Input Layer

- This layer accepts input features.
- It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.

Hidden Layer

- Nodes of this layer are not exposed to the outer world, they are the part of the abstraction provided by any neural network.
- Hidden layer performs all sort of computation on the features entered through the input layer and transfer the result to the output layer.

Output Layer

- This layer returns the final output computed by the network to the application.

Activation Function

- Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

- The purpose of the activation function is to *introduce non-linearity* into the output of a neuron.

Activation Function

- Neural network has neurons that work in correspondence of *weight, bias* and their respective activation function.
- In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output.
- This process is known as *back-propagation*.
- Activation functions make the back-propagation possible

Activation Function

- **Sigmoid Function :**
- Usually used in output layer of a binary classification, where result is either 0 or 1
- As value for sigmoid function lies between 0 and 1 only
- Result can be predicted easily to be 1 if value is greater than 0.5 and 0 otherwise.

Activation Function

- **tanh() Function :**
- Hyperbolic tangent function (also known as tanh)
- The hyperbolic tangent function outputs in the range (-1, 1), thus mapping strongly negative inputs to negative values.

Activation Function

- **RELU (***Rectified linear unit***)**

- It is the most widely used activation function.

- ReLu is less computationally expensive because it involves simpler mathematical operations.

Gradient Descent

- Gradient Descent is used while training a machine learning model.
- A gradient measures how much the output of a function changes if you change the inputs a little bit.
- It simply measures the change in all weights with regard to the change in error.

Types of Gradient descents:

- **Batch Gradient Descent:** Parameters are updated after computing the gradient of error with respect to the entire training set

- **Stochastic Gradient Descent:** Parameters are updated after computing the gradient of error with respect to a single training example

- **Mini-Batch Gradient Descent:** Parameters are updated after computing the gradient of error with respect to a subset of the training set

Optimization techniques for Gradient Descent

- **Momentum method**: This method is used to accelerate the gradient descent algorithm by taking into consideration the exponentially weighted average of the gradients.

- **new weight ←(old weight)-(learning rate)(gradient)**
- **new weight ←(old weight)-(learning rate)(gradient) + past gradient**
- **(accumulator)←(old accumulator)(momentum)+gradient**
- **momentum → weighted average of past gradients**
- **new weight ←(old weight)-(learning rate)(accumulator)**

..

