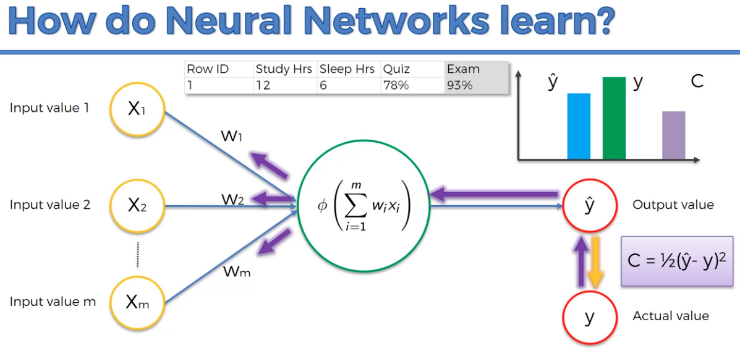
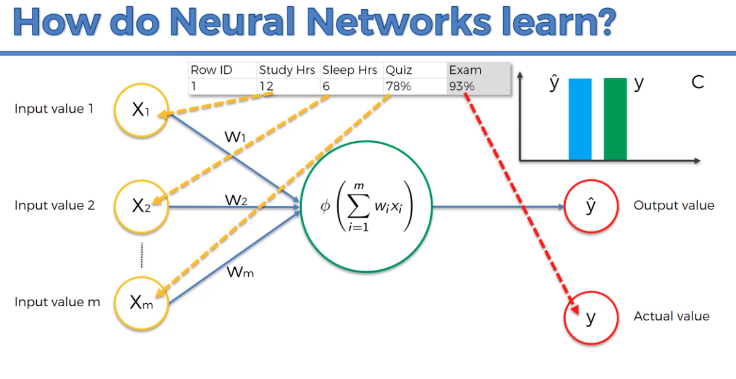
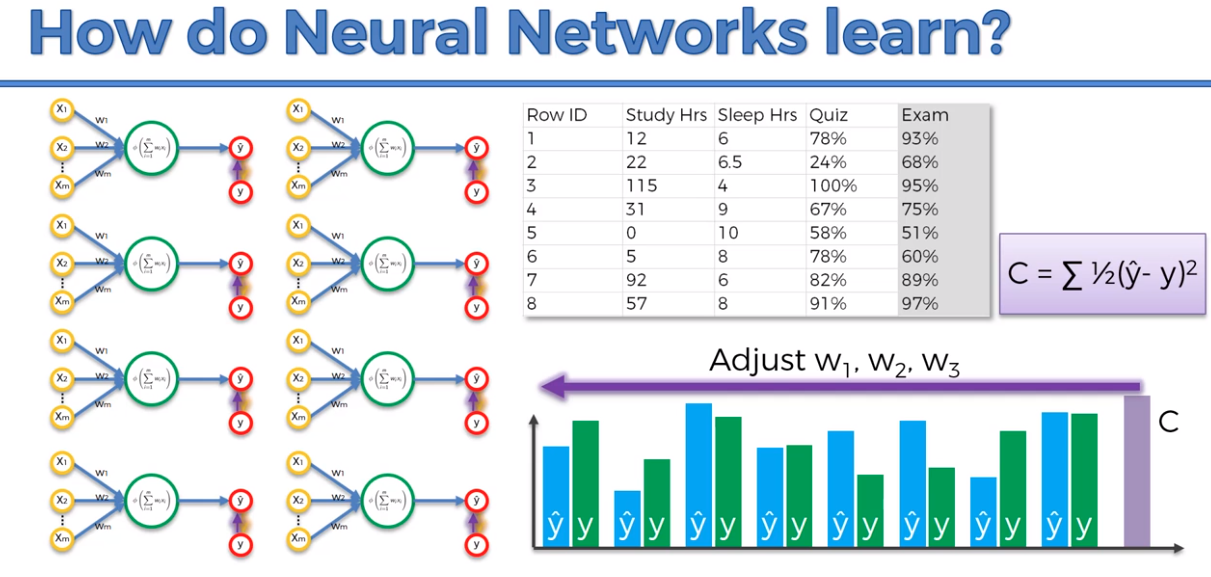
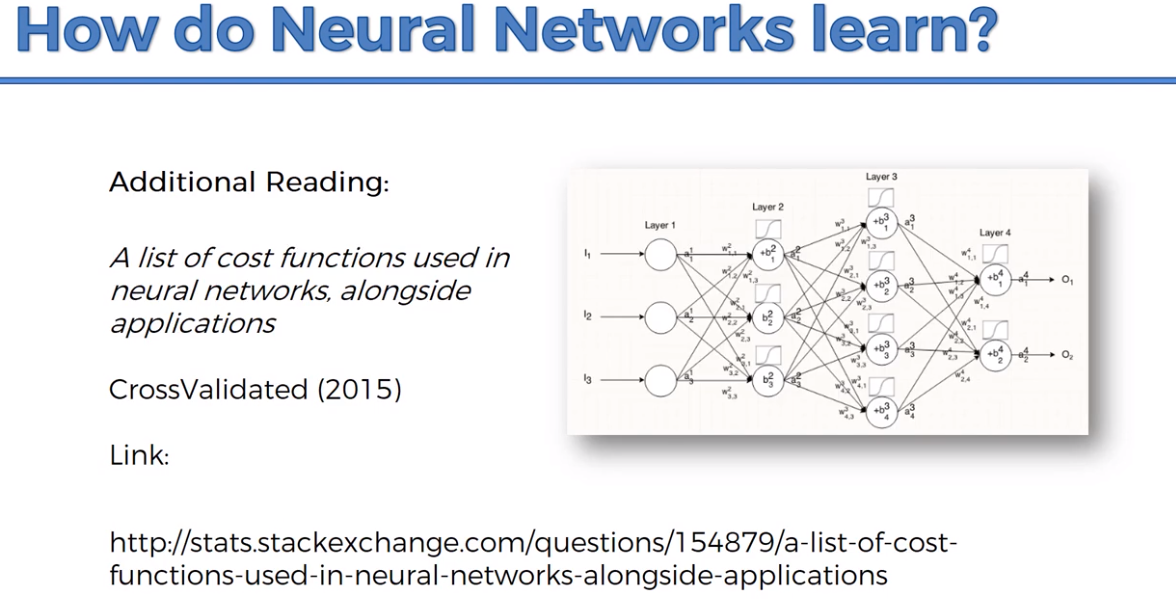


