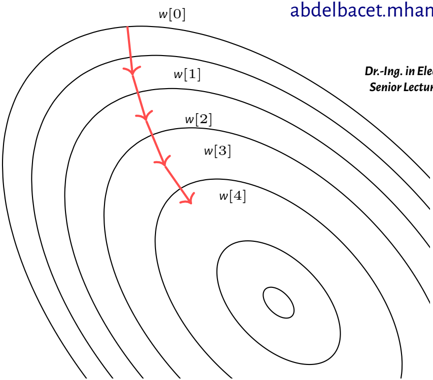
An Introduction To Machine Learning Sorcery

(A Demystification Draft)[1]

Abdelbacet Mhamdi
abdelbacet.mhamdi@bizerte.r-iset.tn

Dr.-Ing. in Electrical Engineering
Senior Lecturer at ISET Bizerte



"Computers are able to see, hear and learn. Welcome to the future."

Dave Waters

"This is nothing. In a few years, that bot will move so fast you'll need a strobe light to see it. Sweet dreams..."

Elon Musk

"Machine intelligence is the last invention that humanity will ever need to make."

Nick Bostrom

[1]Available @ https://github.com/a-mhamdi/isetbz/

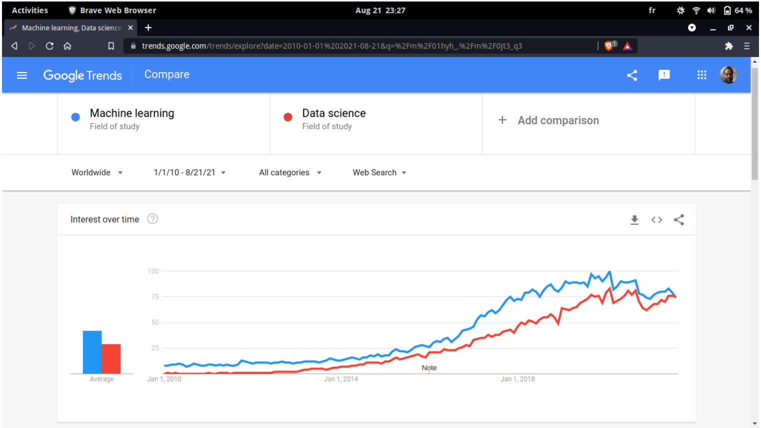
Disclaimer

This document features some material gathered from multiple online sources. Please note no copyright infringement is intended, and I do not own nor claim to own any of the original material. They are used for educational purposes only. I have included links solely as a convenience to the reader. Some links within these slides may lead to other websites, including those operated and maintained by third parties. The presence of such a link does not imply a responsibility for the linked site or an endorsement of the linked site, its operator, or its contents.

Outline

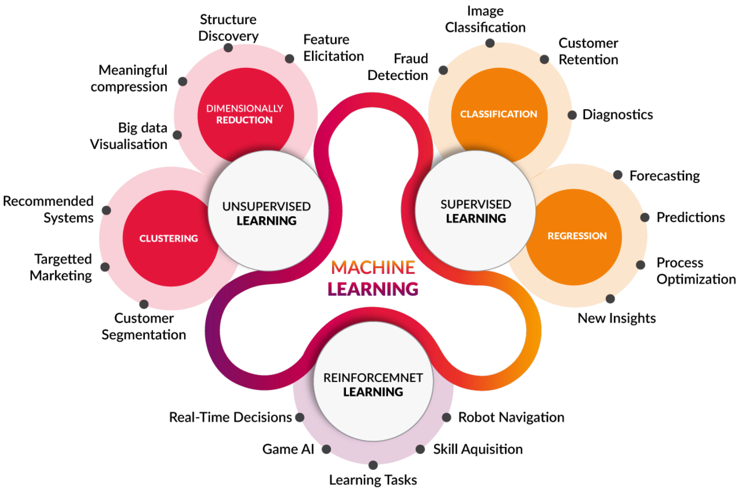
Next...

Top uses

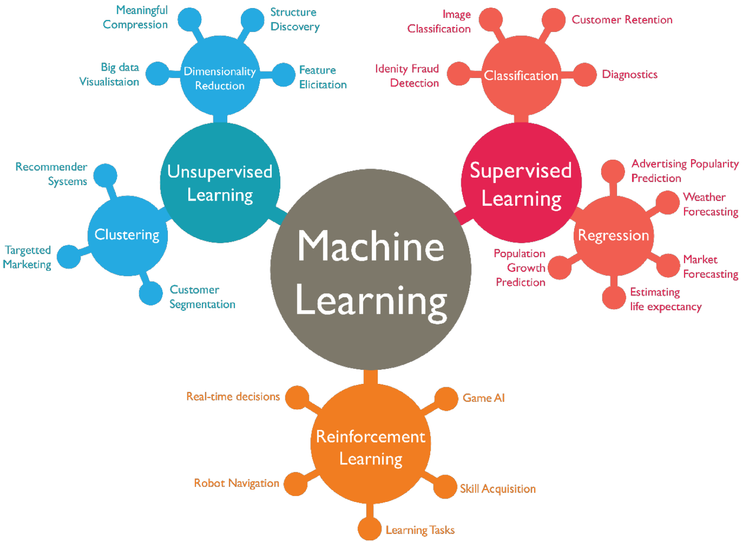


"Numbers represent search interest relative to the highest point on the chart for the given region and time.

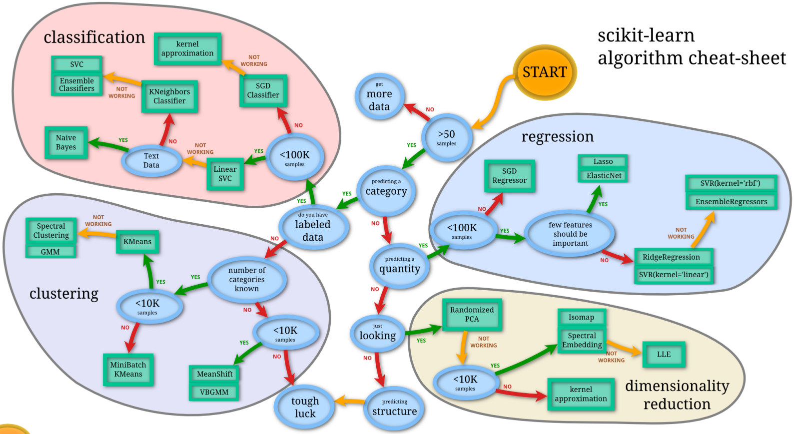
- A value of 100 is the peak popularity for the term;
- A value of 50 means that the term is half as popular;
- A score of 0 means there was not enough data for this term."



https://www.cognub.com/index.php/cognitive-platform/



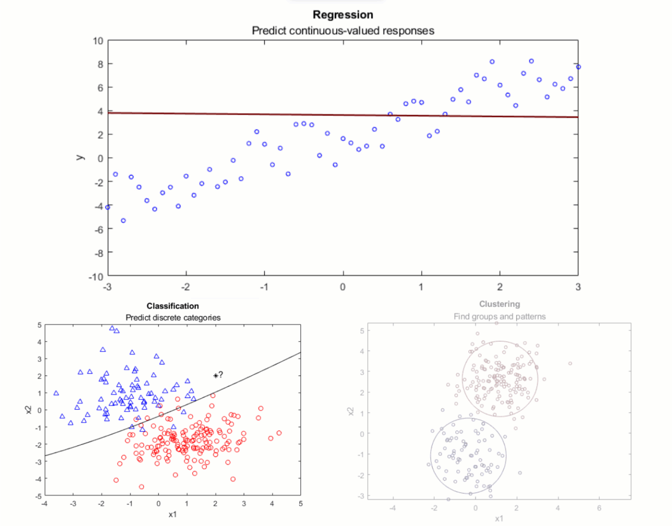
https://vitalflux.com/great-mind-maps-for-learning-machine-learning/



Back

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Regression | Classification | Clustering



https://github.com/MathWorks-Teaching-Resources/Machine-Learning-for-Regression

Machine Learning Definition

Arthur Samuel (1959)

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998)

Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

Machine Learning Definition

Arthur Samuel (1959)

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998)

Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

Task #1

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $\mathcal{T}$ in this setting?

