Machine Learning
MLE & MAP in Python

AmirReza Azari
99101087

Objective

This exercise will help you gain a deeper understanding of, and insights into, Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP) estimation

$Maximum\ Likelihood\ Estimation\ (MLE)\ and\ Maximum\ A\ Posteriori\ (MAP)$:) \ Let's say you have a barrel of apples that are all different sizes. You pick an apple at random, and you want to know its weight. Unfortunately, all you have is a broken scale. answer the questions below.

1) For the sake of this section, lets imagine a farmer tells you that the scale returns the weight of the object with an error of +/- a standard deviation of 5g. We can describe this mathematically as:

$$measurement = weight + \mathcal{N}(0, 5g)$$

You can weigh the apple as many times as you want, so weigh it 100 times. plot its histogram of your 100 measurements. (y axis is the counts and x-axis is the measured weight)

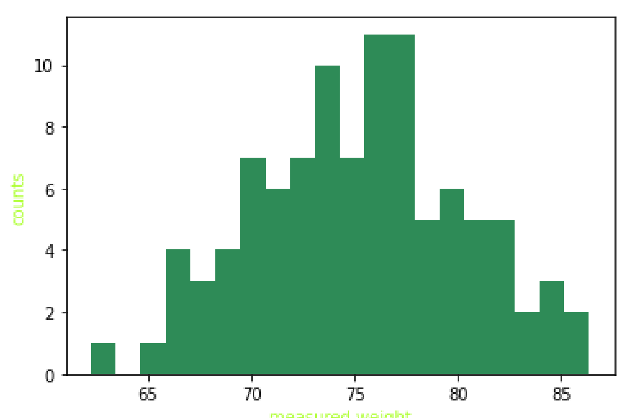
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
# Based on what you said in last cell: "The average apples is between 70-100 g", I cons
weight = 75
data = np.random.normal(0, 5, 100) + 75
data.sort()
print(data)

plt.hist(data, bins=20, color='SeaGreen')
plt.xlabel("measured weight", color='GreenYellow')
plt.ylabel("counts", color='GreenYellow')
plt.show()
```

```
[62.23505092 65.09601766 66.36858699 66.46864905 66.84900827 66.93051076
 67.31878157 67.72817163 67.89991031 68.7360232  68.8258709  69.09683908
 69.1742508  69.64623689 69.75723517 70.1136106  70.46350818 70.52266719
 70.56107126 70.64601425 70.7295213  70.93426859 71.2891749  71.57594955
 71.63769776 71.82838953 72.10575168 72.44597431 72.45173909 72.80962849
 72.98411527 72.99109532 73.06336591 73.18629417 73.20223419 73.26043925
 73.44223734 73.48848625 73.9362986  73.97420868 74.06408075 74.10037582
 74.24321396 74.48390574 74.85908886 75.0525001  75.22879259 75.25972698
 75.28082671 75.33258611 75.60837508 75.63456046 75.64491455 75.72021786
 75.77473713 75.78174485 75.88713071 76.04137489 76.51235949 76.56533851
 76.66837164 76.78183199 76.8908126  76.93451249 77.00078604 77.00994682
 77.01170821 77.05299251 77.14165935 77.21931616 77.31391128 77.3283122
 78.26809298 78.53286584 78.64545281 78.80518863 78.88745178 79.32218099
 79.50413243 79.75044209 79.88319518 79.89368992 80.27225863 80.69700342
 80.89389786 81.01189924 81.11222535 81.1514534  82.27136753 82.34679385
 82.44126097 82.47039537 82.66389607 83.82026173 83.92935247 84.33778995
 84.47944588 84.75387698 86.204466   86.34877312]
```



2) Find the average weight of the apple. Is it a good guess? state your reason.

In [2]:

