Machine Learning (CPS-863 / COS-623 / MAB-608)

- Terceiro Trimestre de 2018
- Professores: Edmundo de Souza e Silva, Daniel S. Menascé Rosa Leão
- 2ª Lista de Exercícios (Graduação e Pós-Graduação)
- **Lucas Lopes Felipe**

Questão 1

Consider a Normal distribution $\mathcal{N}$(μ, σ2) and N data samples ($\mathcal{D} = x_1, x_2, \ldots, x_N$).

1. What is the likelihood function $\mathcal{L}(\theta, \mathcal{D})$ in this case? Recall that *θ* is the parameter vector. How many elements vector *θ* has?

Dado que a distribuição do *dataset* é uma Normal, a *Probability Density Function* (PDF) é dada por:

$$f(\mathcal{D}, \theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

Logo, a função de *Likelihood* é obtida através do produtório entre a Gaussiana de cada elemento do conjunto, pois considera-se que a probabilidades das amostras são indepentes entre si. A expressão pode ser representada da seguinte forma:

$$\mathcal{L}(\theta, \mathcal{D}) = \prod_{i=1}^{n} \frac{e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

E também computacionalmente em *Python*:

In [1]:
```
import math # Biblioteca para obter os valores de Pi e do Número de Eu
ler

# Função Likelihood para uma distribuição Normal
def likelihood_normal(theta, dataset):
    L = 1
    for data in dataset:
        L *= normal(theta, data)
    return L

# Distribuição Guassiana (normal)
def normal(theta, data):
    mean = theta[0] # média
    stdv = theta[1] # desvio padrão
    return (math.e ** ((-(data - mean) ** 2) / (2 * (stdv ** 2)))) / (
stdv * math.sqrt(2 * math.pi))
```

O vetor θ contém os parâmetros do modelo, onde neste caso por ser uma distribuição normal, possui 2 elementos. São eles:

1. Média
2. Desvio Padão

2. Show all the steps to obtain the MLE in this case. Compare your result and your proof with those in Theorem 4.1.1 of Murphy.

Se queremos obter o *Maximum Likelihood Estimate* precisamos descobrir os parâmetros do vetor θ (μ, σ^2) que produzam o maior *Likelihood* dentre qualquer outras combinação de valores possíveis.

Isso é o mesmo que descobrir o ponto onde a derivada da função equivale a 0, pois este é o topo da função, e logo, o valor que maxiza a mesma.

Porém, derivar a função *Likelihood* não é trivial, uma boa solução é derivar o logaritmo natural da função, pois além de transformar o log do produtório em um somatório dos logs (o que deixa o problema mais fácil), é uma função monotônica crescente, o que significa que possui o topo no mesmo ponto que a função original. O *Log-Likelihood* fica portanto da seguinte maneira:

$$\mathcal{L}(\theta, \mathcal{D}) = \prod_{i=1}^{n} \frac{e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

$$\ln(\mathcal{L}(\theta, \mathcal{D})) = \sum_{i=1}^{n} \ln\left(\frac{e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}\right)$$

$$\Rightarrow \sum_{i=1}^{n} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \ln\left(e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right)$$

$$\Rightarrow \sum_{i=1}^{n} \ln\left((2\pi\sigma^2)^{-\frac{1}{2}}\right) - \frac{(x_i-\mu)^2}{2\sigma^2}\ln(e)$$

$$\Rightarrow \sum_{i=1}^{n} -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

$$\Rightarrow \sum_{i=1}^{n} -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma^2) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

$$\Rightarrow \sum_{i=1}^{n} -\frac{1}{2}\ln(2\pi) - \frac{2}{2}\ln(\sigma) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

$$\Rightarrow -\frac{n}{2}\ln(2\pi) - n\ \ln(\sigma) - \sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2}$$