1. Classifying emails as spam or not spam;
2. Watching you label emails as spam or not spam;
3. The number (or fraction) of emails correctly classified as spam/not spam;
4. None of the above-this not a machine learning problem.

Machine Learning Definition

Arthur Samuel (1959)

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998)

Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

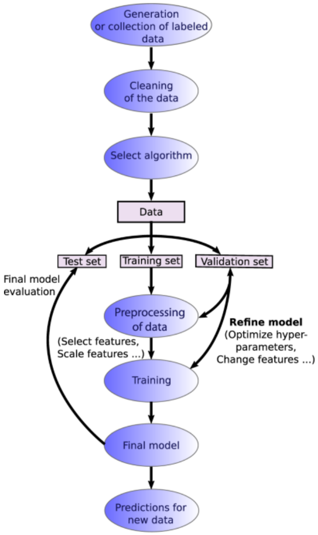
Task #1

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $\mathcal{T}$ in this setting?

1. Classifying emails as spam or not spam;
2. Watching you label emails as spam or not spam;
3. The number (or fraction) of emails correctly classified as spam/not spam;
4. None of the above-this not a machine learning problem.

Next...

Overall Methodology [Sch+19]



Data Preprocessing Template (1/7)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/Data.csv')
df.head()
```

[2]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

Extracting independant and dependant variables

[3]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Data Preprocessing Template (2/7)

[4]:
```
print(X) # Features
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

[5]:
```
print(y) # Target
```

```
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

Data Preprocessing Template (3/7)

Imputation transformer for completing missing values

https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

[6]:
```
from sklearn.impute import SimpleImputer
```

[7]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
```

[8]:
```
print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 63777.77777777778]
 ['France' 35.0 58000.0]
 ['Spain' 38.77777777777778 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

Data Preprocessing Template (4/7)

How to encode categorical data?

Case of Independent Variable

[9]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

[10]:
```
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[0])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

[11]:
```
print(X)
```

```
[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.77777777777778 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

Data Preprocessing Template (5/7)

Case of Dependent Variable

[12]:
```
from sklearn.preprocessing import LabelEncoder
```

[13]:
```
le = LabelEncoder()
y = le.fit_transform(y)
```

[14]:
```
print(y)
```

```
[0 1 0 0 1 1 0 1 0 1]
```

Splitting the dataset into Training set and Test set

[15]:
```
from sklearn.model_selection import train_test_split
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪2, random_state=1)
```

[17]:
```
print(X_train)
```

Data Preprocessing Template (6/7)

```
[[0.0 0.0 1.0 38.77777777777778 52000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 35.0 58000.0]]
```

[18]:
```
print(X_test)
```

```
[[0.0 1.0 0.0 30.0 54000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

[19]:
```
print(y_train)
```

```
[0 1 0 0 1 1 0 1]
```

[20]:
```
print(y_test)
```

```
[0 1]
```

Data Preprocessing Template (7/7)

Scaling of features

[21]:
```
from sklearn.preprocessing import StandardScaler
```

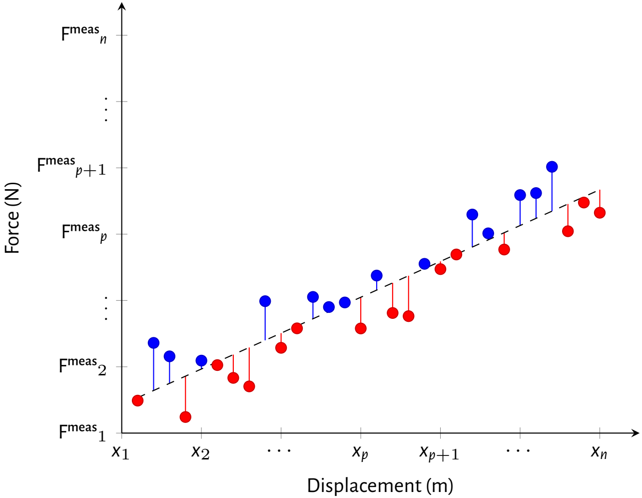
[22]:
```
sc = StandardScaler()
X_train[:, 3:] = sc.fit_transform(X_train[:, 3:])
X_test[:, 3:] = sc.transform(X_test[:, 3:])
```

[23]:
```
print(X_train)
```

```
[[0.0 0.0 1.0 -0.19159184384578545 -1.0781259408412425]
 [0.0 1.0 0.0 -0.014117293757057777 -0.07013167641635372]
 [1.0 0.0 0.0 0.566708506533324 0.633562432710455]
 [0.0 0.0 1.0 -0.30453019390224867 -0.30786617274297867]
 [0.0 0.0 1.0 -1.9018011447007988 -1.420463615551582]
 [1.0 0.0 0.0 1.1475343068237058 1.232653363453549]
 [0.0 1.0 0.0 1.4379472069688968 1.5749910381638885]
 [1.0 0.0 0.0 -0.7401495441200351 -0.5646194287757332]]
```

[24]:
```
print(X_test)
```

```
[[0.0 1.0 0.0 -1.4661817944830124 -0.9069571034860727]
 [1.0 0.0 0.0 -0.44973664397484414 0.2056403393225306]]
```


$F^{meas}{}_n$
$\vdots$
$F^{meas}{}_{p+1}$
$F^{meas}{}_p$
$\vdots$
$F^{meas}{}_2$
$F^{meas}{}_1$
Force (N)
x_1
x_2
$\cdots$
x_p
x_{p+1}
$\cdots$
x_n
Displacement (m)

Consider the example of a spring. Our main goal is to determine the stiffness k of this spring, given some experimental data. The mathematical model *(Hooke's law)*:

$$F = kx \tag{1}$$

Restoring force is proportional to displacement.

Table: Measurements of couple $(x_i,\ \mathrm{F^{meas}}_i)$

x_i	x_1	$\cdots$	x_p	$\cdots$	x_n
$\mathrm{F^{meas}}_i$	$\mathrm{F^{meas}}_1$	$\cdots$	$\mathrm{F^{meas}}_p$	$\cdots$	$\mathrm{F^{meas}}_n$

$$\begin{aligned} \mathrm{F^{meas}}_i &= \mathrm{F}_i + \varepsilon_i \\ &= kx_i + \varepsilon_i, \end{aligned} \tag{2}$$

