IIT ROORKEE

FREE ONLINE EDUCATION swayam शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSE

Maximum Likelihood Estimation-II

Dr. A. Ramesh

DEPARTMENT OF MANAGEMENT STUDIES

Agenda

- This lecture will provide understanding intuition behind the MLE using Theory and examples.

Example1: **Estimation of parameters of normal distribution**

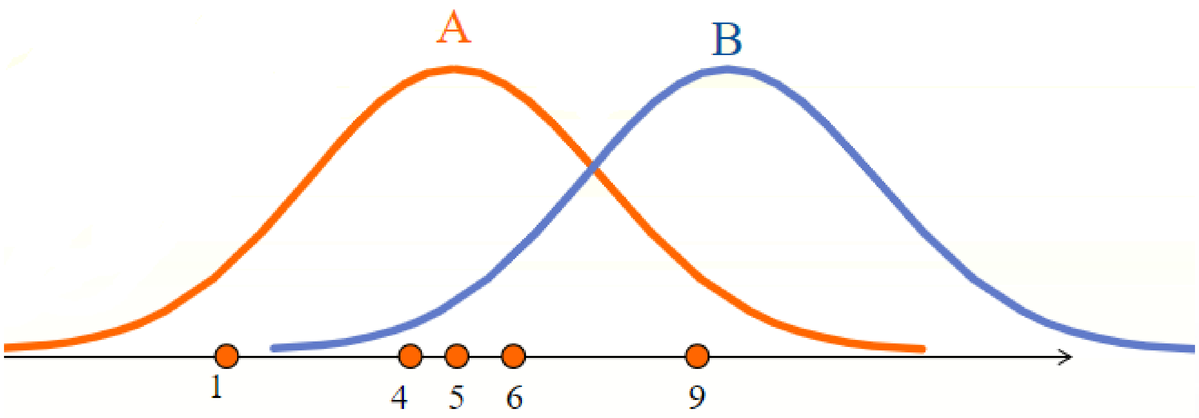
- Let us explain basic idea of MLE using simple problems.
- Let us make assumption that variable x follows normal distributed
- Density function of normal distribution with mean μ and variance σ^2 is given by:

Id	x
1	1
2	4
3	5
4	6
35	9

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \quad \text{for} \quad -\infty < x < \infty$$

Example 1: **Estimation of parameters of normal distribution**

- The data is plotted on a horizontal line
- Think which distribution, either A or B, is more likely to have generated the data?



Interpretation

- Answer to this question is A, because the data are cluster around the center of the distribution A, but not around the center of the distribution B
- This example illustrate that, by looking at the data, it is possible to find the distribution that is most likely to have generated the data
- Now, I will explain exactly how to find the distribution in practice

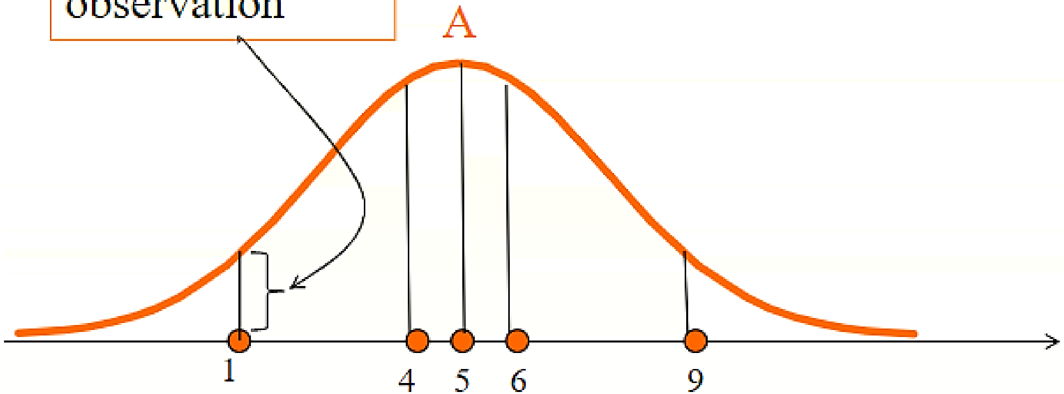
The illustration of the estimation procedure

- MLE starts with computing the likelihood contribution of each observation
- The likelihood contribution is the height of the density function.
- We use L_i to denote the likelihood contribution of i^{th} observation.

