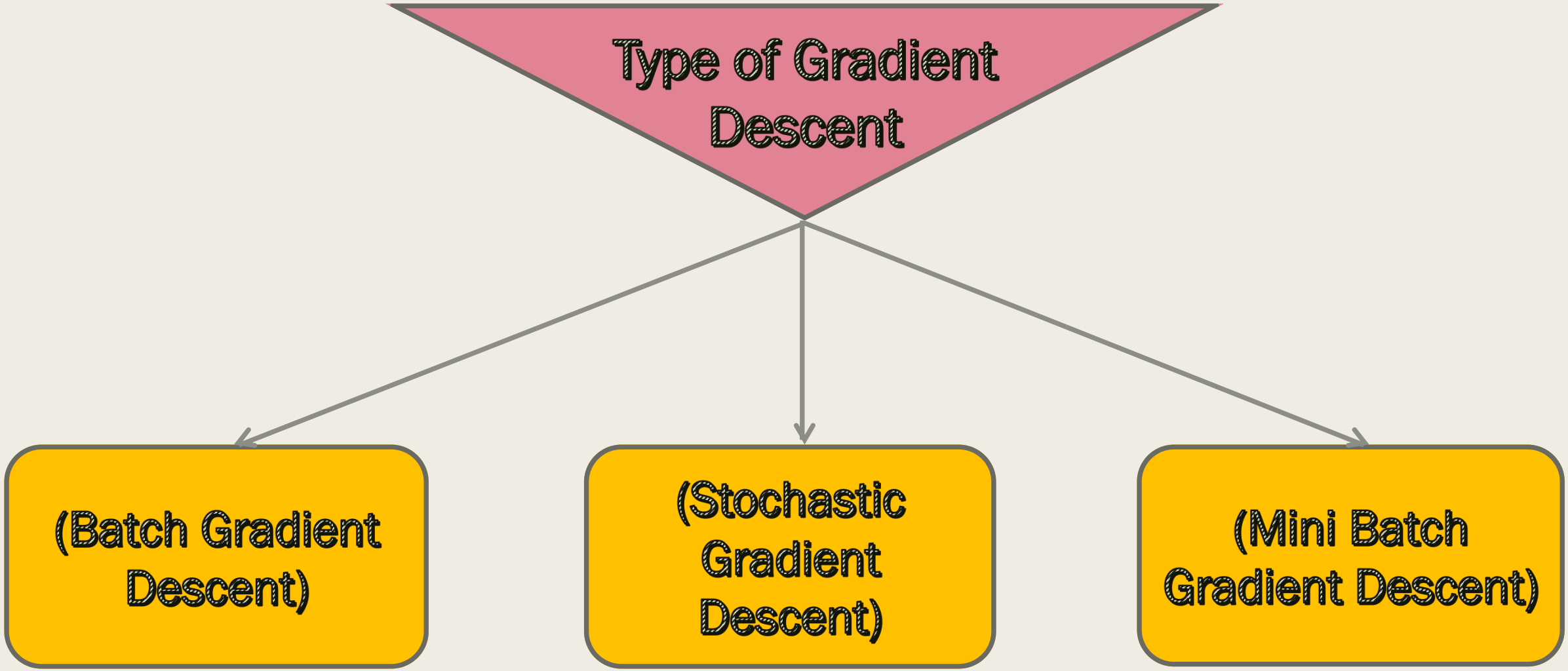
ML

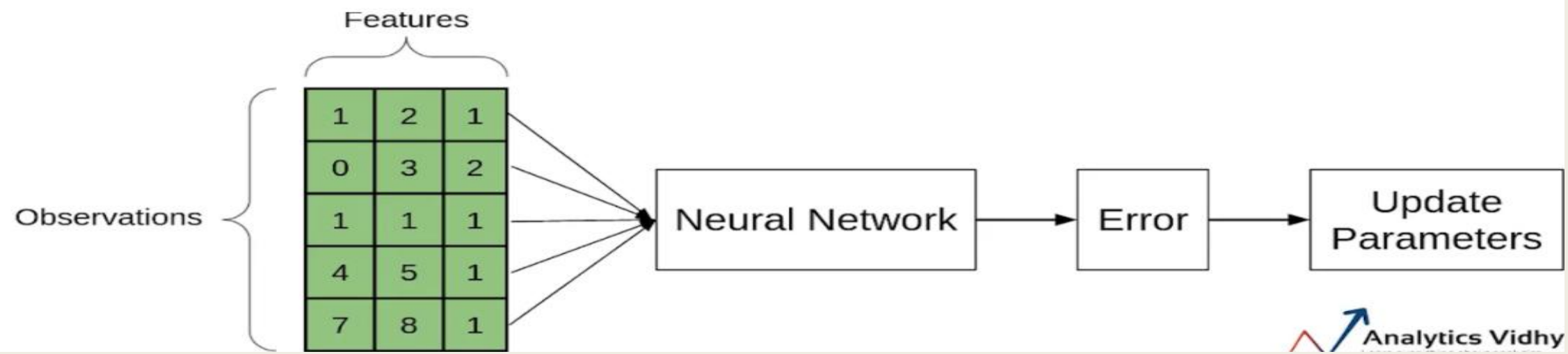
LECTURE-6

BY
Dr. Ramesh Kumar Thakur
Assistant Professor (II)
School Of Computer Engineering


Type of Gradient Descent
(Batch Gradient Descent)
(Stochastic Gradient Descent)
(Mini Batch Gradient Descent)

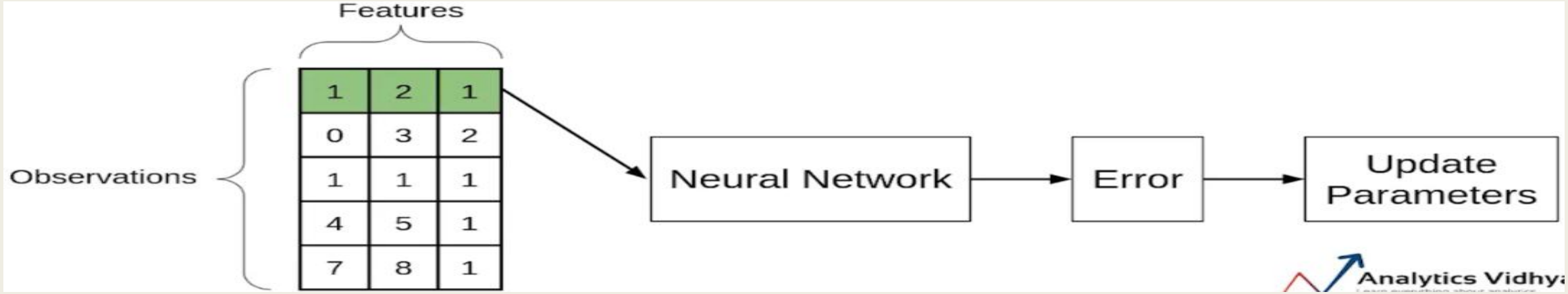
Batch Gradient Descent

- In Batch Gradient Descent, all the training data is taken into consideration to take a single step.
- We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that's just one step of gradient descent in one epoch.
- In batch Gradient Descent since we are using the entire training set, the parameters will be updated only once per epoch.
- Batch Gradient Descent is great for convex or relatively smooth error manifolds.
- In this case, we move somewhat directly towards an optimum solution.
- The graph of cost vs epochs is also quite smooth because we are averaging over all the gradients of training data for a single step. The cost keeps on decreasing over the epochs.



