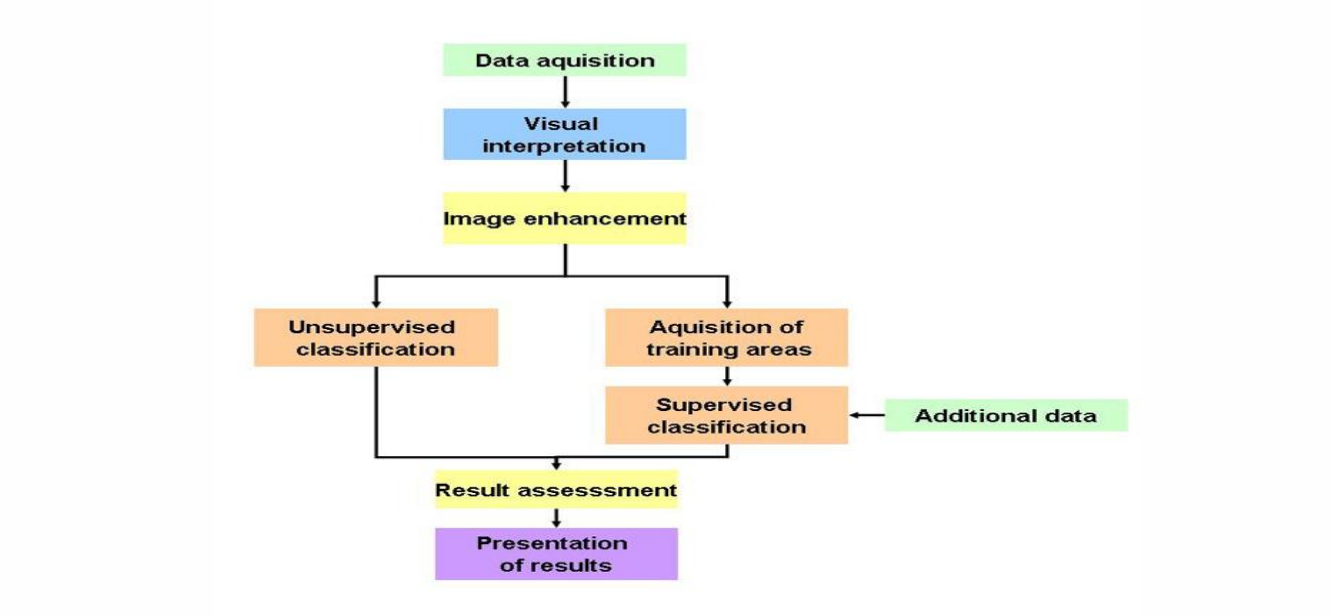
IMAGE CLASSIFICATION

WHAT IS IMAGE CLASSIFICATION?

Image classification refers to a process in computer vision that can classify an image according to its visual content. For example, an image classification algorithm may be designed to tell if an image contains a human figure or not. While detecting an object is trivial for humans, robust image classification is still a challenge in computer vision applications. Image classification is the process of assigning land cover classes to pixels. For example, classes include water, urban, forest, agriculture and grassland.

Image classification refers to the task of extracting information classes from a multiband raster image. The resulting raster from image classification can be used to create thematic maps. Depending on the interaction between the analyst and the computer during classification, there are two types of classification: supervised and unsupervised.

Image classification is assigning pixels in the image to categories or classes of interest Examples: built-up areas, water body, green vegetation, bare soil, rocky areas, cloud, shadow etc. in order to classify a set of data into different classes or categories, the relationship between the data and the classes into which they are classified must be well understood. To achieve this by computer, the computer must be (i) trained Training is key to the success of classification, (ii) Classification techniques were originally developed (iii) Out of research in Pattern Recognition field