**Goal: Minimize Cost Function**

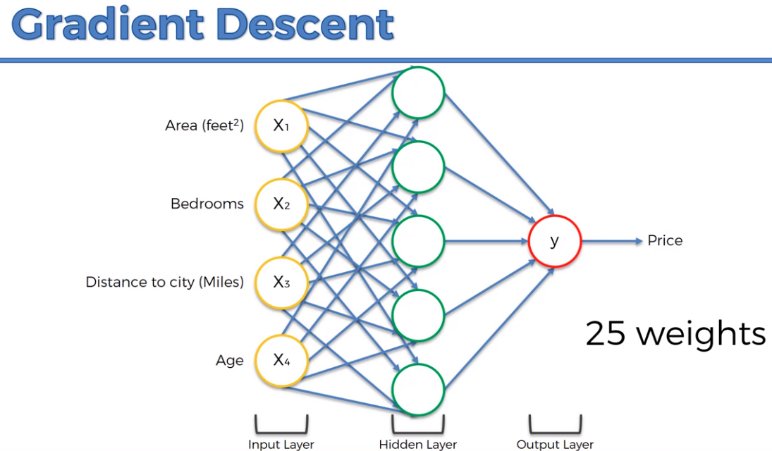


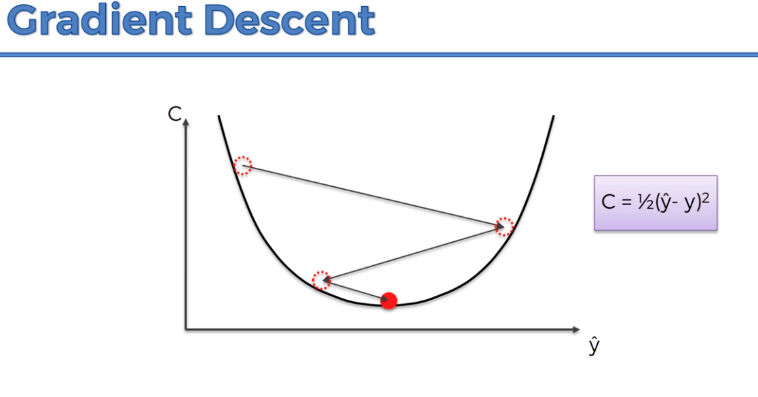
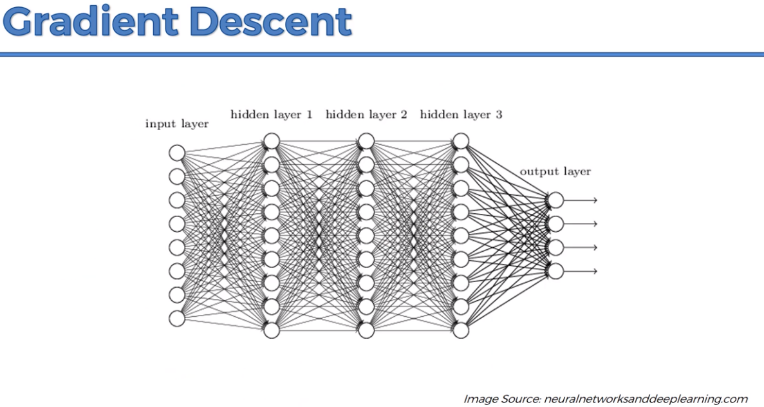


Here we have only 1 NN. But, 8 NNs are shown to have 8 different rows from the dataset to calculate the Cost Function and the adjustment is done based on that. A number of times this whole process i.e., **epoch** is run and optimized for best possible minimal value of cost function. Row ID has no contribution is predicting ‘Exam’ feature so, it is not included.