Stochastic Gradient Descent

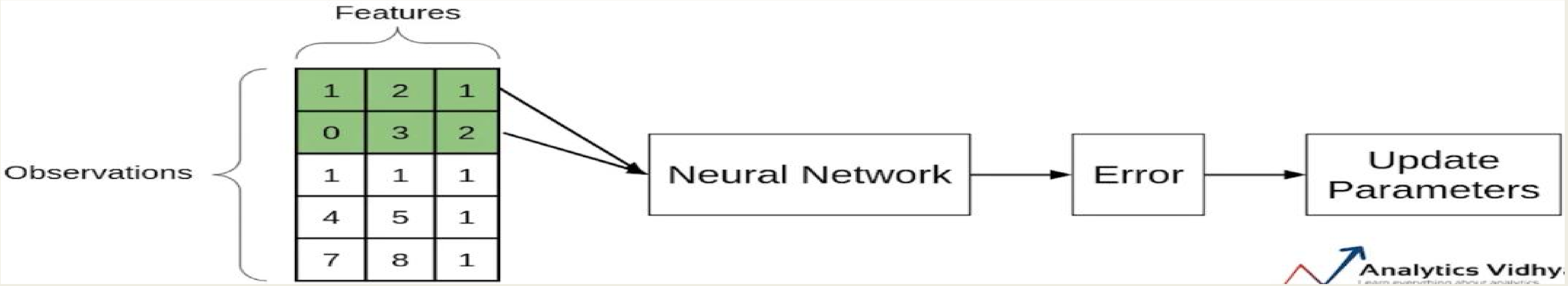
- In Batch Gradient Descent we were considering all the examples for every step of Gradient Descent. But what if our dataset is very huge.
- Suppose our dataset has 5 million examples, then just to take one step the model will have to calculate the gradients of all the 5 million examples.
- This does not seem an efficient way. To tackle this problem we have Stochastic Gradient Descent.
- In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step.
- Since we are considering just one example at a time the cost will fluctuate over the training examples and it will not necessarily decrease. But in the long run, you will see the cost decreasing with fluctuations.
- Because the cost is so fluctuating, it will never reach the minima but it will keep dancing around it.
- SGD can be used for larger datasets. It converges faster when the dataset is large as it causes updates to the parameters more frequently.