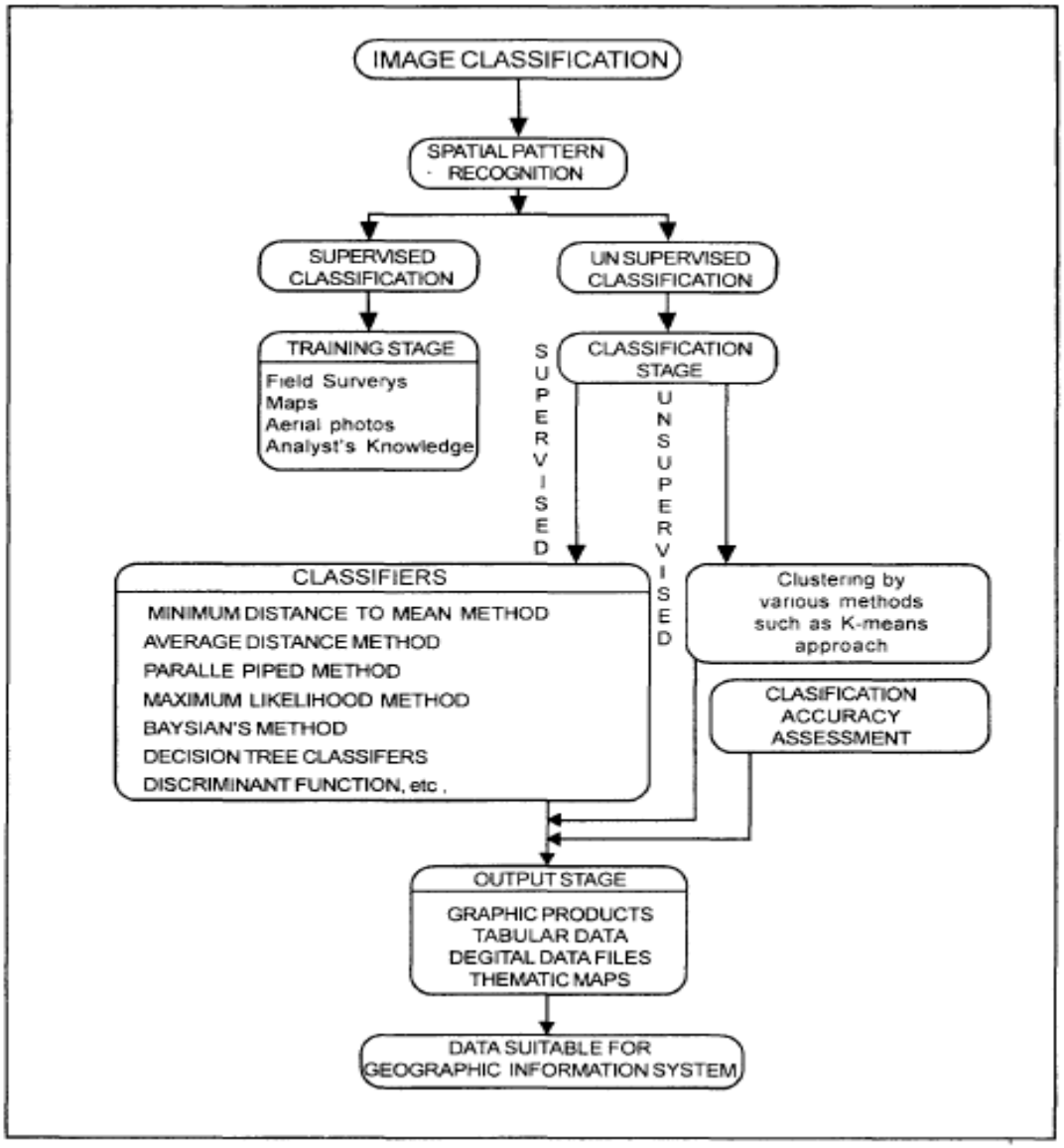


Digital Image Processing and Result Assessment

Computer classification of remotely sensed images involves the process of the computer program learning the relationship between the data and the information classes.

Image classification is a procedure to automatically categorize all pixels in an Image of a terrain into land cover classes. Normally, multispectral data are used to Perform the classification of the spectral

pattern present within the data for each pixel is used as the numerical basis for categorization. This concept is dealt under the Broad subject, namely, Pattern Recognition. Spectral pattern recognition refers to the Family of classification procedures that utilizes this pixel-by-pixel spectral information as the basis for automated land cover classification. Spatial pattern recognition involves the categorization of image pixels on the basis of the spatial relationship with pixels surrounding them. Image classification techniques are grouped into two types, namely supervised and unsupervised. The classification process may also include features, Such as, land surface elevation and the soil type that are not derived from the image.



