Summary Report [Week 5]. MLE vs MAP

20190106 KimByungjun (김병준)

● Exercises

이번 Exercise 에선 coin toss 상황에서 다른 prior(uniform, gaussian, beta) distribution 에 대해 MAP 가 어떻게 변하는 지 scipy 라이브러리를 이용하여 구현하는 게 목표이다. MAP 에 대한 설명 이전에 **1), 2)**는 MLE 에 대한 내용이다.

1) Implement likelihood

```python
def likelihood(x, n, p):
    ########## implement here ##########
    return (p**x)*((1-p)**(n-x))
    ####################################
```

[Fig 1] Code for Exercise 1 (x: head, n: total toss, p: probability 1d array)

Likelihood 란, observed data 에 대해 주어진 data 를 가장 잘 표현하는 parameter 를 구하는 방법 중 하나로, observed data 가 어떤 확률분포에서 나왔는가에 대응되는 값이다. Coin toss 의 경우, x=1(head)가 나올 확률 p, x=0(tail)이 나올 확률 1-p 를 가지므로, Bernoulli distribution 을 따른다고 볼 수 있으며 단일 시행에 대해 $L(x_i|p) = p^{x_i}(1-p)^{x_i}$ $(x_i = 0,1)$의 likelihood 를 가진다. n 회의 coin toss 를 시행 시, likelihood 는 $L(x|p) = \prod_i L(x_i|p) = \prod_i p^{x_i}(1-p)^{x_i} = p^x(1-p)^{n-x}$ $(x\text{: }head$ 횟수$)$ 로 나타낼 수 있다. 이에 대한 code 는 **[Fig 1]**과 같다.

Explanation: code 그대로 $p^x(1-p)^{n-x}$ 를 반환한다. (* p 는 [0,1] 내 동일 간격의 10001 개 values 에 대한 np.darray 이다)

2) Implement calculate_MLE

```python
def calculate_MLE(x, n, p):
    ########## implement here ##########
    L = likelihood(x,n,p)
    return p[np.argmax(L)]
    ####################################
## print(calculate_MLE(50,100,p))
```

[Fig 2] Code for Exercise 2 (x: head, n: total toss, p: probability 1d array)

1)에서 구한 likelihood 함수에 np.ndarray type 의 parameter p 를 input 에 넣으면, 각 element 에 대한 likelihood 를 계산 및 np.ndarray type 으로 return 한다. 그 중, 가장 큰(maximum) 값을 가진 p 를 추출하는 함수이며, 그 p 가 likelihood 를 maximize 하는 parameter 이라 이 과정을 MLE(Maximum Likelihood Estimation)라 부른다. 해당 p 가 해당 observed data 가 가장 잘 나오는 경우에 대응되는 것이다. 이에 대한 code 는 **[Fig 2]**와 같다.

Explanation: 변수 L 이 **1)**에서 구한 likelihood 함수의 np.ndarray type 1D output 이다. np.argmax 는 input ndarray 중 가장 큰 값을 가지는 element 의 index 를 반환한다. 구하고자 하는 것은 그 maximum 상황에서의 parameter p 값이므로, p 배열에서 np.argmax 에 해당하는 index 의 element 를 return 하면 끝나므로, 'return p[np.argmax(L)]' 로 정리할 수 있다.

```python
def prior(p, mode, param_1=1, param_2=1):
    if mode == 'uniform':
        return 1.0  # Uniform prior
    elif mode == 'beta':
        return beta.pdf(p, param_1, param_2)  # Beta prior
    elif mode == 'Gaussian':
        return norm.pdf(p, param_1, param_2)  # Gaussian prior
```

[Fig 3] prior function (includes uniform, beta, gaussian distribution)