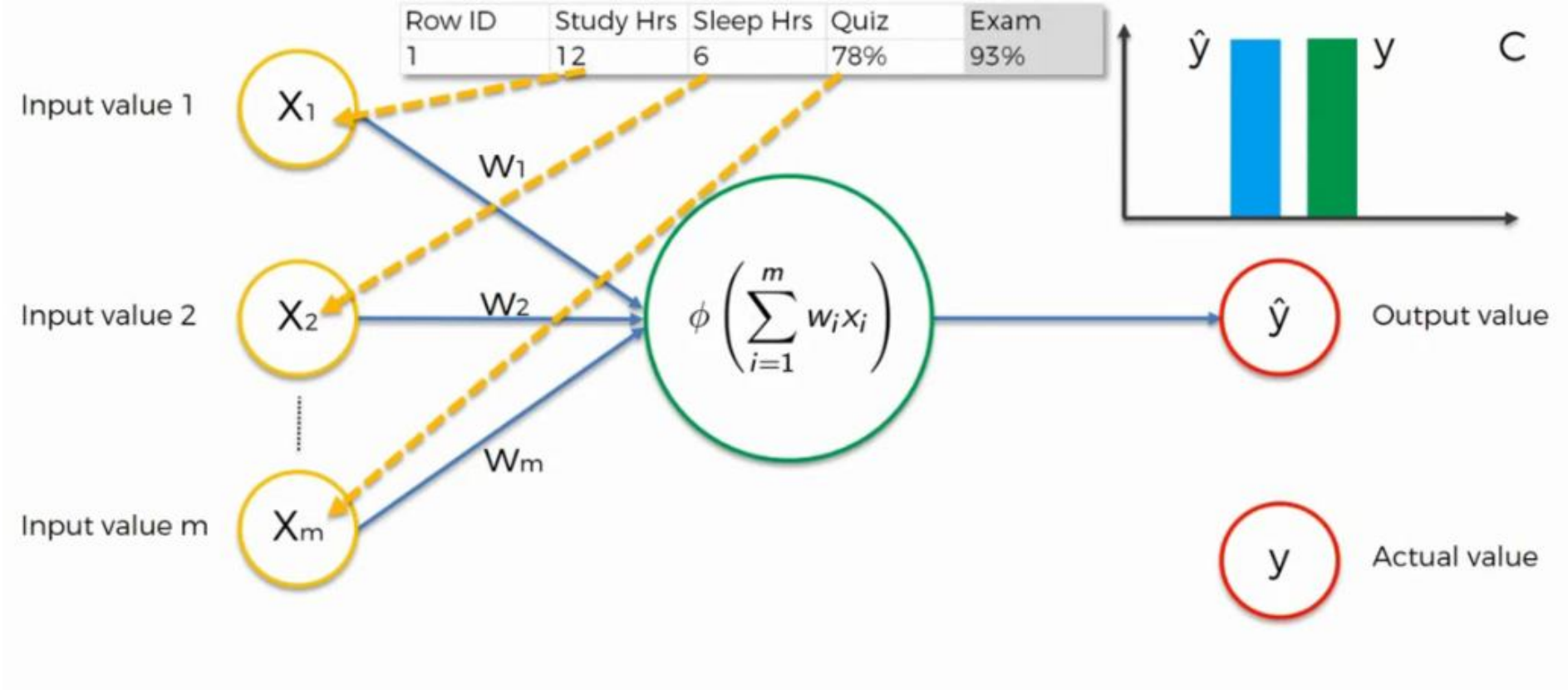
- **RMSprop**: Root Mean Square Propagation
- RMSprop is another adaptive method which retains the learning rate for each parameter but uses a moving average over the gradients to make optimization more suited for more non-convex optimization