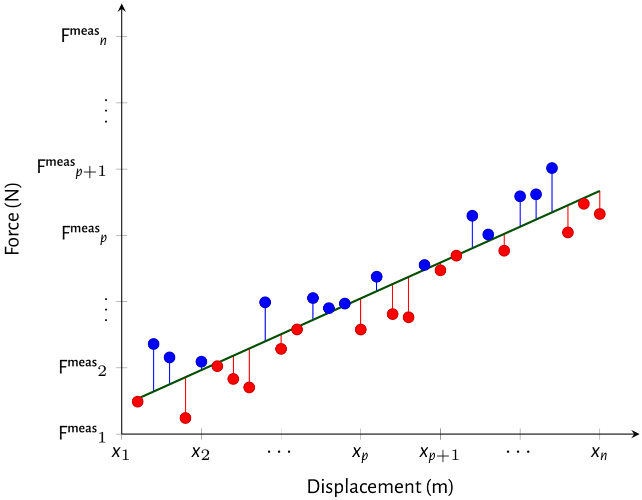
where F_i denotes the unknown real value of the force applied to the spring. In order to estimate the stiffness value k, we can consider the quadratic criterion:

$$\begin{aligned} \mathcal{J} &= \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \sum_{i=1}^{n} \left(\mathrm{F^{meas}}_i - kx_i\right)^2 \end{aligned}$$

$$\frac{\partial \mathcal{J}}{\partial k} = 0 \tag{3}$$

$$\begin{aligned} 2\sum_{i=1}^{n}\left(\mathrm{F^{meas}}_i - kx_i\right)\sum_{i=1}^{n}\frac{\partial\left(\mathrm{F^{meas}}_i - kx_i\right)}{\partial k} &= 0 \\ \sum_{i=1}^{n}\left(\mathrm{F^{meas}}_i - kx_i\right)\sum_{i=1}^{n}x_i &= 0 \end{aligned}$$

$$\sum_{i=1}^{n}\mathrm{F^{meas}}_i\,x_i = k\sum_{i=1}^{n}x_i^2 \iff \boxed{\hat{k} = \frac{\sum_{i=1}^{n}\mathrm{F^{meas}}_i\,x_i}{\sum_{i=1}^{n}x_i^2}}$$


Force (N)
F^meas_n
F^meas_{p+1}
F^meas_p
F^meas_2
F^meas_1
x_1
x_2
...
x_p
x_{p+1}
...
x_n
Displacement (m)

Simple Linear Regression (1/6)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/Salary_Data.csv')
df.head()
```

[2]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

[3]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Splitting the dataset into training set and test set

Simple Linear Regression (2/6)

[4]:
```
from sklearn.model_selection import train_test_split
```

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/
↪3, random_state=0)
```

Training the Simple Linear Regression model on the Training set

[6]:
```
from sklearn.linear_model import LinearRegression
```

[7]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[7]:
```
LinearRegression()
```

Predicting the Test set results

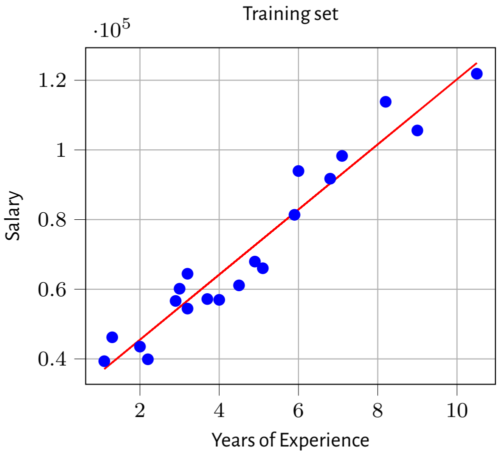
[8]:
```
y_pred = regressor.predict(X_test)
```

Visualising the Training set results

Simple Linear Regression (3/6)

[9]:
```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, regressor.predict(X_train), color='red')
plt.title('Training set')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.grid()
```

Simple Linear Regression (4/6)

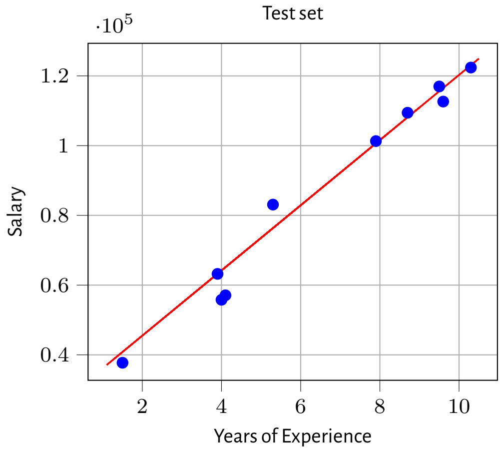


Simple Linear Regression (5/6)

Visualising the Test set results

[10]:
```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_train, regressor.predict(X_train), color='red')
plt.title('Test set')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.grid()
```

Simple Linear Regression (6/6)



This example consists on determining the unknown couple $(y_0,\ v_0)$ of a mobile solid. We assume that the trajectory is linear. The mathematical model that relates the position y to time t is given by this equation:

$$y = y_0 + v_0 t \tag{4}$$

Table: Measurements of position y

t_i	t_1	$\cdots$	t_p	$\cdots$	t_n
$y^{meas}{}_i$	$y^{meas}{}_1$	$\cdots$	$y^{meas}{}_p$	$\cdots$	$y^{meas}{}_n$

$$\begin{aligned} y^{meas}{}_i &= y_i + \varepsilon_i \\ &= y_0 + v_0 t_i + \varepsilon_i, \end{aligned} \tag{5}$$

where y_i denotes the unknown real value of the position y at time point t_i.
In order to estimate the values taken by the couple $\begin{bmatrix} y_0, & v_0 \end{bmatrix}^T$, we consider the quadratic criterion again, as follows:

$$\begin{aligned} \mathcal{J} &= \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \varepsilon^T \times \varepsilon \end{aligned}$$

The vector ε is set by $\varepsilon_i, \ \forall i \geq 1$:

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & \cdots & \varepsilon_n \end{bmatrix}^T$$

$$\frac{\partial \mathcal{J}}{\partial \begin{bmatrix} y_0 \\ \\ v_0 \end{bmatrix}} = 0 \tag{6}$$

Multiple Linear Regression (1/4)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/50_Startups.csv')
df.head()
```

[2]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

[3]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

[4]:
```
print(X[:5])
```

Multiple Linear Regression (2/4)

```
[[165349.2 136897.8 471784.1 'New York']
 [162597.7 151377.59 443898.53 'California']
 [153441.51 101145.55 407934.54 'Florida']
 [144372.41 118671.85 383199.62 'New York']
 [142107.34 91391.77 366168.42 'Florida']]
```

[5]:
```
print(y[:5])
```

```
[192261.83 191792.06 191050.39 182901.99 166187.94]
```

Encoding categorical data

[6]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

[7]:
```
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

[8]:
```
print(X[:5])
```

Multiple Linear Regression (3/4)

```
[[0.0 0.0 1.0 165349.2 136897.8 471784.1]
 [1.0 0.0 0.0 162597.7 151377.59 443898.53]
 [0.0 1.0 0.0 153441.51 101145.55 407934.54]
 [0.0 0.0 1.0 144372.41 118671.85 383199.62]
 [0.0 1.0 0.0 142107.34 91391.77 366168.42]]
```

Splitting the dataset into training set and test set

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪2, random_state=0)
```

Training the multiple linear regression model on the training set

[11]:
```
from sklearn.linear_model import LinearRegression
```

[12]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

[12]:
```
LinearRegression()
```

Making predictions using the X test set and comparison

Multiple Linear Regression (4/4)

[13]:

```
y_pred = lr.predict(X_test)
np.set_printoptions(precision=2)
print(type(y_pred), y_pred.shape)
y_pred = y_pred.reshape(-1,1)
print(type(y_pred), y_pred.shape)
y_test = y_test.reshape(-1,1)
print(np.concatenate((y_pred, y_test), axis=1))
```

```
<class 'numpy.ndarray'> (10,)
<class 'numpy.ndarray'> (10, 1)
[[103015.2  103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1   77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

Consider the following multivariate equation:

$$y = \theta_1 x_{(1)} + \theta_2 x_{(2)} + \cdots + \theta_m x_{(m)} \quad (7)$$

For a particular single measurement, eq. (7) can be updated as

$$y_k = \theta_1 x_{(1,\,k)} + \theta_2 x_{(2,\,k)} + \cdots + \theta_m x_{(m,\,k)} + \varepsilon_k \quad (8)$$

We denote hereafter by θ the vector $\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$. The function y_k becomes:

$$y_k = \underbrace{\left[x_{(1,\,k)},\, x_{(2,\,k)},\, \cdots,\, x_{(m,\,k)}\right]}_{x_k^T} \theta + \varepsilon_k$$

We assume that we have n measurements for y. Then we can transform the previous equation into

$$y = H\theta + \varepsilon,$$

where $y^T = [y_1,\, y_2,\, \cdots,\, y_n]$, $X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$, and $\varepsilon^T = [\varepsilon_1,\, \varepsilon_2,\, \cdots,\, \varepsilon_n]$.