Posterior 란, data 를 관찰한 후 그 data 가 특정 model 에서 발생했을 확률이며 사전 확률(prior probability)가 uniform 하지 않은 경우에 대해 보정이 적용된 것이 사후 확률 posterior probability 이다. 다시 말해, 위 MLE 는 p 에 대한 분포(prior)를 고려하지 않은 경우이고 MAP 와 이점이 크게 다르다. 앞서 말한 prior 에 대한 code 가 **[Fig 3]**과 같다. mode parameter 에는 uniform, beta, gaussian 이 들어갈 수 있으며, uniform 인 경우가 MAP 와 MLE 가 같은 경우에 대응된다. Beta distribution 과 gaussian distribution 의 경우, p 배열을 input 으로 받아, param_1, param_2 의 parameter 로 p 배열에 대한 prior 을 적용한다. 이 때, beta 의 경우 $Beta(p;\, \alpha, \beta)$에서 α, β가, gaussian 인 경우 $N(p; \mu, \sigma^2)$에서 μ, σ가 param_1, param_2 에 해당한다.

3) Implement unnormalized_posterior

Posterior 는 data 를 observed 하였다는 사후 조건의 개념이므로, Bayes' theorem 에 따라 $p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$ 로 계산할 수 있다 (이 Summary 에서는, θ는 1D array p 의 원소들에 해당한다). 이 때, $p(x|\theta)$ 는 위에서 구한 likelihood $L(x|\theta)$에 대응되는 비례 관계의 항이며, $p(x)$는 parameter θ와 무관한 항이므로, 따라 $p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \propto L(x|\theta)p(x)$ 라는

비례 관계가 성립한다. 여기서 $L(x|\theta)p(x)$ 는 unnormalized posterior(integral 시 1 이 되지 않은 unnormalized 상태이면서 posterior 와 비례한 식) 라 부를 수 있다.

```python
def unnormalized_posterior(x, n, p, mode, param_1=1, param_2=1):
    ########## implement here ##########
    return likelihood(x,n,p)*prior(p, mode, param_1, param_2)
    ###########################
```

[Fig 4] Code for Exercise 3 (meaning of the same parameter names by likelihood & prior function)

Explanation: [Fig 4]에 implement 되어 있는 앞선 설명에 따라 unnormalized_posterior 함수는 **1)**에서의 likelihood(x,n,p)와 **[Fig 3]**의 prior(p,mode,param_1,param_2)의 곱으로 표현할 수 있다. 단순히 해당 값을 return 하면 되며, 총 6 개의 parameters 들을 받는다.

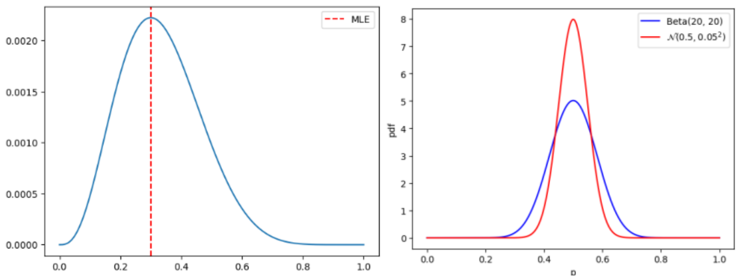
4) Implement calculate_MAP

3)에서 구현한 unnormalized_posterior 은 인자로 받은 p array 각각의 elements 에 대한 $L(x|\theta)p(x)$ 값으로 구성된 1D array 를 output 으로 반환한다. 그 1D array 내 가장 큰 unnormalized_posterior 값을 만들어내는 parameter p 를 MAP 라 한다. 즉, MLE 와 똑같이 np.argmax 함수를 이용하여 maximizer p 를 p array 내에서 찾아내면 된다. Code 진행은 **[Fig 5]**와 같다.

```python
def calculate_MAP(x, n, p, mode, param_1=1, param_2=1):
    ########## implement here ##########
    return p[np.argmax(unnormalized_posterior(x,n,p,mode,param_1,param_2))]
    ####################################
```

[Fig 5] Code for Exercise 4 (same parameters as unnormalized posterior function)

Explanation: unnormalized_posterior 의 maximizer p 를 구하는 것이므로, **2)**와 code flow 는 동일하다.