```
data.mean()
```

```
75.29904007767243
```

Average is 75.299 ~ 75.3 by 100 times weighting.
Actual weight is 75.
So it's a good guess.

3) we are going to use grid approximation for calculating the MLE. here is the link if you wnat to get more fimilar with this technique: [https://www.bayesrulesbook.com/chapter-6 (https://www.bayesrulesbook.com/chapter-6)](https://www.bayesrulesbook.com/chapter-6)

Our end goal is to find the weight of the apple, given the data we have. To formulate it in a Bayesian way: We'll ask what is the probability of the apple having weight, w, given the measurements we took, X. And, because we're formulating this in a Bayesian way, we use Bayes' Law to find the answer:

$$P(w|X) = \frac{P(X|w)P(w)}{P(X)}$$

If we make no assumptions about the initial weight of our apple, then we can drop $P(w)$. We'll say all sizes of apples are equally likely (we'll revisit this assumption in the MAP approximation).

Furthermore, we'll drop $P(X)$ - the probability of seeing our data. This is a normalization constant and will be important if we do want to know the probabilities of apple weights. But, for right now, our end goal is to only to find the most probable weight. $P(X)$ is independent of w, so we can drop it if we're doing relative comparisons.

This leaves us with $P(X|w)$, our likelihood, as in, what is the likelihood that we would see the data, X, given an apple of weight w. If we maximize this, we maximize the probability that we will guess the right weight.

The grid approximation is probably the simplest way to do this. Basically, we'll systematically step through different weight guesses, and compare what it would look like if this hypothetical weight were to generate data. We'll compare this hypothetical data to our real data and pick the one that matches the best.

To formulate this mathematically:

For each of these guesses, we're asking "what is the probability that the data we have, came from the distribution that our weight guess would generate". Because each measurement is independent from another, we can break the above equation down into finding the probability on a per measurement basis:

$$P(X|w) = \prod_i^N p(x_i|w)$$

So, if we multiply the probability that we would see each individual data point - given our weight guess - then we can find one number comparing our weight guess to all of our data.

The peak in the likelihood is the weight of the apple.

To make it computationally easier,

$$\log P(X|w) = \log \prod_i^N p(x_i|w) = \sum_i^N \log p(d_i|w)$$

a) Why did we use log likelihood? Is it ok to do so?

b) do the grid approximation and complete the cell below

a) Why did we use log likelihood? Is it ok to do so?
Log is an increasing function, so it's ok.

In [3]:
```
from scipy.stats import norm

"""
# Calculate the maximum likelihood estimate of a parameter in a normal distribution.
# First calculate the log likelihoods for a range of weight guesses.
# For each weight guess, assume that the data comes from a normal distribution with tha
# Then calculate the log of the probability density function (pdf) of the data under th
# The sum of these log pdf values is the total log likelihood for that weight guess.
# After calculating the log likelihoods for all weight guesses, find the weight guess w
# This is the maximum likelihood estimate of the weight.
"""

weight_grid = np.linspace(0, 100)
print("\"data\" dataframe:\n",data)
print("\n\"wight_grid\" dataframe:\n",weight_grid)

mle_weight = 0
mle_likelihood = float('-inf')
for w in weight_grid:
    likelihood = np.sum(norm.logpdf(data, w, 10))
    if likelihood > mle_likelihood:
        mle_likelihood = likelihood
        mle_weight = w
print("\nMaximum Likelihood Estimate (MLE) Weight in \"data\" dataframe:", mle_weight)

mle_weight = 0
mle_likelihood = float('-inf')
for w in weight_grid:
    likelihood = np.sum(norm.logpdf(weight_grid, w, 10))
    if likelihood > mle_likelihood:
        mle_likelihood = likelihood
        mle_weight = w
print("\nMaximum Likelihood Estimate (MLE) Weight in \"weight_grid\" dataframe:", mle_w
```

