Team members : Sathiya Ramesh, Pradheep Krishna Muthukrishnan Padmanabhan, Naresh Kumar Gurulingan

Task1

Implement in Python (you can use SciPy library) the Maximum Likelihood Estimator to estimate the parameters for example mean and variance of some data. Your steps are:

- Create a data set:
  - Set x-values for example: *x = np.linspace(0, 100, num=100)*,
  - Set observed y-values using a known slope (1.4), intercept (4), and sd (3), for example *y = 4 + 1.4x + np.random.normal(0, 3, 100)*
- Create a likelihood function which arguments is a list of initial parameters
- Test this function on various data sets (Hint: you can use minimize from scipy.optimize and scipy.stats to compute the negative log-likelihood)

In [1]:
```
# https://stackoverflow.com/questions/7718034/maximum-likelihood-estimate-pseudocode
# http://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/06/lecture-06.pdf
import numpy as np
from scipy.optimize import minimize
import scipy.stats as stats
import time
```

In [2]:
```
class MaximumLikelihoodEstimator:

    def __init__(self, x, y_observed):
        self.x = x
        self.y_observed = y_observed

    def calc_log_likelihood(self, args):
        y_intercept = args[0]
        slope = args[1]
        sigma = args[2]

        y_prediction = y_intercept + slope*x
        log_likelihood = -np.sum(stats.norm.logpdf(y, loc=y_prediction, scale=sigma))
        return(log_likelihood)

    def predict(self, initial_parameters):
        results = minimize(self.calc_log_likelihood, initial_parameters, method='nelder-mead')
        return results
```

In [3]:
```
#Testing the estimator with 20 dataset
x = np.linspace(0, 100, num=100)
initial_parameters = [1, 1, 1]
y_intercepts = np.arange(1, 21, 1)
slopes = np.random.rand(20) * 5
sigma = np.random.rand(20)

y_intercepts = [4] + list(y_intercepts) # example y_intercept appended t
o list
slopes = [1.4] + list(slopes) # example slopes appended to list
sigma = [3] + list(sigma) # example sigma appended to list

print('Example y_intercept {}, slope {}, sigma {} appended to dataset ge
neration'
      .format(y_intercepts[0], slopes[0], sigma[0]))
print()

difference = list()
for inter, slope, sig in zip(y_intercepts, slopes, sigma):

    y = inter + slope * x + np.random.normal(0, sig, 100)
    estimator = MaximumLikelihoodEstimator(x, y)
    result = estimator.predict(initial_parameters)
    difference.append(np.abs([inter, slope, sig] - result.x))

difference = np.array(difference)
# average error callculation for  all dataset
print('Average difference between in y_intercepts {}'.format(np.mean(dif
ference[:,0])))
print('Average difference between in slopes {}'.format(np.mean(differenc
e[:,1])))
print('Average difference between in sigmas {}'.format(np.mean(differenc
e[:,2])))
```

```
Example y_intercept 4, slope 1.4, sigma 3 appended to dataset generation

Average difference between in y_intercepts 0.09161781694874446
Average difference between in slopes 0.0015433256685622586
Average difference between in sigmas 0.019451842436567807
```