

Three Types of Learning That Artificial Intelligence (AI) Does

- **Supervised Learning:** The machine has a “teacher” who guides it by providing sample inputs along with the desired output. The machine then maps the inputs and the outputs. This is similar to how we teach very young children with picture books. According to Yann LeCun, all of the AI machines we have today have used this form of learning (from speech recognition to self-driving cars).
- **Reinforcement Learning:** Yann LeCun believes this plays a relatively minor role in training AI and is similar to training an animal. When the animal displays a desired behavior it is given a reward. According to the Wikipedia entry on Machine Learning, reinforcement learning is defined as “a computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal. “
- **Unsupervised Learning:** This is the most important and most difficult type of learning and would be better titled Predictive Learning. In this case the machine is not given any labels for its inputs and needs to “figure out” the structure on its own. This is similar to how babies learn early in life. For example they learn that if an object in space is not supported it will fall

Learning Decision Trees

A decision tree is a simple representation for classifying examples. Decision tree learning is one of the most successful techniques for supervised classification learning. For this section, assume that all of the features have finite discrete domains, and there is a single target feature called the **classification**. Each element of the domain of the classification is called a **class**.

A **decision tree** or a **classification tree** is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

To classify an example, filter it down the tree, as follows. For each feature encountered in the tree, the arc corresponding to the value of the example for that feature is followed. When a leaf is reached, the classification corresponding to that leaf is returned.

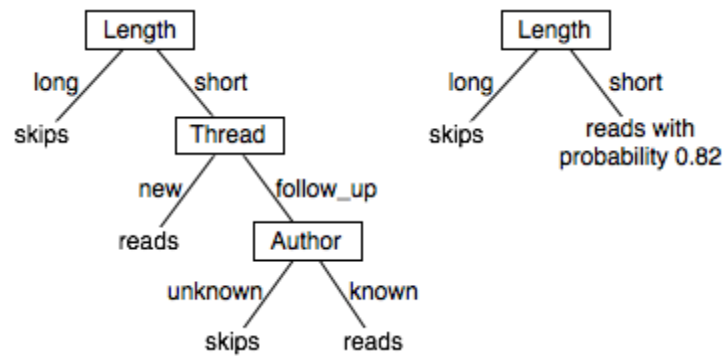


Figure : Two decision trees

Example : Figure shows two possible decision trees for the example of Figure . Each decision tree can be used to classify examples according to the user's action. To classify a new example using the tree on the left, first determine the length. If it is long, predict skips. Otherwise, check the thread. If the thread is new, predict reads. Otherwise, check the author and predict read only if the author is known. This decision tree can correctly classify all examples in Figure.

The tree on the right makes probabilistic predictions when the length is short. In this case, it predicts reads with probability 0.82 and so skips with probability 0.18.

A deterministic decision tree, in which all of the leaves are classes, can be mapped into a set of rules, with each leaf of the tree corresponding to a rule. The example has the classification at the leaf if all of the conditions on the path from the root to the leaf are true.

Example: The leftmost decision tree of Figure can be represented as the following rules:

skips \leftarrow *long*.

reads \leftarrow *short* \wedge *new*.

reads \leftarrow *short* \wedge *follow Up* \wedge *known*.

skips \leftarrow *short* \wedge *follow Up* \wedge *unknown*.

With negation as failure, the rules for either *skips* or *reads* can be omitted, and the other can be inferred from the negation.

To use decision trees as a target representation, there are a number of questions that arise:

- Given some training examples, what decision tree should be generated? Because a decision tree can represent any function of the input features, the bias that is necessary to learn is incorporated into the preference of one decision tree over another. One proposal is to prefer the smallest tree that is consistent with the data, which could mean the tree with the least depth or the tree with the fewest nodes. Which decision trees are the best predictors of unseen data is an empirical question.
- How should an agent go about building a decision tree? One way is to search the space of decision trees for the smallest decision tree that fits the data. Unfortunately the space of decision trees is enormous. A practical solution is to carry out a local search on the space of decision trees, with the goal of minimizing the error. This is the idea behind the algorithm described below.

Support Vector Machine(SVM)

Overview

- Explanation of support vector machine (SVM), a popular machine learning algorithm or classification
- Implementation of SVM in R and Python
- Learn about the pros and cons of Support Vector Machines(SVM) and its different applications

Introduction

Mastering machine learning algorithms isn't a myth at all. Most of the beginners start by learning regression. It is simple to learn and use, but does that solve our purpose? Of course not! Because you can do so much more than just Regression!

Think of machine learning algorithms as an armoury packed with axes, sword, blades, bow, dagger, etc. You have various tools, but you ought to learn to use them at the right time. As an analogy, think of 'Regression' as a sword capable of slicing and dicing data efficiently, but incapable of dealing with highly complex data. On the contrary, 'Support Vector Machines' is like a sharp knife – it works on smaller datasets, but on the complex ones, it can be much stronger and powerful in building machine learning models.