We can consider the mean squared error or quadratic criterion in order to compute the approximated value of θ:

$$\begin{aligned} \mathcal{J} &= \sum_{k=1}^{n} \varepsilon_k^2 \\ &= \varepsilon^T \varepsilon \end{aligned}$$

The best well estimated value of $\hat{\theta}$ corresponds to the absolute minimum of $\mathcal{J}$. This leads to calculate the gradient of $\mathcal{J}$ with respect to θ:

$$\frac{\partial \mathcal{J}}{\partial \theta} = \frac{\partial \left(\varepsilon^T \varepsilon\right)}{\partial \theta} \tag{9}$$

$$\frac{\partial \left(\varepsilon^T \varepsilon\right)}{\partial \theta} = 2 \left(\frac{\partial \varepsilon}{\partial \theta}\right)^T \varepsilon \tag{10}$$

Recall that $\varepsilon = y - X\theta$, the term $\frac{\partial \varepsilon}{\partial \theta}$ hence becomes:

$$\frac{\partial \varepsilon}{\partial \theta} = -X \tag{11}$$

$$\begin{aligned} \frac{\partial J}{\partial \theta} &= 2(-X)^T (y - X\theta) \\ &= 0 \end{aligned}$$

The regressor is given by

$$\boxed{\hat{\theta} = \left(X^T X\right)^{-1} X^T y}$$

$X^T X$ **is not invertible (singular/degenerate)**

- Redundant Features

Some features are linearly dependant, i.e, $\exists$ some $x_p \propto$ some x_l for instance x_p in feet and x_l in m.

- Too many features

Fewer observations compared to the number of features, i.e, $m \geq n$.

- Delete some features
- Add extra observations
- Use regularization

Gradient Descent

$$\boxed{\theta_i = \theta_i - \underbrace{\alpha}_{\text{Learning Rate}} \frac{\partial \mathcal{J}}{\partial \theta_i}}$$

Recall that $\mathcal{J} = {}^{1}/_{2n} \sum_{k=1}^{n} (y_k - h_\theta(x_k))^2 \implies \frac{\partial \mathcal{J}}{\partial \theta_i} = -{}^{1}/_{n} \sum_{k=1}^{n} (y_k - h_\theta(x_k))\, x_{(i,\,k)}$

$$\boxed{\theta_i \leftarrow \theta_i + \alpha \frac{1}{n} \sum_{k=1}^{n} (y_k - h_\theta(x_k))\, x_{(i,\,k)}}$$

$$\boxed{\theta_0 \leftarrow \theta_0 + \alpha \frac{1}{n} \sum_{k=1}^{n} (y_k - h_\theta(x_k))\, x_{(0,\,k)}} \qquad \boxed{\theta_1 \leftarrow \theta_1 + \alpha \frac{1}{n} \sum_{k=1}^{n} (y_k - h_\theta(x_k))\, x_{(1,\,k)}}$$

$$\vdots$$

$$\boxed{\theta_m \leftarrow \theta_m + \alpha \frac{1}{n} \sum_{k=1}^{n} (y_k - h_\theta(x_k))\, x_{(m,\,k)}}$$

Polynomial Regression (1/8)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/Position_Salaries.csv')
df.head()
```

[2]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

[3]:
```
X = df.iloc[:, 1:-1].values
y = df.iloc[:, -1].values
```

[4]:
```
print(type(X), X[:5], sep='\n')
```

Polynomial Regression (2/8)

```
<class 'numpy.ndarray'>
[[1]
 [2]
 [3]
 [4]
 [5]]
```

[5]:
```
print(type(y), y[:5], sep='\n')
```

```
<class 'numpy.ndarray'>
[ 45000  50000  60000  80000 110000]
```

Training the linear regression model on the whole dataset

[6]:
```
from sklearn.linear_model import LinearRegression
```

Training the linear regression model on the whole dataset

[7]:
```
lr_1 = LinearRegression()
lr_1.fit(X, y)
```

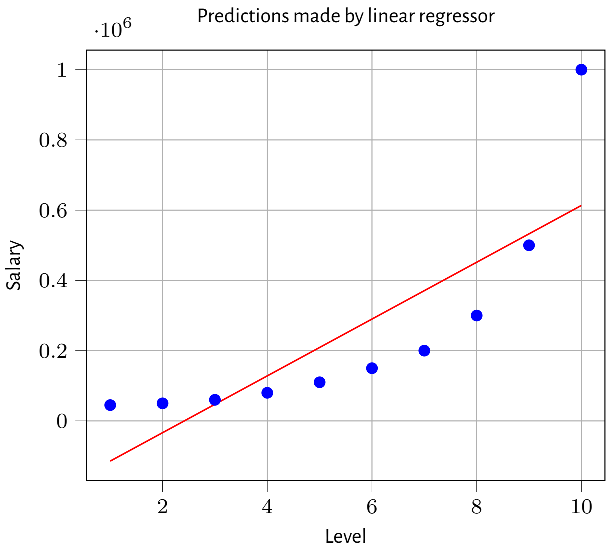
[7]:
```
LinearRegression()
```

Visualising the linear regression predictions

Polynomial Regression (3/8)

[8]:
```
plt.scatter(X, y, color = 'blue')
plt.plot(X, lr_1.predict(X), color = 'red')
plt.title('Predictions made by linear regressor')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.grid()
```

Polynomial Regression (4/8)



Polynomial Regression (5/8)

[9]:
```
from sklearn.preprocessing import PolynomialFeatures
```

[10]:
```
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
print(X_poly[:5])
lr_2 = LinearRegression()
lr_2.fit(X_poly, y)
```

```
[[  1.   1.   1.   1.   1.]
 [  1.   2.   4.   8.  16.]
 [  1.   3.   9.  27.  81.]
 [  1.   4.  16.  64. 256.]
 [  1.   5.  25. 125. 625.]]
```

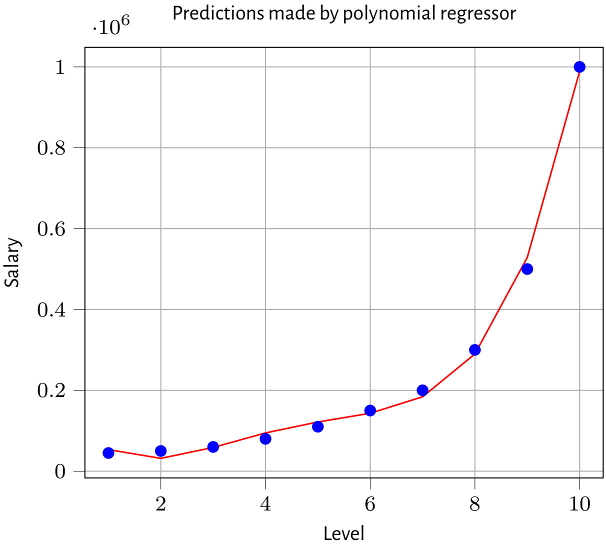
Polynomial Regression (6/8)

[10]: LinearRegression()

Visualising the polynomial regression predictions

[11]:
```
plt.scatter(X, y, color='blue')
plt.plot(X, lr_2.predict(poly_reg.fit_transform(X)), color='red')
plt.title('Predictions made by polynomial regressor')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.grid()
```

Polynomial Regression (7/8)



Polynomial Regression (8/8)

Predicting a new result using the linear regressor

```
[12]: lr_1.predict([[6.5]])
```

```
[12]: array([330378.78787879])
```

Predicting a new result using the polynomial regressor