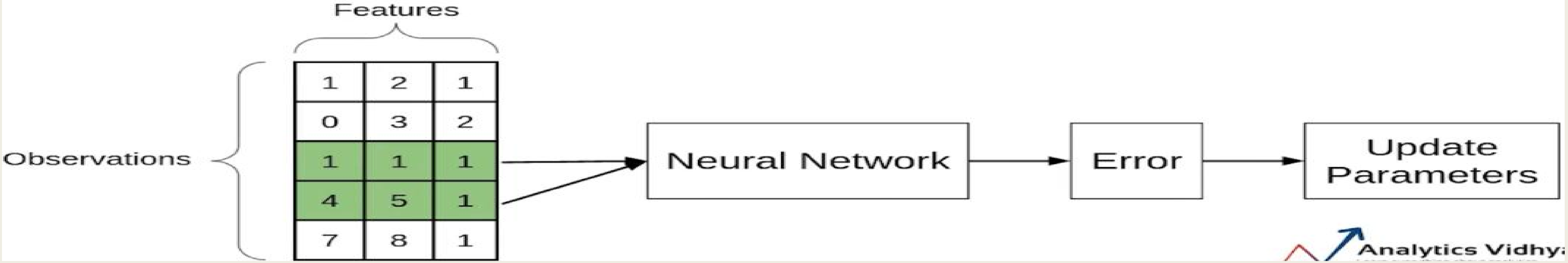


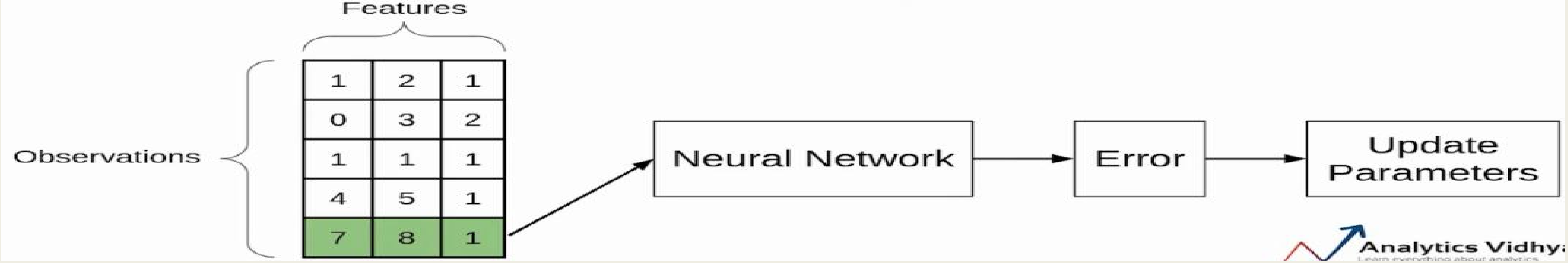
Mini Batch Gradient Descent

- Batch Gradient Descent converges directly to minima. SGD converges faster for larger datasets. But, since in SGD we use only one example at a time, we cannot implement the vectorized implementation on it.
- This can slow down the computations. To tackle this problem, a mixture of Batch Gradient Descent and SGD is used.
- Neither we use all the dataset all at once nor we use the single example at a time.
- We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch. Doing this helps us achieve the advantages of both the former variants (GD and SGD).
- Just like SGD, the average cost over the epochs in mini-batch gradient descent fluctuates because we are averaging a small number of examples at a time.
- When we are using the mini-batch gradient descent we are updating our parameters frequently as well as we can use vectorized implementation for faster computations.