[Fig 6-1] (Left) Likelihood(blue) and MLE(red dot)
[Fig 6-2] (Right) Gaussian(red) and Beta(blue) distribution

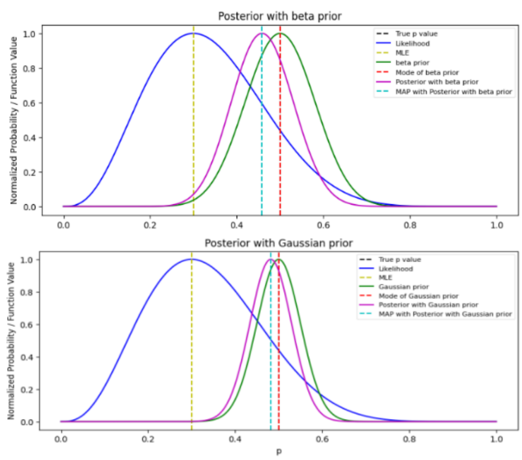
● Discussion

1) Fair coin tosses in small number / large number

10 회의 coin toss 에 대해 fair coin(양면 모두 p_real=0.5) 특정 seed(random 한 상황을 부여하는 난수)를 주고 MLE 와 각기 다른 prior 들을 가진 MAP 들을 비교해볼 것이며, coin 을 10 회가 아닌 100 회를 던졌을 때와는 어떤 차이를 보이는 지 분석하고자 한다. 해당 discussion 에서 seed 는 head 가 3 회 나온 경우가 조성되도록 값이 지정되었다.

MLE 는 logarithm likelihood $ln\,L(x|p) = x\,ln\,p + (n-x)\,ln(1-p)$를 p 에 대해 미분하여 maximizer p 를 구하며, $\frac{d}{dp}(ln\,L) = \frac{x}{p} - \frac{n-x}{1-p} = 0, \boldsymbol{p} = \frac{x}{n}$가 analytic method 로 구한 *maximizer 이다. 실제로, 2)에서 구한 calculate_MLE 함수를 이용하면, **[Fig 6-1]**과 같이 p=0.3 의 MLE 가 도출되었음을 알 수 있다.

이번엔 MAP 계산을 위해 **[Fig 6-2]** 와 같은 bell-shaped 모양의 Gaussian($N(0.5, 0.05^2)$), Beta($Beta(20,20)$) 분포, 그리고 Uniform($U(0,1)$)의 prior 을 사용하며, MAP 와 MLE 를 비교하겠다. Uniform prior 의 경우, 앞서 설명한대로 MLE 와 MAP 에 따른 p 값이 동일한 0.3 의 값으로 나오며, 다른 prior 의 경우 **[Fig 7]**과 같은 모습을 보인다(설명에 앞서, 해석의 편의성을 위해 graph 들의 y 값에 대해 normalization 이 적용되었음을 알린다). 0.3 이라는 동일한 maximizer p 의 파란색 likelihood 와는 거리가 많이 먼 posterior(핑크색) 분포를 보이고 있으며, MAP 값은 beta 의 경우 약 0.4583, gaussian 의 경우 0.4818 의 MLE=0.3 과는 다른 값을 가진다(**[Fig 7]**의 하늘색 점선 참고).



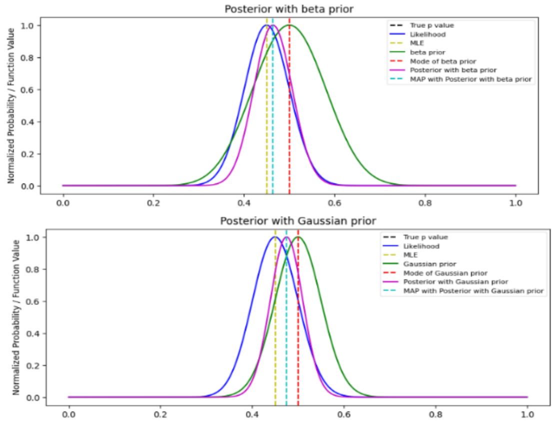
[Fig 7] Posterior(pink), Likelihood(blue) Graph with beta (upper) and gaussian (lower) prior distribution (n=10, x=3)