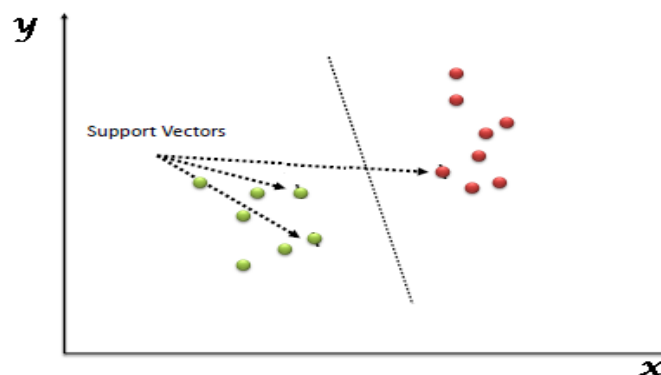
By now, I hope you've now mastered Random Forest, Naive Bayes Algorithm and Ensemble Modeling. If not, I'd suggest you take out a few minutes and read about them as well. In this article, I shall guide you through the basics to advanced knowledge of a crucial machine learning algorithm, support vector machines.

Table of Contents

1. What is Support Vector Machine?
2. How does it work?
3. How to implement SVM in Python and R?
4. How to tune Parameters of SVM?
5. Pros and Cons associated with SVM

What is Support Vector Machine?

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



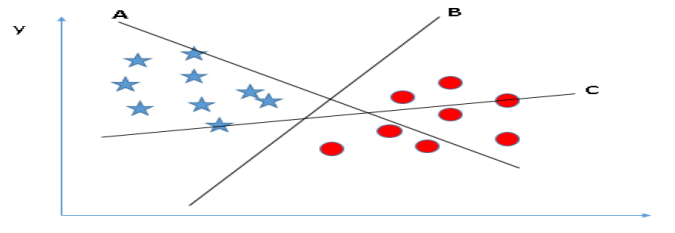
Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

How does it work?

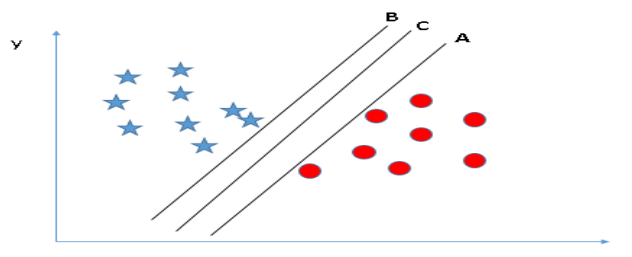
Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?”. Don't worry, it's not as hard as you think!

Let's understand:

- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

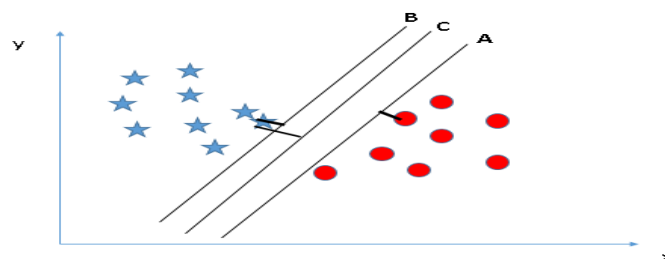


- You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.
- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



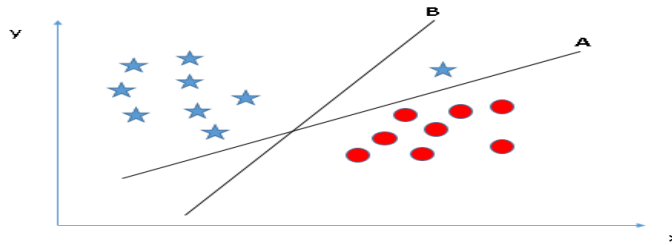
Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

Let's look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):** Hint: Use the rules as discussed in previous section to identify the right hyper-plane

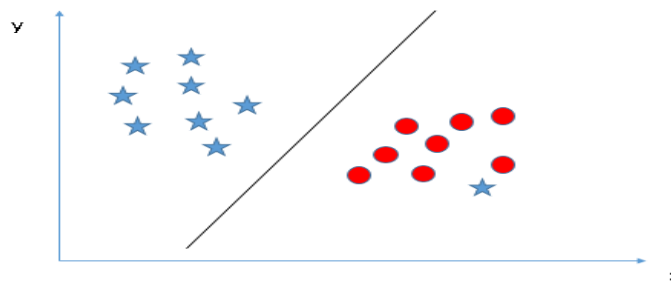


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A**.

- **Can we classify two classes (Scenario-4)?**: Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.