Graphical illustration of likelihood contribution

The likelihood contribution of the first observation $= L_1 = \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-(1-\mu)^2/2\sigma^2}$

Data value



The illustration of the estimation procedure

- Then, you multiply the likelihood contributions of all the observations. this is called the likelihood function. We use the notation L
- Likelihood function L= $\prod_{i=1}^{n} L_i$

This notation means you multiply from i= 1 through n

- In our example, n= 5

The illustration of the estimation procedure

- In our example, the likelihood function looks like:

$$L(\mu,\sigma) = \prod_{i=1}^{5} L_i = L_1 \times L_2 \times L_3 \times L_4 \times L_5$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(1-\mu)^2/\sigma^2} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(4-\mu)^2/\sigma^2}$$

$$\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(5-\mu)^2/\sigma^2} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(6-\mu)^2/\sigma^2}$$

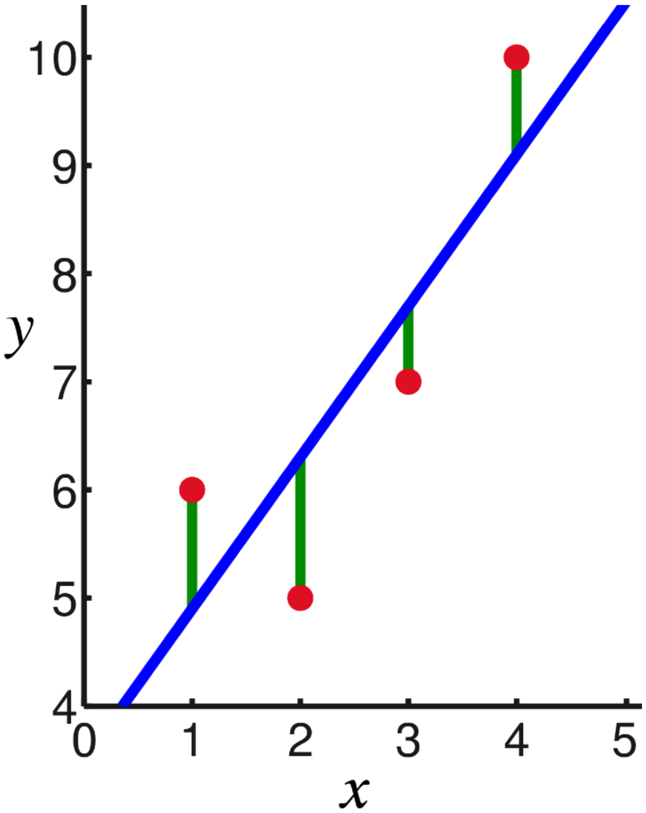
$$\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(9-\mu)^2/\sigma^2}$$

- The likelihood function depends on mean μ and variance σ^2

The illustration of the estimation procedure

- The value of mean μ and σ that maximise the likelihood function is found.
- The values of mean μ and σ which are obtained this way are called the maximum likelihood estimators of mean μ and σ
- Most of the MLE cannot be solved ‘by hand’. Thus, you need to write an iterative procedure to solve it on computer

Method of Least-squares vs MLE



Model for the expectation (fixed part of the model):

$$E[Y_i] = \beta_0 + \beta_1 x_i$$

Residuals: $r_i = y_i - E[Y_i]$

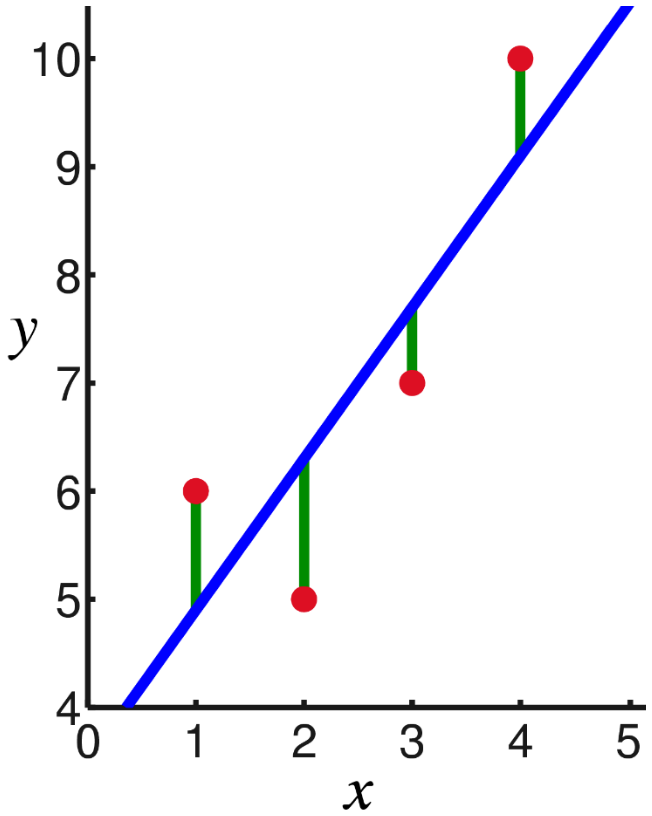
The method of least-squares:

Find the values for the parameters (β_0 and β_1) that makes the sum of the squared residuals (Σr_j^2) as small as possible.

