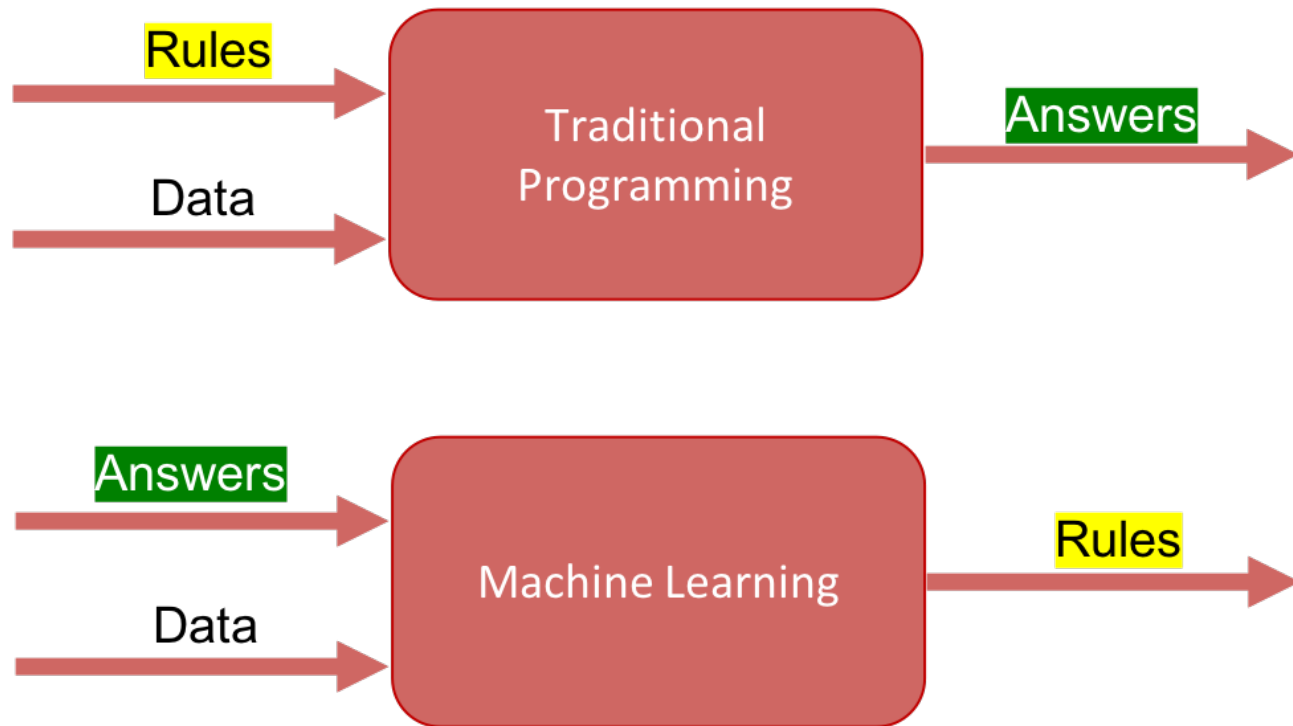


Machine Learning

Unit 4

What is Machine Learning?



Machine learning is an application of AI that enables systems to learn and improve from experience without being explicitly programmed.

What is Machine Learning?

- Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

- IBM

- In 1959, Arthur Samuel (an AI pioneer at IBM), defined the term “Machine Learning” as

“Field of study that gives computers the ability to learn without being explicitly programmed.”

What is Machine Learning? (contd.)

- After that, Tom M. Mitchell in 1997 gave the definition of Machine Learning which is widely quoted all over the globe.
- This is the more formal definition of the algorithms that are studied in the Machine Learning field,

“A computer program is said to learn from experience E with respect to some class of tasks T and a performance measure P if its performance in tasks T , as measured by P , improves with experience E .”

Machine Learning is the subfield of AI and is multidisciplinary

Multidisciplinarity

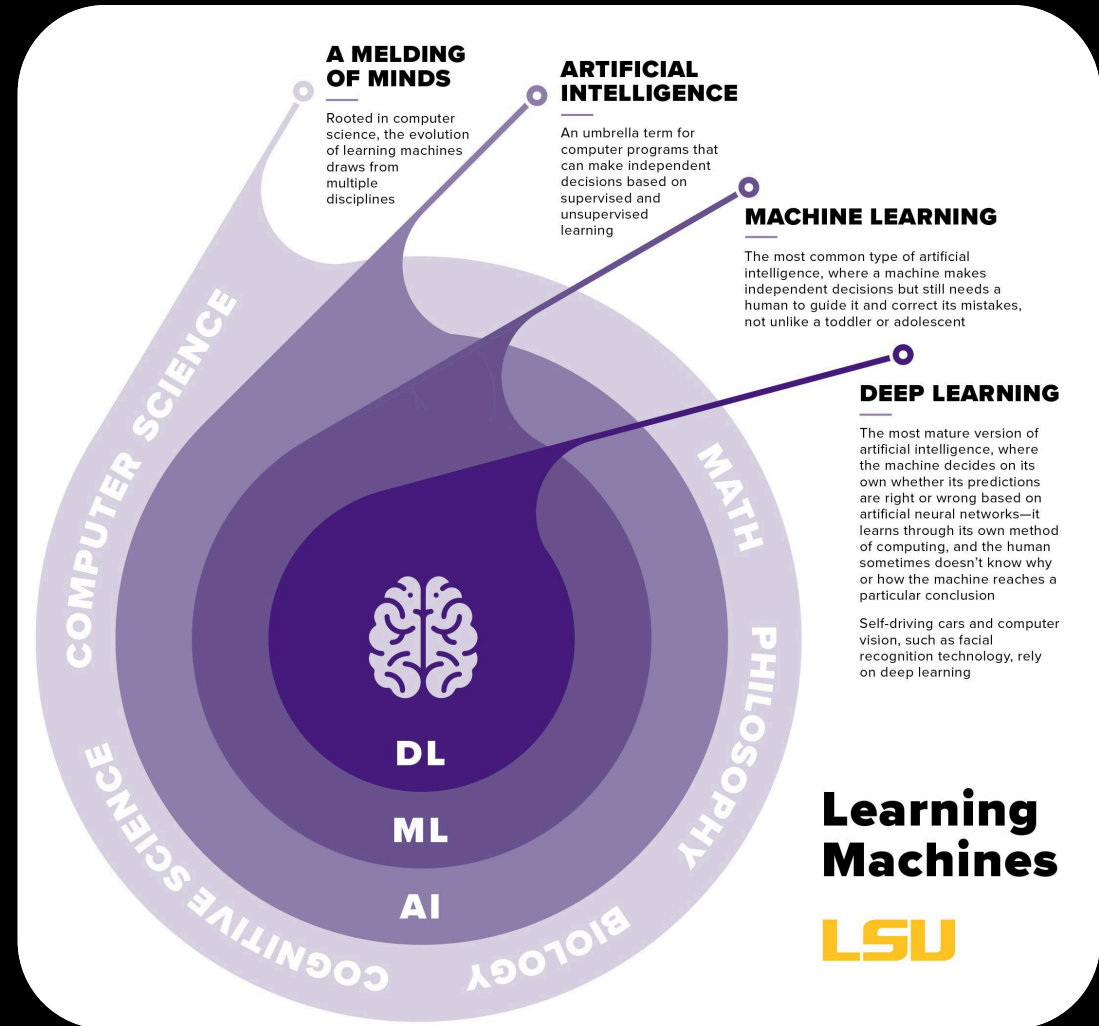
Different academic disciplines working together and drawing on their disciplinary knowledge in parallel, to conduct research on a single problem or theme but without integration.

Interdisciplinarity

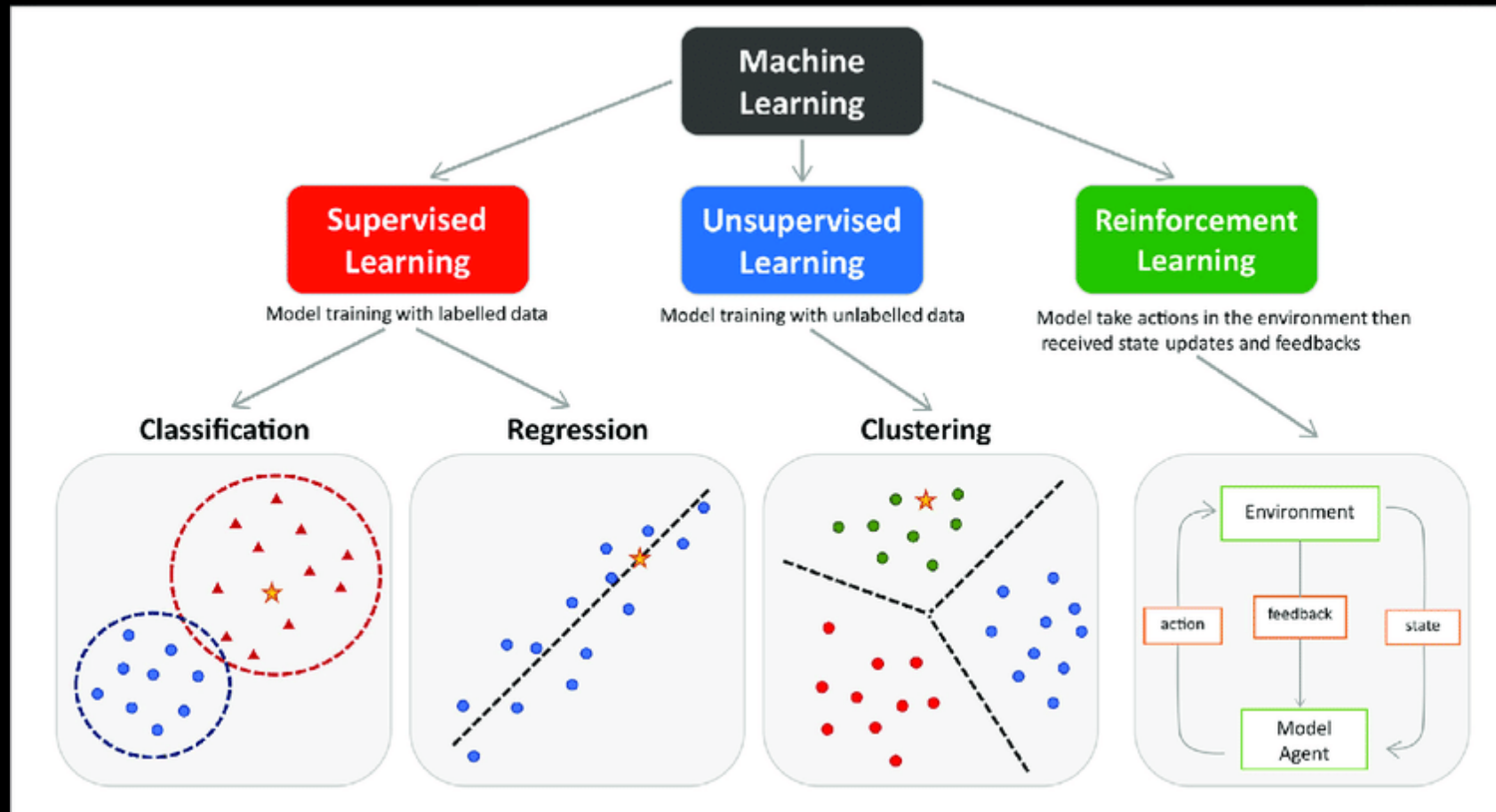
Different academic disciplines working together to integrate disciplinary knowledge and methods, to develop and meet shared research goals achieving a real synthesis of approaches.

Transdisciplinarity

Different academic disciplines working together with non-academic collaborators to integrate knowledge and methods, to develop and meet shared research goals achieving a real synthesis of approaches.



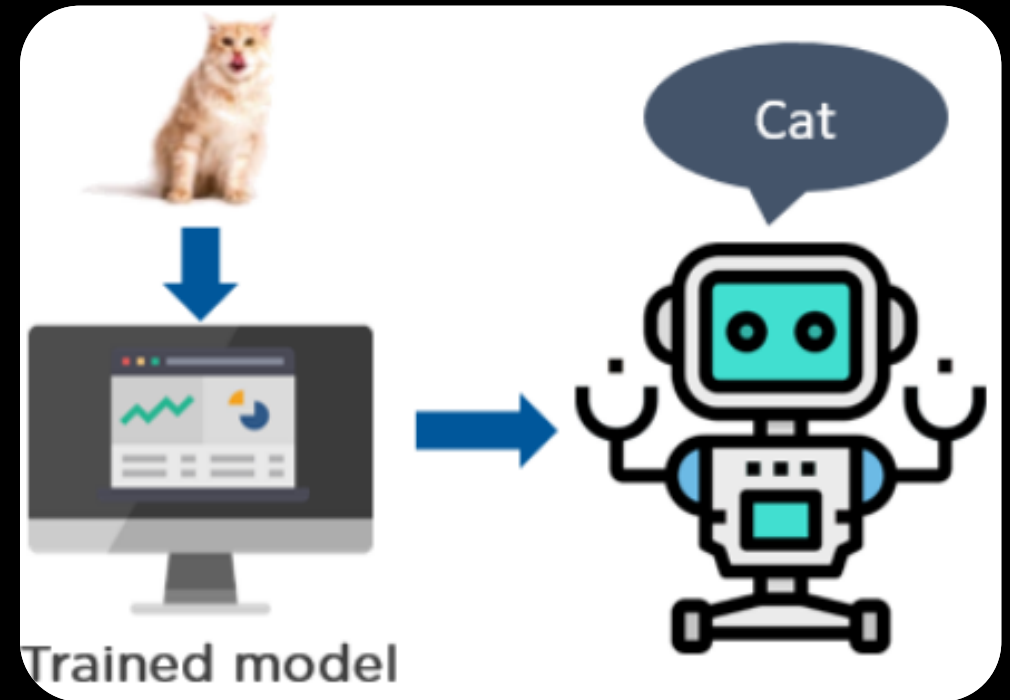
Different Types of Machine Learning



Also, there is something called **Semi Supervised Learning** and **Self Supervised Learning**.

1. Supervised Learning

- A type of learning that uses labelled data (or input with outputs) to train machine learning algorithms.
- The input are provided with corresponding outputs while training ML models.
- Since these algorithm needs external supervision on mapping input and expected output, they are called **supervised**.