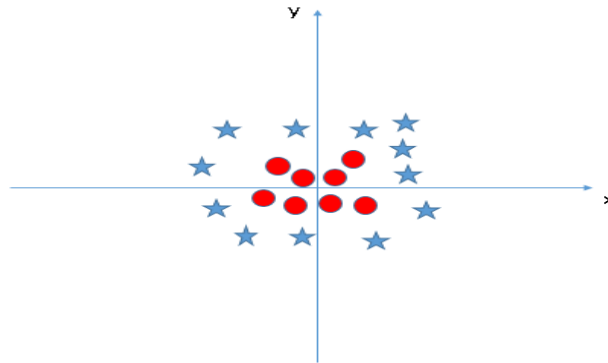


- As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.

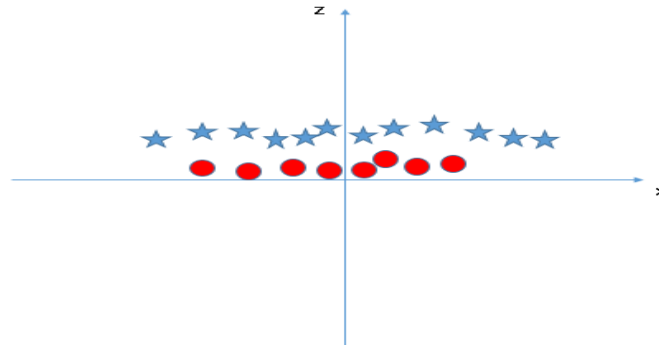


- **Find the hyper-plane to segregate two classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM

- classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z = x^2 + y^2$. Now, let's plot the data points on axis x and z:



In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z .

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the **kernel trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

What is Unsupervised Learning?

Unsupervised learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabelled data.

Unsupervised learning algorithms allows you to perform more complex processing tasks compared to supervised learning. Although, unsupervised learning can be more unpredictable compared with other natural learning methods.

Supervised vs. Unsupervised Machine Learning

Parameters	Supervised machine learning technique	Unsupervised machine learning technique
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data which is not labelled
Computational Complexity	Supervised learning is a simpler method.	Unsupervised learning is computationally complex
Accuracy	Highly accurate and trustworthy method.	Less accurate and trustworthy method.

Applications of unsupervised machine learning

Some applications of unsupervised machine learning techniques are:

- Clustering automatically split the dataset into groups base on their similarities
- Anomaly detection can discover unusual data points in your dataset. It is useful for finding fraudulent transactions
- Association mining identifies sets of items which often occur together in your dataset
- Latent variable models are widely used for data preprocessing. Like reducing the number of features in a dataset or decomposing the dataset into multiple components

Disadvantages of Unsupervised Learning

- You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labeled and not known
- Less accuracy of the results is because the input data is not known and not labeled by people in advance. This means that the machine requires to do this itself.
- The spectral classes do not always correspond to informational classes.
- The user needs to spend time interpreting and label the classes which follow that classification.

- Spectral properties of classes can also change over time so you can't have the same class information while moving from one image to another.

Market Basket Analysis using R and Neural Designer

Our objective is to analyze a dataset from a grocery store to create a recommendation system. This system will be capable of generating accurate recommendations about products that the user may have an interest in.

The type of learning that we use for this example is called "unsupervised learning". It is a type of machine learning technique used to find patterns in data when there are no target values.

In our example the output values will be the percentages that the products have to be in the same shopping basket.

Data

The dataset selected for our example consists of 9835 transactions of a grocery store. Each transaction can be a single product or several products.

From a first look at the database, we cannot know the number of items in the store. Also, the format of the database is difficult to analyze, that is why we need to make a preliminary treatment of the information we have collected before analyzing it.

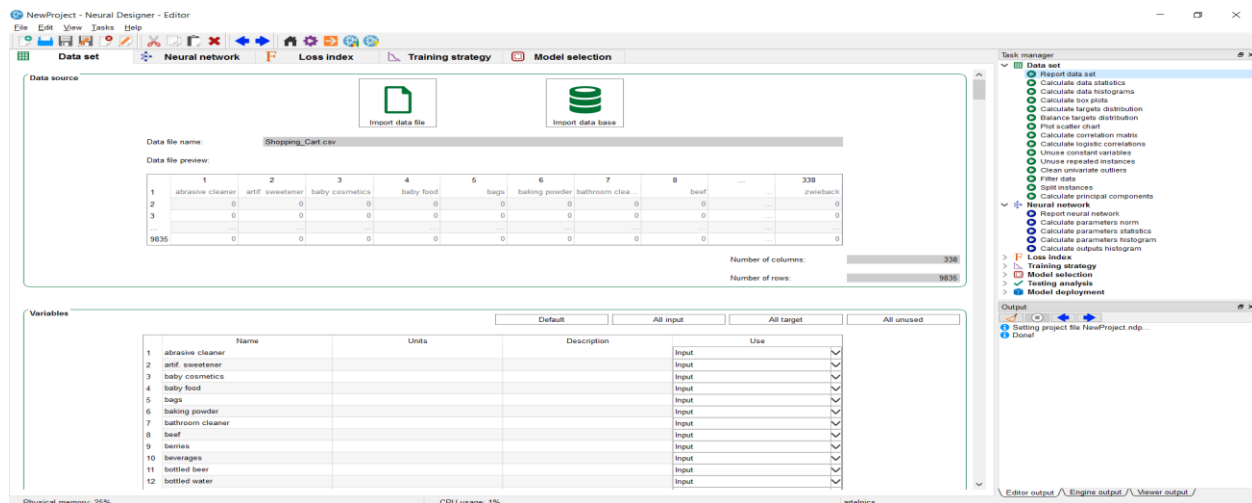
Data preparation using R

As we have said before, it is necessary to pre-process the information that we have in our database and for that purpose we use R.