Can only be used when the error term is normal (residuals are assumed to be drawn from a normal distribution)

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \text{ where } \varepsilon_i \sim N(0, \sigma)$$

Method of Least-squares vs MLE



Model for the expectation (fixed part of the model):

$$E[Y_i] = \beta_0 + \beta_1 x_i$$

Residuals: $r_i = y_i - E[Y_i]$

The maximum likelihood method is more general!

- Can be applied to models with any probability distribution

Estimation of Regression Parameter

- We are interested in estimating a model like this:

$$y=\beta_0+\beta_1 x+u$$

- Estimating such a model can be done using MLE

Estimation of Regression Parameters

- Suppose that we have the following data and we are interested in estimating the model: $y=\beta_0+\beta_1 x+u$
- Let us make an assumption that u follows the normal distribution with mean 0 and variance σ^2

Id	Y	X
1	2	1
2	6	4
3	7	5
4	9	6
5	15	9

Estimation of Regression Parameters

- We can write the model as :
  u = $y-(\beta_0+\beta_1 x)$

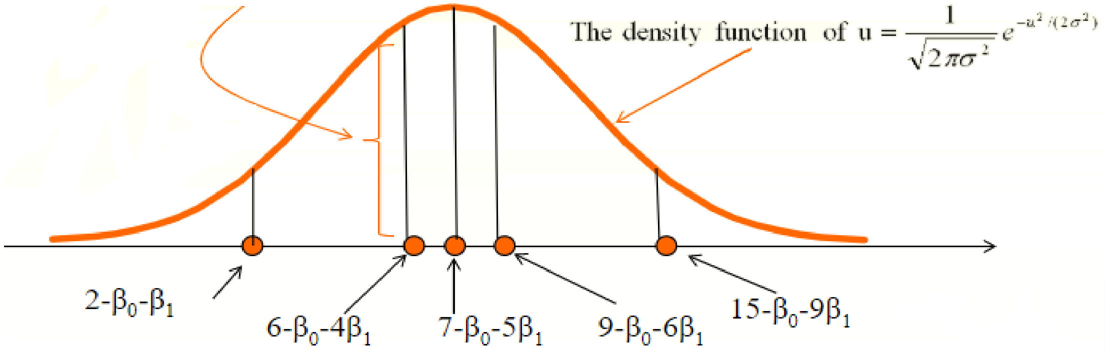
- This means that $y-(\beta_0+\beta_1 x)$ follows the normal distribution with mean 0 and variance σ^2
- The likelihood contribution of each data point is the height of the density function at the data points $(y-\beta_0-\beta_1 x)$

Estimation of Regression Parameters

- The likelihood contribution in this example, of the 2nd observation is given by: $L_2 = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(6-\beta_0-4\beta_1)^2/2\sigma^2}$

Data point

The likelihood contribution of the 2nd observation



Estimation of Regression Parameters

- Then the likelihood function is given by

Id	Y	X
1	2	1
2	6	4
3	7	5
4	9	6
5	15	9

$$L(\beta_0, \beta_1, \sigma) = \prod_{i=1}^{n} L_i = L_1 \times L_2 \times L_3 \times L_4 \times L_5$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left.(2-\beta_0-\beta_1)^2\right/\sigma^2} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left.(6-\beta_0-4\beta_1)^2\right/\sigma^2}$$

$$\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left.(7-\beta_0-5\beta_1)^2\right/\sigma^2} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left.(9-\beta_0-6\beta_1)^2\right/\sigma^2}$$

$$\times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left.(15-\beta_0-9\beta_1)^2\right/\sigma^2}$$

- The likelihood function is a function of β_0 , β_1 and σ.

Estimation of Regression Parameters

- You choose the values of β_0 , β_1 and σ that maximizes the likelihood function.