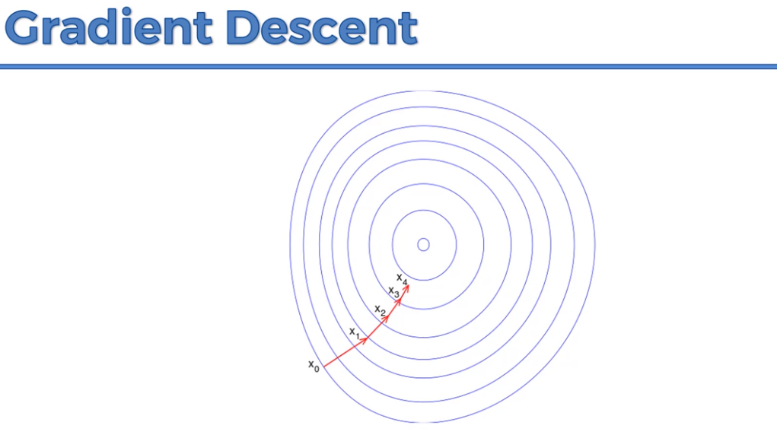
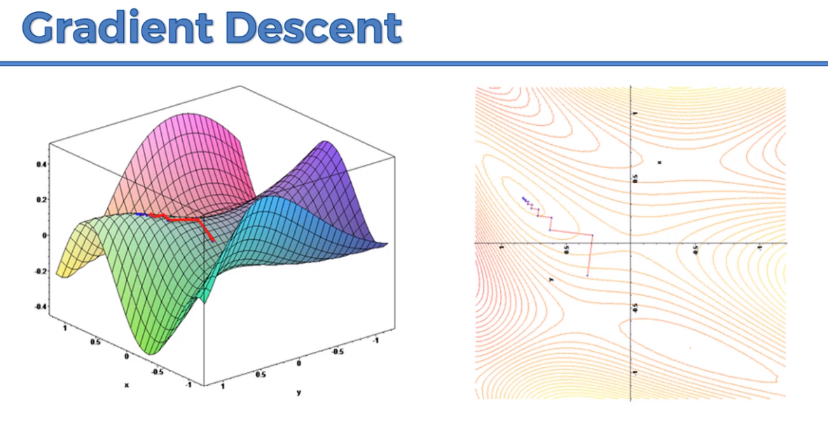
**Example**



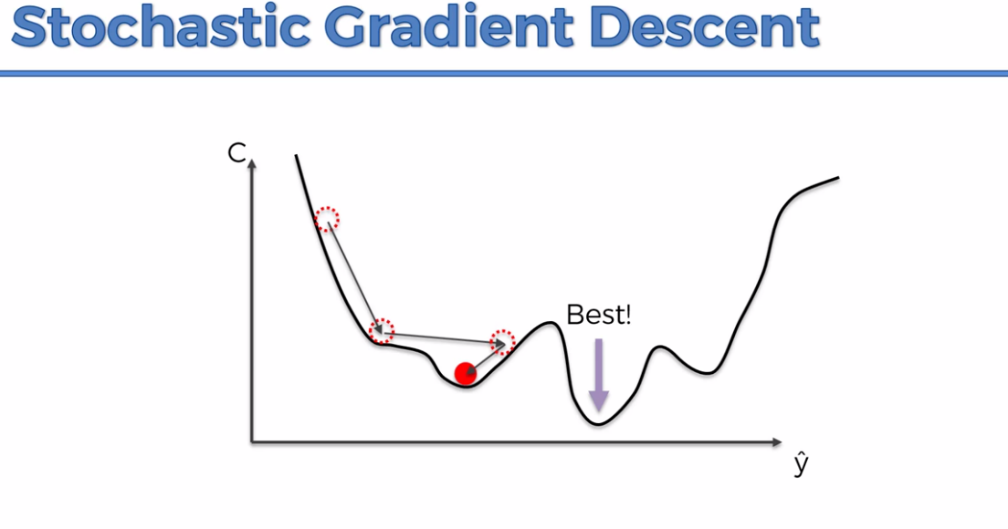


**Gradient Descent Finds Local Minima**

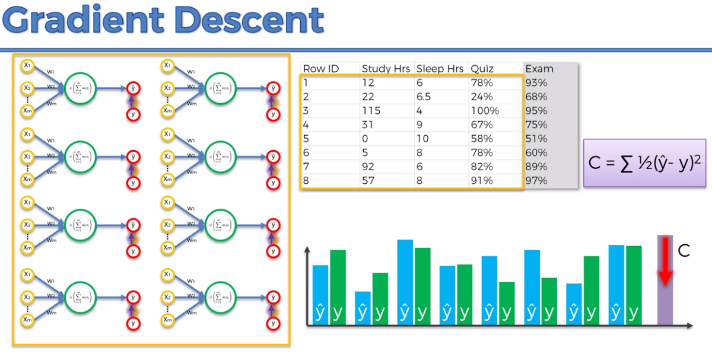
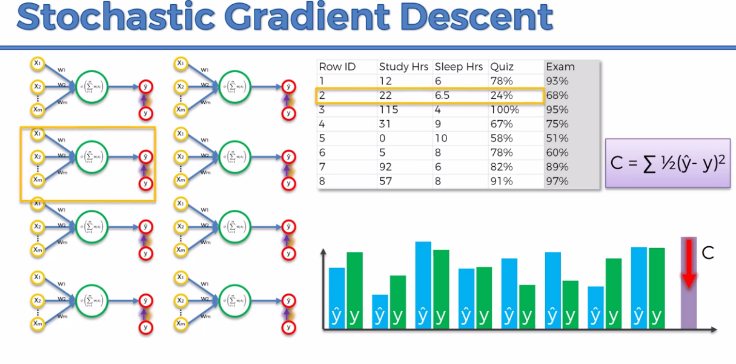
In 2D In 3D