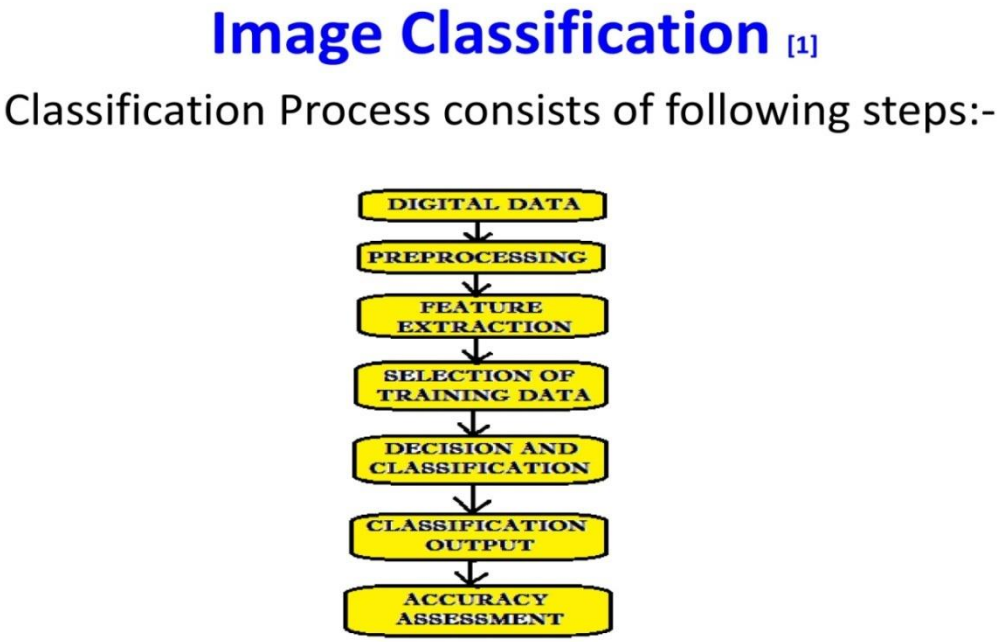
Flow Chart showing Image Classification

With the ArcGIS Spatial Analyst extension, there is a full suite of tools in the Multivariate toolset to perform supervised and unsupervised classification. The classification process is a multi-step workflow; therefore, the Image Classification toolbar has been developed to provide an integrated environment to perform classifications with the tools. Not only does the toolbar help with the workflow for performing unsupervised and supervised classification, it also contains additional functionality for analyzing input data, creating training samples and signature files, and determining the quality of the training samples

and signature files. The recommended way to perform classification and multivariate analysis is through the Image Classification toolbar.

The amount of image data that is received from satellite is constantly increasing. For example, nearly 3 terabytes of data are being sent to Earth by NASA's satellites every day. Advances in satellite technology and computing power have enabled the study of multi-modal, multi-spectral, multi-resolution and multi-temporal data sets for applications such as urban land use monitoring and management, Geographic Information System (GIS) and mapping, environmental change, site suitability, agricultural and ecological studies. Automatic content extraction, classification and contentbased retrieval have become highly desired goals for developing intelligent systems for effective and efficient processing of remotely sensed data sets. Gong and Howarth (1992) discussed the classification of remote sensing data is a complex process and requires consideration of many factors. The user's need, scale of the study area, economic condition, and analyst's skills are important factors influencing the selection of remotely sensed data, the design of the classification procedure and the quality of the classification results. The major steps of image classification may include image preprocessing, feature extraction, selection of training samples, selection of suitable classification approaches, post-classification processing, and accuracy assessment.
Phinn et al (2000) described classification of remote sensing data is used to assign corresponding labels with respect to homogeneous characteristics of groups. The main aim of classification is to discriminate multiple objects from each other within the image. Classification will be executed on the base of spectral or spectrally defined features, such as density, texture etc., in the feature space. It can be said that classification divides the feature space into several classes based on a decision rule. In many cases, classification will be undertaken using a computer, with the use of mathematical classification techniques.

 # STEPS IN IMAGE CLASSIFICATION:



Step 1: Definition of Classification Classes Depending on the objective and the characteristics of the image data, the classification classes should be clearly defined.

Step 2: Selection of Features to discriminate between the classes should be established using multi-spectral or multi-temporal characteristics, colour, textures etc.