Python Demo for MLE

In [1]:
```
import numpy as np
from scipy.optimize import minimize
import scipy.stats as stats
```

In [2]:
```
import pandas as pd
tbl = pd.read_excel('mle.xlsx')
tbl
```

Out[2]:

	Id	Y	X
0	1	2	1
1	2	6	4
2	3	7	5
3	4	9	6
4	5	15	9

In [3]:

```
import statsmodels.api as sm
x= tbl['X']
y =tbl['Y']
x2 = sm.add_constant(x)
modl = sm.OLS(y,x2)
modl2 = modl.fit()
print(modl2.summary())
```

```
C:\Users\HP\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The
recated and will be removed in a future version. Please use the pandas.tseries module inste
  from pandas.core import datetools
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.980
Model:                            OLS   Adj. R-squared:                  0.973
Method:                 Least Squares   F-statistic:                     145.9
Date:                Wed, 11 Sep 2019   Prob (F-statistic):            0.00122
Time:                        10:05:16   Log-Likelihood:                -4.5811
No. Observations:                   5   AIC:                             13.16
Df Residuals:                       3   BIC:                             12.38
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.2882      0.755     -0.382      0.728      -2.692       2.115
X              1.6176      0.134     12.079      0.001       1.191       2.044
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   1.405
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.551
Skew:                           0.089   Prob(JB):                        0.759
Kurtosis:                       1.384   Cond. No.                         12.5
==============================================================================
```

b0 = -0.2882 and b1 = 1.6176

Parameter estimation by MLE

In [9]:
```
e=modl2.resid
```

In [10]:
```
e
```

Out[10]:
```
0    0.670588
1   -0.182353
2   -0.800000
3   -0.417647
4    0.729412
dtype: float64
```

In [6]:
```
np.std(e)
```

Out[6]:
```
0.6048820983804831
```

Parameter estimation by MLE

In [11]:
```
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

def lik(parameters):
    m = parameters[0]
    b = parameters[1]
    sigma = parameters[2]
    for i in np.arange(0, len(x)):
        y_exp = m * x + b
    L = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
      (2 * sigma ** 2) * sum((y - y_exp) ** 2))
    return L

x = np.array([1,4,5,6,9])
y = np.array([2,6,7,9,15])
lik_model = minimize(lik, np.array([2,2,2]), method='L-BFGS-B')
```

In [12]:
```
lik_model
```

Out[12]:
```
      fun: 4.581084072762135
 hess_inv: <3x3 LbfgsInvHessProduct with dtype=float64>
      jac: array([1.24344979e-06, 2.84217094e-06, 1.33226763e-06])
  message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL'
     nfev: 108
      nit: 17
   status: 0
  success: True
        x: array([ 1.61764689, -0.28823426,  0.60488214])
```

Example 2

In [1]:
```
import numpy as np
from scipy.optimize import minimize
import scipy.stats as stats
```

In [2]:
```
import pandas as pd
tbl = pd.read_excel('regcar.xlsx')
tbl
```

Out[2]:

	TV Ads	car Sold
0	1	14
1	3	24
2	2	18
3	1	17
4	3	27

In [3]:
```python
import statsmodels.api as sm
x= tbl['TV Ads']
y =tbl['car Sold']
x2 = sm.add_constant(x)
modl = sm.OLS(y,x2)
modl2 = modl.fit()
print(modl2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               car Sold   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.836
Method:                 Least Squares   F-statistic:                     21.43
Date:                Wed, 11 Sep 2019   Prob (F-statistic):             0.0190
Time:                        11:11:23   Log-Likelihood:                -9.6687
No. Observations:                   5   AIC:                             23.34
Df Residuals:                       3   BIC:                             22.56
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         10.0000      2.366      4.226      0.024       2.469      17.531
TV Ads         5.0000      1.080      4.629      0.019       1.563       8.437
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   1.214
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.674
Skew:                           0.256   Prob(JB):                        0.714
Kurtosis:                       1.276   Cond. No.                         6.33
==============================================================================
```

b0 = 10 and b1 = 5

In [4]:
```
e=modl2.resid
```

In [5]:
```
e
```

Out[5]:
```
0   -1.0
1   -1.0
2   -2.0
3    2.0
4    2.0
dtype: float64
```

In [9]:
```
np.std(e)
```

Out[9]:
```
1.6733200530681507
```

In [10]:
```python
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

def lik(parameters):
    m = parameters[0]
    b = parameters[1]
    sigma = parameters[2]
    for i in np.arange(0, len(x)):
        y_exp = m * x + b
    L = (len(x)/2 * np.log(2 * np.pi) + len(x)/2 * np.log(sigma ** 2) + 1 /
         (2 * sigma ** 2) * sum((y - y_exp) ** 2))
    return L

x = np.array([1,3,2,1,3])
y = np.array([14,24,18,17,27])
lik_model = minimize(lik, np.array([2,2,2]), method='L-BFGS-B')
```

In [11]:
```python
lik_model
```

Out[11]:
```
      fun: 9.668741208976929
 hess_inv: <3x3 LbfgsInvHessProduct with dtype=float64>
      jac: array([-5.32907052e-07, -1.77635684e-07, -2.30926389e-06])
  message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL'
     nfev: 104
      nit: 22
   status: 0
  success: True
        x: array([ 5.        ,  9.99999989, -1.67332066])
```

Thank you