* **If Cost Function is not ins standard form or in multi-dimension**
* **Gradient Descent requires cost function to be convex.**

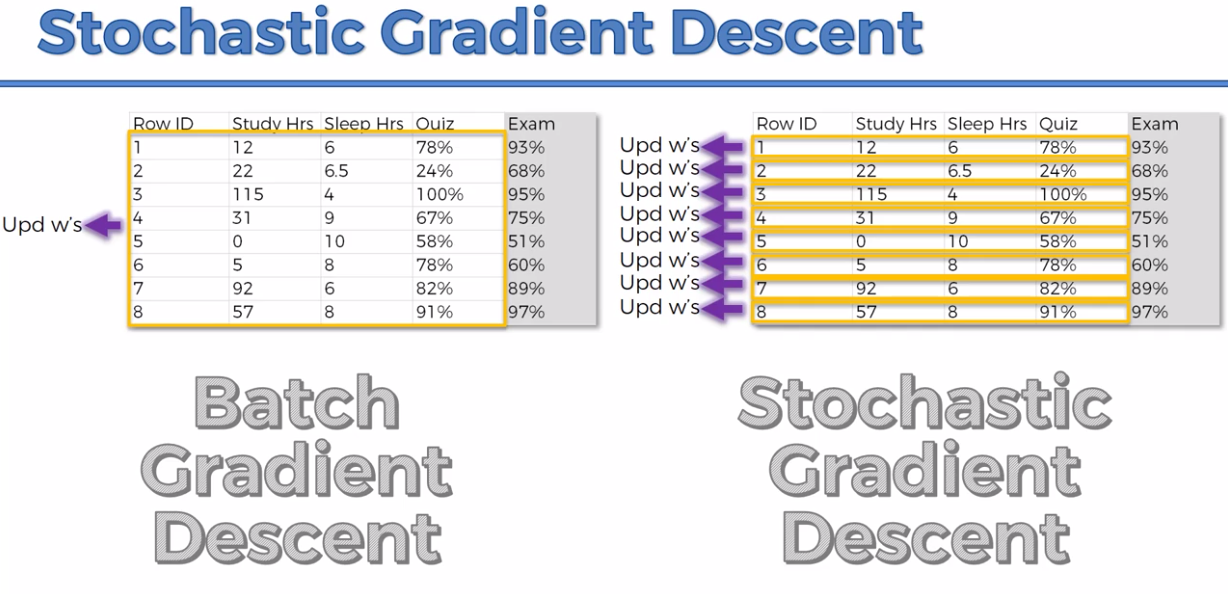


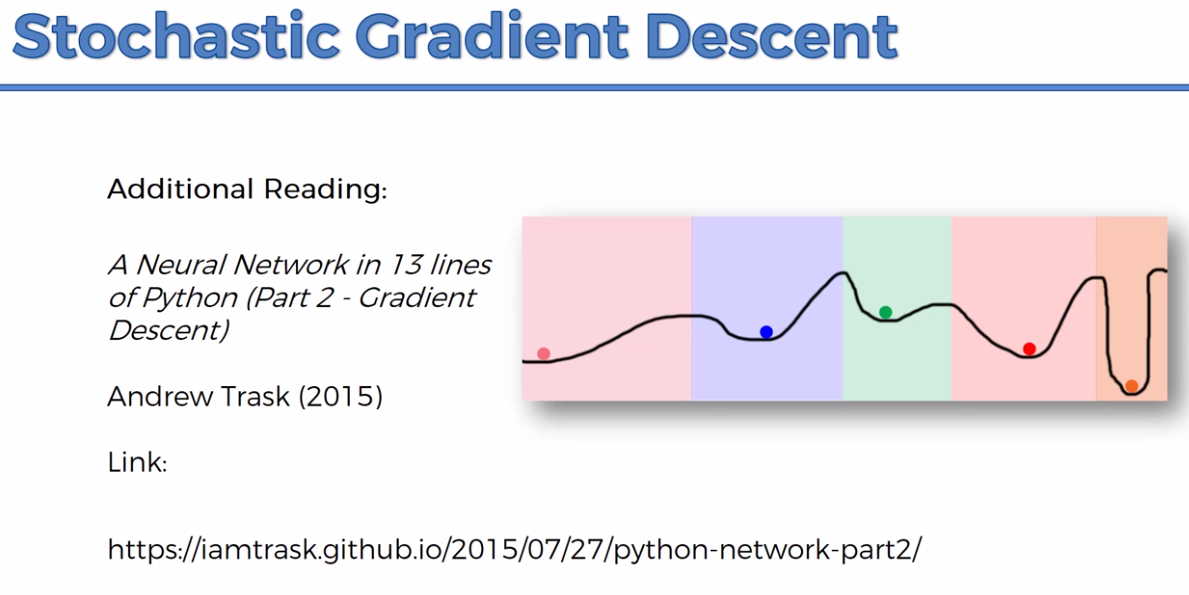
**We take whole dataset or whole BATCH of our sample**