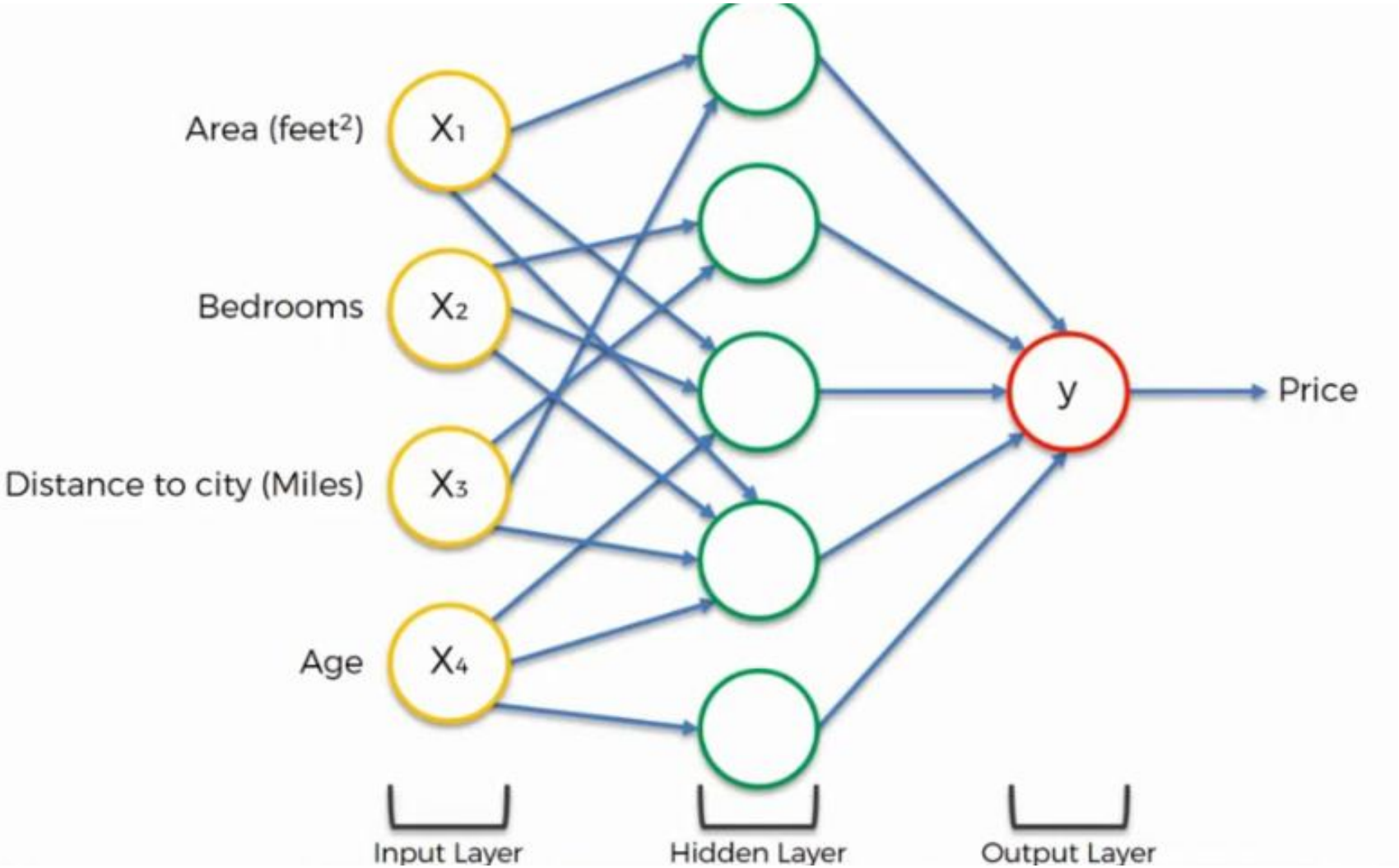
- **Adam optimizer**: Another algorithm that uses adaptive method. Adam stands for Adaptive momentum estimation, it tends to combine the best part of RMSprop & momentum optimizer

