frac{\sigma^4}{8s} \hat{h}^{(4)} \right)$$

- If only the first order term in the above approximation is used

$$I(s) = \int_{-\infty}^{\infty} \exp(-sh(y)) dy \approx e^{-s\hat{h}} \sqrt{2\pi\sigma} s^{-1/2} \quad \text{for a large } s$$

then the approximation is known as [Laplace approximation](#).

- Essentially, many posteriors can be approximated locally near the mode by a normal distribution, when we have enough data, the posterior is dominated near the mode, thus the approximation becomes better and better $s \rightarrow \infty$.
- It can be extended to high dimensional cases as well as the following case

$$I(s) = \int_{-\infty}^{\infty} g(y) \exp(-sh(y)) dy$$