Step 3: Sampling of Training Data Training data should be sampled in order to determine appropriate decision rules. Classification techniques such as supervised or unsupervised learning will then be selected on the basis of the training data sets. 1

Step 4: Finding of proper decision rule Various classification techniques will be compared with the training data, so that an appropriate decision rule is selected for subsequent classification.

Step 5: Classification depending upon the decision rule, all pixels are classified in a single class. There are two methods of pixel by pixel classification and per-field classification, with respect to segmented areas.

Step 6: Verification of Results The classified results should be checked and verified for their accuracy and reliability.

IMAGE CLASSIFICATION TECHNIQUES:

The learning algorithms are broadly classified into supervised and unsupervised learning techniques. The distinction is drawn from how the learner classifies data. In supervised learning, the classes are predetermined. These classes can be conceived of as a finite set, previously arrived at by a human. In practice, a certain classes of data will be labeled with these classifications. Althausen (2002) reviewed the classes are then evaluated based on their predictive capacity in relation to measures of variance in the data itself. Some of the examples of supervised classification techniques are Back Propagation Network (BPN), Learning Vector Quantization (LVQ), Self Organizing Map (SOM), Support Vector Machine (SVM), etc., The basic task of unsupervised learning is to develop classification labels automatically. Unsupervised algorithms seek out similarity between pieces of data in order to determine whether that can be characterized as forming a group. These groups are termed clusters. Unsupervised classification, often called as clustering, the system is not informed how the pixels are grouped. The task of clustering is to arrive at some grouping of the data. One of the very common of cluster analysis is K-means clustering.

The intent of the classification process is to categorize all pixels in a digital image into one of several land cover classes, or *"themes"*. This categorized data may then be used to produce thematic maps of the land cover present in an image. Normally, multispectral data are used to perform the classification and, indeed, the spectral pattern present within the data for each pixel is used as the numerical basis for categorization (Lillesand and Kiefer, 1994). The objective of image classification is to identify and portray, as a unique gray level (or color), the features occurring in an image in terms of the object or type of land cover these features actually represent on the ground.

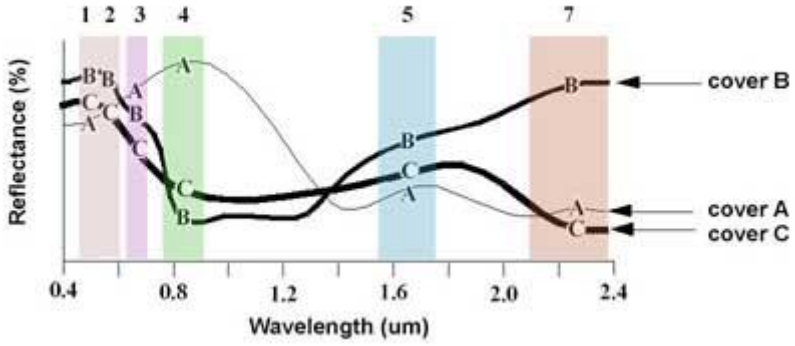


Figure: Spectral Reflectance curve of 3 land covers

Image classification is perhaps the most important part of digital image analysis. It is very nice to have a *"pretty picture"* or an image, showing a magnitude of colors illustrating various features of the underlying terrain, but it is quite useless unless to know what the colors mean. (PCI, 1997). Two main classification methods in GIS are *Supervised Classification* and *Unsupervised Classification.* Two major categories of image classification techniques include **unsupervised** (calculated by software) and **supervised** (human-guided) classification.

The 2 main image classification techniques in remote sensing are:

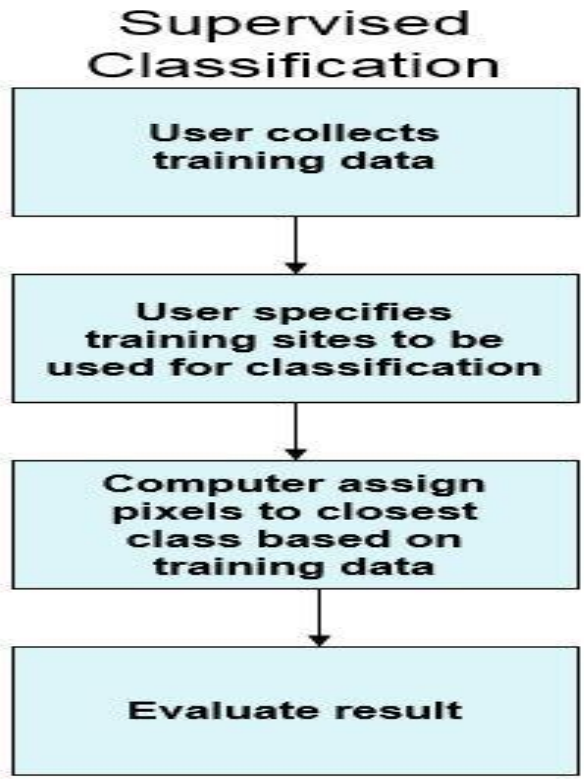
- **Unsupervised image classification**
- **Supervised image classification**

Unsupervised and supervised image classification is the two most common approaches. However, object-based classification has gained more popularity because it's useful for high-resolution data.

1. Supervised classification

Supervised classification uses the spectral signatures obtained from training samples to classify an image. With the assistance of the Image Classification toolbar, you can easily create training samples to represent the classes you want to extract. You can also easily create a signature file from the training samples, which is then used by the multivariate classification tools to classify the image.

The classifier has the advantage of an analyst or domain knowledge using which the classifier can be guided to learn the relationship between the data and the classes. The number of classes, prototype pixels for each class can be identified using this prior knowledge. When prior knowledge is available for some classes, and not for others,⁄ For some dates and not for others in a multitemporal⁄ dataset, Combination of supervised and unsupervised methods can be employed for partially supervised classification of images.



Supervised classification is based on the idea that a user can select sample pixels in an image that are representative of specific classes and then direct the image processing software to use these training sites as references for the classification of all other pixels in the image. Training sites (also known as testing sets or input classes) are selected based on the knowledge of the user. The user also sets the bounds for how similar other pixels must be to group them together. These bounds are often set based on the spectral characteristics of the training area, plus or minus a certain increment (often based on "brightness" or strength of reflection in specific spectral bands). The user also designates the number of classes that the image is classified into. Many analysts use a combination of supervised and unsupervised classification processes to develop final output analysis and classified maps. In supervised classification the user or image analyst "supervises" the pixel classification process. The user specifies the various pixels values or spectral signatures that should be associated with each class. This is done by selecting representative sample sites of a known cover type called **Training Sites or Areas**. The computer algorithm then uses the spectral signatures from these training areas to classify the whole image. Ideally, the classes should not overlap or should only minimally overlap with other classes.

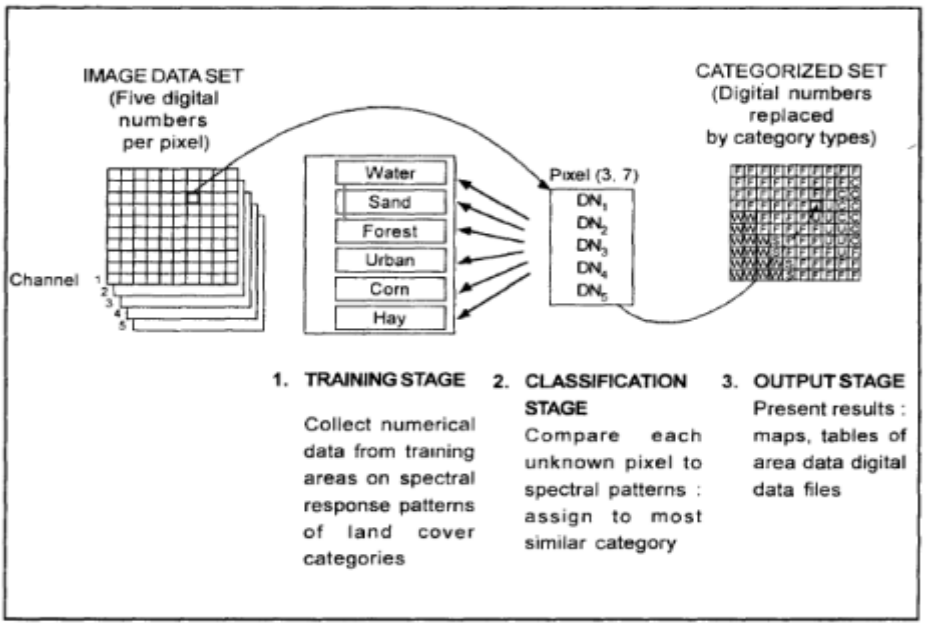
Huang et al (2009) presented supervised classification identifies the specific area of interest i.e., water body and non-water body in the image. The supervised classification is much more accurate for mapping classes, but depends heavily on the cognition and skills of the image specialist. The strategy is simple: the specialist must recognize conventional classes or meaningful classes in a scene from prior knowledge, such as personal experience with what's present in the scene, or more generally, the region it's located in, by experience with thematic maps, or by on-site visits. This familiarity allows the individual making the classification to choose and set up discrete classes and then, assign them category names. As a rule, the classifying person also locates specific training sites on the image to identify the classes. The resulting training sites are areas representing each known land cover category that appear fairly homogeneous on the image. For each class thus outlined, mean values and variances