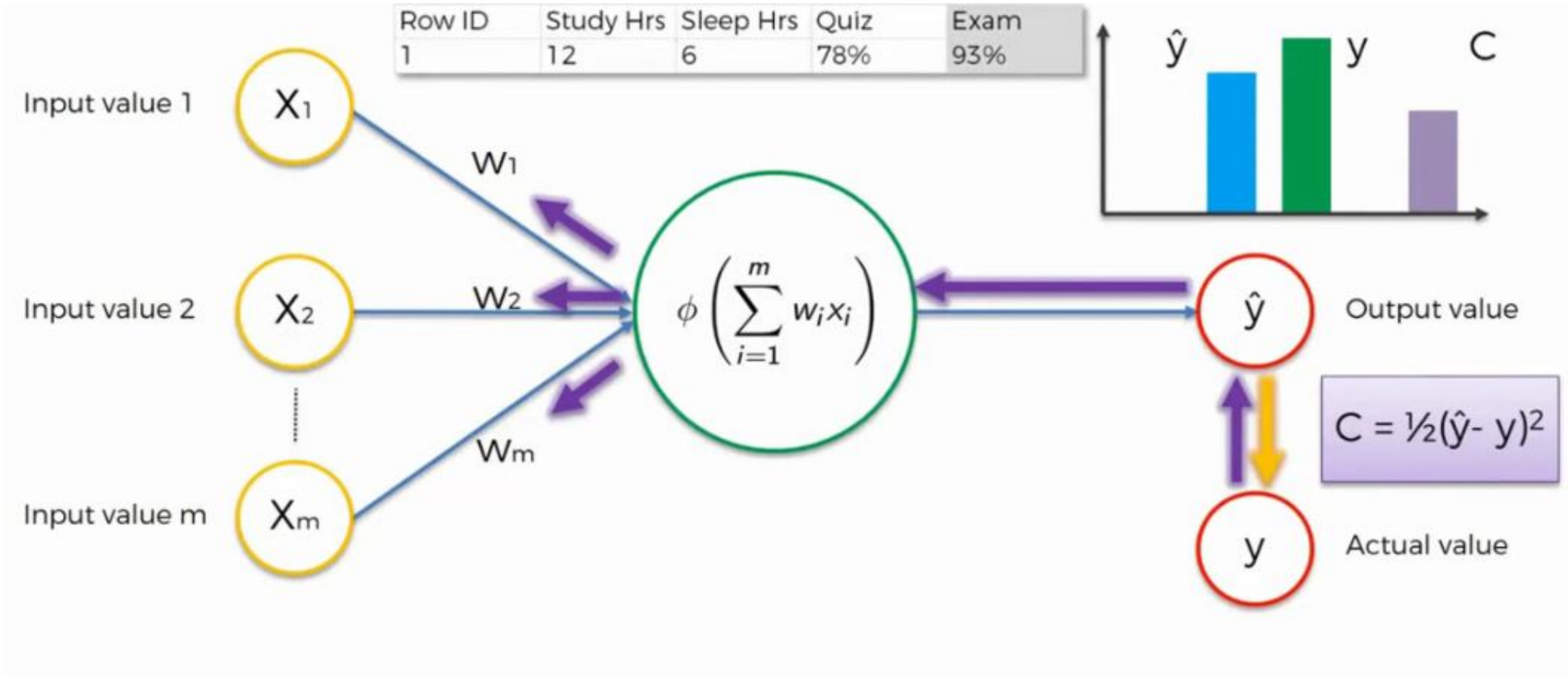
Loss Functions

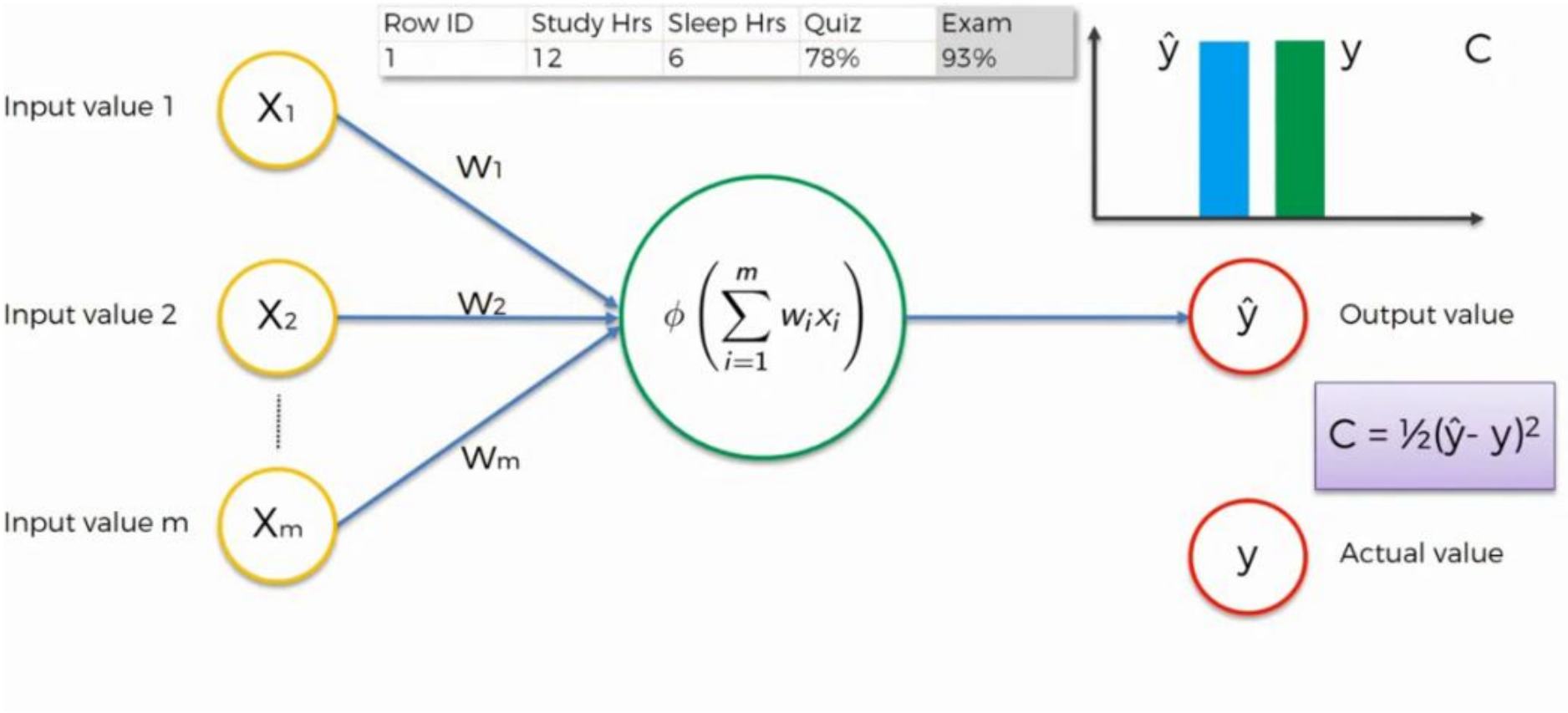
- **Regression Loss Functions**
- **Mean Squared Error Loss**
  - Mean squared error is calculated as the average of the squared differences between the predicted and actual values.
  - ‘*mean_squared_error*‘
- **Mean Absolute Error Loss**
  - ‘*mean_absolute_error*‘