```
[13]: lr_2.predict(poly_reg.fit_transform([[6.5]]))
```

```
[13]: array([158862.45265155])
```

Task #2

The yield y of a chemical process is a random variable whose value is considered to be a linear function of the temperature x. The following data of corresponding values of x and y is found:

Temperature in °C (x)	0	25	50	75	100
Yield in grams (y)	14	38	54	76	95

The linear regression model $y = \theta_0 + \theta_1 x$ is used. Determine the values of $\theta_0,\ \theta_1$.

1. Using normal equation,
2. Using gradient descent for 5 iterations.

$$y = \begin{bmatrix} 14 \\ 38 \\ 54 \\ 76 \\ 95 \end{bmatrix} \quad \text{et} \quad X = \begin{bmatrix} 1 & 0 \\ 1 & 25 \\ 1 & 50 \\ 1 & 75 \\ 1 & 100 \end{bmatrix} \implies X^T X = \begin{bmatrix} 5 & 250 \\ 250 & 18750 \end{bmatrix}$$

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \end{bmatrix} = \begin{bmatrix} 15.4 \\ 0.8 \end{bmatrix}$$

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

>>> X = np.array([[1,0], [1,25], [1,50], [1,75], [1,100]], dtype=np.float32)
>>> y = np.array([[14], [38], [54], [76], [95]])

>>> # NORMAL EQUATION
>>> XtX = X.T.dot(X)
>>> invXtX = np.linalg.inv(XtX)
>>> t_ne = invXtX.dot(np.matmul(X.T, y))

>>> print(t_ne)
[[15.39999944]
 [ 0.79999982]]

>>> X[:,1]= (X[:,1]-X[:,1].min())/X[:,1].max()
>>> y = (y-y.min())/y.max()

>>> # GRADIENT DESCENT
>>> t_gd = np.array([[1], [1]])
>>> alpha = .1
>>> vect = np.zeros(shape=(2, 1001))
>>> vect[:,0] = t_gd[[0,1],[0]]
>>> lost = []
```

```
>>> for k in range(1000):
...     eps = y-np.matmul(X, t_gd)
...     lost.append(1/(2*len(y))*eps.T.dot(eps)[[0],[0]][0])
...     t_gd = t_gd+alpha*1/len(y)*np.matmul((eps).T, X).T
...     vect[:,k+1] = t_gd[[0,1],[0]]
...
>>> print(vect[:, -1])
[0.01474565 0.84208938]

>>> plt.plot(vect[0, :], label=r'$\hat{\theta}_0$')
[<matplotlib.lines.Line2D object at 0x7f8188fc5f10>]
>>> plt.plot(vect[1, :], label=r'$\hat{\theta}_1$')
[<matplotlib.lines.Line2D object at 0x7f8188fd52e0>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x7f8188fd5520>
>>> plt.grid()
>>> #plt.show()

>>> plt.plot(lost)
[<matplotlib.lines.Line2D object at 0x7f8188fe66a0>]
>>> plt.grid()
>>> #plt.show()
```

F1-Score, **Accuracy**, **Recall** and **Precision** are calculated as follow:

$$f1 - \text{score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

$f1 -$ score denotes the *Harmonic Mean of Recall & Precision*

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

It denotes the ratio of how much we got right over all cases. Recall, on the other hand, designates the ratio of how much positives do we got right over all actual positive cases.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

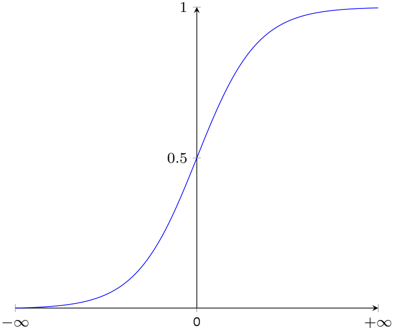
Precision, at last, is how much positives we got right over all positive predictions. It is given by:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Method	Pros		Cons	
Logistic Regression	▲	Probabilistic	▼	almost linearly separable data
K-NN	▲	Simple	▼	Number of neighbors k
	▲	Fast		
	▲	Efficient		
SVM	▲	***	▼	***
	▲	***	▼	***
Kernel SVM	▲	***	▼	***
	▲	***	▼	***
Naive Bayes	▲	***	▼	***
	▲	***	▼	***
Decision Tree Classification	▲	***	▼	***
	▲	***	▼	***
Random Forest Classification	▲	***	▼	***
	▲	***	▼	***

Logistic or S-shaped function σ

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$



- σ squashes range of distance from $]-\infty, +\infty[$ to $[0, 1]$
- σ is differentiable and easy to compute: $\boxed{\dot{\sigma} = \sigma \times (1 - \sigma)}$

Decision boundary

$$y = \sigma\left(\theta_1 x_{(1)} + \theta_2 x_{(2)} + \cdots + \theta_m x_{(m)}\right)$$

$$y = \frac{1}{1 + \mathrm{e}^{-\theta^T x}}$$

Hypothesis:

$$h_\theta\left(x\right) = \frac{1}{1 + \mathrm{e}^{-\theta^T x}} \qquad h_\theta\left(x_k\right) = \frac{1}{1 + \mathrm{e}^{-\theta^T x_k}}$$

Cost function:

$$\mathcal{J} = \begin{cases} -\ln\left(h_\theta(x)\right) & \text{if} \quad y = 1 \\ -\ln\left(1 - h_\theta(x)\right) & \text{if} \quad y = 0 \end{cases}$$

$$\boxed{\mathcal{J} = -y \ln\left(h_\theta(x)\right) - (1 - y)\ln\left(1 - h_\theta(x)\right)}$$

Gradient Descent

$$\boxed{\theta_i \;=\; \theta_i - \underbrace{\alpha}_{\text{Learning Rate}} \frac{\partial \mathcal{J}}{\partial \theta_i}}$$

Generalizing $\mathcal{J}$ yields: $\mathcal{J} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k \ln\left(h_\theta(x_k)\right) + (1-y_k)\ln\left(1-h_\theta(x_k)\right)\right)$

$$\Longrightarrow \quad \frac{\partial \mathcal{J}}{\partial \theta_i} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta(x_k)\right) x_{(i,\,k)}$$

$$\boxed{\theta_i \;\leftarrow\; \theta_i + \alpha \frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right) x_{(i,\,k)}}$$

$$\boxed{\theta_0 \;\leftarrow\; \theta_0 + \alpha \frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right) x_{(0,\,k)}} \qquad \boxed{\theta_1 \;\leftarrow\; \theta_1 + \alpha \frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right) x_{(1,\,k)}}$$

$$\vdots$$

$$\boxed{\theta_m \;\leftarrow\; \theta_m + \alpha \frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right) x_{(m,\,k)}}$$

Logistic Regression (1/8)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/Social_Network_Ads.csv')
df.head()
```

[2]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

[3]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

Logistic Regression (2/8)

[4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪25, random_state=0)
```

Feature Scaling

[5]:
```
from sklearn.preprocessing import StandardScaler
```

[6]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the Logistic Regression model on the Training set

[7]:
```
from sklearn.linear_model import LogisticRegression
```

[8]:
```
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

[8]:
```
LogisticRegression(random_state=0)
```

Predicting a new result

Logistic Regression (3/8)

[9]:
```
print(classifier.predict(sc.transform([[30,87000]])))
```