1. Supervised Learning (contd.)

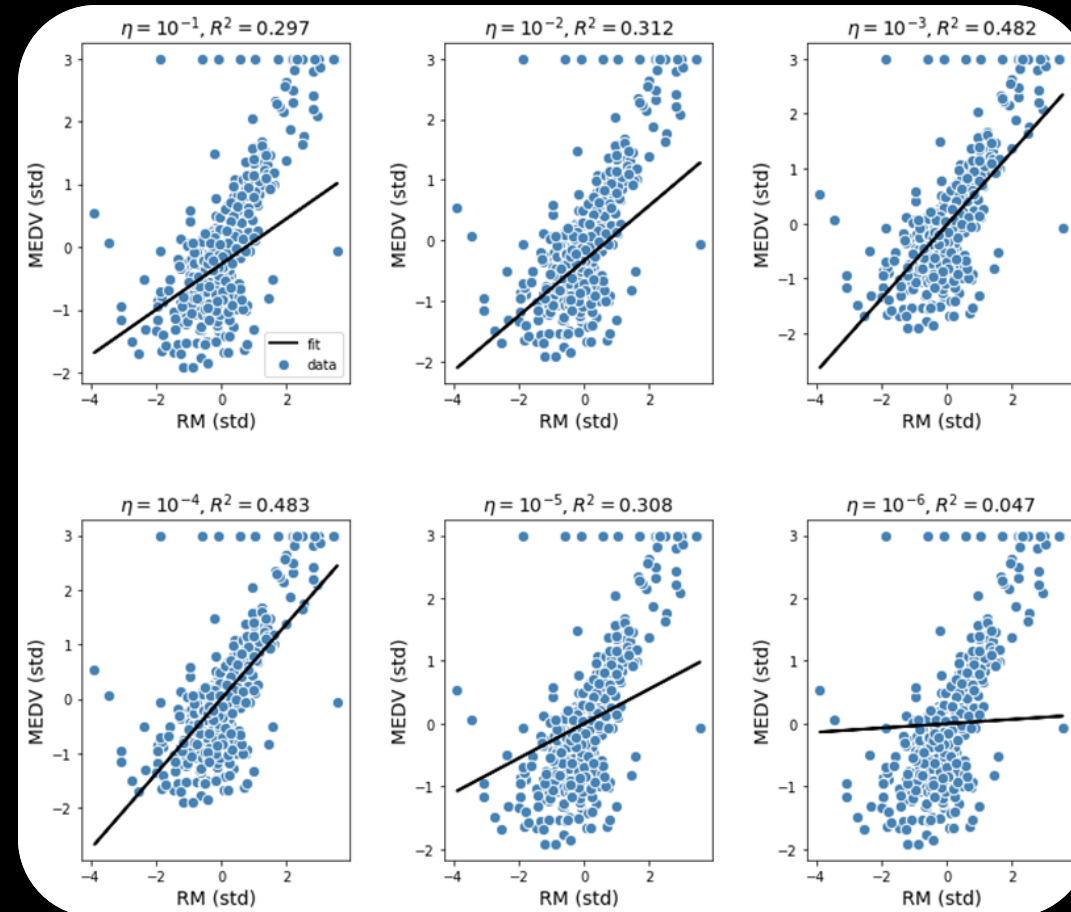
- Even though the data needs to be labeled accurately for this method to work, usually done by experts, supervised learning is extremely powerful when used in the right circumstances.

1. Supervised Learning (contd.)

Working Mechanism

- In supervised learning, the ML algorithm is given a training dataset to work with.
- This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with.
- The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem.

1. Supervised Learning (contd.)



1. Supervised Learning (contd.)

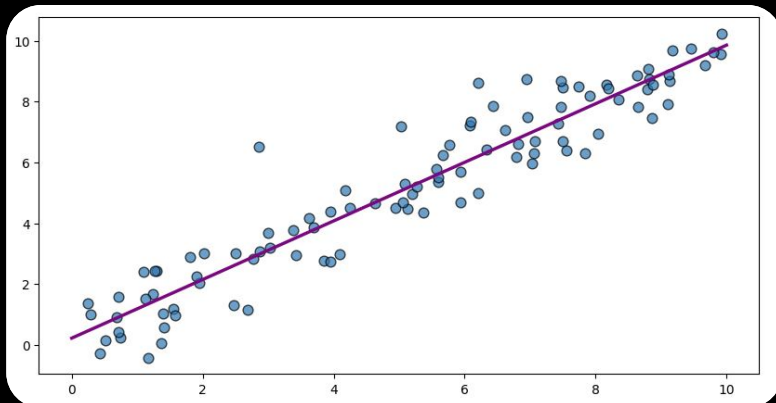
- Working Mechanism

- The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset.
- At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.
- This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset.
- This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

Types of Supervised Learning

Regression

- Learning for prediction of value
 - Has continuous output
- Needs corresponding output with input



Classification

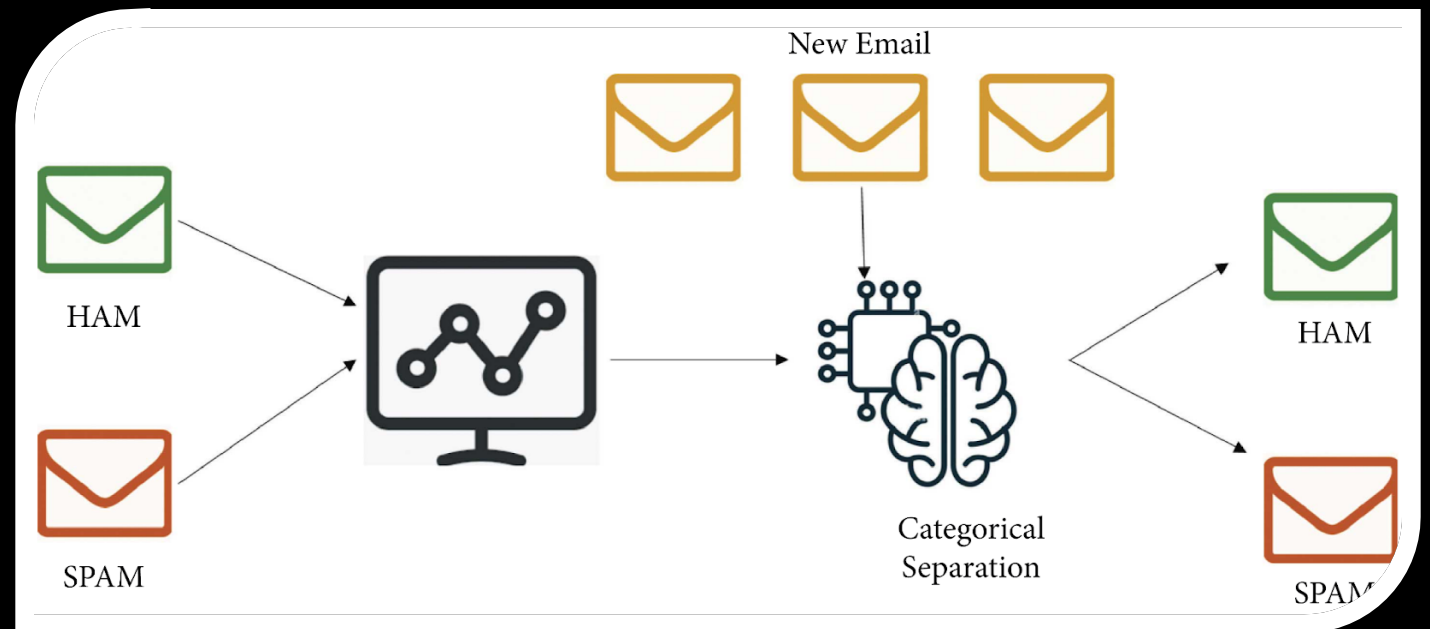
- Learning for classification of objects
- Has discrete output
- Needs Labelled data with input



1. Supervised Learning (contd.)

- Applications

- Price Prediction
- Image classification and segmentation
- Disease identification and medical diagnosis
- Fraud detection
- Spam detection
- Speech recognition
- Sentiment Analysis
- Image Captioning
- Image Generation
- Text Generation etc.



1. Supervised Learning (contd.)

- **Algorithms**

- Linear regression
- Logistic regression
- Naive Bayes
- Linear discriminant analysis
- Decision trees
- K-nearest neighbor algorithm
- Neural networks (Multilayer perceptron)
- Random forest algorithm
- Support Vector Machine etc.

2. Unsupervised Algorithm

- A class of algorithm which has input data but no corresponding output or label.
- The goal for unsupervised learning is to **model/ understand/ manipulate** the underlying structure or distribution of the data in order to learn more about the data.
- There is no supervision from expert through label or output, thus, called unsupervised algorithm.
- Unsupervised machine learning purports to uncover previously unknown patterns in data, but most of the time these patterns are poor approximations of what supervised machine learning can achieve.

Types of Unsupervised Algorithms

- **Clustering** allows you to automatically split the dataset into groups according to similarity. Often, however, cluster analysis overestimates the similarity between groups and doesn't treat data points as individuals. For this reason, cluster analysis is a poor choice for applications like customer segmentation and targeting.
- **Anomaly detection** can automatically discover unusual data points in your dataset. This is useful in pinpointing fraudulent transactions, discovering faulty pieces of hardware, or identifying an outlier caused by a human error during data entry.
- **Association mining** identifies sets of items that frequently occur together in your dataset. Retailers often use it for basket analysis, because it allows analysts to discover goods often purchased at the same time and develop more effective marketing and merchandising strategies.
- **Latent variable models** are commonly used for data preprocessing, such as reducing the number of features in a dataset (dimensionality reduction) or decomposing the dataset into multiple components.

2. Unsupervised Algorithms (contd.)

- Applications

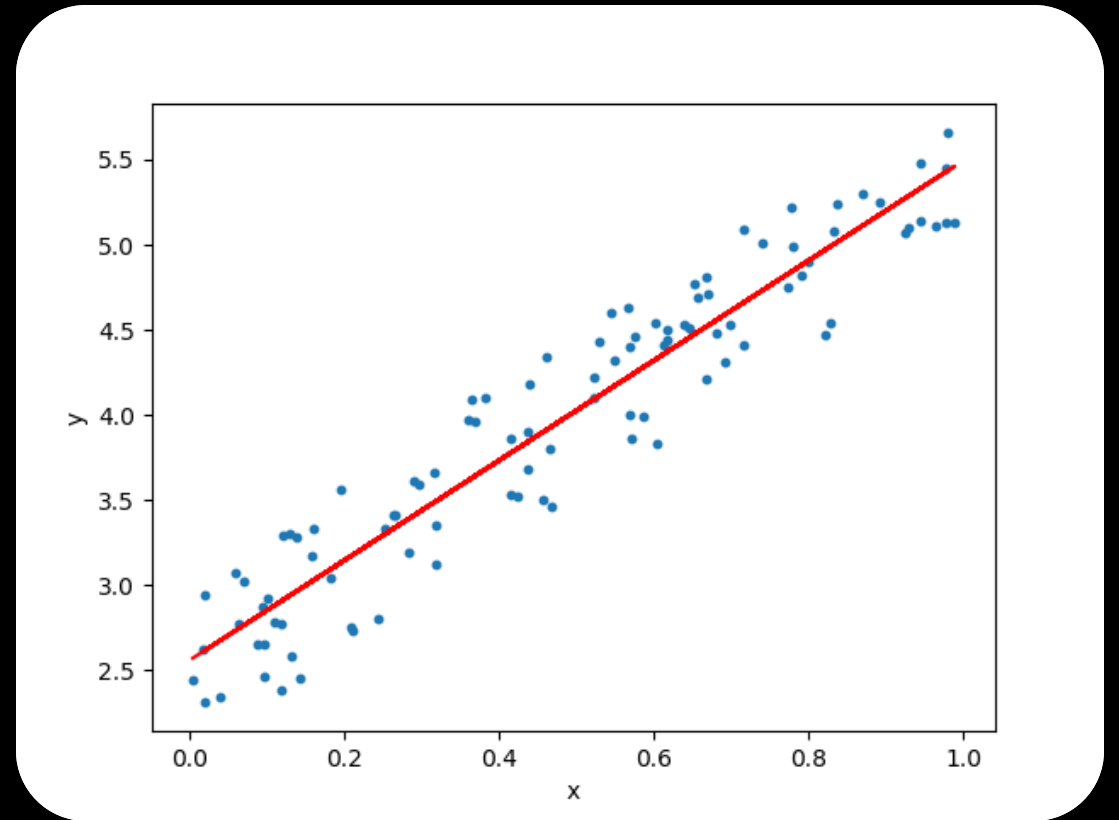
- Market Basket Analysis
- Semantic Clustering
- Delivery Store Optimization
- Identifying Accident Prone Areas
- Customer Segmentation
- Dimensionality Reduction
- Image Segmentation
- Audience segmentation
- Market research
- Recommendation System

2. Unsupervised Algorithms (contd.)

- Algorithms
 - K-means clustering
 - Hierarchical clustering
 - Gaussian Mixture Models
 - Apriori algorithms
 - FP Growth
 - Principal Component Analysis
 - Singular Value Decomposition
 - Autoencoders
 - Local Outlier Factor
 - Expected-Maximization