Using R we convert the database into a binary matrix, which is the format we need to be able to perform the data advanced analysis with Neural Designer. The following script is the one we used to make this change.

Advanced Analytics using Neural Designer

When data has already been pre-processed, it is time to add it to Neural Designer to make our recommendation system. Neural Designer provides an easy way for analyzing and deploying advanced analytics models. The next picture shows the data set tab in Neural Designer.



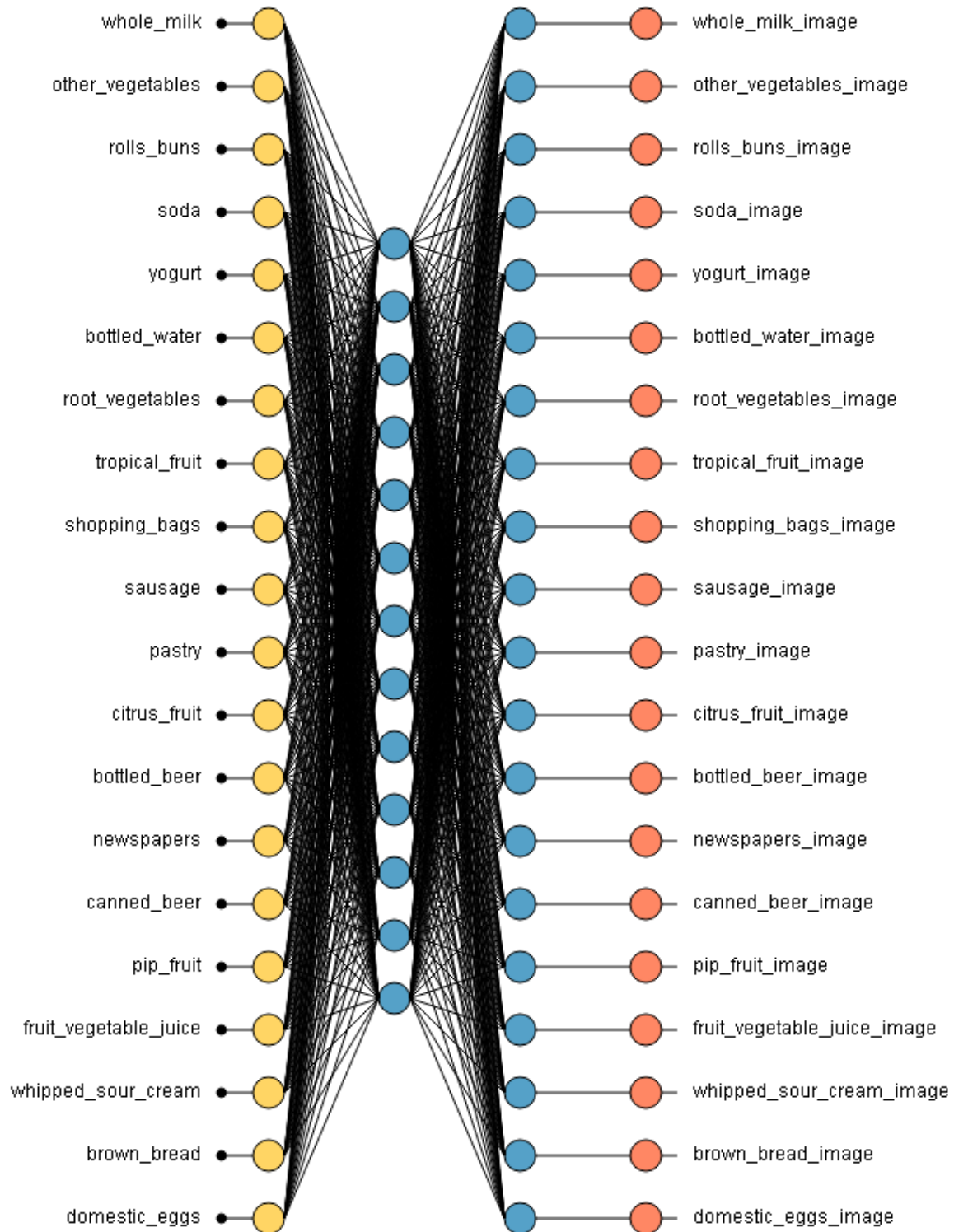
Now the data is loaded into the software, we can check basic statistics of each of the variables. The table below shows the minimums, maximums, means and standard deviations of the top 20 variables in this data set.