두 Beta 분포의 곱(합성)은 다른 Beta 분포를 도출한다는 성질에 따라, 여기서 likelihood $L(x|p) = p^x(1-p)^{n-x} \propto Beta(x+1, n-x+1)$ 또한 일종의 beta 함수 형태를 가지므로, prior 가 beta 분포인 경우엔 posterior 도 beta 분포를 띈다. **[Fig 7]**의 upper graph 에서 파란색 likelihood 와 초록색 prior 의 곱이 분홍색 posterior 을 만드는 것이며, 그래서 posterior graph 가 likelihood 와

* 유일한 극대이자, $\boldsymbol{p} = \frac{x}{n}$ 이전으로는 증가, 후로는 감소하는 graph 이므로 최대이다.

prior graph 사이에 형성되며 MAP 는 MLE 인 0.3 과, prior 중심인 0.5 사이에 형성된 것이다. Gaussian 의 경우 exact 한 분포를 구할 순 없지만, likelihood graph 와 prior graph 의 곱이 unnormalized posterior($L(x|\theta)p(x)$)임은 동일하므로, 그 둘 사이의 posterior graph 를 얻었다고 유추할 수 있고, beta prior 인 경우보다 더 sharp 한 prior 를 gaussian prior 에서 갖기 때문에, posterior 가 likelihood 보다 prior 쪽에 더 가깝게 위치, 즉 MAP 의 값 측면에서 gaussian 인 경우가 beta 인 경우보다 더 크게 나온 사실과 부합한다.

n (#coin toss)값을 10 에서 100 으로 늘린 경우, **[Fig 8]**의 result 를 얻을 수 있다. **[Fig 8]**의 결과는 **100 번의 fair coin toss 중 head 가 45 번 나온 경우를** 바탕으로 나온 결과임에 유의하자. MLE 는 0.45 가 나오며, n 이 커지면서 likelihood graph 가 p=0.45 를 중심으로 narrow 해졌음을 알 수 있고, 그렇기 때문에 이전과 동일한 prior 을 곱하면, **[Fig 7]**과 같은 맥락으로 봉우리가 있는 posterior 을 얻을 수 있다. MAP 는 beta 에선 0.4638, gaussian 에선 0.475 이 나왔다. 추가로, MLE 의 변화량이 MAP 보다 더 dramatic 함을 확인할 수 있으며, n 이 더 커진다면, MLE 는 실제 coin 의 확률(p_real)인 0.5 에 근접해질 것이고, MAP 또한 0.5 에 같이 근접해짐을 알 수 있다.

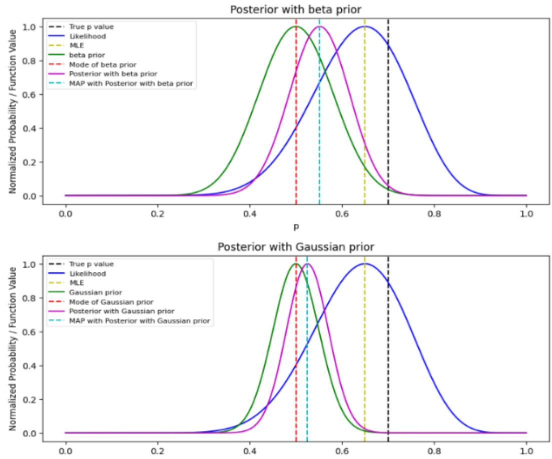


[Fig 8] Resulting graph in large number of coin toss (n=100, x=45) (same labels as [Fig 7])