Linear Regression

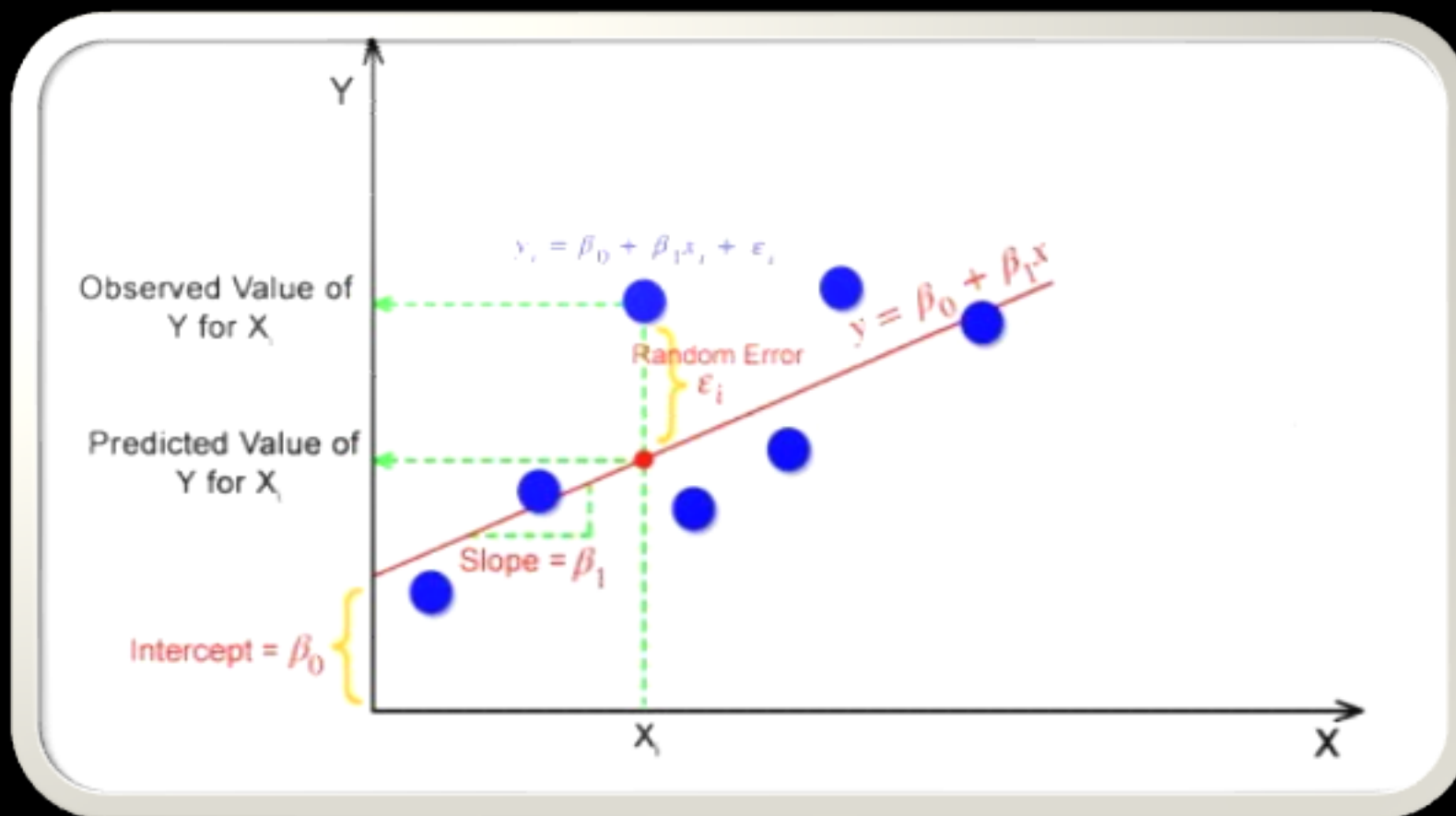
- Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data.
- One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.
- For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.



Linear Regression (contd.)

- Linear Regression is a simple yet powerful and mostly used algorithm in data science.
- Linear regression shows the linear relationship between the independent(predictor) variable i.e. X-axis and the dependent(output) variable i.e. Y-axis, called linear regression.
- If there is a single input variable X(independent variable), such linear regression is called simple linear regression.
- The above graph presents the linear relationship between the output(y) variable and predictor(X) variables.
- The red line is referred to as the best fit straight line.

Linear Regression (contd.)



Linear Regression (contd.)

- The goal of the linear regression algorithm is to get the best values for β_0 and β_1 to find the best fit line.
- In simple terms, the best fit line is a line that fits the given scatter plot in the best way.
- The best fit line is a line that has the least error which means the error between predicted values and actual values should be minimum.

Linear Regression (contd.)

Random Error(Residuals)

- In regression, the difference between the observed value of the dependent variable(y_i) and the predicted value is called the **residuals**.

$$\varepsilon = y_{predicted} - y_{actual}$$

- Then we can define the RSS (Residual Sum of Squares) as

$$RSS = e_1^2 + e_2^2 + \dots \dots \dots + e_m^2$$

Linear Regression (contd.)

- In Linear Regression, generally **Mean Squared Error (MSE)** cost function is used, which is the average of squared error that occurred between the $y_{predicted}$ and y_i .

$$MSE = 1/m \sum_{i=1}^m (y_i - (\beta_0 + B_1 x))^2$$

- We use different optimization techniques to minimize the error or RSS. Some of them are ordinary least square method, gradient descent etc.

X Ordinary Least Square Method

- This is a statistical Method for performing linear regression.
- Ordinary least square error is given by:

$$OLSE = \frac{1}{2} \sum_{i=1}^m \left(h_{\beta}(x^i) - y(i) \right)^2$$

- Using Least Square method, we can write

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^m (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^m (x_i - \bar{x})^2}$$

$$\widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x}$$

X Ordinary Least Square Method

Derivation of OLS Estimator

In class we set up the minimization problem that is the starting point for deriving the formula for the OLS intercept and slope coefficient. That problem was,

$$\min_{\beta_0, \beta_1} \sum_{i=1}^N (y_i - \beta_0 - \beta_1 x_i)^2. \quad (1)$$

As we learned in calculus, a univariate optimization problem is solved by taking the derivative and setting equal to 0. Similarly, this minimization problem above is solved by setting the partial derivatives equal to 0. That is, take the derivative of (1) with respect to β_0 and set it equal to 0. We then do the same thing for β_1 . This gives us,

$$\frac{\partial W}{\partial \beta_0} = \sum_{i=1}^N -2(y_i - \beta_0 - \beta_1 x_i) = 0 \quad (2)$$

and,

$$\frac{\partial W}{\partial \beta_1} = \sum_{i=1}^N -2x_i(y_i - \beta_0 - \beta_1 x_i) = 0 \quad (3)$$

Note that I have used W to denote $\sum_{i=1}^N (y_i - \beta_0 - \beta_1 x_i)^2$. Now our task is to solve (2) and (3) using some algebraic tricks and some properties of summations. Let's start with the first order condition for β_0 (this is Equation (2)). We can immediately get rid of the -2 and write $\sum_{i=1}^N (y_i - \beta_0 - \beta_1 x_i) = 0$. Now let's rearrange this expression and make use of the algebraic fact that $\sum_{i=1}^N 1 = N$. This leaves us with,

$$N\beta_0 = N\bar{y} - N\beta_1\bar{x}. \quad (4)$$

We simply divide everything by N and, assuming, we have the formula that Professor Saloner gave in lecture! That is,

$$\beta_0 = \bar{y} - \beta_1\bar{x}. \quad (5)$$

Now let's consider solving for β_1 . This one is a bit more tricky. We can first get rid of the -2 and rearrange Equation (3) to get $\sum_{i=1}^N x_i(y_i - \beta_0 - \beta_1 x_i) = 0$. Now let's substitute our result for β_0 into this expression and this gives us,

$$\sum_{i=1}^N x_i(y_i - (\bar{y} - \beta_1\bar{x}) - \beta_1 x_i) = 0 \quad (6)$$

Note that the summation is applying to everything in the above equation. We can distribute the sum to each term to get,

$$\sum_{i=1}^N x_i y_i - \bar{y} \sum_{i=1}^N x_i + \beta_1 \bar{x} \sum_{i=1}^N x_i - \beta_1 \sum_{i=1}^N x_i^2 = 0. \quad (7)$$

We have of course used the property that you can always pull a constant term out in front of a summation. Let's again use the property that $\sum_{i=1}^N 1 = N$ (and of course this also means that $\sum_{i=1}^N x_i = N\bar{x}$). We apply these facts to Equation (7) and solve for β_1 . This gives,

$$\beta_1 = \frac{\sum_{i=1}^N x_i y_i - N\bar{y}\bar{x}}{\sum_{i=1}^N x_i^2 - N\bar{x}^2}. \quad (8)$$

Doesn't quite look like the formula from class, right? Well, let us just use a couple more tricks. You can either look up or derive for yourself that $\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^N x_i y_i - N\bar{y}\bar{x}$. You can also easily derive that $\sum_{i=1}^N (x_i - \bar{x})^2 = \sum_{i=1}^N x_i^2 - N\bar{x}^2$. These two can be derived very easily using algebra. Now we substitute these two properties into (8) and we have something that looks very, very familiar:

$$\beta_1 = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^N (x_i - \bar{x})^2}. \quad (9)$$

$$\beta^2 = \frac{\sum_{i=1}^N (y_i - \bar{y})^2}{\sum_{i=1}^N (x_i - \bar{x})^2 + \sum_{i=1}^N (y_i - \bar{y})^2} \quad (10)$$

Equation (10) can be derived from (9) and the fact that $\sum_{i=1}^N (y_i - \bar{y})^2 = \sum_{i=1}^N y_i^2 - N\bar{y}^2$.

Equation (11) can be derived from (9) and the fact that $\sum_{i=1}^N (x_i - \bar{x})^2 = \sum_{i=1}^N x_i^2 - N\bar{x}^2$. Equation (12) can be derived from (9) and the fact that $\sum_{i=1}^N (y_i - \bar{y})^2 = \sum_{i=1}^N y_i^2 - N\bar{y}^2$.

X Gradient Descent

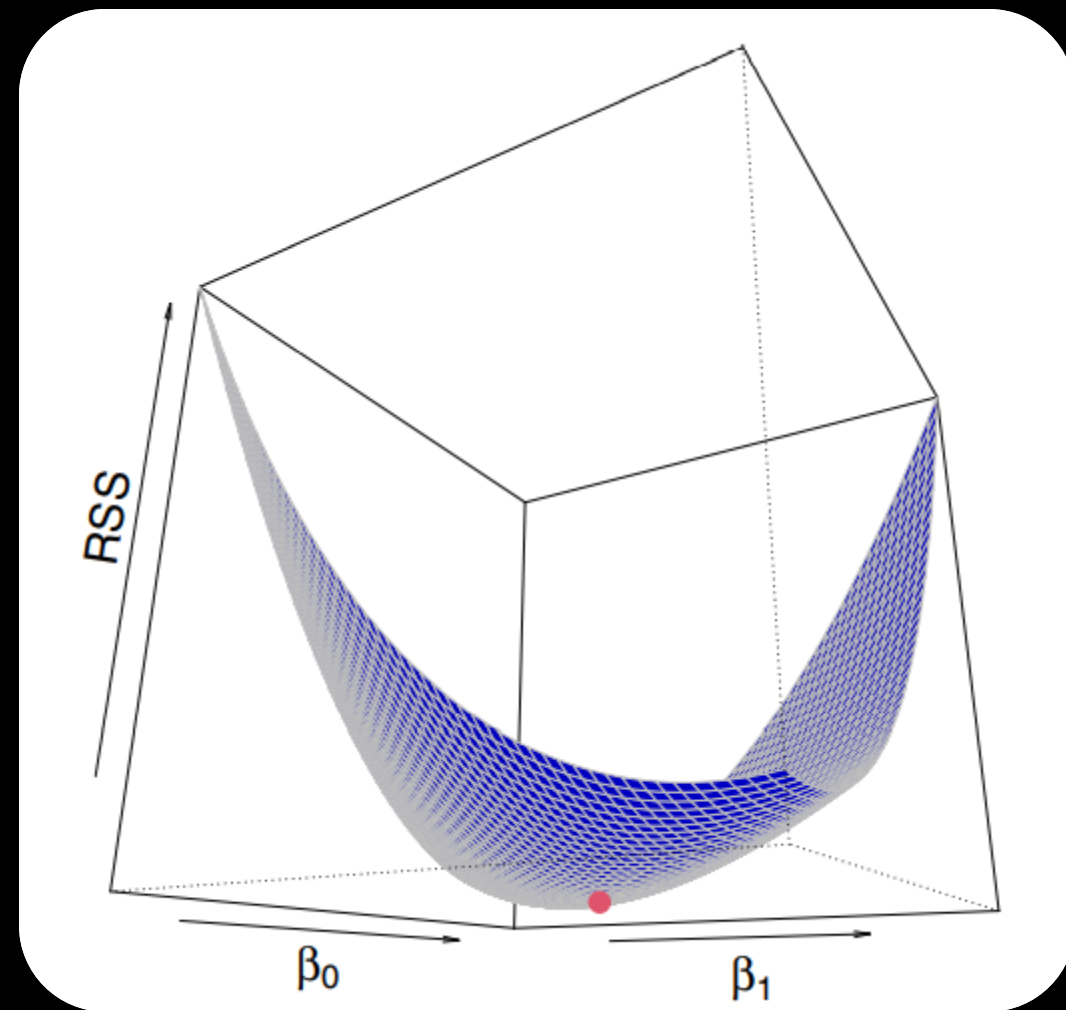
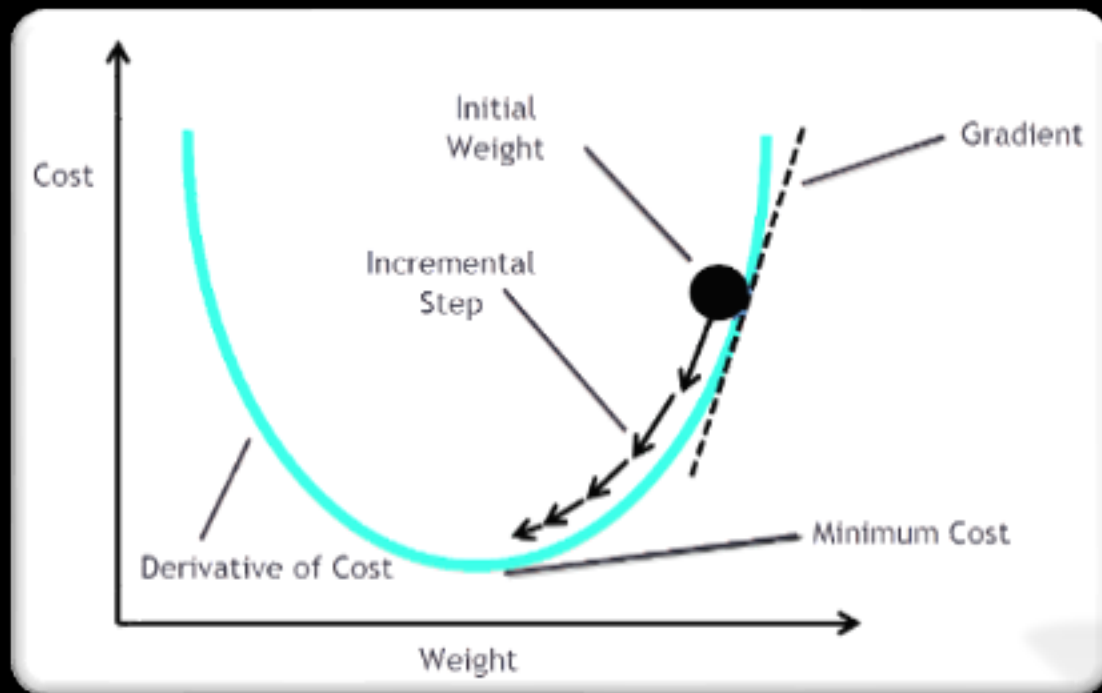
- We use linear regression to predict the dependent continuous variable Y on the basis of independent X. It assumes the relationship between independent and dependent variables to be linear as such:

$$y_i = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

- Gradient Descent is one of the optimization algorithms that optimize the cost function(objective function) to reach the optimal minimal solution.
- To find the optimum solution we need to reduce the cost function(MSE) for all data points.
- This is done by updating the values of β_0 and β_1 iteratively until we get an optimal solution.

$$J(\beta) = \frac{1}{2m} \sum_{i=1}^m (h_{\beta}(x^{(i)}) - y^{(i)})^2$$

X Gradient Descent



X Gradient Descent (contd.)