	Minimum	Maximum	Mean	Deviation
whole milk	0	1	0.25552	0.43617
other vegetables	0	1	0.19349	0.39506
rolls/buns	0	1	0.18393	0.38745
soda	0	1	0.17438	0.37945
yogurt	0	1	0.1395	0.34649
bottled water	0	1	0.11052	0.31356
root vegetables	0	1	0.109	0.31165
tropical fruit	0	1	0.10493	0.30648
shopping bags	0	1	0.098526	0.29804
sausage	0	1	0.09395	0.29177
pastry	0	1	0.088968	0.28471
citrus fruit	0	1	0.082766	0.27554
bottled beer	0	1	0.080529	0.27212
newspapers	0	1	0.079817	0.27102
canned beer	0	1	0.077682	0.26768
pip fruit	0	1	0.075648	0.26445
fruit/vegetable juice	0	1	0.072293	0.25899
whipped/sour cream	0	1	0.071683	0.25798
brown bread	0	1	0.06487	0.24631
domestic eggs	0	1	0.063447	0.24378

to develop our recommendation system, we use neural networks, the machine learning technique that Neural Designer implements. The neural network defines the predictive model as a multidimensional function containing adjustable parameters. The first step to create our recommendation system is to choose a neural network architecture that represents the classification function.

Because the neuronal network of this problem is very complex, 169 inputs 25 neurons in the hidden layer and 169 outputs, the following image represents what would be a neural network if the study was with the 20 most influence variables.

The next step to carry out is to train the neural network mentioned above. For this purpose we apply Quasi-Newton method to obtain a good model that will recommend us the shopping basket more suitable for customer. To know the types of algorithms that can be used to train a neural network you can read 5 algorithms to train a neural network.



What are Artificial Neural Networks (ANNs)?

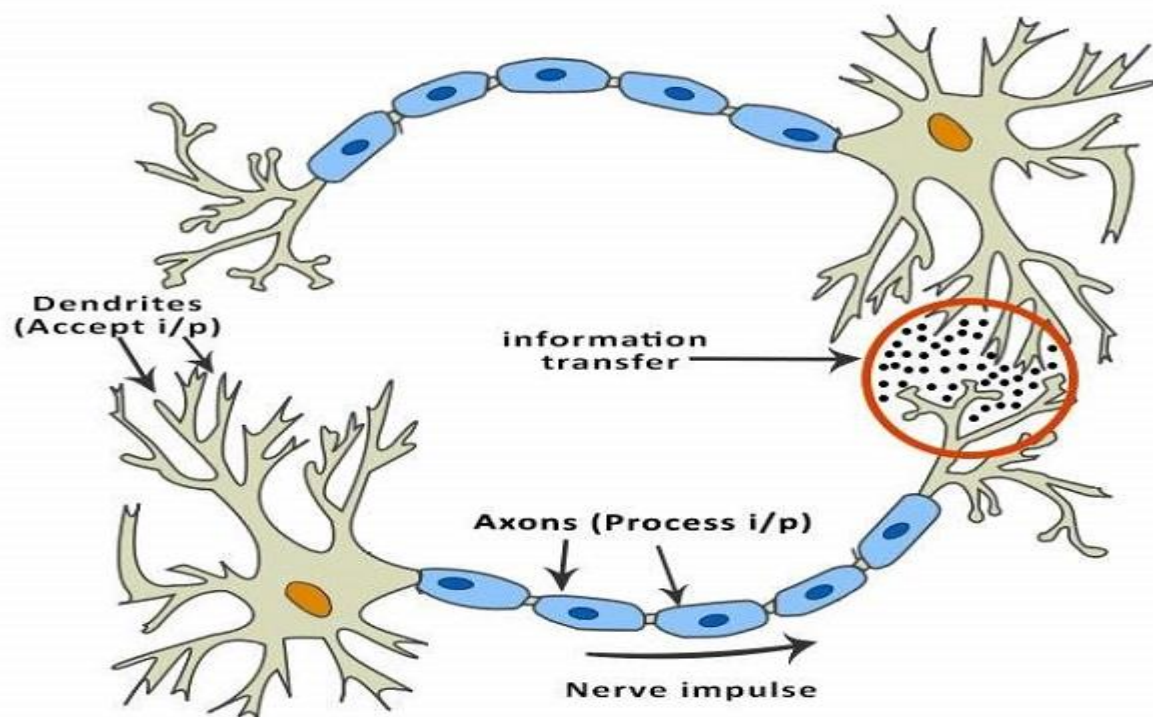
The inventor of the first neurocomputer, Dr. Robert Hecht-Nielsen, defines a neural network as –