Mini Batch Gradient Descent

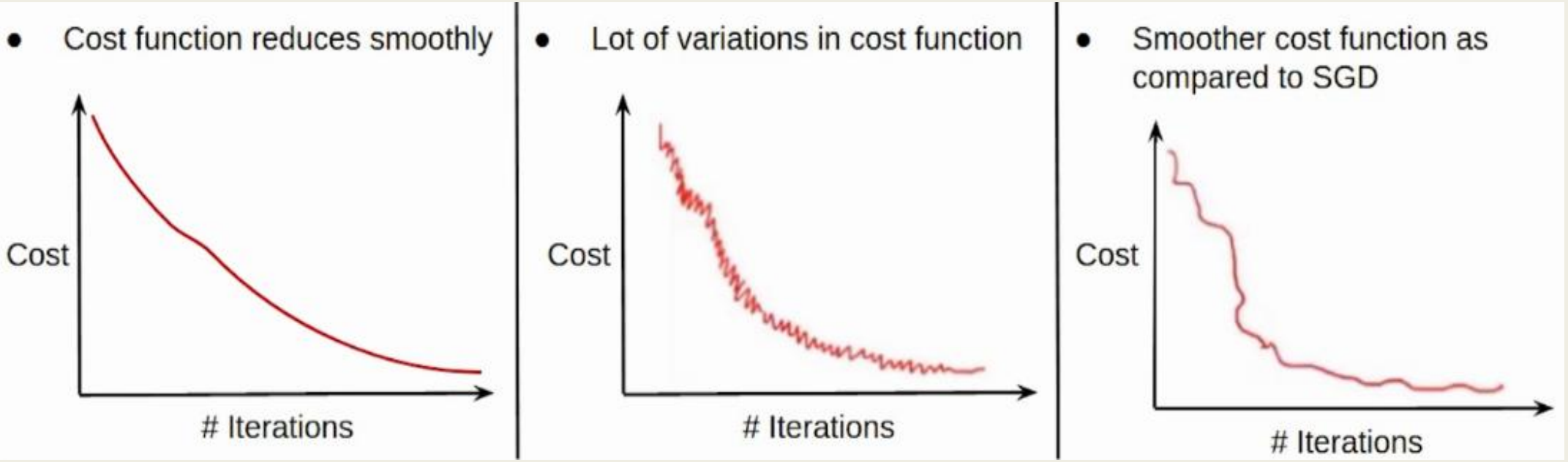






Comparison between different type of Gradient Descent

Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
• Entire dataset for updation	• Single observation for updation	• Subset of data for updation
• Cost function reduces smoothly	• Lot of variations in cost function	• Smoother cost function as compared to SGD
• Computation cost is very high	• Computation time is more	• Computation time is lesser than SGD
		• Computation cost is lesser than Batch Gradient Descent



Feature Scaling (Need)

- In practice, we often encounter different types of variables in the same dataset.
- A significant issue is that the range of the variables may differ a lot.
- Using the original scale may put more weights on the variables with a large range.
-
- In order to deal with this problem, we need to apply the technique of features rescaling to independent variables or features of data in the step of data pre-processing.
- The terms normalisation and standardisation are sometimes used interchangeably, but they usually refer to different things.
- The goal of applying Feature Scaling is to make sure features are on almost the same scale so that each feature is equally important and make it easier to process by most ML algorithms

Feature Scaling (Example)

- This is a dataset that contains a dependent variable (Purchased) and 3 independent variables (Country, Age, and Salary). We can easily notice that the variables are not on the same scale because the range of Age is from 27 to 50, while the range of Salary going from 48 K to 83 K. The range of Salary is much wider than the range of Age. This will cause some issues in our models since a lot of machine learning models such as k-means clustering and nearest neighbour classification are based on the Euclidean Distance.

	Country	Age	Salary	Purchased
1	France	44	72000	No
2	Spain	27	48000	Yes
3	Germany	30	54000	No
4	Spain	38	61000	No
5	Germany	40		Yes
6	France	35	58000	Yes
7	Spain		52000	No
8	France	48	79000	Yes
9	Germany	50	83000	No
10	France	37	67000	Yes

```
dataset['Age'].min()
```

```
27.0
```

```
dataset['Salary'].min()
```

```
48000.0
```

```
dataset['Age'].max()
```

```
50.0
```

```
dataset['Salary'].max()
```

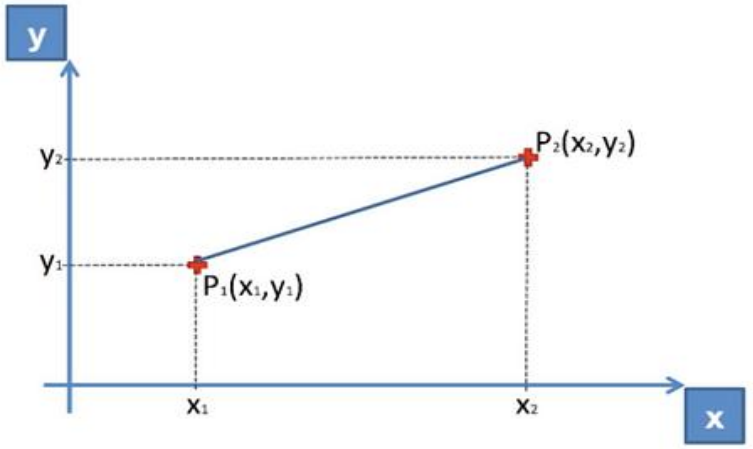
```
83000.0
```

The range of Age: 27 - 50

The range of Salary:48,000 - 83,000

Feature Scaling (Example)

- When we calculate the equation of Euclidean distance, the number of $(x2-x1)^2$ is much bigger than the number of $(y2-y1)^2$ which means the Euclidean distance will be dominated by the salary if we do not apply feature scaling.
- The difference in Age contributes less to the overall difference.
- Therefore, we should use Feature Scaling to bring all values to the same magnitudes and, thus, solve this issue.
- To do this, there are primarily two methods called **Standardisation and Normalisation.**