of the each band used to classify them are calculated from all the pixels enclosed in each site. More than one polygon is usually drawn for any class. The classification program then acts to classify the data representing each class. When the classification for a class is plotted as a function of the band the result is a spectral signature or spectral response curve for that class. The multiple spectral signatures so obtained are for all of the materials within the site that interact with the incoming radiation.

Classification now proceeds by statistical processing in which every pixel is compared with the various signatures and assigned to the class whose signature comes closest. Many of the classes for the satellite images are almost self-evident in portraying ocean water, waves, beach, marsh, shadows. For example, difference between ocean and bay waters, but their gross similarities in spectral properties would probably make separation difficult. Some classes are broad-based, representing two or more related surface materials that might be separable at high resolution but are inexactly expressed in the image. Thus, the supervised classification has an edge over the unsupervised methodology. Learning Vector Quantization (LVQ) and the Support Vector Machines (SVM) are the two supervised classification methodologies implemented to perform the analysis for the water body and non-water body classification.

Steps involved in Supervised Classification:

A supervised classification algorithm requires a training sample for each class, that is, a collection of data points known to have come from the class of interest. The classification is thus based on how "close" a point to be classified is to each training sample. We shall not attempt to define the word "close" other than to say that both Geometric and statistical distance measures are used in practical pattern recognition algorithms. The training samples are representative of the known classes of interest to the analyst. Classification methods that relay on use of training patterns are called supervised classification methods. In supervised classification, you select representative samples for each land cover class. The software then uses these "training sites" and applies them to the entire image. The three basic steps involved in a typical supervised classification procedure are as follows:

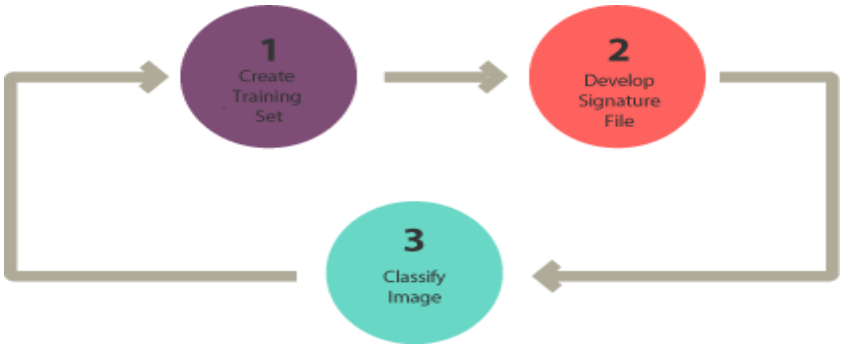


Basic steps of supervised classification

(i) Training stage: The analyst identifies representative training areas and develops numerical descriptions of the spectral signatures of each land cover type of interest in the scene. Training sites are areas that are known to be representative of a particular land cover type. The computer determines the spectral signature of the pixels within each training area, and uses this information to define the statistics, including the mean and variance of each of the classes. Preferably the location of the training sites should be based on field collected data or high resolution reference imagery. It is important to choose training sites that cover the full range of variability within each class to allow the software to accurately classify the rest of the image. If the training areas are not representative of the range of variability found within a particular land cover type, the classification may be much less accurate. Multiple, small training sites should be selected for each class. The more time and effort spent in collecting and selecting training site the better the classification results.