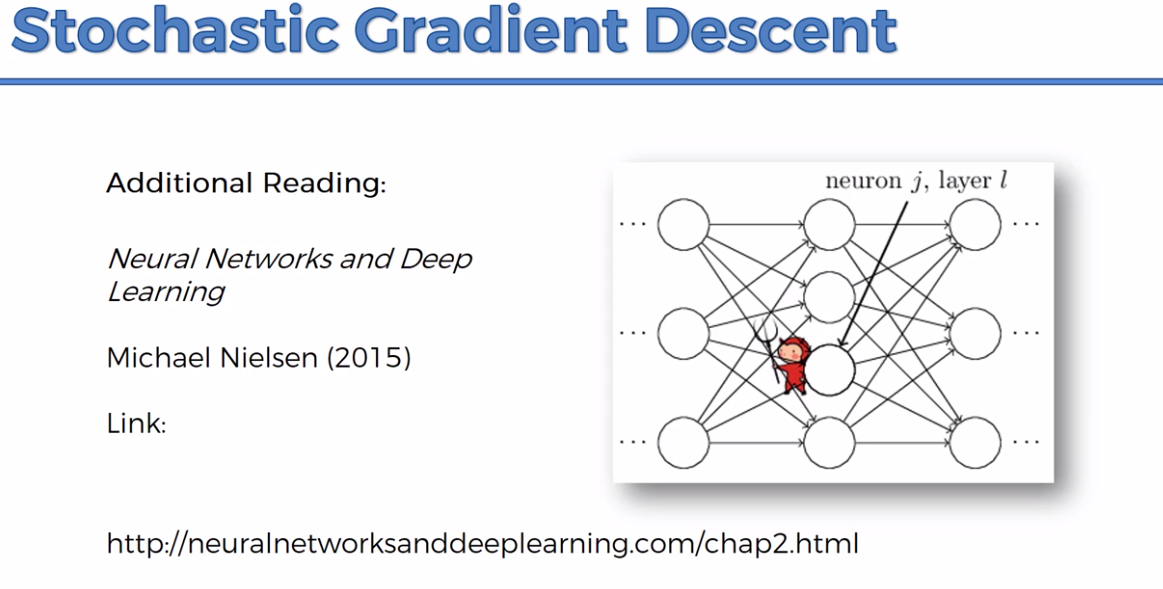
 

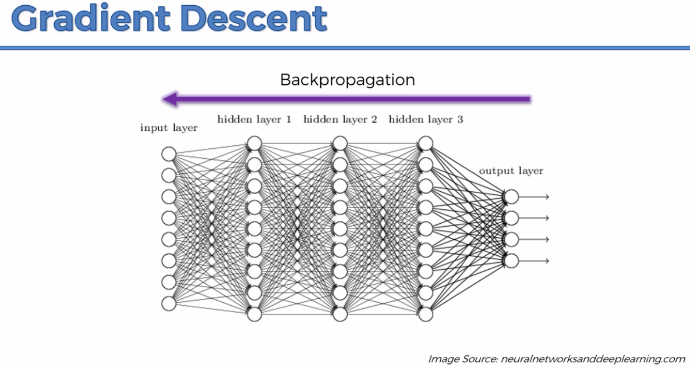
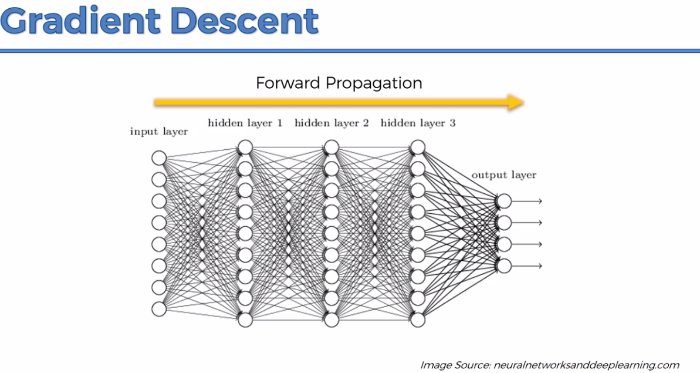
**We take one row at a time and adjust the weight**