```
[0]
```

Predicting the Test set results

[10]:
```
y_pred = classifier.predict(X_test)
#print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
↪reshape(len(y_test),1)),1))
```

Making the Confusion Matrix

[11]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

[12]:
```
confusion_matrix(y_test, y_pred)
```

[12]:
```
array([[65,  3],
       [ 8, 24]])
```

[13]:
```
accuracy_score(y_test, y_pred)
```

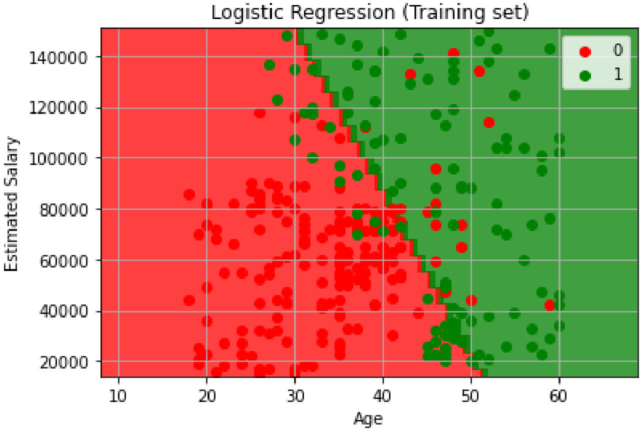
[13]:
```
0.89
```

Logistic Regression (4/8)

Map of Training set results

[14]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop =␣
↪X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop␣
↪= X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.
↪ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c =␣
↪ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.grid()
```

Logistic Regression (5/8)



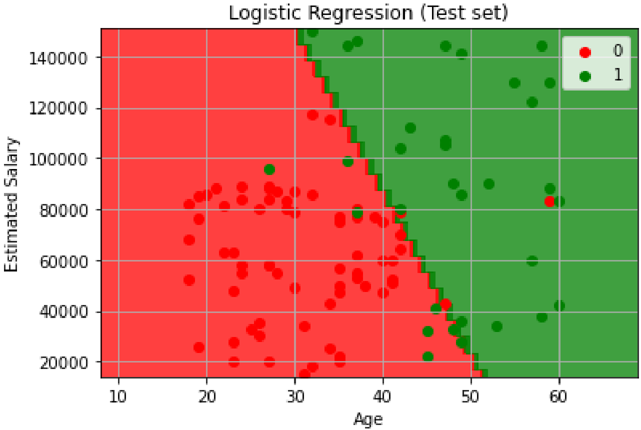
Logistic Regression (6/8)

Map of Test set results

Logistic Regression (7/8)

[ ]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.grid()
```

Logistic Regression (8/8)



K Nearest Neighbors (1/10)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/Social_Network_Ads.csv')
df.head()
```

[2]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

[3]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

K Nearest Neighbors (2/10)

[4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪25, random_state=0)
```

Feature Scaling

[5]:
```
from sklearn.preprocessing import StandardScaler
```

[6]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the K-NN model on the Training set

[7]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[8]:
```
classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski',␣
↪p=2)
```

[9]:
```
classifier.fit(X_train, y_train)
```

K Nearest Neighbors (3/10)

[9]:
```
KNeighborsClassifier()
```

Predicting a new result

[10]:
```
print(classifier.predict(sc.transform([[30,87000]])))
```

```
[0]
```

Predicting the Test set results

[11]:
```
y_pred = classifier.predict(X_test)
#print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
↪reshape(len(y_test),1)),1))
```

Displaying the Confusion Matrix

[12]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

[13]:
```
confusion_matrix(y_test, y_pred)
```

[13]:
```
array([[64,  4],
       [ 3, 29]])
```

K Nearest Neighbors (4/10)

[14]:
```
accuracy_score(y_test, y_pred)
```

[14]: 0.93

Map of Training set results

K Nearest Neighbors (5/10)

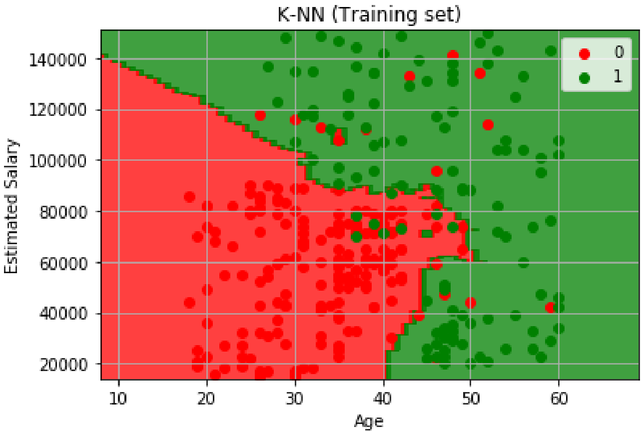
[15]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 1),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 1))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.grid()
```

K Nearest Neighbors (6/10)

```
*c* argument looks like a single numeric RGB or RGBA sequence, which␣
↪should be
avoided as value-mapping will have precedence in case its length␣
↪matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D␣
↪array with a
single row if you intend to specify the same RGB or RGBA value for all␣
↪points.
*c* argument looks like a single numeric RGB or RGBA sequence, which␣
↪should be
avoided as value-mapping will have precedence in case its length␣
↪matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D␣
↪array with a
single row if you intend to specify the same RGB or RGBA value for all␣
↪points.
```

m

K Nearest Neighbors (7/10)



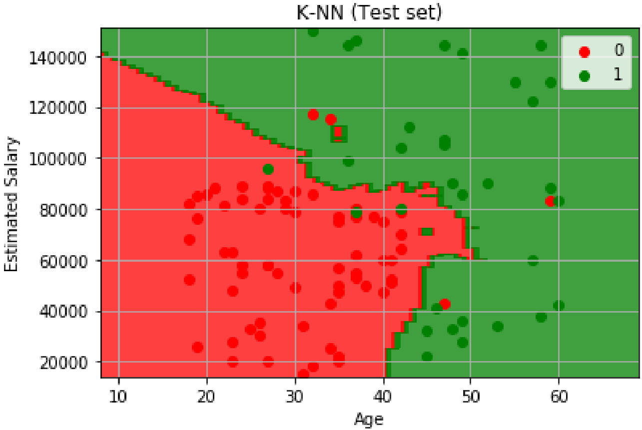
K Nearest Neighbors (8/10)

Map of Test set results

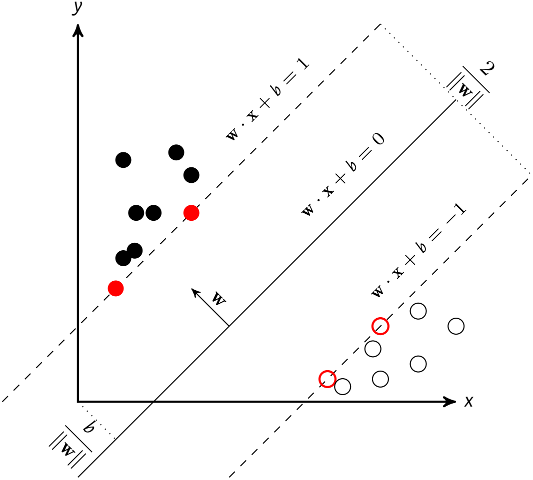
K Nearest Neighbors (9/10)