```
"data" dataframe:
 [62.23505092 65.09601766 66.36858699 66.46864905 66.84900827 66.93051076
 67.31878157 67.72817163 67.89991031 68.7360232  68.8258709  69.09683908
 69.1742508  69.64623689 69.75723517 70.1136106  70.46350818 70.52266719
 70.56107126 70.64601425 70.7295213  70.93426859 71.2891749  71.57594955
 71.63769776 71.82838953 72.10575168 72.44597431 72.45173909 72.80962849
 72.98411527 72.99109532 73.06336591 73.18629417 73.20223419 73.26043925
 73.44223734 73.48848625 73.9362986  73.97420868 74.06408075 74.10037582
 74.24321396 74.48390574 74.85908886 75.0525001  75.22879259 75.25972698
 75.28082671 75.33258611 75.60837508 75.63456046 75.64491455 75.72021786
 75.77473713 75.78174485 75.88713071 76.04137489 76.51235949 76.56533851
 76.66837164 76.78183199 76.8908126  76.93451249 77.00078604 77.00994682
 77.01170821 77.05299251 77.14165935 77.21931616 77.31391128 77.3283122
 78.26809298 78.53286584 78.64545281 78.80518863 78.88745178 79.32218099
 79.50413243 79.75044209 79.88319518 79.89368992 80.27225863 80.69700342
 80.89389786 81.01189924 81.11222535 81.1514534  82.27136753 82.34679385
 82.44126097 82.47039537 82.66389607 83.82026173 83.92935247 84.33778995
 84.47944588 84.75387698 86.204466   86.34877312]

"wight_grid" dataframe:
 [  0.           2.04081633   4.08163265   6.12244898   8.16326531
  10.20408163  12.24489796  14.28571429  16.32653061  18.36734694
  20.40816327  22.44897959  24.48979592  26.53061224  28.57142857
  30.6122449   32.65306122  34.69387755  36.73469388  38.7755102
  40.81632653  42.85714286  44.89795918  46.93877551  48.97959184
  51.02040816  53.06122449  55.10204082  57.14285714  59.18367347
  61.2244898   63.26530612  65.30612245  67.34693878  69.3877551
  71.42857143  73.46938776  75.51020408  77.55102041  79.59183673
  81.63265306  83.67346939  85.71428571  87.75510204  89.79591837
  91.83673469  93.87755102  95.91836735  97.95918367 100.        ]

Maximum Likelihood Estimate (MLE) Weight in "data" dataframe: 75.51020408163265

Maximum Likelihood Estimate (MLE) Weight in "weight_grid" dataframe: 48.9795918367347
```

Play around with the code and try to answer the following questions regarding MLE and MAP. You can draw plots to visualize as well.

In [9]:

```
import numpy as np
from scipy.stats import norm, invgamma, beta


# The barrel of apples
# The average apples is between 70-100 g
BARREL = np.random.normal(loc=85, scale=20, size=100)
# Grid
WEIGHT_GUESSES = np.linspace(1, 200, 100)
ERROR_GUESSES = np.linspace(.1, 50, 100)

# NOTE: Try changing the scale error
# in practice, you would not know this number
SCALE_ERR = 5

# NOTE: Try changing the number of measurements taken
N_MEASURMENTS = 10

# NOTE: Try changing the prior values and distributions
PRIOR_WEIGHT = norm(50, 1).logpdf(WEIGHT_GUESSES)
PRIOR_ERR = invgamma(4).logpdf(ERROR_GUESSES)

LOG_PRIOR_GRID = np.add.outer(PRIOR_ERR, PRIOR_WEIGHT)


def read_scale(apple):
    return apple + np.random.normal(loc=0, scale=SCALE_ERR)


def get_log_likelihood_grid(measurments):
    log_liklelihood = [
        [
            norm(weight_guess, error_guess).logpdf(measurments).sum()
            for weight_guess in WEIGHT_GUESSES
        ]
        for error_guess in ERROR_GUESSES
    ]
    return np.asarray(log_liklelihood)


def get_mle(measurments):
    """
    Calculate the log-likelihood for each measurement in the grid.
    Find the index of the maximum log-likelihood in the grid.
    Return the weight guess corresponding to the maximum log-likelihood.
    """
    grid = get_log_likelihood_grid(measurments)
    max_index = np.unravel_index(grid.argmax(), grid.shape)
    return WEIGHT_GUESSES[max_index[1]]
```

```
def get_map(measurements):
    """
    Calculate the log-likelihood for each measurement in the grid.
    Add the log prior to the log likelihood to get the log posterior.
    Find the index of the maximum log posterior in the grid.
    Return the weight guess corresponding to the maximum log posterior.
    """
    grid = get_log_likelihood_grid(measurements)
    grid += LOG_PRIOR_GRID
    max_index = np.unravel_index(grid.argmax(), grid.shape)
    return WEIGHT_GUESSES[max_index[1]]