- We used the update rule as:

$$\beta_j = \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta)$$

And we get the derivate as:

$$\begin{aligned} \frac{\partial}{\partial \beta_j} J(\beta) &= \frac{\partial}{\partial \beta_j} (h_{\beta}(x) - y)^2 \\ &= \frac{1}{2} \frac{\partial}{\partial \beta_j} (h_{\beta}(x) - y)^2 \\ &= \frac{1}{2} \frac{\partial (h_{\beta}(x) - y)^2}{\partial (h_{\beta}(x) - y)} \cdot \frac{\partial}{\partial \beta_j} (h_{\beta}(x) - y) \\ &= \frac{1}{2} 2 (h_{\beta}(x) - y) \cdot \frac{\partial}{\partial \beta_j} \left(\sum_{i=0}^n \beta_i x_i - y \right) \\ &= (h_{\beta}(x) - y) \cdot x_j \end{aligned}$$

X Gradient Descent (contd.)

- Now, we can rewrite the update rule as:

$$\beta_j = \beta_j - \alpha(h_{\beta}(x) - y) \cdot x_j$$

$$\beta_j = \beta_j + \alpha(y - h_{\beta}(x)) \cdot x_j$$

Which is the required form.

Neural Network

Notations

$x_1 \dots x_n$ - n input features

\mathbf{x} – **Input Vector**

m observations

w – weight (scalar)

\mathbf{w} – **weight (vector)**

b – bias

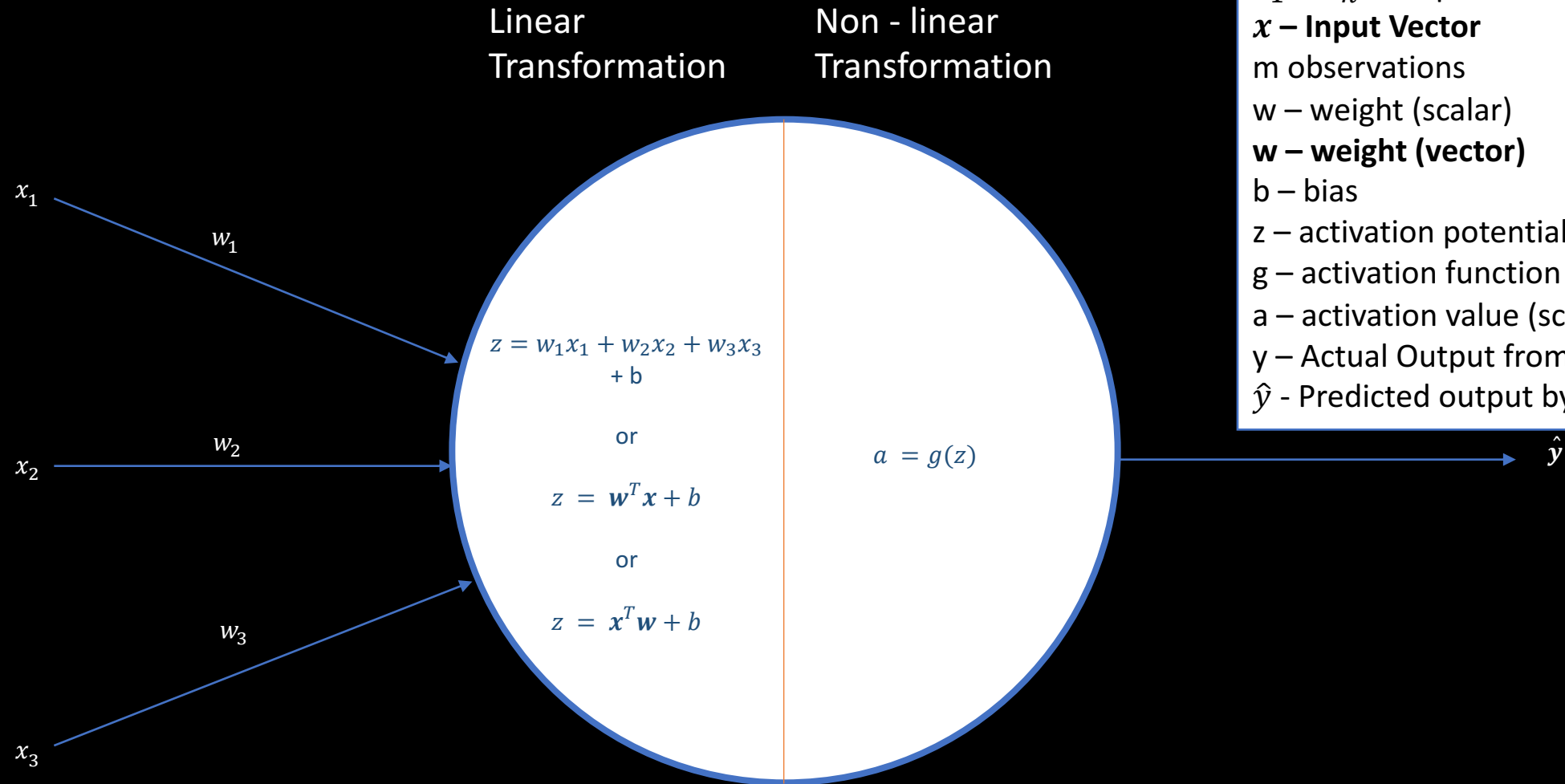
z – activation potential

g – activation function

a – activation value (scalar)

y – Actual Output from input data

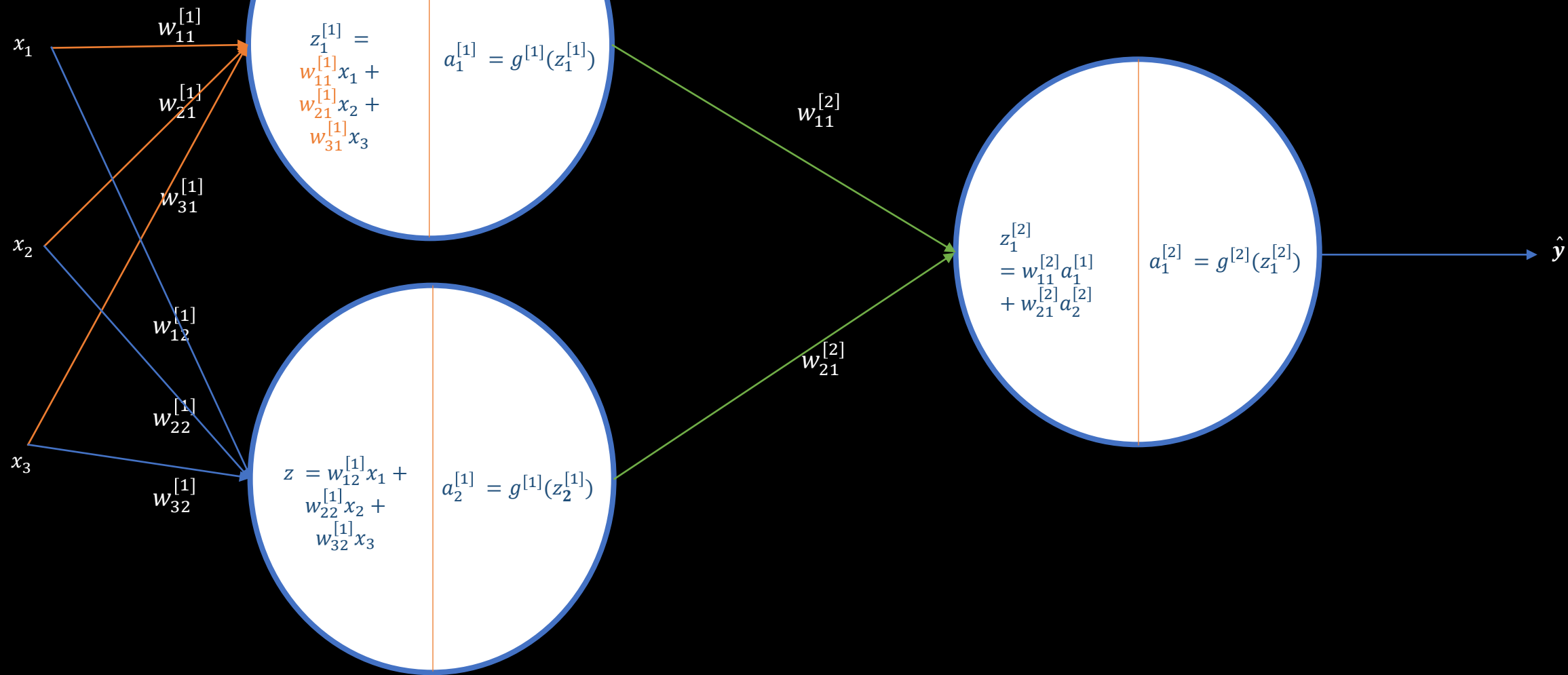
\hat{y} - Predicted output by network



Forward Propagation

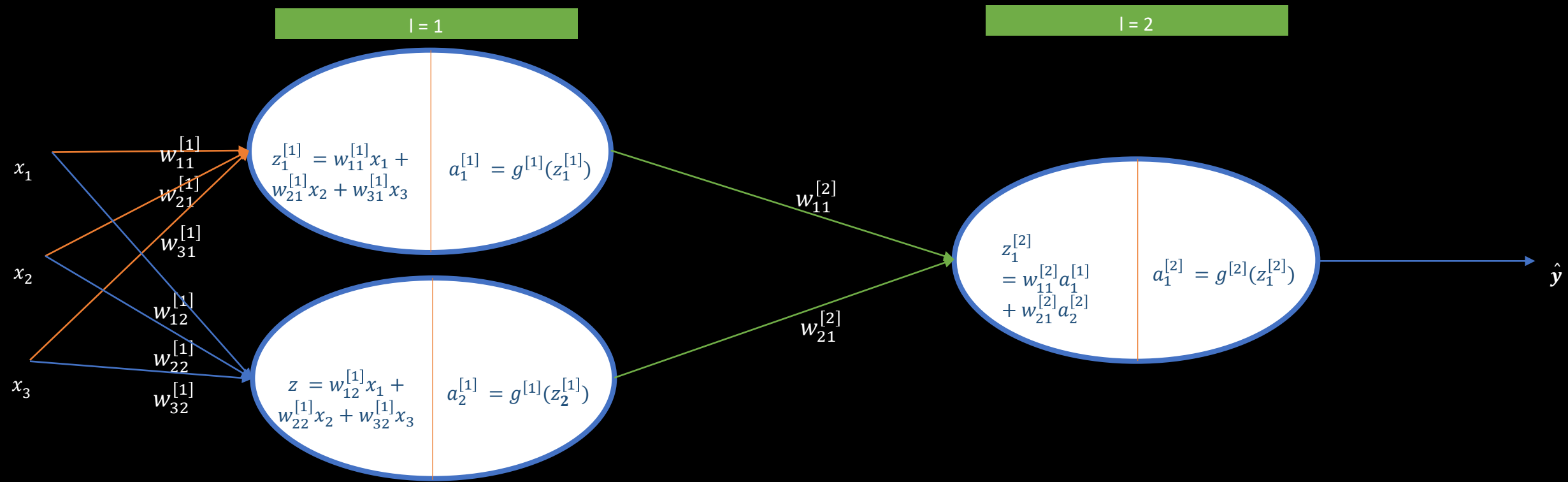
$l = 1$

$l = 2$



Forward Propagation

Vectorization Technique I



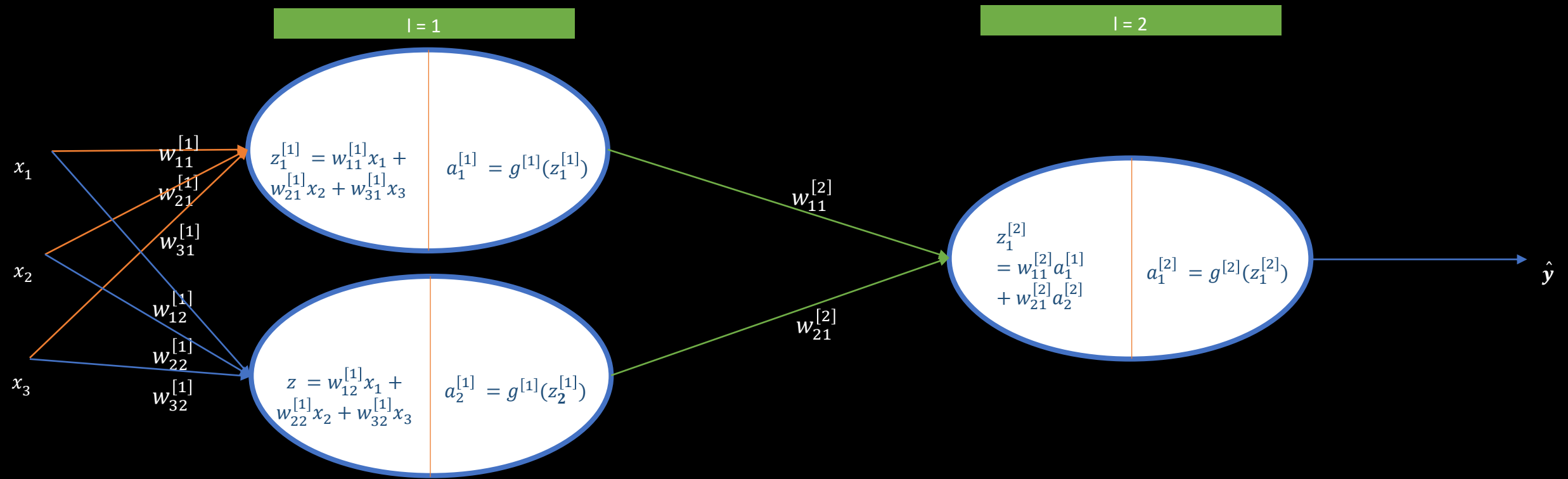
Layer 1 Computation

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \end{bmatrix}$$

$$\mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} \end{bmatrix}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix}$$

$$\begin{aligned} \mathbf{Z}^{[1]} &= \mathbf{XW}^{[1]} + \mathbf{b}^{[1]} \\ \mathbf{A}^{[1]} &= g^{[1]}(\mathbf{Z}^{[1]}) \end{aligned}$$