[16]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 1),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 1))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.grid()
```

K Nearest Neighbors (10/10)



Support Vector Machine (SVM)



Next...

KMeans (1/6)

Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the dataset

[2]:
```
df = pd.read_csv('datasets/Mall_Customers.csv')
df.head()
```

[2]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

[3]:
```
X = df.iloc[:, [3, 4]].values
```

Import `KMeans` **class**

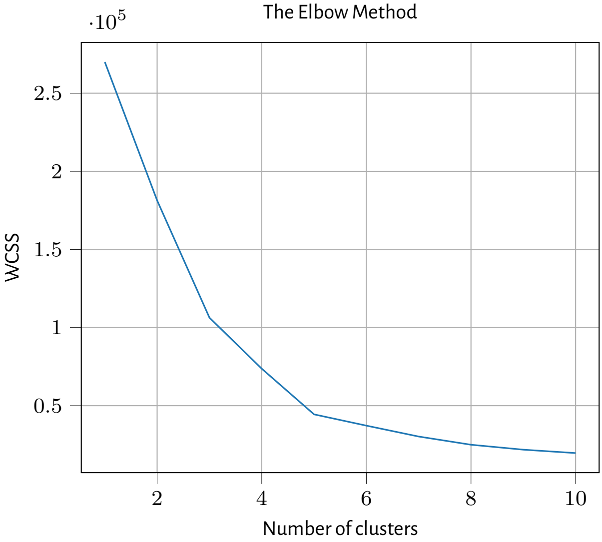
KMeans (2/6)

[4]:
```
from sklearn.cluster import KMeans
```

Using the elbow method to find the optimal number of clusters

[5]:
```
wcss = []
for i in range(1, 11):
kmeans = KMeans(n_clusters=i,
    init='k-means++', # Init method
    random_state=42) # Random seed for reproducibility
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid()
```

KMeans (3/6)


The Elbow Method
$\cdot 10^5$
WCSS
2.5
2
1.5
1
0.5
2
4
6
8
10
Number of clusters

KMeans (4/6)

Training the K-Means model on the dataset

[6]:
```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

Visualizing the clusters

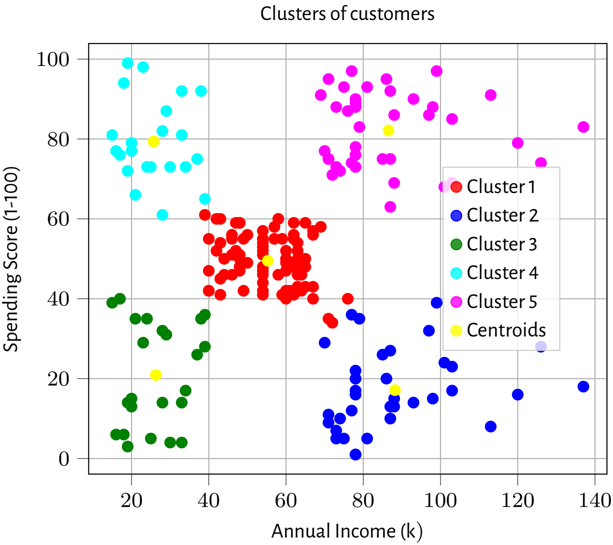
KMeans (5/6)

[7]:
```
plt.scatter(X[y_kmeans==0,0], X[y_kmeans==0,1],
    s=100, c='red', label='Cluster 1')
plt.scatter(X[y_kmeans==1,0], X[y_kmeans==1,1],
    s=100, c='blue', label='Cluster 2')
plt.scatter(X[y_kmeans==2,0], X[y_kmeans==2,1],
    s=100, c='green', label='Cluster 3')
plt.scatter(X[y_kmeans==3,0], X[y_kmeans==3,1],
    s=100, c='cyan', label='Cluster 4')
plt.scatter(X[y_kmeans==4,0], X[y_kmeans==4,1],
    s=100, c='magenta', label='Cluster 5')

plt.scatter(kmeans.cluster_centers_[:, 0],
    kmeans.cluster_centers_[:, 1],
    s=300, c='yellow', label='Centroids')

plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid()
```

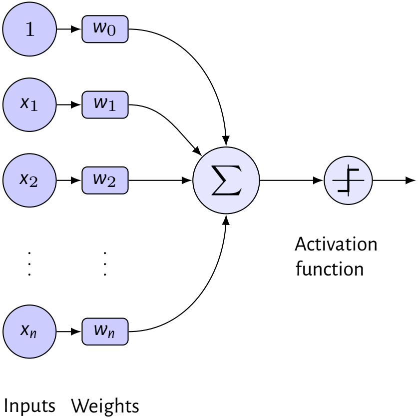
KMeans (6/6)


Clusters of customers
Spending Score (1-100)
Annual Income (k)
0
20
40
60
80
100
20
40
60
80
100
120
140
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids

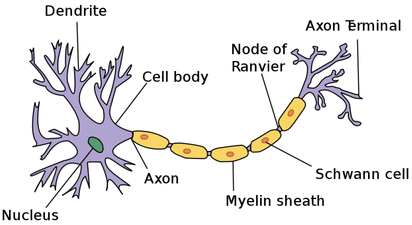
Next...

Fundamental unit of a neural network $(1/2)$

Artificial neuron

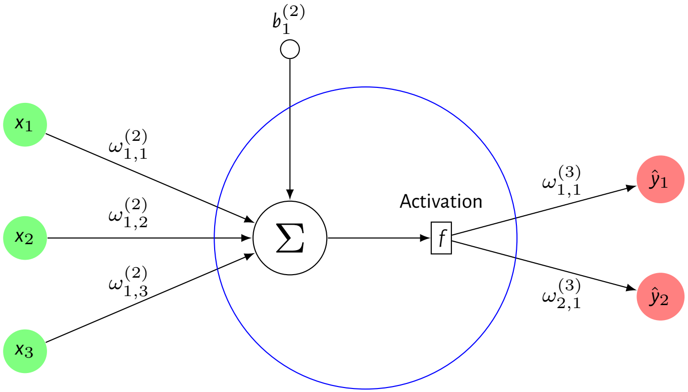


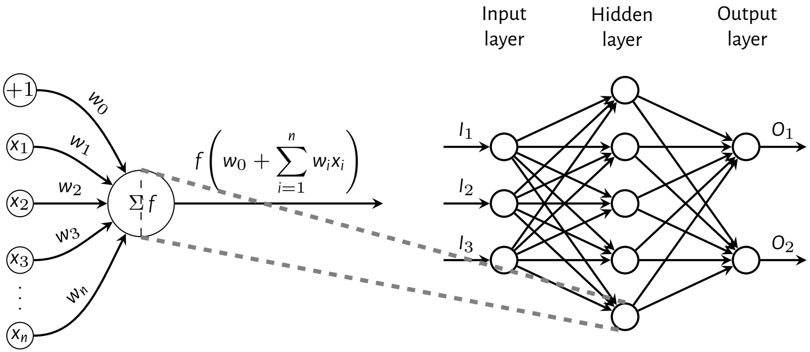
Biological neuron



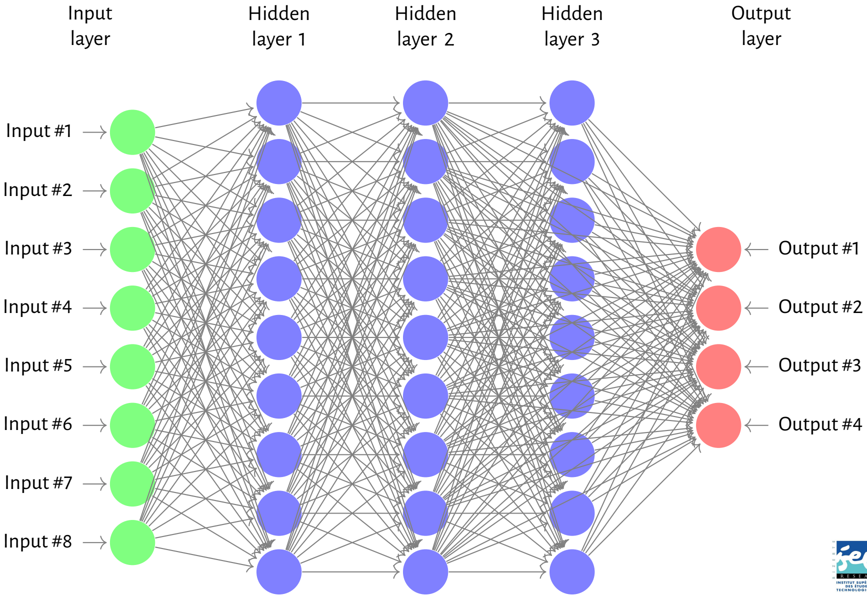
https://id.wikipedia.org/wiki/Sel_saraf

Fundamental unit of a neural network (2/2)