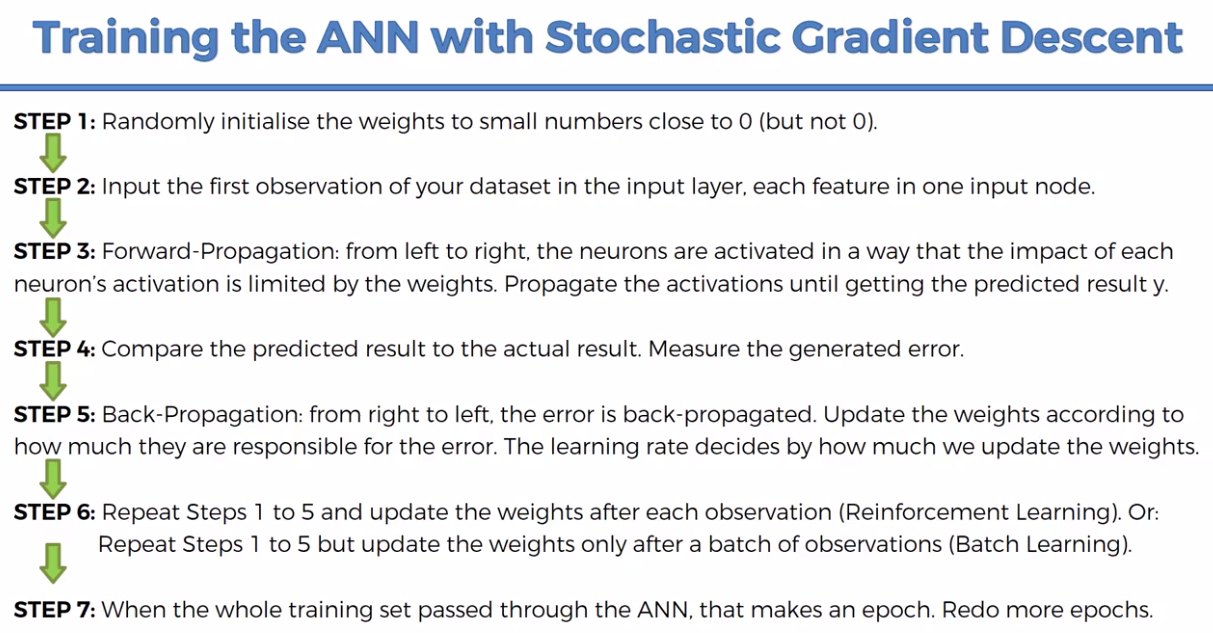
**Stochastic Descent Gradient finds global minimum and faster than gradient descent.**





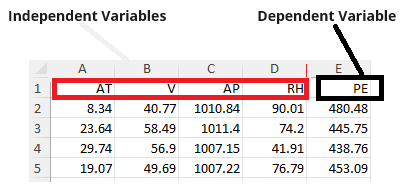






FREE COURSE on ANN for Linear Regression

[**https://www.udemy.com/course/linear-regression-with-artificial-neural-network/**](https://www.udemy.com/course/linear-regression-with-artificial-neural-network/)



**# -\*- coding: utf-8 -\*-**

**# Artificial Neural Network**

**### Importing the libraries**

import numpy as np

import pandas as pd

import tensorflow as tf

tf.\_\_version\_\_

**"""## Part 1 - Data Preprocessing**

**### Importing the dataset**

dataset = pd.read\_excel('Folds5x2\_pp.xlsx')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

**"""### Splitting the dataset into the Training set and Test set"""**

from sklearn.model\_selection import train\_test\_split

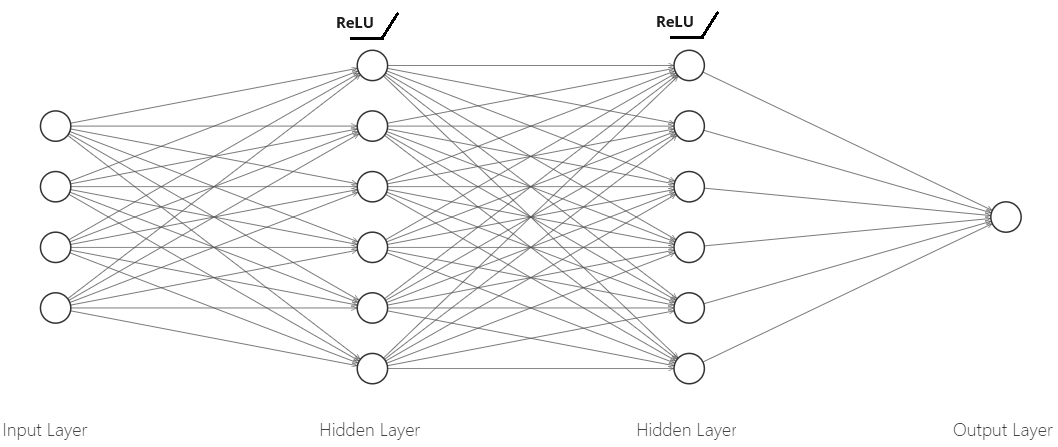
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

**"""## Part 2 - Building the ANN**

**### Initializing the ANN**

**"""**

**###https://alexlenail.me/NN-SVG/index.html**



ann = tf.keras.models.Sequential()

**"""### Adding the input layer and the first hidden layer"""**

ann.add(tf.keras.layers.Dense(units=6, activation='**relu**'))

**"""### Adding the second hidden layer"""**

ann.add(tf.keras.layers.Dense(units=6, activation='**relu**'))

**"""### Adding the output layer"""**

ann.add(tf.keras.layers.Dense(units=1))

**"""## Part 3 - Training the ANN**

**### Compiling the ANN**

**"""**

ann.compile(optimizer = 'adam', loss = '**mean\_squared\_error'**)

**"""### Training the ANN model on the Training set"""**

ann.fit(X\_train, y\_train, batch\_size = 32, epochs = 100)