Layer 2 Computation

$$\mathbf{A}^{[1]} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \end{bmatrix}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} b_1^{[2]} \end{bmatrix}$$

$$\mathbf{Z}^{[2]} = \mathbf{A}^{[1]} \mathbf{W}^{[2]} + \mathbf{b}^{[2]}$$

$$\mathbf{A}^{[2]} = g^{[2]}(\mathbf{Z}^{[2]})$$

Layer 1 Computation

$$\mathbf{A}_0 = \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \end{bmatrix} \quad \mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} \end{bmatrix} \quad \mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix} \quad \begin{aligned} \mathbf{Z}^{[1]} &= \mathbf{XW}^{[1]} + \mathbf{b}^{[1]} \\ \mathbf{A}^{[1]} &= g^{[1]}(\mathbf{Z}^{[1]}) \end{aligned}$$

Layer 2 Computation

$$\mathbf{A}^{[1]} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \quad \mathbf{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \end{bmatrix} \quad \mathbf{b}^{[2]} = \begin{bmatrix} b_1^{[2]} \end{bmatrix} \quad \begin{aligned} \mathbf{Z}^{[2]} &= \mathbf{A}^{[1]}\mathbf{W}^{[2]} + \mathbf{b}^{[2]} \\ \mathbf{A}^{[2]} &= g^{[2]}(\mathbf{Z}^{[2]}) \end{aligned}$$

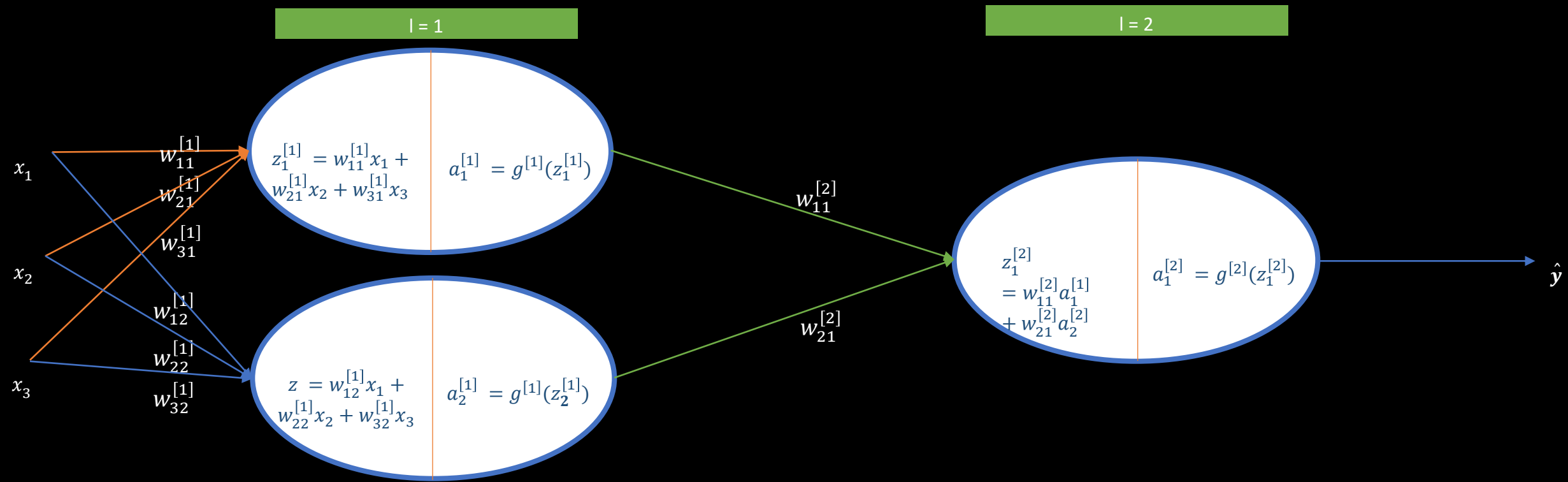
Layer lth Computation

$$\begin{aligned} \mathbf{Z}^{[l]} &= \mathbf{A}^{[l-1]}\mathbf{W}^{[l]} + \mathbf{b}^{[l]} \\ \mathbf{A}^{[l]} &= g^{[l]}(\mathbf{Z}^{[l]}) \end{aligned}$$

```
for l = 1 ... L
{
    Z[l] = numpy.matmul(A[l-1], W[l]) + b[l]
    A[l] = sigmoid(Z[l])
}
```

Forward Propagation

Vectorization Technique II



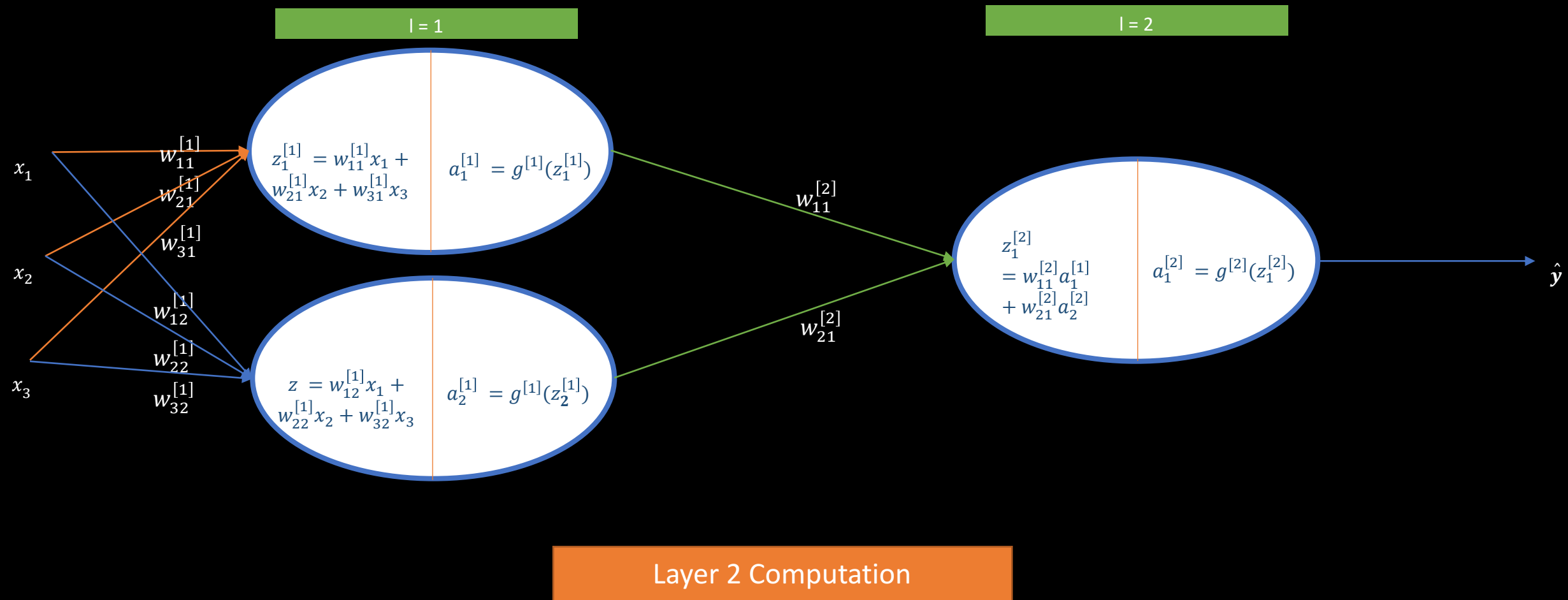
Layer 1 Computation

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} \end{bmatrix}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} & b_2^{[1]} \end{bmatrix}$$

$$\mathbf{Z}^{[1]} = (\mathbf{W}^{[1]})^T \mathbf{X} + \mathbf{b}^{[1]}$$

$$\mathbf{A}^{[1]} = g^{[1]}(\mathbf{Z}^{[1]})$$



$$\mathbf{A}^{[1]} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \end{bmatrix}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} b_1^{[2]} \end{bmatrix}$$

$$\begin{aligned} \mathbf{Z}^{[2]} &= (\mathbf{W}^{[2]})^T \mathbf{A}^{[1]} + \mathbf{b}^{[2]} \\ \mathbf{A}^{[2]} &= g^{[2]}(\mathbf{Z}^{[2]}) \end{aligned}$$

Layer 1 Computation

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{13} & x_{23} & x_{33} & x_{43} \end{bmatrix}$$

$$\mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} \end{bmatrix}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} & b_2^{[1]} \end{bmatrix}$$

$$\begin{aligned} \mathbf{Z}^{[1]} &= (\mathbf{W}^{[1]})^T \mathbf{X} + \mathbf{b}^{[1]} \\ \mathbf{A}^{[1]} &= g^{[1]}(\mathbf{Z}^{[1]}) \end{aligned}$$

Layer 2 Computation

$$\mathbf{A}^{[1]} = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \end{bmatrix}$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \end{bmatrix}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} b_1^{[2]} \end{bmatrix}$$

$$\begin{aligned} \mathbf{Z}^{[2]} &= (\mathbf{W}^{[2]})^T \mathbf{A}^{[1]} + \mathbf{b}^{[2]} \\ \mathbf{A}^{[2]} &= g^{[2]}(\mathbf{Z}^{[2]}) \end{aligned}$$

Layer 1 Computation

$$\mathbf{A}^{[0]} = \mathbf{X} = \begin{bmatrix} x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{13} & x_{23} & x_{33} & x_{43} \end{bmatrix}$$

$$\mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} \end{bmatrix}$$

$$\mathbf{b}^{[1]} = [b_1^{[1]} \quad b_2^{[1]}]$$

$$\begin{aligned} \mathbf{Z}^{[1]} &= (\mathbf{W}^{[1]})^T \mathbf{X} + \mathbf{b}^{[1]} \\ \mathbf{A}^{[1]} &= g^{[1]}(\mathbf{Z}^{[1]}) \end{aligned}$$

Layer 2 Computation

$$\mathbf{A}^{[1]} = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \end{bmatrix}$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \end{bmatrix}$$

$$\mathbf{b}^{[2]} = [b_1^{[2]}]$$

$$\begin{aligned} \mathbf{Z}^{[2]} &= (\mathbf{W}^{[2]})^T \mathbf{A}^{[1]} + \mathbf{b}^{[2]} \\ \mathbf{A}^{[2]} &= g^{[2]}(\mathbf{Z}^{[2]}) \end{aligned}$$

Layer lth Computation

$$\begin{aligned} \mathbf{Z}^{[l]} &= (\mathbf{W}^{[l]})^T \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \\ \mathbf{A}^{[l]} &= g^{[l]}(\mathbf{Z}^{[l]}) \end{aligned}$$

```

for l = 1 ... L
{
    Z[l] = numpy.matmul(W[l].T, A[l-1]) + b[l]
    A[l] = sigmoid(Z[l])
}

```

Backward Propagation

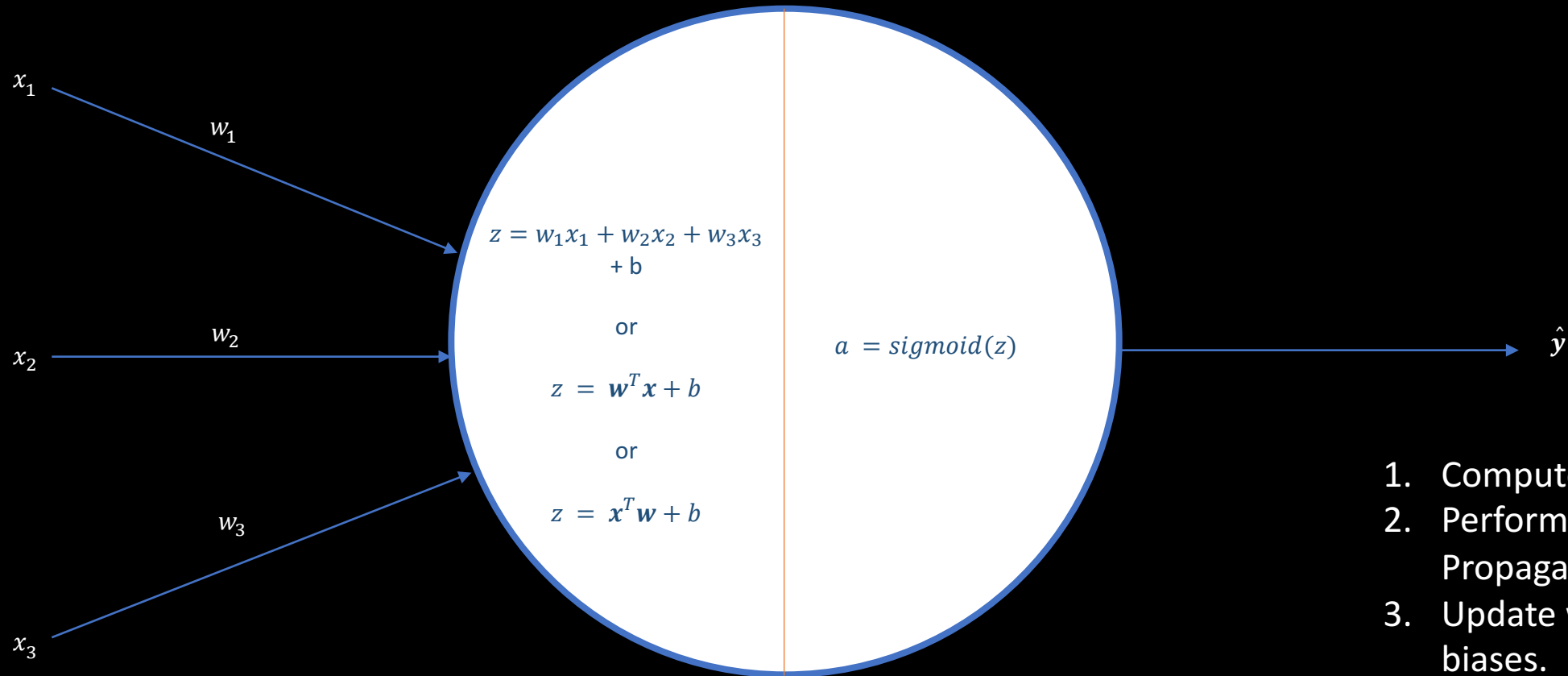
Backward Propagation

- Backpropagation, short for *backward propagation of errors*.
- It is a widely used method for calculating derivatives inside deep feedforward neural networks.
- Backpropagation forms an important part of a number of supervised learning algorithms for training feedforward neural networks, such as stochastic gradient descent.
- When training a neural network by gradient descent, a loss function is calculated, which represents how far the network's predictions are from the true labels.