# Pick an apple at random
apple = np.random.choice(BARREL)

# weight the apple
measurments = np.asarray([read_scale(apple) for _ in range(N_MEASURMENTS)])
print("SIZE = 100")
print("PRIOR = INVGAMMA")
print("SCALE_ERR = ", SCALE_ERR, end=", ")
print("N_MEASURMENTS = ", N_MEASURMENTS)
print(f"Average measurement: {measurments.mean():.3f} g")
print(f"Maximum Likelihood estimate: {get_mle(measurments):.3f} g")
print(f"Maximum A Posterior estimate: {get_map(measurments):.3f} g")
print(f"The true weight of the apple was: {apple:.3f} g")

print("\n*****************************************************\n")

WEIGHT_GUESSES = np.linspace(1, 200, 50)
ERROR_GUESSES = np.linspace(.1, 50, 50)
PRIOR_WEIGHT = norm(50, 1).logpdf(WEIGHT_GUESSES)
PRIOR_ERR = invgamma(4).logpdf(ERROR_GUESSES)
LOG_PRIOR_GRID = np.add.outer(PRIOR_ERR, PRIOR_WEIGHT)
print("SIZE = 50")
print("PRIOR = INVGAMMA")
print("SCALE_ERR = ", SCALE_ERR, end=", ")
print("N_MEASURMENTS = ", N_MEASURMENTS)
print(f"Average measurement: {measurments.mean():.3f} g")
print(f"Maximum Likelihood estimate: {get_mle(measurments):.3f} g")
print(f"Maximum A Posterior estimate: {get_map(measurments):.3f} g")
print(f"The true weight of the apple was: {apple:.3f} g")

print("\n*****************************************************\n")

SCALE_ERR = 5
N_MEASURMENTS = 20
WEIGHT_GUESSES = np.linspace(1, 200, 100)
ERROR_GUESSES = np.linspace(.1, 50, 100)
```

```
PRIOR_WEIGHT = norm(50, 1).logpdf(WEIGHT_GUESSES)
PRIOR_ERR = invgamma(4).logpdf(ERROR_GUESSES)
LOG_PRIOR_GRID = np.add.outer(PRIOR_ERR, PRIOR_WEIGHT)
measurments = np.asarray([read_scale(apple) for _ in range(N_MEASURMENTS)])
print("SIZE = 100")
print("PRIOR = INVGAMMA")
print("SCALE_ERR = ", SCALE_ERR, end=", ")
print("N_MEASURMENTS = ", N_MEASURMENTS)
print(f"Average measurement: {measurments.mean():.3f} g")
print(f"Maximum Likelihood estimate: {get_mle(measurments):.3f} g")
print(f"Maximum A Posterior estimate: {get_map(measurments):.3f} g")
print(f"The true weight of the apple was: {apple:.3f} g")

print("\n*******************************************************\n")

PRIOR_ERR = beta(4, 4).logpdf(ERROR_GUESSES)
LOG_PRIOR_GRID = np.add.outer(PRIOR_ERR, PRIOR_WEIGHT)
SCALE_ERR = 5
N_MEASURMENTS = 20
print("SIZE = 100")
print("PRIOR = BETA")
print("SCALE_ERR = ", SCALE_ERR, end=", ")
print("N_MEASURMENTS = ", N_MEASURMENTS)
print(f"Average measurement: {measurments.mean():.3f} g")
print(f"Maximum Likelihood estimate: {get_mle(measurments):.3f} g")
print(f"Maximum A Posterior estimate: {get_map(measurments):.3f} g")
print(f"The true weight of the apple was: {apple:.3f} g")

print("\n*******************************************************\n")