**"""## Part 4 – Prediction**

**“””### Predict for a single input”””**

print(ann.predict([[8.34,40.77,1010.84,90.01]]))

**"""### Predicting the results of the Test set"""**

y\_pred = ann.predict(X\_test)

np.set\_printoptions(precision=2)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

|  |
| --- |
| [[431.56 431.23]  [462.62 460.01]  [466.12 461.14]  ...  [473.34 473.26]  [440.16 438. ]  [459.35 463.28]] |

**""""# Step 5. Measuring Accuracy"""**

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

**# Predicting the results of the Test set**

y\_pred = ann.predict(X\_test)

**# Calculating the Mean Absolute Error**

mae = mean\_absolute\_error(y\_test, y\_pred)

print("Mean Absolute Error:", mae)

**# Calculating the Mean Squared Error**

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean Squared Error:", mse)

**# Calculating the Root Mean Squared Error**

rmse = np.sqrt(mse)

print("Root Mean Squared Error:", rmse)

**# Calculating the R-squared Score**

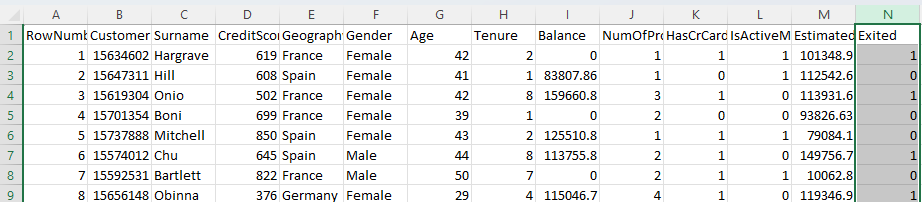
r2 = r2\_score(y\_test, y\_pred)

print("R-squared Score:", r2)

|  |
| --- |
| Mean Absolute Error: 3.957771647262972  Mean Squared Error: 24.650398030113276  Root Mean Squared Error: 4.9649167193532335  R-squared Score: 0.9157216299687292 |

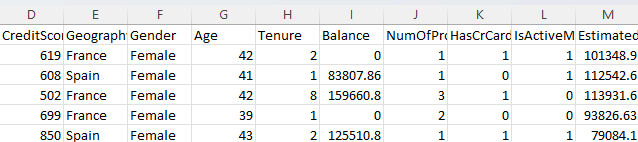
**Binary Classification**

Initially, 13 features including target feature.



Dependent Variable (DV) or Target variable is “Exited”.

We select below features as Independent Variable (IV).



Now, we consider 10 features.

**# Artificial Neural Network**

**# Importing the libraries**

import numpy as np

import pandas as pd

import tensorflow as tf

# tf.\_\_version\_\_

**# Part 1 - Data Preprocessing**

**# Importing the dataset**

dataset = pd.read\_csv('Churn\_Modelling.csv')

X = dataset.iloc[:, 3:-1].values

y = dataset.iloc[:, -1].values

print(X)

|  |
| --- |
| [[619 'France' 'Female' ... 1 1 101348.88]  [608 'Spain' 'Female' ... 0 1 112542.58]  [502 'France' 'Female' ... 1 0 113931.57]  ...  [709 'France' 'Female' ... 0 1 42085.58]  [772 'Germany' 'Male' ... 1 0 92888.52]  [792 'France' 'Female' ... 1 0 38190.78]] |

print(y)

|  |
| --- |
| [1 0 1 ... 1 1 0] |

**# Encoding categorical data**

**# Label Encoding the "Gender" column**

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X[:, 2] = le.fit\_transform(X[:, 2])

print(X)

|  |
| --- |
| [[619 'France' 0 ... 1 1 101348.88]  [608 'Spain' 0 ... 0 1 112542.58]  [502 'France' 0 ... 1 0 113931.57]  ...  [709 'France' 0 ... 0 1 42085.58]  [772 'Germany' 1 ... 1 0 92888.52]  [792 'France' 0 ... 1 0 38190.78]]  # It makes the ‘Gender’ feature two labels such as 0 for ‘Female’ and 1 for ‘Male’ |

**# One Hot Encoding the "Geography" column**

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

|  |
| --- |
| **[[1.0 0.0 0.0** ... 1 1 101348.88]  [**0.0 0.0 1.0** ... 0 1 112542.58]  [**1.0 0.0 0.0** ... 1 0 113931.57]  ...  [1.0 0.0 0.0 ... 0 1 42085.58]  [0.0 1.0 0.0 ... 1 0 92888.52]  [1.0 0.0 0.0 ... 1 0 38190.78]]  #We get dummy variables for each of the ‘Geography’ features such as ‘France’, ‘Spain’, ‘Germany’ such as  ‘1 0 0’ for ‘France’  ‘0 0 1’ for ‘Spain’  ‘0 1 0’ for ‘Germany’  So, now total features i.e., columns are 12.  9 for other features  3 new dummy for ‘Geography’ |