Agora que temos a o log da função, vamos derivar em função de μ para descobrir o primeiro parâmetro:

$$
\begin{aligned}
\frac{\partial}{\partial \mu} \ln(\mathcal{L}(\theta, \mathcal{D})) &= -\frac{n}{2}\ln(2\pi) - n\ \ln(\sigma) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \\
&\Rightarrow 0 - 0 - \sum_{i=1}^{n} \frac{2(x_i - \mu)}{2\sigma^2} \\
&\Rightarrow \sum_{i=1}^{n} \frac{(x_i - \mu)}{\sigma^2} \\
&\Rightarrow \frac{1}{\sigma^2} \sum_{i=1}^{n} x_i - \mu \\
&\Rightarrow \frac{-n\,\mu}{\sigma^2} \sum_{i=1}^{n} x_i
\end{aligned}
$$

Uma vez que temos a derivada da *Log-Likelihood*, para que encontremos o topo da função, devemos igualar a 0:

$$
\begin{aligned}
\frac{\partial}{\partial \mu} \ln(\mathcal{L}(\theta, \mathcal{D})) = 0 \\
\frac{-n\,\mu}{\sigma^2} \sum_{i=1}^{n} x_i = 0 \\
-n\,\mu \sum_{i=1}^{n} x_i = 0 \\
n\,\mu = \sum_{i=1}^{n} x_i \\
\mu = \frac{\sum_{i=1}^{n} x_i}{n}
\end{aligned}
$$

Ou seja, o melhor valor para o pâmetro μ é a média dos dados. Que pode ser representado pela função em Python:

In [16]:
```
# Cálculo da média
def mean(data):
    total = 0
    for d in data:
        total += d
    return total / len(data)
```

Agora que encontramos o 1° parâmentro, vamos descobrir o segundo. Para isso, vamos derivar em função de σ:

$$
\begin{aligned}
\frac{\partial}{\partial \sigma}\ln(\mathcal{L}(\theta, \mathcal{D})) &= -\frac{n}{2}\ln(2\pi) - n\ \ln(\sigma) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2} \\
&\Rightarrow 0 - \frac{n}{\sigma} - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2}\sigma^{-2} \\
&\Rightarrow -\frac{n}{\sigma} - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2}(-2)\sigma^{-3} \\
&\Rightarrow -\frac{n}{\sigma} + \sum_{i=1}^{n}(x_i-\mu)^2\sigma^{-3} \\
&\Rightarrow -\frac{n}{\sigma} + \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{\sigma^3} \\
&\Rightarrow -\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(x_i-\mu)^2
\end{aligned}
$$

E novamente, igualar a 0, para encontrar o ponto em que a função está no topo:

$$
\begin{aligned}
\frac{\partial}{\partial \sigma}\ln(\mathcal{L}(\theta, \mathcal{D})) &= 0 \\
-\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(x_i-\mu)^2 &= 0 \\
-n + \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 &= 0 \\
n &= \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 \\
n\,\sigma^2 &= \sum_{i=1}^{n}(x_i-\mu)^2 \\
\sigma &= \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}}
\end{aligned}
$$

Ou seja, o melhor valor para o pâmetro σ é o desvio padrão dos dados. Que pode ser representado pela função em Python:

In [17]:

```python
# Cálculo do Desvio Padrão
def stdv(data, mean):
    total = 0
    for d in data:
        total += (d - mean) ** 2
    return math.sqrt(total / (len(data) - 1))
```

Questão 2

> For $\mathcal{D}$ you are given 2 datasets: *D1* = `data-l2-p-1a.txt` and *D2* = `data-l2-p-1b.txt`. You know that one of the two datasets are samples obtained from a normal distribution and the other from a uniform distribution.