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

Basic Structure of ANNs

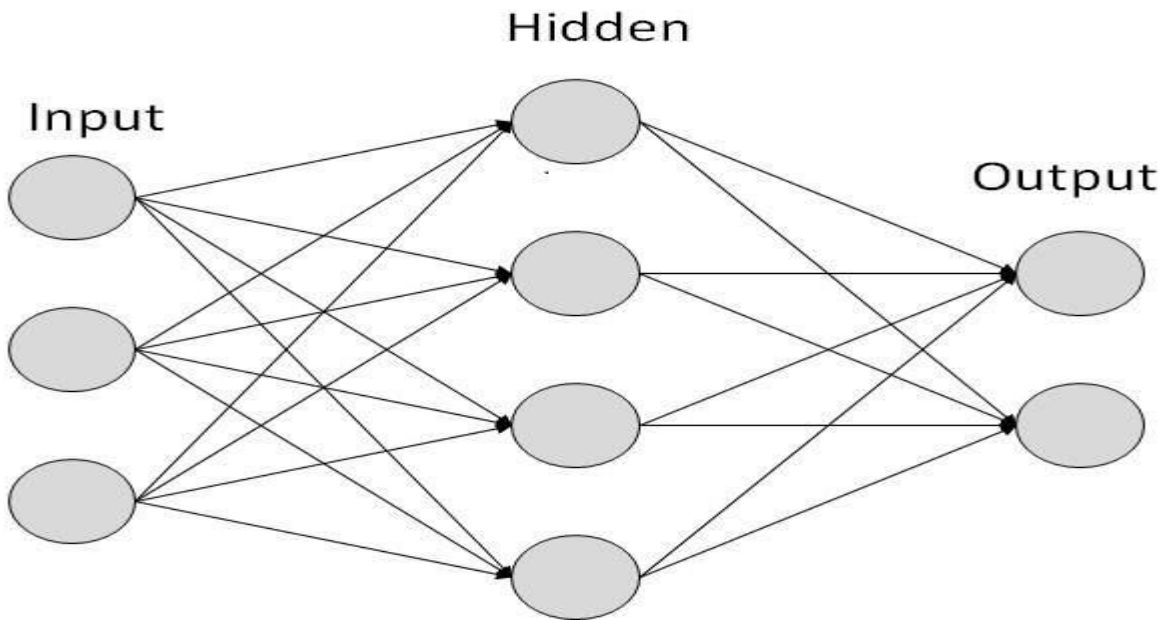
The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living **neurons** and **dendrites**.

The human brain is composed of 86 billion nerve cells called **neurons**. They are connected to other thousand cells by **Axons**. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.



ANNs are composed of multiple **nodes**, which imitate biological **neurons** of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its **activation** or **node value**.

Each link is associated with **weight**. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN –

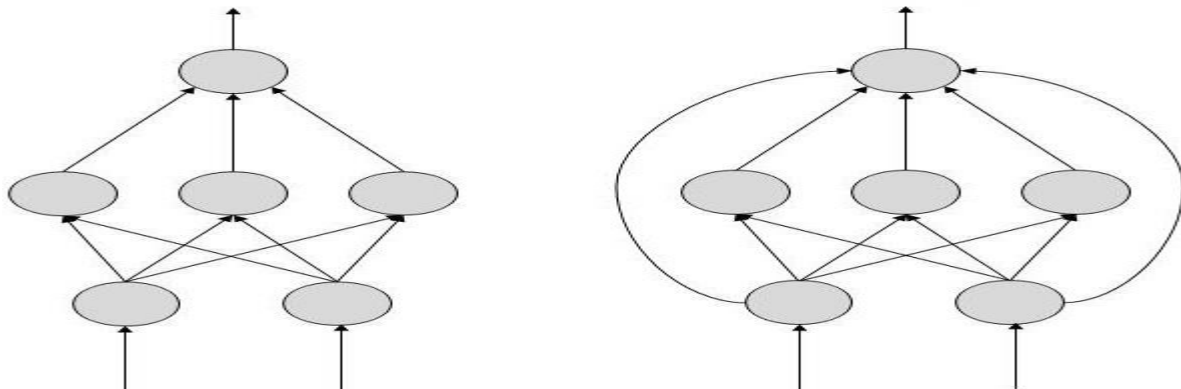


Types of Artificial Neural Networks

There are two Artificial Neural Network topologies – **FeedForward** and **Feedback**.

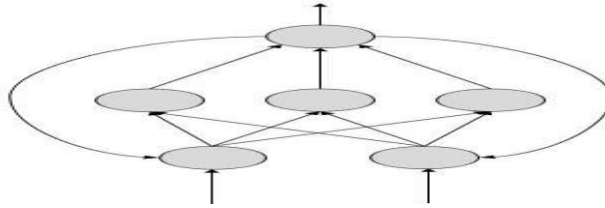
FeedForward ANN

In this ANN, the information flow is unidirectional. A unit sends information to other unit from which it does not receive any information. There are no feedback loops. They are used in pattern generation/recognition/classification. They have fixed inputs and outputs.