- **Binary Classification Loss Functions**
- **Binary Cross-Entropy Loss**
  - It is intended for use with binary classification where the target values are in the set {0, 1}
  - '*binary_crossentropy*'
- **Hinge Loss**
  - It is intended for use with binary classification where the target values are in the set {-1, 1}.
  - 'hinge'


Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value

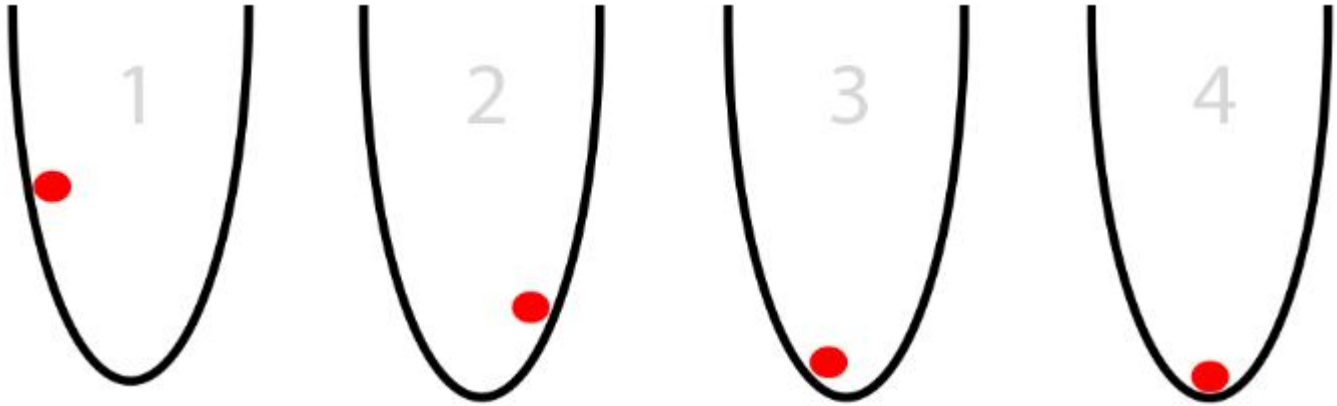

Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Price
Input Layer
Hidden Layer
Output Layer


Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value


Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value

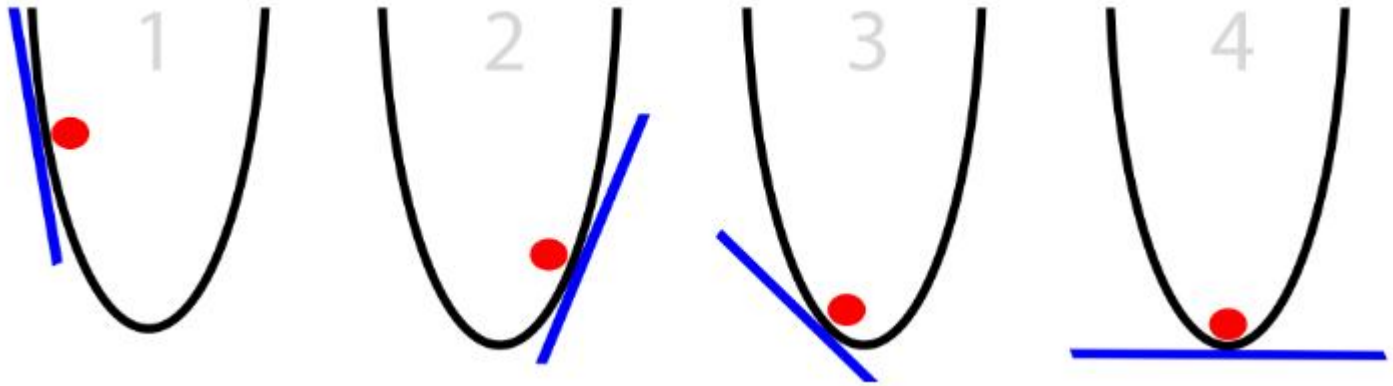
Gradient Descent

- Imagine a red ball inside of a rounded bucket like in the picture below. Imagine further that the red ball is trying to find the bottom of the bucket. This is **optimization**. In our case, the ball is optimizing it's position (from left to right) to find the lowest point in the bucket. The ball is optimizing it's position (from left to right) to find the lowest point in the bucket



So, what information does the ball use to adjust its position to find the lowest point? The only information it has is the **slope** of the side of the bucket at its current position

This is a sub-field of optimization called **gradient optimization**.



Oversimplified Gradient Descent:

- Calculate slope at current position
- If slope is negative, move right
- If slope is positive, move left
- (Repeat until slope == 0)

NLP

- TF-IDF
- TF => Trem Frequency
- IDF => Inverse document frequency

TF	=	Term Frequency
IDF	=	Inverse Document Frequency
TF-TDF	=	TF * IDF

TF

$$\frac{(Number\ of\ occurrences\ of\ a\ word\ in\ a\ document)}{(Number\ of\ words\ in\ that\ document)}$$

"to be or not to be"

$$to = \frac{1+1}{6}$$

to = 0.33

be = 0.33

or = 0.16

"It is going to rain today"

"Today I am not going outside"

"I am going to watch the season premiere"

Sentence 1	Sentence 2	Sentence 3
it	today	i
is	i	am
going	am	going
to	not	to
rain	going	watch
today	outside	the
		season
		premiere

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

IDF

Formula

$$log\left(\frac{(Number\ of\ documents)}{(Number\ of\ documents\ containing\ word)}\right)$$

$$log(\frac{(Number\ of\ documents)}{(Number\ of\ documents\ containing\ word)})$$

"to be or not to be"
"i have to be"
"you got to be"

to = $\log\left(\frac{3}{3}\right)$

to = 0

be = $\log(\frac{3}{3})$

be = 0

have = $\log(\frac{3}{1})$

Words	IDF Value
going	log(3/3)
to	log(3/2)
today	log(3/2)
i	log(3/2)
am	log(3/2)
It	log(3/1)
is	log(3/1)
rain	log(3/1)

"it is going to rain today"

"today i am not going outside"

"i am going to watch the season premiere"

Words	IDF Value
going	0
to	0.41
today	0.41
i	0.41
am	0.41
It	1.09
is	1.09
rain	1.09

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

$$TFIDF(Word) = TF(Document, Word) * IDF(Word)$$

Latent Dirichlet Allocation (LDA) for Topic Modelling

- LDA is a topic model that generates topics based on word frequency from a set of documents
- LDA is particularly useful for finding reasonably accurate mixtures of topics within a given document

How does LDA work

- Create a collection of documents from news articles.
- Each document represents a news article.
- Data cleaning is the next step:
- Tokenizing, Stop words elimination, Stemming