**# Splitting the dataset into the Training set and Test set**

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

**# Feature Scaling**

from sklearn.preprocessing import StandardScaler

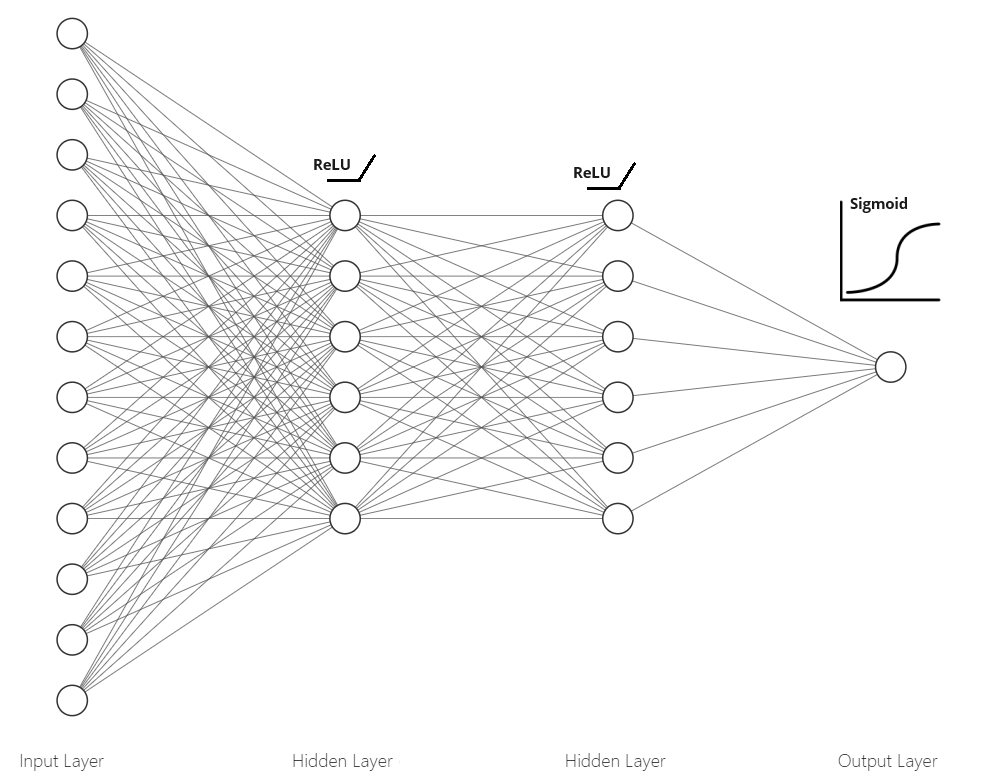
sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

**# Part 2 - Building the ANN**

**#Fully Connected Artificial Neural Network with two 6 neuron hidden layers and 1 output layer**



**###https://alexlenail.me/NN-SVG/index.html**

**# Initializing the ANN**

ann = tf.keras.models.Sequential()

**# Adding the input layer and the first hidden layer**

ann.add(tf.keras.layers.Dense(units=6, activation='**relu**'))

**# Adding the second hidden layer**

ann.add(tf.keras.layers.Dense(units=6, activation='**relu**'))

**# Adding the output layer**

ann.add(tf.keras.layers.Dense(units=1, activation='**sigmoid**'))

**############## Binary Classification:** activation = ‘sigmoid’

**############## Non-Binary Classification:** activation = ‘softmax’

**# Part 3 - Training the ANN**

**# Compiling the ANN**

ann.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

**############## Binary Classification:** loss = 'binary\_crossentropy'

**############## Non-Binary Classification:** loss = ‘categorical\_crossentropy’

**# Training the ANN on the Training set**

ann.fit(X\_train, y\_train, batch\_size = 32, epochs = 100)

**# Part 4 - Making the predictions and evaluating the model**

**# Predicting the result of a single observation**

"""

Homework:

Use our ANN model to predict if the customer with the following informations will leave the bank:

Geography: France

Credit Score: 600

Gender: Male

Age: 40 years old

Tenure: 3 years

Balance: $ 60000

Number of Products: 2

Does this customer have a credit card? Yes

Is this customer an Active Member: Yes

Estimated Salary: $ 50000

So, should we say goodbye to that customer?

Solution:

"""

print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])) > 0.5)

"""

Therefore, our ANN model predicts that this customer stays in the bank!

Important note 1: Notice that the values of the features were all input in a double pair of square brackets. That's because the "predict" method always expects a 2D array as the format of its inputs. And putting our values into a double pair of square brackets makes the input exactly a 2D array.

Important note 2: Notice also that the "France" country was not input as a string in the last column but as "1, 0, 0" in the first three columns. That's because of course the predict method expects the one-hot-encoded values of the state, and as we see in the first row of the matrix of features X, "France" was encoded as "1, 0, 0". And be careful to include these values in the first three columns, because the dummy variables are always created in the first columns.

"""

**# Predicting the Test set results**

y\_pred = ann.predict(X\_test)

y\_pred = (y\_pred > 0.5)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

|  |
| --- |
| 63/63 [==============================] - 0s 1ms/step  [[0 0]  [0 1]  [0 0]  ...  [0 0]  [0 0]  [0 0]] |

**# Making the Confusion Matrix**

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

|  |
| --- |
| [[1511 84]  [ 197 208]]  Out[20]: 0.8595 |