FeedBack ANN

Here, feedback loops are allowed. They are used in content addressable memories.



Working of ANNs

In the topology diagrams shown, each arrow represents a connection between two neurons and indicates the pathway for the flow of information. Each connection has a weight, an integer number that controls the signal between the two neurons.

If the network generates a “good or desired” output, there is no need to adjust the weights. However, if the network generates a “poor or undesired” output or an error, then the system alters the weights in order to improve subsequent results.

Machine Learning in ANNs

ANNs are capable of learning and they need to be trained. There are several learning strategies –

- **Supervised Learning** – It involves a teacher that is scholar than the ANN itself. For example, the teacher feeds some example data about which the teacher already knows the answers.

For example, pattern recognizing. The ANN comes up with guesses while recognizing. Then the teacher provides the ANN with the answers. The network then compares its guesses with the teacher’s “correct” answers and makes adjustments according to errors.
- **Unsupervised Learning** – It is required when there is no example data set with known answers. For example, searching for a hidden pattern. In this case, clustering i.e. dividing a set of elements into groups according to some unknown pattern is carried out based on the existing data sets present.
- **Reinforcement Learning** – This strategy built on observation. The ANN makes a decision by observing its environment. If the observation is negative, the network adjusts its weights to be able to make a different required decision the next time.

Back Propagation Algorithm

It is the training or learning algorithm. It learns by example. If you submit to the algorithm the example of what you want the network to do, it changes the network’s weights so that it can produce desired output for a particular input on finishing the training.

Back Propagation networks are ideal for simple Pattern Recognition and Mapping Tasks.

Bayesian Networks (BN)

These are the graphical structures used to represent the probabilistic relationship among a set of random variables. Bayesian networks are also called **Belief Networks** or **Bayes Nets**. BNs reason about uncertain domain.

In these networks, each node represents a random variable with specific propositions. For example, in a medical diagnosis domain, the node Cancer represents the proposition that a patient has cancer.

The edges connecting the nodes represent probabilistic dependencies among those random variables. If out of two nodes, one is affecting the other then they must be directly connected in the directions of the effect. The strength of the relationship between variables is quantified by the probability associated with each node.

There is an only constraint on the arcs in a BN that you cannot return to a node simply by following directed arcs. Hence the BNs are called Directed Acyclic Graphs (DAGs).

BNs are capable of handling multivalued variables simultaneously. The BN variables are composed of two dimensions –

- Range of prepositions
- Probability assigned to each of the prepositions.

Consider a finite set $X = \{X_1, X_2, \dots, X_n\}$ of discrete random variables, where each variable X_i may take values from a finite set, denoted by $Val(X_i)$. If there is a directed link from variable X_i to variable X_j , then variable X_i will be a parent of variable X_j showing direct dependencies between the variables.

The structure of BN is ideal for combining prior knowledge and observed data. BN can be used to learn the causal relationships and understand various problem domains and to predict future events, even in case of missing data.

Building a Bayesian Network

A knowledge engineer can build a Bayesian network. There are a number of steps the knowledge engineer needs to take while building it.

Example problem – Lung cancer. A patient has been suffering from breathlessness. He visits the doctor, suspecting he has lung cancer. The doctor knows that barring lung cancer, there are various other possible diseases the patient might have such as tuberculosis and bronchitis.

Gather Relevant Information of Problem

- Is the patient a smoker? If yes, then high chances of cancer and bronchitis.
- Is the patient exposed to air pollution? If yes, what sort of air pollution?
- Take an X-Ray positive X-ray would indicate either TB or lung cancer.

Identify Interesting Variables

The knowledge engineer tries to answer the questions –

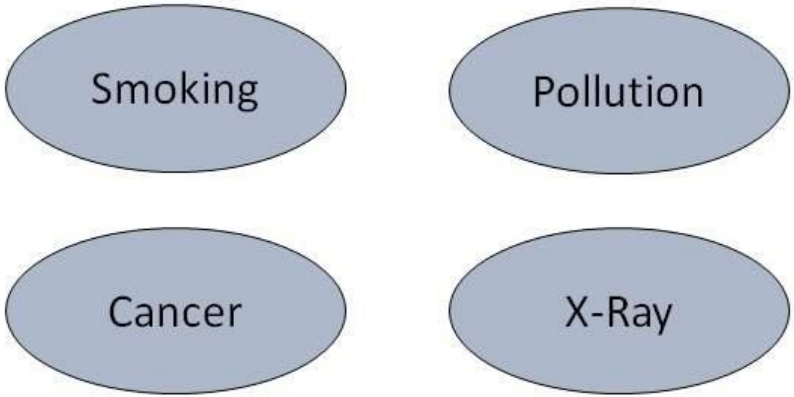
- Which nodes to represent?
- What values can they take? In which state can they be?

For now let us consider nodes, with only discrete values. The variable must take on exactly one of these values at a time.