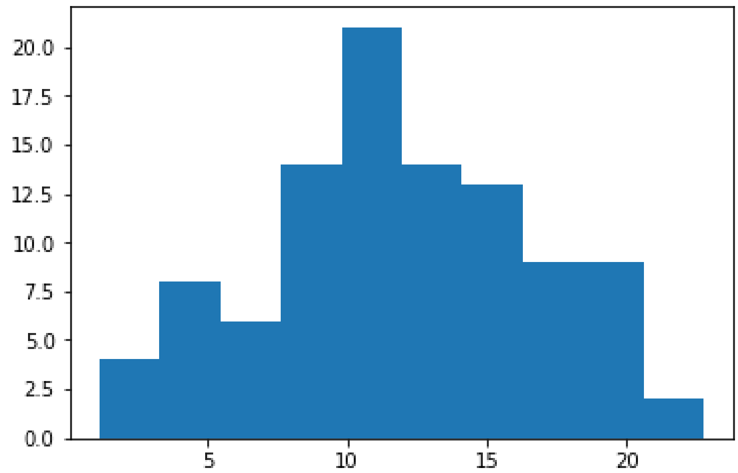
In [4]:
```
from matplotlib import pyplot as plt # Biblioteca para Plot
%matplotlib inline

# Converter Lista de String para Lista de Floats
def string2float(data):
    for i in range(len(data)):
        data[i] = float(data[i])
    return data

D1 = string2float(open('./data-sets/D2/data-l2-p-1a.txt', 'r').read().
split('\n')[10:-1])
D2 = string2float(open('./data-sets/D2/data-l2-p-1b.txt', 'r').read().
split('\n')[10:-1])
```

1. Consider dataset D1

In [5]:
```
plt.hist(D1)
plt.show()
```



a) Obtain the MLE parameters assuming the model is a normal distribution.

In [6]:
```
mean = mean(D1)
stdv = stdv(D1, mean)

theta = [mean, stdv]

li_D1_normal = likelihood_normal(theta, D1)
```

b) Obtain the MLE parameters assuming the model is a uniform distribution

A *Probability Density Function* (PDF) da distribuição Uniforme é dada por:

$$f(\mathcal{D}, \theta) = \frac{1}{\theta} = \theta^{-1}$$

Onde os valores do *Dataset* devem ser superiores a 0 e inferiores a θ. Portanto, os parâmetros para a distribuição devem ser o **menor** e **maior** valor do conjunto de dados. E a função *Likelihood* é dada pelo produtório entre eles:

$$\mathcal{L}(\theta, \mathcal{D}) = \prod_{i=1}^{n} \theta^{-1} = \theta^{-n}$$

Que pode ser representada pela função em Python:

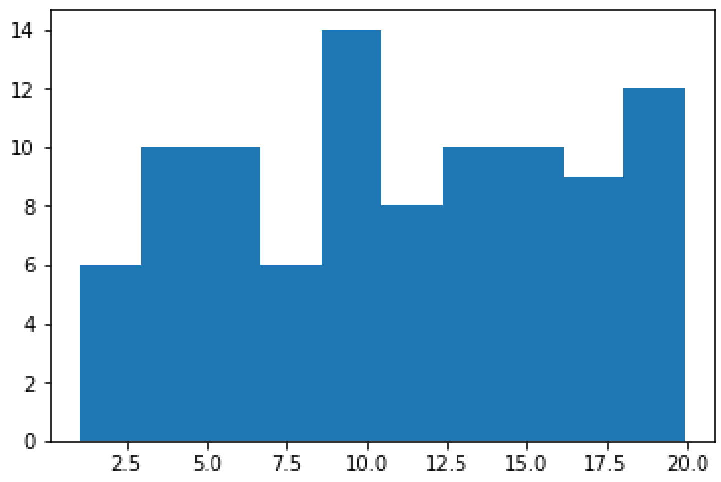
In [7]:
```
def likelihood_uniform(theta, dataset):
    return (theta[1] - theta[0]) ** - len(dataset)

theta = [min(D1), max(D1)]

li_D1_uniform = likelihood_uniform(theta, D1)
```

2. Consider dataset D2

In [8]:
```
plt.hist(D2)
plt.show()
```



a) Obtain the MLE parameters assuming the model is a normal distribution.