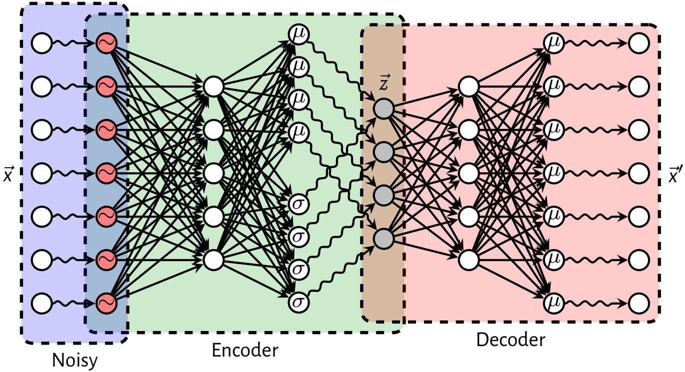



+1
x_1
x_2
x_3
x_n
w_0
w_1
w_2
w_3
w_n
Σf
$f\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$
Input layer
Hidden layer
Output layer
I_1
I_2
I_3
O_1
O_2

Multilayer Perceptron (MLP)



Variational Auto-Encoder



Next...

MCQ (1/13)

Features of Machine Learning are ...

1. Automation
2. Improved customer experience
3. Business intelligence
4. All of the above

The term machine learning was coined by ...

1. James Gosling
2. Arthur Samuel
3. Guido van Rossum
4. None of the above

Which among the following algorithms are used in Machine learning?

1. Naive Bayes
2. Support Vector Machines
3. K-Nearest Neighbors
4. All of the above

MCQ (2/13)

... is the machine learning algorithms that can be used with labeled data.

1. Regression algorithms
2. Clustering algorithms
3. Association algorithms
4. All of the above

Replace missing values with mean/median/mode helps to handle missing or corrupted data in a dataset. True/False?

1. True
2. False

The Real-world machine learning use cases are

1. Digital assistants
2. Chatbots
3. Fraud detection
4. All of the above

MCQ (3/13)

... is a part of machine learning that works with neural networks.

1. Deep learning
2. Artificial intelligence
3. All of the above
4. None of the above

The supervised learning problems can be grouped as ...

1. Regression problems
2. Classification problems
3. All of the above
4. None of the above

The unsupervised learning problems can be grouped as ...

1. Clustering
2. Association
3. All of the above
4. None of the above

MCQ (4/13)

Overfitting is a type of modelling error which results in the failure to predict future observations effectively or fit additional data in the existing model. Yes/No?

1. Yes
2. No
3. Can not say
4. Probably

... is the scenario when the model fails to decipher the underlying trend in the input data.

1. Underfitting
2. Overfitting
3. All of the above
4. None of the above

MCQ (5/13)

Machine learning approaches can be traditionally categorized into ... categories.

1. 3
2. 4
3. 7
4. 9

The categories in which Machine learning approaches can be traditionally categorized are ...

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning
4. All of the above

In general, to have a well-defined learning problem, we must identity which of the following

1. The class of tasks
2. The measure of performance to be improved
3. The source of experience
4. All of the above

MCQ (6/13)

The average positive difference between computed and desired outcome values

1. Root Mean Squared Error
2. Mean Squared Error
3. Mean Absolute Error
4. Mean Positive Error

... is used as an input to the machine learning model for training and prediction purposes.

1. Target variable
2. Feature vector
3. All of the above
4. None of the above

Simple regression assumes a ... relationship between the input attribute and output attribute.

1. linear
2. quadratic
3. reciprocal
4. inverse

MCQ (7/13)

What is Machine Learning (ML)?

1. The autonomous acquisition of knowledge through the use of manual programs
2. The selective acquisition of knowledge through the use of computer programs
3. The selective acquisition of knowledge through the use of manual programs
4. The autonomous acquisition of knowledge through the use of computer programs

The correlation between the number of years an emplyee has worked for a company and the salary of the employee is 0.75. What can be said about emplyee salary and years worked?

1. The is no relationship between salary and years worked.
2. Individuals that have worked for the company the longest have higher salaries.
3. Individuals that have worked for the company the longest have lower salaries.
4. The majority of employees have been with the company a long time.

MCQ (8/13)

Successful applications of ML

1. Learning to recognize spoken words
2. Learning to drive an autonomous vehicle
3. Learning to classify new astronomical structures
4. Learning to play world-class backgammon
5. All of the above

Which machine learning models are trained to make a series of decisions based on the rewards and feedback they receive for their actions?

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning
4. All of the above

MCQ (9/13)

Which of the following is not a type of supervised learning?

1. Classification
2. Regression
3. Clustering
4. None of the above

As the amount of training data increases

1. Training error usually increases and generalization error usually increases
2. Training error usually increases and generalization error usually decreases
3. Training error usually decreases and generalization error usually decreases
4. Training error usually decreases and generalization error usually increases

Which of the following are not classification tasks?

1. Find the gender of a person by analyzing his writing style
2. Detect Pneumonia from Chest X-ray images
3. Predict the price of a house based on floor area, number of rooms, etc.
4. Predict whether there will be abnormally heavy rainfall next year

MCQ (10/13)

Which of the following is a categorical feature?

1. Height of a person
2. Price of petroleum
3. Amount of rainfall in a day
4. Mother tongue of a person

What is the use of Validation dataset in Machine Learning?

1. To train the machine learning model.
2. To tune the hyperparameters of the machine learning model
3. To evaluate the performance of the machine learning model
4. None of the above

Which of the following criteria is typically used for optimizing in linear regression.

1. Minimize the number of points it touches.
2. Maximize the number of points it touches.
3. Minimize the squared distance from the points.
4. Minimize the maximum distance of a point from a line.

MCQ (11/13)

For two runs of K-Mean clustering, is it expected to get same clustering results?

1. Yes
2. No

Which of the following can act as possible termination conditions in K-Means?

a) For a fixed number of iterations

b) Assignment of observations to clusters does not change between iterations. Except for cases with a bad local minimum.

c) Centroids do not change between successive iterations

d) Terminate when RSS falls below a threshold

1. a, c & d
2. a, b & c
3. a, b & d
4. All of the above

MCQ (12/13)

In training a neural network, we notice that the loss does not increase in the first few starting epochs: What is the reason for this?

1. The learning rate is low
2. Regularization parameter is high
3. Stuck at the local minima
4. All of the above

Which of the following is true about model capacity *(where model capacity means the ability of neural network to approximate complex functions)*?

1. As number of hidden layers increase, model capacity increases
2. As dropout ratio increases, model capacity increases
3. As learning rate increases, model capacity increases
4. None of these

MCQ (13/13)

When there is noise in data, which of the following options would improve the performance of the KNN algorithm?

1. Increase the value of k
2. Decrease the value of k
3. Changing value of k will not change the effect of the noise
4. None of these

Logistic Regression is used for ...

1. regression purposes
2. classification purposes
3. all of the above
4. none of the above

Which of the following methods do we use to best fit the data in Logistic Regression?

1. Least Squared Error
2. Maximum Likelihood
3. Jaccard distance

Further Reading

I. El Naqa and M. J. Murphy. "What Is Machine Learning?" In: *Machine Learning in Radiation Oncology: Theory and Applications*. Ed. by I. El Naqa, R. Li, and M. J. Murphy. Cham: Springer International Publishing, 2015, pp. 3–11. DOI: 10.1007/978-3-319-18305-3_1.

J. Schmidt et al. "Recent advances and applications of machine learning in solid-state materials science". In: *npj Computational Materials* 5.1 (Aug. 2019). DOI: 10.1038/s41524-019-0221-0 (cit. on p. 15).