Backward Propagation

- Backpropagation allows us to calculate the gradient of the loss function with respect to each of the weights of the network.
- This enables every weight to be updated individually to gradually reduce the loss function over many training iterations.
- Backpropagation involves the calculation of the gradient proceeding backwards through the feedforward network from the last layer through to the first.
- To calculate the gradient at a particular layer, the gradients of all following layers are combined via the chain rule of calculus.

Forward Propagation



1. Compute *Loss*
2. Perform Backward Propagation
3. Update weights and biases.

Backward Propagation

Backward Propagation

1. Compute *Loss*

$$Loss = y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

This is called cross entropy loss commonly used in classification. We may use other loss function e.g. MSE as well.

2. Perform Backward Propagation

a. Compute $\frac{\partial Loss}{\partial \hat{y}}$

b. Compute $\frac{\partial Loss}{\partial a} = \frac{\partial Loss}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a}$

c. Compute $\frac{\partial Loss}{\partial z} = \frac{\partial Loss}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z}$

d. Compute $\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w}$

e. Compute $\frac{\partial Loss}{\partial b} = \frac{\partial Loss}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial b}$

3. Update Weights and Biases

$$w = w - \alpha \frac{\partial Loss}{\partial w}$$

$$b = b - \alpha \frac{\partial Loss}{\partial b}$$

KNN (K – Nearest Neighbor) Algorithm

- K-Nearest Neighbor (KNN) is a non-parametric machine learning algorithm used for classification and regression tasks.
- It is a supervised learning technique.
- It is based on the idea of finding the K nearest neighbors to a new observation in the training set and assigning the observation to the class that has the majority of neighbors.
- K-NN algorithm assumes the similarity between the new data points and available data points and put the new point into the class that is most similar to one of the available classes.

KNN (contd.)

- It is also called a **lazy learner** algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset into memory and when it gets new data, then it classifies that data into a class that is much similar to the new data.

How KNN works?

- The KNN algorithm works by finding the distance between the new observation and all the other observations in the training set.
- The distance is usually calculated using Euclidean distance.
- The algorithm then selects the K nearest neighbors to the new observation based on their distances.
- Finally, the new observation is assigned to the class that has the majority of neighbors among the K nearest ones.

Working of KNN Algorithm

- Given training data is:

| RIGHTNESS | SATURATION | CLASS |
|-----------|------------|-------|
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |

The new data point to classify is:

| BRIGHTNESS | SATURATION | CLASS |
|------------|------------|-------|
| 20 | 35 | ? |

Working of KNN Algorithm (contd.)

- Next, we compute a distance between new point and the existing points using Euclidean distance.

$$dist = \sqrt{\{(x_2 - x_1)^2 + (y_2 - y_1)^2\}}$$

| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
|------------|------------|-------|----------|
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | 68.01 |
| 10 | 25 | Red | 10 |
| 70 | 70 | Blue | 61.03 |
| 60 | 10 | Red | 47.17 |
| 25 | 80 | Blue | 45 |

Working of KNN Algorithm (contd.)

- Now, we rearrange the distances in ascending order:

| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
|------------|------------|-------|----------|
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |
| 70 | 70 | Blue | 61.03 |
| 60 | 90 | Blue | 68.01 |

Since we chose 5 as the value of **K**, we'll only consider the first five rows.

As you can see above, the majority class within the 5 nearest neighbors to the new entry is **Red**.

Pros and Cons of KNN

Pros

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Cons

- Always needs to determine the value of K which may be complex some time.
- The curse of dimensionality can affect the performance of the algorithm in high-dimensional feature spaces.
- The computation cost is high because of calculating the distance between the data points for all the training samples.
- Doesn't scale

How to choose the value of K in KNN?

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as $K=1$ or $K=2$, can be noisy and lead to inaccurate predictions due to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties. However, always use an odd number as the value of K.
- Use Hierarchical clustering.

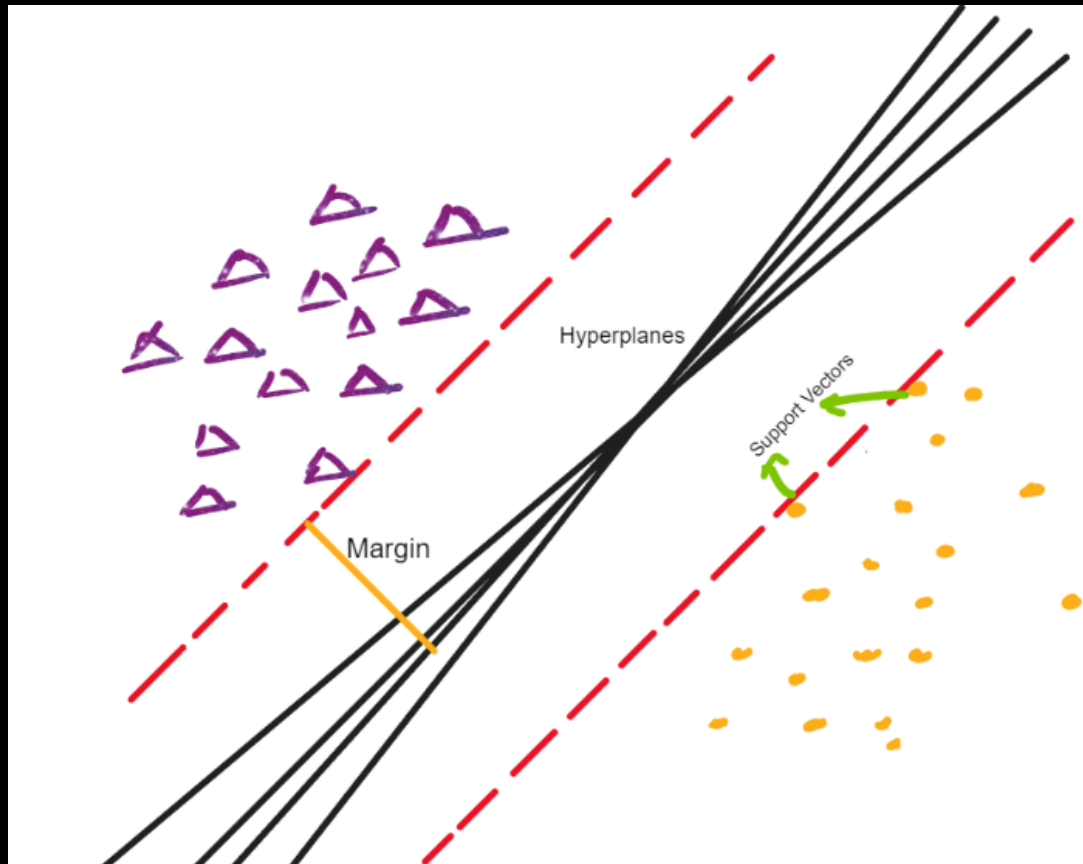
Applications of K-Nearest Neighbor Algorithm

- KNN can be used in a variety of applications, such as image recognition, spam filtering, and recommendation systems.
- It is particularly useful when the underlying data distribution is unknown or when the dataset is small.

Support Vector Machines (SVM)

- Support Vector Machines (SVM) is a powerful supervised machine learning algorithm used for classification and regression analysis.
- Support Vector Machine (SVM) is non parametric supervised machine learning algorithm that fits best for classification problem. But it also works well with regression problem.
- In N-dimensional input space, our goal using SVM is to learn a **hyperplane** that best separates the input space into classes.
 - Hyperplane is a line in 2D space and just a point in 1D space.
- Multiple hyperplane can exists in given input space that separates classes. And, the optimal hyperplane is the one among all plausible hyperplanes that separates classes with largest margin.

SVM (contd.)



- Here, the distance between hyperplane and the closest data point is called **margin**.
- The margin is computed only with the closest points called '**support vectors**' as the perpendicular distance from these support vectors with the hyperplane.

SVM

- The distance of any line, $ax + by + c = 0$ from a given point say, (x_0, y_0) is given by d .

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

- Similarly, the distance of a hyperplane equation $w^T X + b = 0$ from a given point x_0 can be written as:

$$d = \frac{|w^T x + b|}{\|w\|_2}$$

SVM (contd.)

- They are called `**support vectors**` because they support the construction of hyperplane determining their orientation.
- Thus in SVM, the optimal hyperplane is obtained from the learning process using training data through the maximization of margin.
- SVM is not just limited to linear classification, SVM's can efficiently perform a non-linear classification, implicitly mapping their inputs into high dimensional feature space.

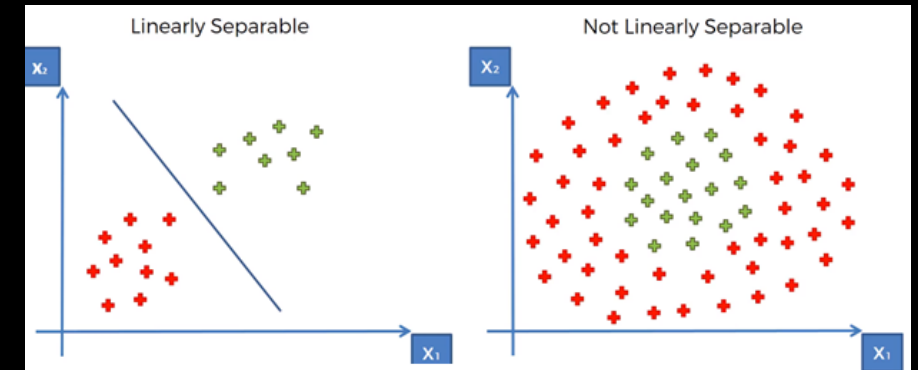
Hard Margin vs. Soft Margin

- We have already defined margin as the perpendicular distance between support vectors and the hyperplanes. Also, we choose the hyperplane with maximal margin as the optimal one.
- But, in SVM, we have notion of two different margin viz. Hard Margin and Soft Margin.
- In SVM, the hyperplane is expected to separate the classes clearly such that the points belonging to one class is on one side of the hyperplane only and the points belonging to the another class is on other side of the hyperplane only. This is called **Hard Margin** which is default state of SVM.
- On the other hand, if we allow some data points to cross the boundary set by hyperplane i.e. we allow some misclassification of points by relaxing the hard constraints set by hyperplane. This is called **Soft Margin**.

Hard Margin vs. Soft Margin (contd.)

- We incorporate soft margin by introducing slack variables C . This is a regularization parameter that defines how much misclassification are we allowing for i.e. C determines the no. of observations allowed to cross the boundary set by hyperplane.
- The smaller the value of C the more violations of boundary or misclassification is allowed and is more sensitive to the training data. This is the case of higher variance and lower bias.
- Conversely, the larger the value of the algorithm is more sensitive towards training data causing lower variance and higher bias.
- Conclusion is, hard margin implementation has larger value of C and Soft margin implementation has lesser value of C .

Types of SVM



Linear SVM

- Linear SVM is used for linearly separable data.
- It finds the optimal hyperplane that separates the two classes using a linear function.
- The hyperplane with the maximum margin is selected as the best classifier.

Non Linear SMV

- Nonlinear SVM is used for non-linearly separable data.
- It uses a kernel function to transform the data into a higher-dimensional feature space, where a linear boundary can be used to separate the classes.
- The kernel function can be linear, polynomial, Gaussian, or sigmoid.

Kernel in SVM

- We have already mentioned above that the SVM not only works with linear data but also with non linear data. It does so by transforming lower dimension data into higher dimension such that classes are linearly separable. And this is achieved through the use of so called `Kernel`.
- In SVM, a kernel is a function that performs mapping of the input data from the original feature space to a higher-dimensional space.
- The kernel trick enables SVM to find a nonlinear decision boundary in the feature space by transforming the data into a higher-dimensional space where a linear boundary can be used.
- The learning of hyperplane explained above is the result of linear kernel.

Types of Kernel

1. Linear Kernel

The linear kernel is the dot product between new data point and the support vectors.

For each data point x_j in X , compute

$$K(x_i, x_j) = x_i \cdot x_j$$

where, x_i is the support vector.

Types of Kernel (contd.)

2. Polynomial Kernel

The polynomial kernel is given as:

foreach data point x_j in X , compute

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$$

where d is the degree of polynomial and is a hyperparameter which can't be learned through training.

Types of Kernel (contd.)

3. Gaussian Kernel

It is general purpose Kernel and used when there is no prior knowledge about data. It is given as:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Types of Kernel (contd.)

4. Gaussian Radial Basis Function (RBF) Kernel

It is also general purpose Kernel and used when there is no prior knowledge about data. It is given as:

$$K(x_i, x_j) = \exp(\gamma * \sum(x_j - x_i)^2)$$

Where, gamma is the hyperparameter which defines how far the single training example influence is. Thus, higher value of gamma consider points closer to the plausible hyperparameter and conversely, lower value of gamma consider farther points.

Types of Kernel (contd.)

5. Laplace RBF Kernel

It is also general purpose Kernel and used when there is no prior knowledge about data. It is given as:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|}{\sigma}\right)$$

Pros and Cons of SVM

Pros

- SVM works well when margin is clearly distinguishable.
- SVM is more effective in high dimensional spaces.
- SVM works in case where the number of dimensions is larger than the number of samples.
- SVM is also memory efficient algorithm since this algorithm uses only the subject of observations called support vectors to determine hyperplane and is suitable for both classification and regression.

Pros and Cons of SVM (contd.)