In [18]:
```
mean = mean(D2)
stdv = stdv(D2, mean)

theta = [mean, stdv]

li_D2_normal = likelihood_normal(theta, D2)
```

b) Obtain the MLE parameters assuming the model is a uniform distribution.

In [19]:
```
theta = [max(D2), min(D2)]

li_D2_uniform = likelihood_uniform(theta, D2)
```

3. Using what you learned about the *likelihood*, which of the two datasets was generated by a normal distribution, and which one was generated by a uniform distribution. Comment your answer and indicate how sure are you concerning your answer.

In [20]:
```
D1_ratio = li_D1_uniform / li_D1_normal
D2_ratio = li_D2_uniform / li_D2_normal

print('Para o Dataset 1, a distribuição Normal é {}x em relação a Unif
orme'.format(D1_ratio))
print('Para o Dataset 2, a distribuição Normal é {}x em relação a Unif
orme'.format(D2_ratio))
```

```
Para o Dataset 1, a distribuição Normal é 8.806811404648603e-06x em r
elação a Uniforme
Para o Dataset 2, a distribuição Normal é -835750.9346326572x em rela
ção a Uniforme
```

Logo podemos concluir que:

- D1 foi gerado por uma *Distribuição Normal*
- D2 foi gerado por uma *Distribuição Uniforme*

Questão 3

> Para exemplificar a aplicação de regressão linear, utilizaremos um pequeno subconjunto de dados reais de [vendas de imóveis em Iowa (https://www.kaggle.com/c/house-prices-advanced-regression-techniques)](https://www.kaggle.com/c/house-prices-advanced-regression-techniques). Consideraremos apenas 1 *feature*, a área da casa vendida. O *target* a ser estimado é o preço de venda. A Tabela 1 possui 6 pontos obtidos a partir deste dataset. A Tabela 2 possui 2 pontos para os quais se deseja prever o valor de venda.

Área do imóvel (sq ft)	Preço de venda ($)
334	39300
438	60000
520	68500
605	86000
672	113000
767	133000

Tabela 1: Imóveis com preço de venda conhecido.

Área do imóvel (sq ft)	Preço de venda ($)
848	y_1
912	y_2

Tabela 2: Imóveis com preço de venda desconhecido.

In [21]:
```python
tabela1_area  = [334  ,   438,    520,    605,     672,     767]
tabela1_preco = [39300, 60000, 68500, 86000, 113000, 133000]

tabela2_area  = [848, 912]
```

1. Obtenha a partir dos dados da Tabela 1 os modelos *M1* e *M2* utilizando a técnica de regressção linear. Para o modelo *M1* utilize ϕ1 = [1, x]. Para o modelo *M2*, utilize ϕ2 = [1, x, x², x³, x^4].

In [22]:
```python
import numpy as np

def phi1(x): return [1, x]
def phi2(x): return [1, x, x ** 2, x ** 3, x ** 4]

def inputs(vector, model):
    new = []
    if model == 1:
        for x in vector:
            new.append(phi1(x))
    elif model == 2:
        for x in vector:
            new.append(phi2(x))
    else: new = None
    return np.array(new)

def weights(X, y):
    return np.dot(np.float_power(np.dot(np.transpose(X), X), -1), np.d
ot(np.transpose(X), y))

def model(X, w):
    return np.dot(X, w)
```

2. Utilizando os modelos obtidos na questão anterior, calcule o erro quadrático médio para *M1* e *M2* considerando apenas os dados da Tabela 1. Calcule também a razão entre o erro quadrático médio de *M1* e *M2*.