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Let x be the no. of Salary and y be the no. of Age

Example: x1&y1 are in row 2, x2&y2 are in row 9

(x2-x1)^2= (83000-48000)^2

=1225000000

(y2-y1)^2=(50-27)^2

=529

Standardisation

- The result of standardization (or **Z-score normalization**) is that the features will be rescaled to ensure the **mean and the standard deviation to be 0 and 1, respectively**. The equation is shown below:

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

- This technique is to re-scale features value is useful for the optimization algorithms, such as gradient descent, that are used within machine learning algorithms that weight inputs (e.g., regression and neural networks).
- Rescaling is also used for algorithms that use distance measurements, for example, K-Nearest-Neighbours (KNN).

Normalization

- Another common approach is the so-called **Max-Min Normalization (Min-Max scaling)**.
- This technique is to re-scales features with a distribution value between 0 and 1.
- For every feature, the minimum value of that feature gets transformed into 0, and the maximum value gets transformed into 1. The general equation is shown below:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Standardisation vs Max-Min Normalization

- In contrast to standardisation, we will obtain smaller standard deviations through the process of Max-Min Normalisation. Let me illustrate more in this area using the above dataset.

Standardisation

	Age	Salary
0	0.758874	7.494733e-01
1	-1.711504	-1.438178e+00
2	-1.275555	-8.912655e-01
3	-0.113024	-2.532004e-01
4	0.177609	6.632192e-16
5	-0.548973	-5.266569e-01
6	0.000000	-1.073570e+00
7	1.340140	1.387538e+00
8	1.630773	1.752147e+00
9	-0.258340	2.937125e-01

Max-Min Normalization

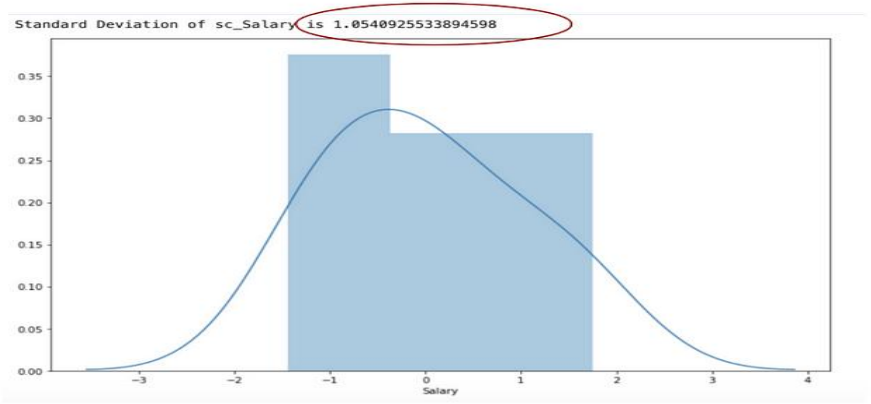
	Age	Salary
0	0.739130	0.685714
1	0.000000	0.000000
2	0.130435	0.171429
3	0.478261	0.371429
4	0.565217	0.450794
5	0.347826	0.285714
6	0.512077	0.114286
7	0.913043	0.885714
8	1.000000	1.000000
9	0.434783	0.542857