Random Topic Initialization

- We want to find out 3 major topics from the news articles/documents the we have collected

Document

3	2	1	3	1
Cricket	World Cup	2019	Winners	India

Document to Topic count

	Topic 1	Topic 2	Topic 3
Doc i	2	1	2

Random Topic Initialization

	Topic 1	Topic 2	Topic 3
Cricket	1	0	35
World Cup	18	8	1
2019	42	1	0
Winners	0	0	20
India	50	0	1

Count of how many times a word is associated to the topic

Converge Words to Signify the Topic

- **Consider the word - World Cup**
- **We want to reassign the topic for the given word.**
- **Decrementing counts after removing current assignments**

3	2	1	3
Cricket	World Cup	2019	Winn

	Topic 1	Topic 2	Topic
Doc i	2	1	2

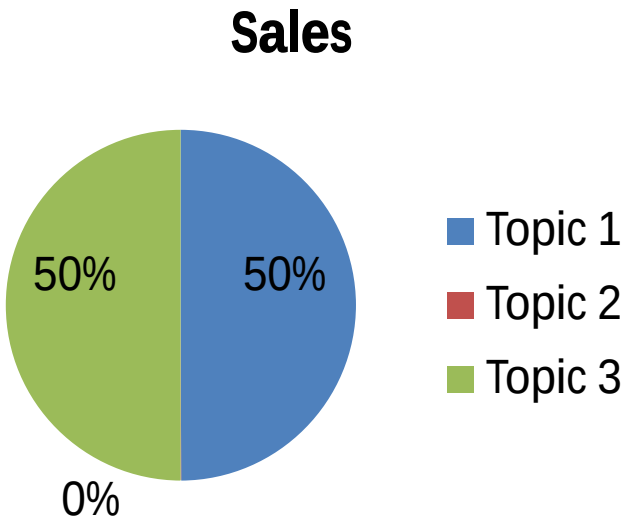
	Topic 1	Topic 2	Topic 3
Cricket	1	0	35
World Cup	18	7	1
2019	42	1	0
Winners	0	0	20
India	50	0	1

After Removing

	Topic 1	Topic 2	Topic 3
Doc i	2	0	2

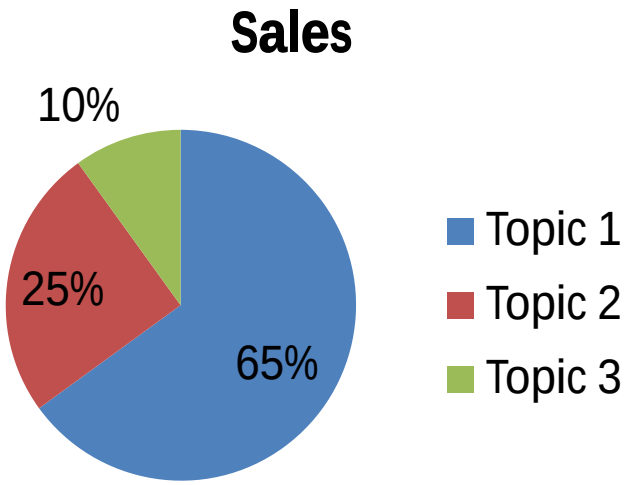
- Reassign the topic based on probability calculation

How much doc likes each topic based on other assignment in doc



	Topic 1	Topic 2	Topic 3
Doc i	2	0	2

- Reassign the topic based on probability calculation



How much topic likes the word “World cup” based on assignment in other doc

	Topic 1	Topic 2	Topic 3
World Cup	18	7	1

How much topic likes the word “World cup” based on assignment in other doc

This will happen for each word in multiple passes of LDA

	Topic 1	Topic 2	Topic 3
World Cup	18	7	1

3	1	1	3	1
Cricket	World Cup	2019	Winners	India

	Topic 1	Topic 2	Topic 3
Doc i	2	0	2

Topic 1 is assigned to world cup , where initially it was assigned Topic 2

How much doc likes each topic based on other assignment in doc