In [23]:
```
def mean_squared_error(predict, y):
    total_error = 0
    for i in range(len(y)):
        total_error += ((predict[i] - y[i]) ** 2)
    return total_error / len(y)

M1_input   = inputs(tabela1_area, 1)
M1_weights = weights(M1_input, tabela1_preco)
M1_predict = model(M1_input, M1_weights)
M1_error   = mean_squared_error(M1_predict, tabela1_preco)

M2_input   = inputs(tabela1_area, 2)
M2_weights = weights(M2_input, tabela1_preco)
M2_predict = model(M2_input, M2_weights)
M2_error   = mean_squared_error(M2_predict, tabela1_preco)

print('Erro Quadrático Médio de M1:', M1_error)
print('Erro Quadrático Médio de M2:', M2_error)
```

```
Erro Quadrático Médio de M1: 67987115556.2
Erro Quadrático Médio de M2: 4.51974619605e+20
```

3. Calcule o valor estimado para y1 e y2 utilizando *M1* e *M2*.

In [24]:
```
tabela2_area = [848, 912]

M1_input_tab2   = inputs(tabela2_area, 1)
M1_predict_tab2 = model(M1_input_tab2, M1_weights)

M2_input_tab2   = inputs(tabela2_area, 2)
M2_predict_tab2 = model(M2_input_tab2, M2_weights)

print('M1: y1 =', M1_predict_tab2[0], 'e y2 =', M1_predict_tab2[1])
print('M2: y1 =', M2_predict_tab2[0], 'e y2 =', M2_predict_tab2[1])
```

```
M1: y1 = 432422.859944 e y2 = 451871.542374
M2: y1 = 62407239915.4 e y2 = 83487201783.1
```

4. Descobre-se que y1 = 155900 e y2 = 156000. A partir desta informação calcule o erro quadrático médio para *M1* e *M2*, considerando apenas os dados da Tabela 2. Calcule também a razão entre o erro quadrático médio de *M1* e *M2*. Houve mudança significativa na razão? Explique o resultado obtido.

In [26]:
```
y_real = [155900, 156000]

M1_error_tab2 = mean_squared_error(M1_predict_tab2, y_real)
M2_error_tab2 = mean_squared_error(M2_predict_tab2, y_real)

print('Erro Quadrático Médio de M1 da Tabela 2:', M1_error_tab2)
print('Erro Quadrático Médio de M2 da Tabela 2:', M2_error_tab2)

ratio = M2_error_tab2 / M1_error_tab2
print('\n O erro de M2 é', ratio, 'vezes maior que o de M1')
```

```
Erro Quadrático Médio de M1 da Tabela 2: 82002430829.0
Erro Quadrático Médio de M2 da Tabela 2: 5.43236547445e+21

 O erro de M2 é 66246395619.4 vezes maior que o de M1
```

O erro de M2 é muito maior pois houve problema de *overfitting* já que utilizou-se uma função polinomial de alto grau.

Questão 4

> O objetivo deste problema é estudar *logistic regression*. Considere o dataset da questão 3 (https://www.kaggle.com/c/house-prices-advanced-regression-techniques). Considere apenas 1 *feature*, a área da casa vendida. O objetivo é calcular a probabilidade do preço ser superior a **dado valor** *T* a partir da área do imóvel. Importante: escolha um valor para *T* que seja do seu interesse a partir dos dados.

In [ ]:
```
import pandas as pd # Biblioteca para importar CSV

imoveis_train = pd.read_csv('./data-sets/Iowa/train.csv')[['LotArea',
'SalePrice']] # Dados de Treinamento
imoveis_test  = pd.read_csv('./data-sets/Iowa/test.csv')['LotArea'] #
 Dados de Teste
imoveis_train.head() # Exibindo os 5 primeiros itens do Dataset de Tre
inamento
```

1. A partir da função *sigmoid* mostre como obter o log likelihood para uma amostra de tamanho *N*.

In [ ]:
```

```

2. Como se acha os parâmetros do problema?

In [ ]:
```

```

3. A partir dos dados do site acima, usando wT x = [w0, w1x] ache a função sigm(wT x). Plote essa função.