Standardisation vs Max-Min Normalization

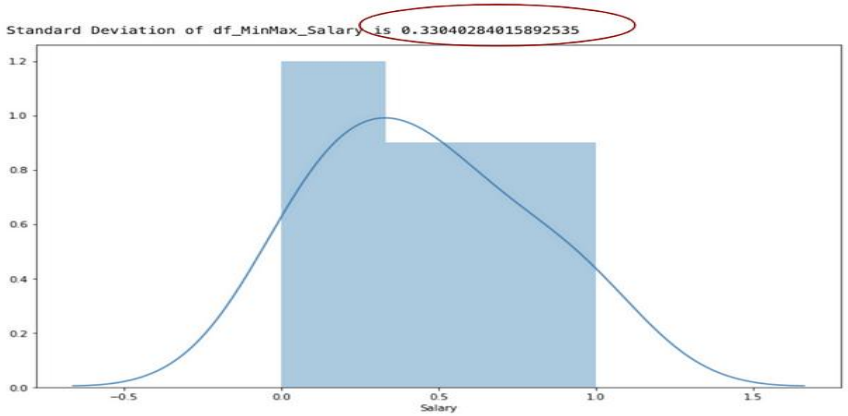
Column: Salary

Standard Deviation (Salary):
Max-Min Normalization (0.33) < Standardisation (1.05)

Standardisation



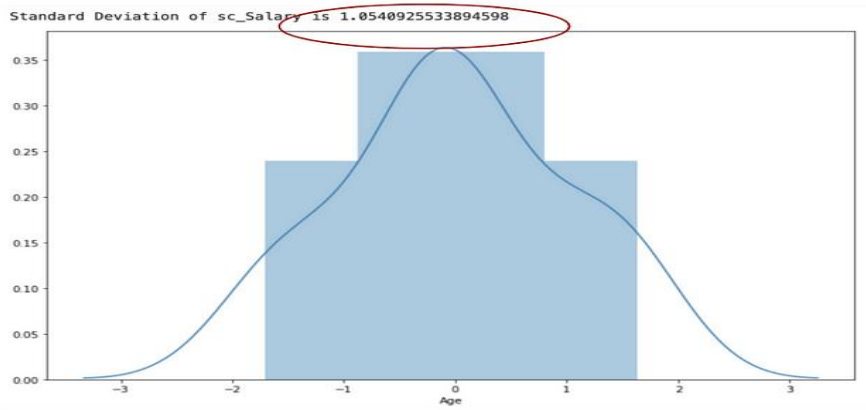
Max-Min Normalisation



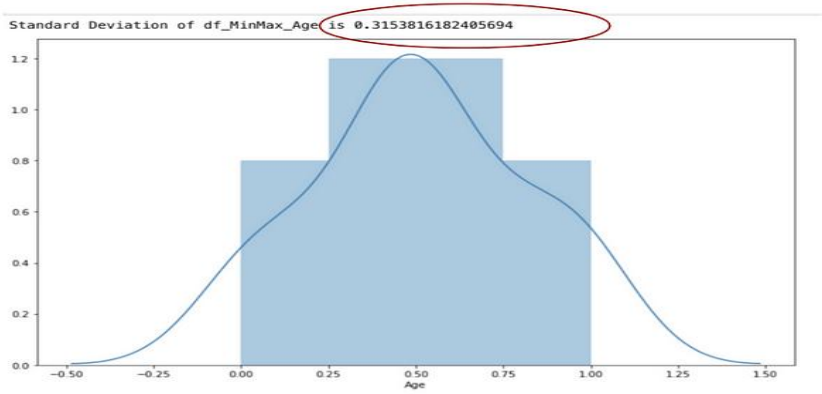
Column:Age

Standard Deviation (Age):
Max-Min Normalization (0.315) < Standardisation (1.05)