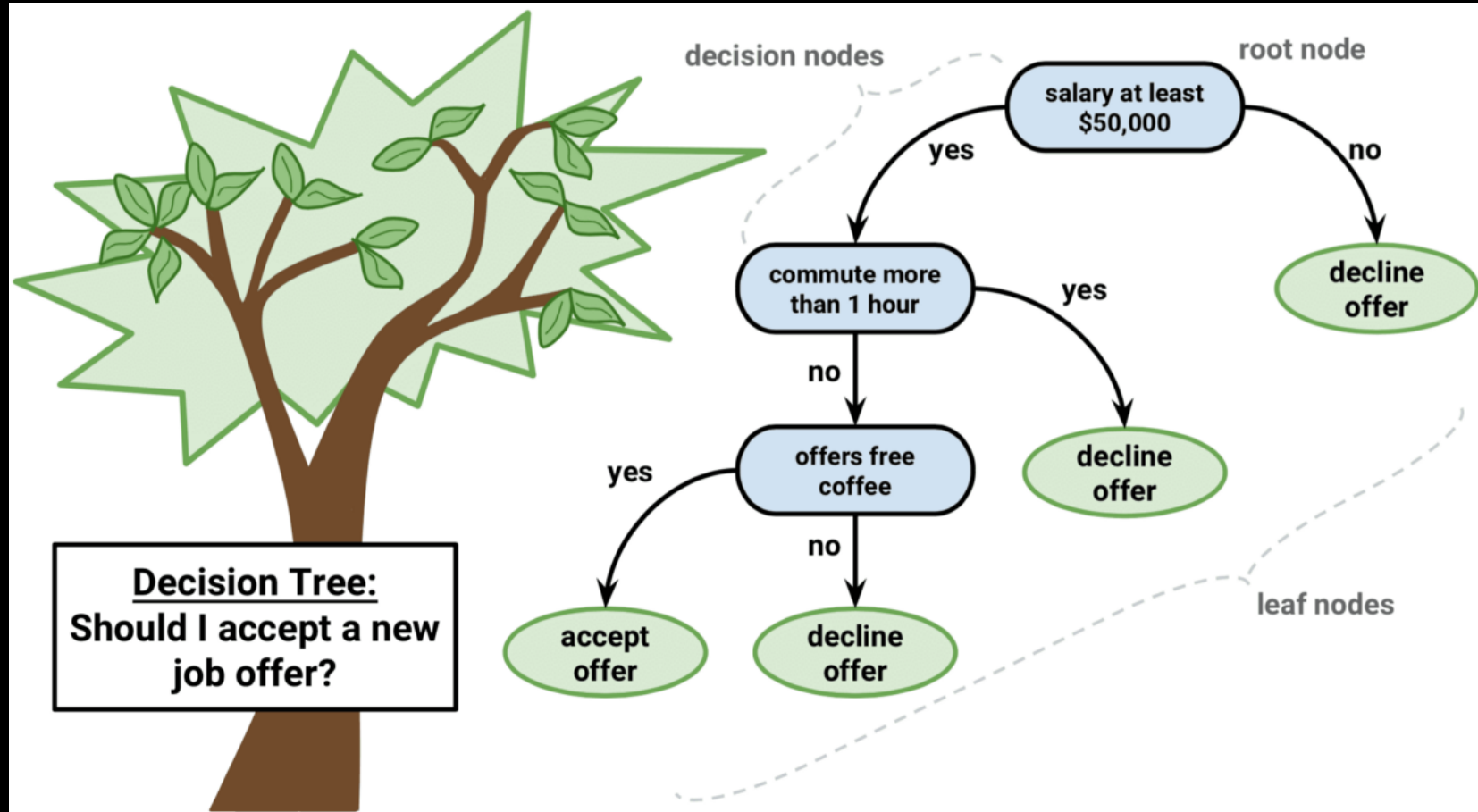
Cons

- SVM is not preferred with very large data sets.
- SVM does not work well with overlapping classes.
- SVM doesn't work well when number of observations are too much lesser than the number of features.

Decision Tree

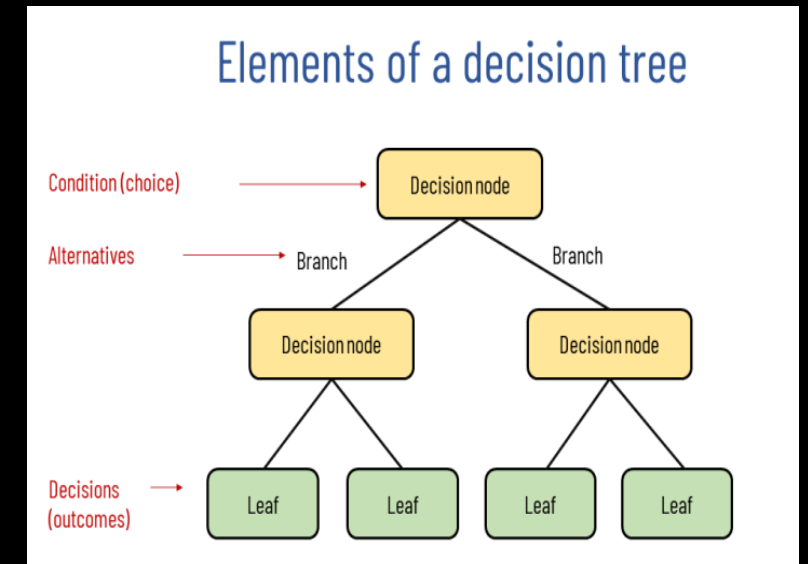
- Decision Trees are also a non-parametric model used for both regression and classification.
- Depiction of inverted tree like structure gave it a name tree and since it is used to make decisions, thus it is called **decision tree**.
- Each internal node denotes an attribute with some branching condition, if it has, extending to child nodes.
- Decision tree is a graphical representation of all possible solutions to a decision.
- In ML, It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

Decision Tree



Decision Tree

- During training, the Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets.
- The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.
- Decision trees also provide the foundation for more advanced ensemble methods such as **bagging**, **random forests** and **gradient boosting**.



Construction of Decision Tree

- There are various techniques to construct decision tree:
 - CART (Classification and Regression Trees)
 - ID3 (Iterative Dichotomizer 3)
 - C4.5
 - CHAID (Chi-Squared Automatic Interaction Detection)

Classification and Regression Tree (CART)

- CART is simple to understand, interpret, visualize and requires little or no effort for data preparation.
- Moreover, it performs feature selection.
- **Regression trees** are mainly used when the target variable is numerical.
- Here value obtained by a terminal node is always the mean or average of the responses falling in that region. As a result, if any unseen data or observation will predict with the mean value.
- **Classification** is used when the target variable is categorical.
- Here value obtained by a terminal node is the mode of response falling in that region and any unseen data or observation in this region will make a prediction based on the mode value.

CART (contd.)

- The decision to make a strategic split heavily affects the accuracy of the tree and the decision criteria for regression and classification trees will be different.
- Entropy/Information gain or Gini Index can be used for choosing the best split.
- For a given dataset with different features, to decide which feature to be considered as the root node and which feature should be the next decision node and so on, information gain of each feature should be known.
- The feature which has maximum information gain will be considered as the root node.

CART (contd.)

- Entropy: Entropy is a measure of disorder or impurity in the given dataset.
- In the decision tree, messy data are split based on values of the feature vector associated with each data point.
- With each split, the data becomes more homogenous which will decrease the entropy. However, some data in some nodes will not be homogenous, where the entropy value will not be small.
- The higher the entropy, the harder it is to draw any conclusion. When the tree finally reaches the terminal or leaf node maximum purity is added.

CART (contd.)

Information gain indicates how much information a particular variable or feature gives us about the final outcome. It can be found out by subtracting the entropy of a particular attribute inside the data set from the entropy of the whole data set.

- For a dataset that has C classes and the probability of randomly choosing data from class i is p_i . Then entropy $E(S)$ can be mathematically represented as

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Then information gain is given as:

$$\text{Gain}(A, S) = E(S) - E(A, S)$$

Where,

$$E(A, S) = \sum_{j=1}^v \frac{|S_j|}{|S|} \cdot E(S_j)$$

$H(S)$: entropy of whole data set S

$|S_j|$: number of instances with j value of an attribute A

$|S|$: total number of instances in the dataset

v - set of distinct values of an attribute A

$E(S_j)$ - entropy of subset of instances for attribute A

$E(A, S)$ - entropy of an attribute A

CART (contd.)

- Let's take an example:

| Day | Outlook | Humidity | Wind | Play |
|-----|----------|----------|--------|------|
| 1 | Sunny | High | Weak | No |
| 2 | Sunny | High | Strong | No |
| 3 | Overcast | High | Weak | Yes |
| 4 | Rain | High | Weak | Yes |
| 5 | Rain | Normal | Weak | Yes |
| 6 | Rain | Normal | Strong | No |
| 7 | Overcast | Normal | Strong | Yes |
| 8 | Sunny | High | Weak | No |
| 9 | Sunny | Normal | Weak | Yes |
| 10 | Rain | Normal | Weak | Yes |
| 11 | Sunny | Normal | Strong | Yes |
| 12 | Overcast | High | Strong | Yes |
| 13 | Overcast | Normal | Weak | Yes |
| 14 | Rain | High | Strong | No |

CART (contd.)

First in order to find the root node, we calculate the entropy of our dataset as

$$E(S) = -\left(\frac{9}{14}\log_2\left(\frac{9}{14}\right) + \frac{5}{14}\log_2\left(\frac{5}{14}\right)\right) = 0.94$$

How?

Check on the decision node, there are two events as outcomes. One is **yes** and the other is **no**. Out of 14 records, 9 are **yes** and 5 are **no**. Thus,

$$P(\text{yes}) = 9/14$$

$$P(\text{no}) = 5/14$$

CART (contd.)

Next, we compute the information gain for each feature. For that compute,

$$\begin{aligned} & E(Outlook, S) \\ &= -\frac{5}{14} \left(-\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5} \right) \right) + \frac{4}{14} \left(-\left(\frac{4}{4} \log_2 \frac{4}{4} \right) \right) \\ &+ -\frac{5}{14} \left(-\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5} \right) \right) = 0.693 \end{aligned}$$

CART (contd.)

Now, Information Gain

$$IG(Outlook) = E(S) - E(Outlook, S) = 0.94 - 0.693 = 0.247$$

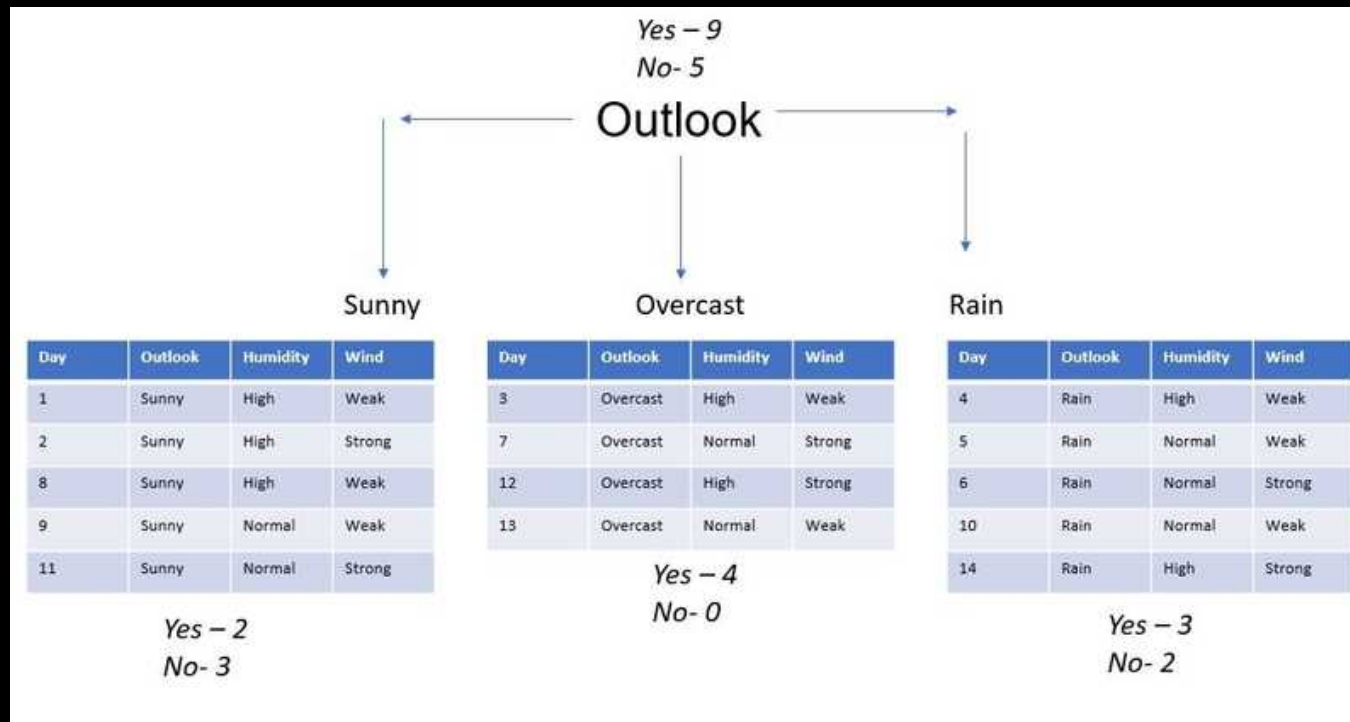
Repeat same for other attributes as well. So,

$$IG(Wind) = 0.94 - \frac{8}{14} \left(- \left(\frac{6}{8} \log_2 \frac{6}{8} + \frac{2}{8} \log_2 \frac{2}{8} \right) \right) + \frac{6}{14} \left(- \left(\frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6} \right) \right) = 0.048$$

$$IG(Humidity) = 0.94 - \frac{7}{14} \left(- \left(\frac{3}{7} \log_2 \frac{3}{7} + \frac{4}{7} \log_2 \frac{4}{7} \right) \right) + \frac{7}{14} \left(- \left(\frac{6}{7} \log_2 \frac{6}{7} + \frac{1}{7} \log_2 \frac{1}{7} \right) \right) = 0.151$$