WEIGHT_GUESSES = np.linspace(1, 200, 50)
ERROR_GUESSES = np.linspace(.1, 50, 50)
PRIOR_WEIGHT = norm(50, 1).logpdf(WEIGHT_GUESSES)
PRIOR_ERR = beta(4, 4).logpdf(ERROR_GUESSES)
LOG_PRIOR_GRID = np.add.outer(PRIOR_ERR, PRIOR_WEIGHT)
SCALE_ERR = 5
N_MEASURMENTS = 20
print("SIZE = 50")
print("PRIOR = BETA")
print("SCALE_ERR = ", SCALE_ERR, end=", ")
print("N_MEASURMENTS = ", N_MEASURMENTS)
print(f"Average measurement: {measurments.mean():.3f} g")
print(f"Maximum Likelihood estimate: {get_mle(measurments):.3f} g")
print(f"Maximum A Posterior estimate: {get_map(measurments):.3f} g")
print(f"The true weight of the apple was: {apple:.3f} g")

print("\n*******************************************************\n")

WEIGHT_GUESSES = np.linspace(1, 200, 50)
ERROR_GUESSES = np.linspace(.1, 50, 50)
```

```
PRIOR_WEIGHT = norm(50, 1).logpdf(WEIGHT_GUESSES)
PRIOR_ERR = invgamma(4).logpdf(ERROR_GUESSES)
LOG_PRIOR_GRID = np.add.outer(PRIOR_ERR, PRIOR_WEIGHT)
SCALE_ERR = 5
N_MEASURMENTS = 20
print("SIZE = 50")
print("PRIOR = INVGAMMA")
print("SCALE_ERR = ", SCALE_ERR, end=", ")
print("N_MEASURMENTS = ", N_MEASURMENTS)
print(f"Average measurement: {measurments.mean():.3f} g")
print(f"Maximum Likelihood estimate: {get_mle(measurments):.3f} g")
print(f"Maximum A Posterior estimate: {get_map(measurments):.3f} g")
print(f"The true weight of the apple was: {apple:.3f} g")

print("\n******************************************************\n")
```

```
SIZE = 100
PRIOR = INVGAMMA
SCALE_ERR =  5, N_MEASURMENTS =  10
Average measurement: 85.014 g
Maximum Likelihood estimate: 85.424 g
Maximum A Posterior estimate: 51.253 g
The true weight of the apple was: 86.488 g

******************************************************

SIZE = 50
PRIOR = INVGAMMA
SCALE_ERR =  5, N_MEASURMENTS =  10
Average measurement: 85.014 g
Maximum Likelihood estimate: 86.286 g
Maximum A Posterior estimate: 49.735 g
The true weight of the apple was: 86.488 g

******************************************************

SIZE = 100
PRIOR = INVGAMMA
SCALE_ERR =  5, N_MEASURMENTS =  20
Average measurement: 85.797 g
Maximum Likelihood estimate: 85.424 g
Maximum A Posterior estimate: 51.253 g
The true weight of the apple was: 86.488 g

******************************************************

SIZE = 100
PRIOR = BETA
SCALE_ERR =  5, N_MEASURMENTS =  20
Average measurement: 85.797 g
Maximum Likelihood estimate: 85.424 g
Maximum A Posterior estimate: 85.424 g
The true weight of the apple was: 86.488 g

******************************************************

SIZE = 50
PRIOR = BETA
SCALE_ERR =  5, N_MEASURMENTS =  20
Average measurement: 85.797 g
Maximum Likelihood estimate: 86.286 g
Maximum A Posterior estimate: 86.286 g
The true weight of the apple was: 86.488 g

******************************************************

SIZE = 50
PRIOR = INVGAMMA
SCALE_ERR =  5, N_MEASURMENTS =  20
Average measurement: 85.797 g
Maximum Likelihood estimate: 86.286 g
Maximum A Posterior estimate: 49.735 g
```

```
The true weight of the apple was: 86.488 g

*****************************************************
```

Questions

1. How sensitive is the MAP measurement to the choice of prior?

It's very sensetive based on last part.
For example Beta improve the guess and it's really better than invgamma.

2. How sensitive is the MLE and MAP answer to the grid size?

MLE is more sensetive to grid size than MAP.
Generally if we increase the size of the grid, the estimation with be much closer to the real weight of the apple.