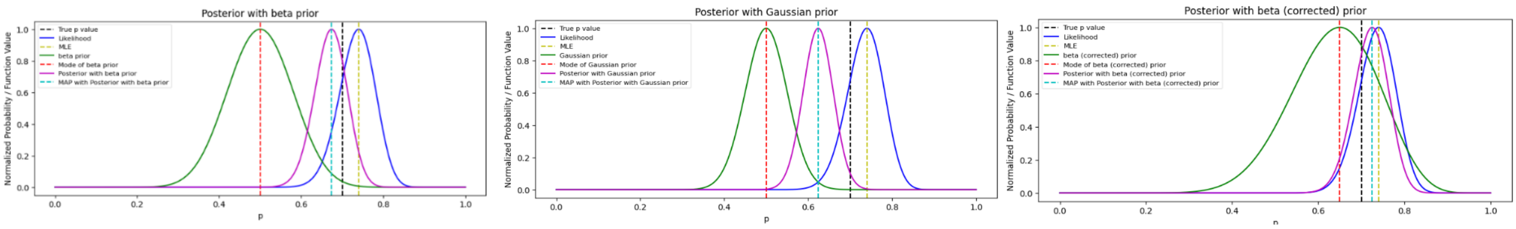
2) Unfair coin tosses in medium number / large number with prior correction

이번엔 p_real=0.7 인 경우에 대해 observed data 가 coin toss n=20, head x=13 인 경우 결과는 어떠할 지 확인해보고자 한다. 현재 prior 의 parameter 는 **1)**과 똑같이 Gaussian($N(0.5, 0.05^2)$), Beta($Beta(20,20)$) 이다. 결과는 **[Fig 9]**와 같으며, MAP 의 경우 beta 에선 0.5517, gaussian 에선 0.5251 이 나왔다.

p_real 인 0.7(검은색 점선)과 MAP(하늘색 점선)의 위치가 두 경우 모두 크게 다른 것을 알 수 있는데, prior 의 parameter 설정이 잘못되었음을 알 수 있다. Beta 분포에서 현재 n=20, x=13 이므로 $Beta(x+1=14, n-x+1=8)$를 가지며 Beta 분포에서 mode 란 maximum 이 생기는 p 를 의미하며, 앞서 말한 MLE 와 동일한 개념이다. Mode 의 값 또한 MLE 와 동등한 derivation 및 value 인 $p=\frac{x}{n}$ 을 가지며, 여기선 prior_mode 값을 $\frac{13}{20}=0.65$ 로 바꿔(correction)줘야 한다.



[Fig 9] Resulting graph in unfair coin toss case (p_real=0.7, n= 20, x=13) (same labels as [Fig 7])



[Fig 10] Resulting graph in unfair coin toss case (p_real=0.7, n= 100, x=74) (same labels as [Fig 7])
(left) : beta prior without correction ($Beta(20,20)$), (middle) gaussian prior ($N(0.5, 0.05^2)$), (right) beta prior with correction ($Beta(14,8)$),

해당 correction 을 적용한 상태에서 n 을 20 에서 100 으로 늘리면서 head x 가 74 인 상황(p_real 은 0.7 그대로)을 관찰했을 때, MLE=0.74 이며 posterior 계산 시 **[Fig 10]**과 같은 결과를 얻을 수 있다. Correction 이 적용된 beta prior 을 보면 p=0.5 를 기준으로 symmetric 했던 분포가 이제는 p=0.65 에서 maxima 를 가지는 오른쪽으로 쏠린 beta prior(초록색) 분포를 보인다. 우편향된 prior 와 역시 우편향된 likelihood 의 중첩으로, posterior 역시 다른 case 들과 다르게 p=0.7~0.8 부근에 봉우리를 자리잡고 있다. **[Fig 10]**에서 순서대로 MAP 는 0.6739, 0.6238, 0.725 가 나왔으며, correction 이 적용된 beta prior 의 posterior case 가 가장 p_real 에 근접한 model 임을 결론 지을 수 있다.