Common types of discrete nodes are –

- **Boolean nodes** – They represent propositions, taking binary values TRUE (T) and FALSE (F).
- **Ordered values** – A node *Pollution* might represent and take values from {low, medium, high} describing degree of a patient's exposure to pollution.
- **Integral values** – A node called *Age* might represent patient's age with possible values from 1 to 120. Even at this early stage, modeling choices are being made.

Possible nodes and values for the lung cancer example –

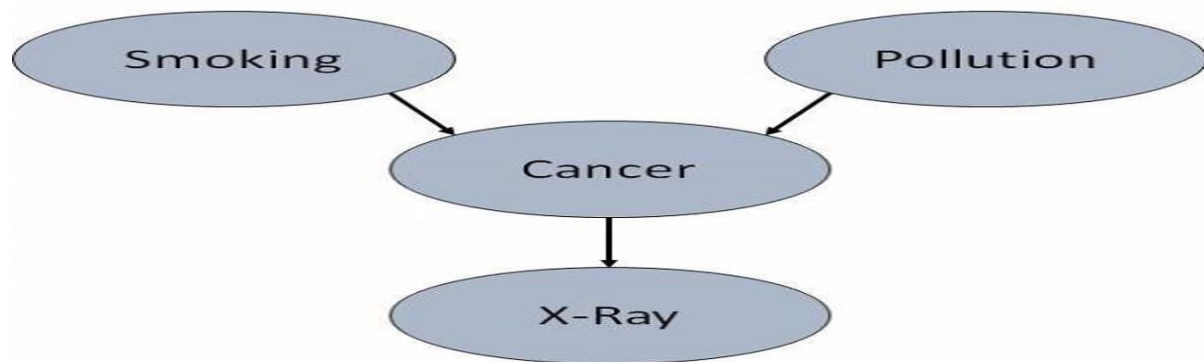
Node Name	Type	Value	Nodes Creation	
Polution	Binary	{LOW, HIGH, MEDIUM}		
Smoker	Boolean	{TRUE, FASLE}		
Lung-Cancer	Boolean	{TRUE, FASLE}		
X-Ray	Binary	{Positive, Negative}		

Create Arcs between Nodes

Topology of the network should capture qualitative relationships between variables.

For example, what causes a patient to have lung cancer? - Pollution and smoking. Then add arcs from node *Pollution* and node *Smoker* to node *Lung-Cancer*.

Similarly if patient has lung cancer, then X-ray result will be positive. Then add arcs from node *Lung-Cancer* to node *X-Ray*.



Specify Topology

Conventionally, BNs are laid out so that the arcs point from top to bottom. The set of parent nodes of a node X is given by $\text{Parents}(X)$.

The *Lung-Cancer* node has two parents (reasons or causes): *Pollution* and *Smoker*, while node *Smoker* is an **ancestor** of node *X-Ray*. Similarly, *X-Ray* is a child (consequence or effects) of node *Lung-Cancer* and **successor** of nodes *Smoker* and *Pollution*.

Conditional Probabilities

Now quantify the relationships between connected nodes: this is done by specifying a conditional probability distribution for each node. As only discrete variables are considered here, this takes the form of a **Conditional Probability Table (CPT)**.

First, for each node we need to look at all the possible combinations of values of those parent nodes. Each such combination is called an **instantiation** of the parent set. For each distinct instantiation of parent node values, we need to specify the probability that the child will take.

For example, the *Lung-Cancer* node's parents are *Pollution* and *Smoking*. They take the possible values = { (H,T), (H,F), (L,T), (L,F)}. The CPT specifies the probability of cancer for each of these cases as <0.05, 0.02, 0.03, 0.001> respectively.

Each node will have conditional probability associated as follows –

Smoking	
$P(S = T)$	
0.30	

Pollution	
$P(P = L)$	
0.90	

Lung-Cancer		
P	S	P (C = T P, S)
H	T	0.05
H	F	0.02
L	T	0.03
L	F	0.001

X-Ray	
C	X = (Pos C)
T	0.90
F	0.20

Applications of Neural Networks

They can perform tasks that are easy for a human but difficult for a machine –

- **Aerospace** – Autopilot aircrafts, aircraft fault detection.
- **Automotive** – Automobile guidance systems.
- **Military** – Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.
- **Electronics** – Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.
- **Financial** – Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.
- **Industrial** – Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modeling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.
- **Medical** – Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
- **Speech** – Speech recognition, speech classification, text to speech conversion.
- **Telecommunications** – Image and data compression, automated information services, real-time spoken language translation.
- **Transportation** – Truck Brake system diagnosis, vehicle scheduling, routing systems.
- **Software** – Pattern Recognition in facial recognition, optical character recognition, etc.

- **Time Series Prediction** – ANNs are used to make predictions on stocks and natural calamities.
- **Signal Processing** – Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.
- **Control** – ANNs are often used to make steering decisions of physical vehicles.
- **Anomaly Detection** – As ANNs are expert at recognizing patterns, they can also be trained to generate an output when something unusual occurs that misfits the pattern.