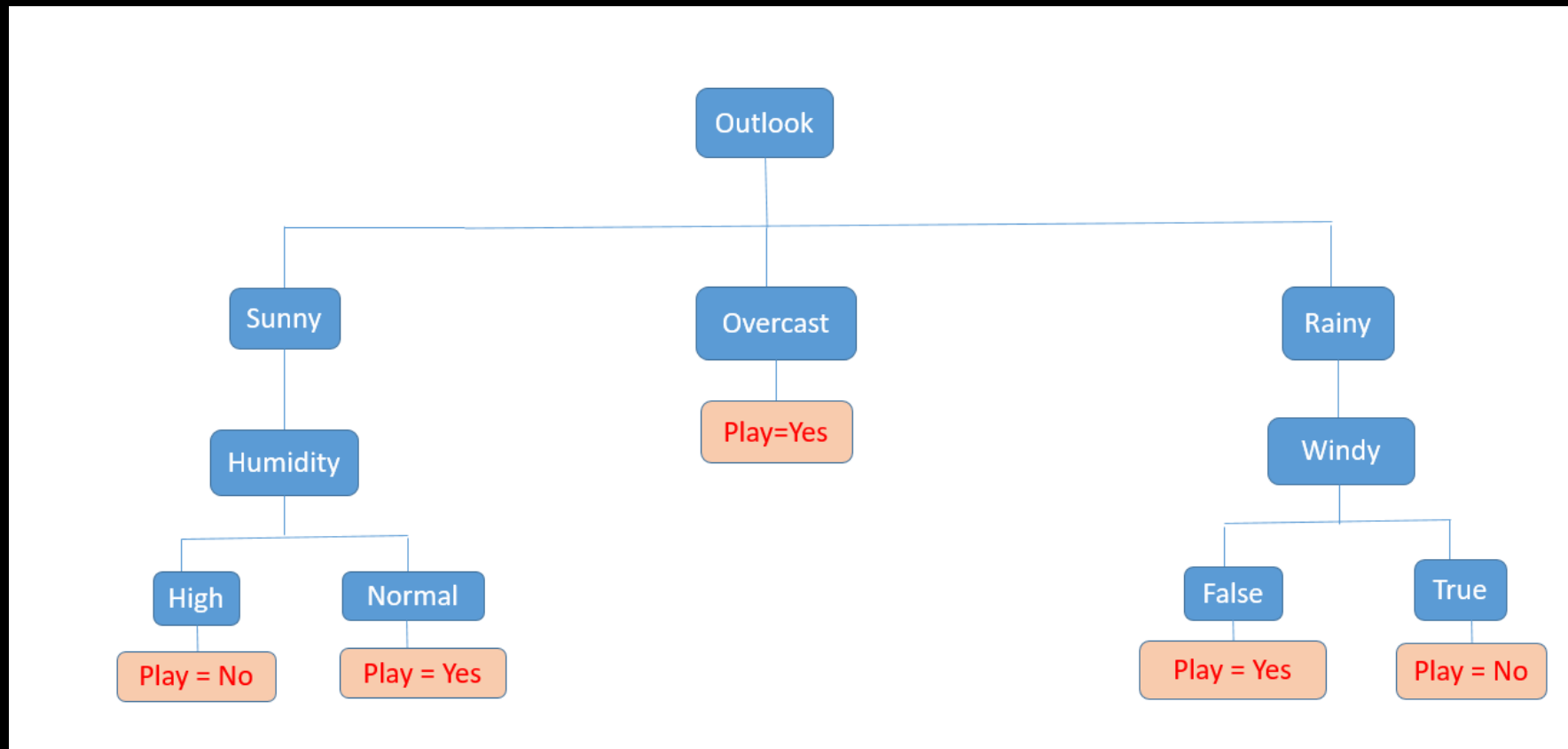
CART (contd.)

Since, the information gain of the Outlook is highest, Outlook will be the root of the tree and we get tree as



CART (contd.)

- Repeat the process until you get decision on leaf node as below:



Pros and Cons of Decision Tree

Pros

- Intuitive and easy to understand
- Non-parametric in nature and less number of data propagation required.

Pros and Cons of Decision Tree

Cons

- High chance of overfitting as it is a high variance algorithm.
- Unstable as small change in data can dramatically change the predictions produced by the model.
- Less effective in their ability to represent complex relationship between variables, particularly when dealing with nonlinear data.
- Although used for regression, but it is less appropriate for the prediction of continuous value.

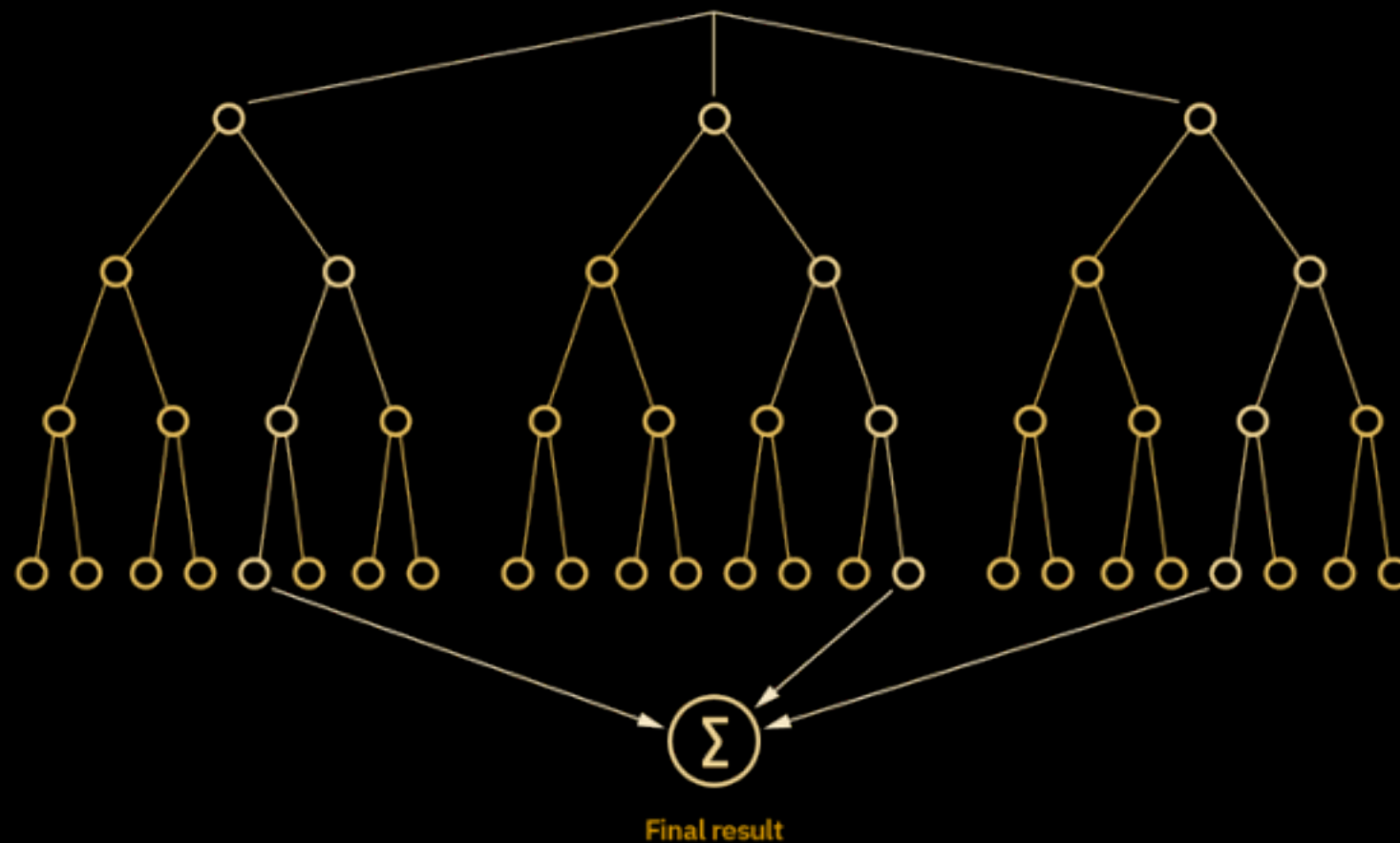
Random Forest

- Random Forest is a commonly-used ML algorithm which gives the foundation to the rise of **ensemble method**.
- Forest basically means the multiple trees.
- Thus, Random Forest combines the output of multiple decision tree to determine single outcome with feature randomness to create an uncorrelated forest of decision trees.
- Feature randomness generates a random subset of features, which ensures low correlation among decision trees. *(This marks the key difference between decision trees and random forests.)*

Random Forest (contd.)

- One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification.
- It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

Random Forest (contd.)



Essential Features of Random Forest

- **Miscellany**: Each tree has a unique attribute, variety and features concerning other trees. Not all trees are the same.
- **Immune to the curse of dimensionality**: Since a tree is a conceptual idea, it requires no features to be considered. Hence, the feature space is reduced.
- **Parallelization**: We can fully use the CPU to build random forests since each tree is created autonomously from different data and features.
- **Train-Test split**: In a Random Forest, we don't have to differentiate the data for train and test because the decision tree never sees 30% of the data.
- **Stability**: The final result is based on Bagging, meaning the result is based on majority voting or average.

Steps Involved in Random Forest Algorithm

1. In the Random forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.
2. Individual decision trees are constructed for each sample.
3. Each decision tree will generate an output.
4. Final output is considered based on **Majority Voting or Averaging** for Classification and regression, respectively.

When to Avoid Using Random Forests?

Random Forests Algorithms are not ideal in the following situations:

- **Extrapolation:** Random Forest regression is not ideal in the extrapolation of data. Unlike linear regression, which uses existing observations to estimate values beyond the observation range.
- **Sparse Data:** Random Forest does not produce good results when the data is sparse. In this case, the subject of features and bootstrapped sample will have an invariant space. This will lead to unproductive splits, which will affect the outcome.

Pros and Cons of Random Forest

Pros

- **Reduced risk of overfitting:** Decision trees run the risk of overfitting as they tend to tightly fit all the samples within training data. However, when there's a robust number of decision trees in a random forest, the classifier won't overfit the model since the averaging of uncorrelated trees lowers the overall variance and prediction error.
- **Provides flexibility:** Since random forest can handle both regression and classification tasks with a high degree of accuracy, it is a popular method among data scientists. Feature bagging also makes the random forest classifier an effective tool for estimating missing values as it maintains accuracy when a portion of the data is missing.
- **Easy to determine feature importance:** Random forest makes it easy to evaluate variable importance, or contribution, to the model. There are a few ways to evaluate feature importance. **Gini importance** and mean decrease in impurity (MDI) are usually used to measure how much the model's accuracy decreases when a given variable is excluded. However, permutation importance, also known as **mean decrease accuracy (MDA)**, is another importance measure. MDA identifies the average decrease in accuracy by randomly permutating the feature values in oob samples.

Pros and Cons of Random Forest

Cons

- **Time-consuming process:** Since random forest algorithms can handle large data sets, they can provide more accurate predictions, but can be slow to process data as they are computing data for each individual decision tree.
- **Requires more resources:** Since random forests process larger data sets, they'll require more resources to store that data.
- **More complex:** The prediction of a single decision tree is easier to interpret when compared to a forest of them.

Naïve Bayes

- Naive Bayes is a probabilistic machine learning algorithm used for classification tasks.
- It is based on Bayes' theorem and assumes feature independence.
- Bayes' theorem is given as:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

Where

$P(H|D)$ is the **posterior probability distribution** such that $P(H)$ is the **prior probability distribution** and $P(D|H)$ is the **likelihood**.

Naïve Bayes (contd.)

- There's a consumer of my Computer shop, who is 35 years old with a income of 40,000 and fair credit rating.
- Will he buy new computer?

- **Hypothesis:**

H_0 : He will not buy computer

H_1 : He will buy computer

Then,

Naïve Bayes (contd.)

$P(H|D)$ is the probability that the customer will buy a computer given that we know his age, income and credit rating. This is the posterior probability of H)

$P(H)$ is the prior probability of H that the customer will buy a computer regardless of age, income and credit rating.

$P(D|H)$ is the probability that the customer is 35 years old, have fair credit rating and earns 40,000, given that he has bought a computer

$P(D)$ is the probability that a person from our set of customers is 35 years old, have fair rating and earns Rs. 40,000. This is the marginal probability.

Naïve Bayes (contd.)

- Let D be a dataset and C_1, C_2, \dots, C_m are m classes. Then Bayes rule can be written as:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

- Given a tuple, X , the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X .
- Thus, we need to maximize $P(C_i|X)$. As $P(X)$ is constant for all classes, it can be omitted in calculation.

Naïve Bayes (contd.)

- Let X is the set of attributes $\{x_1, x_2, \dots, x_n\}$ where attributes are independent of one another.
- Now the posterior probability $P(X|C_i)$ is given by the equation given below:

$$P(X|C_i) = \prod_{j=1}^n P(x_j|C_i) = P(x_1|C_i) * P(x_2|C_i) * \dots * P(X_n|C_i)$$

Naïve Bayes (contd.)

| Type of family structure | Age group | Income status | Will they buy a car? |
|--------------------------|-------------|---------------|----------------------|
| Nuclear | Young | Low | Yes |
| Extended | Old | Low | No |
| Childless | Middle-aged | Low | No |
| Childless | Young | Medium | Yes |
| Single Parent | Middle-aged | Medium | Yes |
| Childless | Young | Low | No |
| Nuclear | Old | High | Yes |
| Nuclear | Middle-aged | Medium | Yes |
| Extended | Middle-aged | High | Yes |
| Single Parent | Old | Low | No |

Will a person buy a car when Single Parent, Young and Low are the inputs?

Naïve Bayes (contd.)

- Firstly, let's compute the probability of the output labels $P(C_i)$ given the data.

$$\begin{aligned}P(No) &= 4/10 \\ P(Yes) &= 6/10\end{aligned}$$

Naïve Bayes (contd.)

- Now let's calculate the probability of the likelihood of the evidence. Given the inputs Childless, Young, and Low, we'll calculate the probability with respect to both class labels as follows:

$$P(\text{"Single Parent"}|Yes) = 1/6$$

$$P(\text{"Single Parent"}|No) = 1/4$$

$$P(Young|Yes) = 2/6$$

$$P(Young|No) = 1/4$$

$$P(Low|Yes) = 1/6$$

$$P(Low|No) = 4/4$$

Naïve Bayes (contd.)

- Now the posterior probability is calculated as:

$$P(Yes|X) = P(\textit{"Single Parent"}|Yes) * P(Young|Yes) * P(Low|Yes) = 1/6 * 2/6 * 1/6 = 0.0063$$

$$P(No|X) = P(\textit{"Single Parent"}|No) * P(Young|No) * P(Low|No) = 1/4 * 1/4 * 4/4 = 0.0625$$

Naïve Bayes (contd.)

The final probabilities are:

$$P(Yes|X) = 0.0063 / (0.0063 + 0.0625) = 0.09$$

$$P(No|X) = 0.0625 / (0.0063 + 0.0625) = 0.91$$

Thus, the results clearly show that the car probably will not be purchased.

Pros and Cons of Naïve Bayes

Pros

- This algorithm works very fast and can easily predict the class of a test dataset.
- You can use it to solve multi-class prediction problems as it's quite useful with them.
- Naive Bayes classifier performs better than other models with less training data if the assumption of independence of features holds.
- If you have categorical input variables, the Naive Bayes algorithm performs exceptionally well in comparison to numerical variables.
- It can be used for Binary and Multi-class Classifications.
- It effectively works in Multi-class predictions.

Pros and Cons of Naïve Bayes

Cons

- If your test data set has a categorical variable of a category that wasn't present in the training data set, the Naive Bayes model will assign it zero probability and won't be able to make any predictions in this regard. This phenomenon is called 'Zero Frequency,' and you'll have to use a smoothing technique to solve this problem.
- It assumes that all the features are independent. While it might sound great in theory, in real life, you'll hardly find a set of independent features.

End of the Chapter