Standardisation

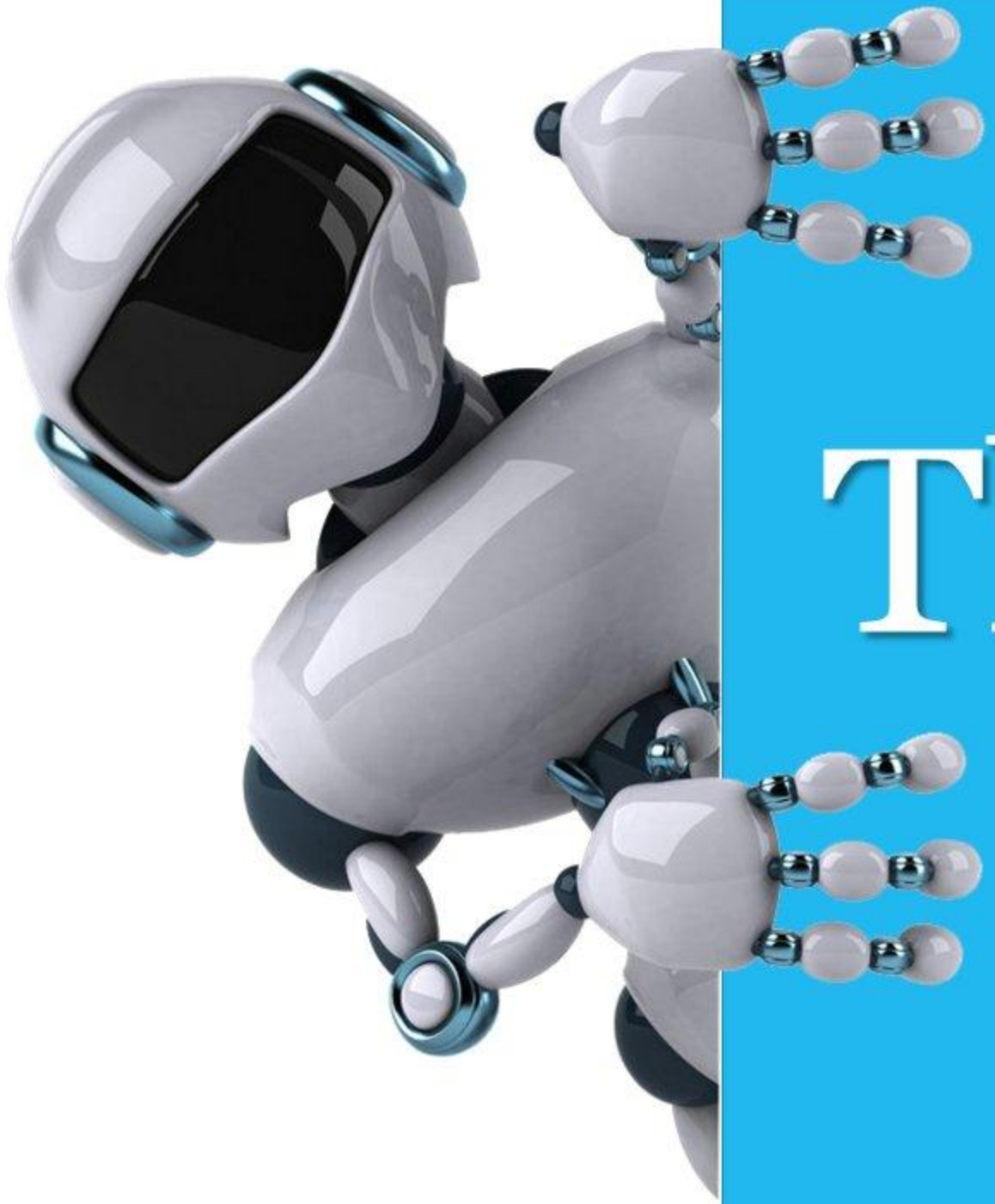


Max-Min Normalisation



Standardisation vs Max-Min Normalization

- Max-Min Normalisation typically allows us to transform the data with varying scales so that no specific dimension will dominate the statistics, and it does not require making a very strong assumption about the distribution of the data, such as k-nearest neighbours and artificial neural networks.
- However, Normalisation does not treat outliners very well.
- On the contrary, standardisation allows users to better handle the outliers and facilitate convergence for some computational algorithms like gradient descent.
- Therefore, we usually prefer standardisation over Min-Max Normalisation.



Thank you