(ii) The classification stag (Decision Rule) or Generate signature file: Each pixel in the image data set IS categorized into the land cover class it most closely resembles. If the pixel is insufficiently similar to any training data set it is usually labeled 'Unknown'.

(iii) The output stage or Classify: The results may be used in a number of different ways. Three typical forms of output products are thematic maps, tables and digital data files which become input data for GIS. The output of image classification becomes input for GIS for spatial analysis of the terrain. Fig. 2 depicts the flow of operations to be performed during image classification of remotely sensed data of an area which ultimately leads to create database as an input for GIS. Plate 6 shows the land use/ land cover color coded image, which is an output of image



For supervised image classification, you first create training samples. For example, you mark urban areas by marking them in the image. Then, you would continue adding training sites representative in the entire image.

For each land over class, you continue creating training samples until you have representative samples for each class. In turn, this would generate a signature file, which stores all training samples spectral information.

Supervised Classification Principles:

The classifier learns the characteristics of different thematic classes – forest, marshy vegetation, agricultural land, turbid water, clear water, open soils, manmade objects, desert etc. This happens by means of analyzing the statistics of a small sets of pixels in each class that are reliably selected by a human analyst through experience or with the help of a map of the area

With supervised classification, we identify examples of the Information classes (i.e., land cover type) of interest in the image. These are called *"training sites"*. The image processing software system is then used to develop a statistical characterization of the reflectance for each information class. This stage is often called "*signature analysis"* and may involve developing a characterization as simple as the mean or the rage of reflectance on each bands, or as complex as detailed analyses of the mean, variances and covariance over all bands. Once a statistical characterization has been achieved for each information class, the image is then classified by examining the reflectance for each pixel and making a decision about which of the signatures it resembles most. (Eastman, 1995)

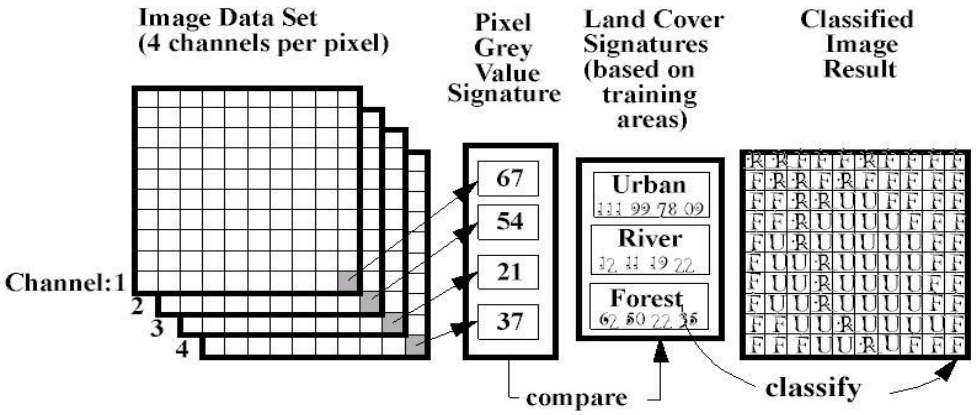


Figure: Steps in Supervised classification

Supervised Classification Algorithms:

Finally, the last step would be to use the signature file to run a classification. From here, you would have to pick a classification algorithms such as:

- Maximum likelihood
- Minimum-distance
- Principal components
- Support vector machine (SVM)
- Iso cluster
- Artificial Neural Networks (ANN)
- Parallel piped
- **Mahalanobis Distance**

As shown in several studies, **SVM is one of the best classification algorithms** in remote sensing. But each option has its own advantages, which you can test for yourself.

(i) Maximum likelihood Classification

Maximum likelihood Classification is a statistical decision criterion to assist in the classification of overlapping signatures; pixels are assigned to the class of highest probability. Assumes that the statistics for each class in each band are normally distributed and calculates the probability that a given pixel belongs to a specific class. Each pixel is assigned to the class that has the highest probability (that is, the maximum likelihood). This is the default.

The maximum likelihood classifier is considered to give more accurate results than parallelepiped classification however it is much slower due to extra computations. We put the word *`accurate'* in quotes because this assumes that classes in the input data have a Gaussian distribution and that signatures were well selected; this is not always a safe assumption.

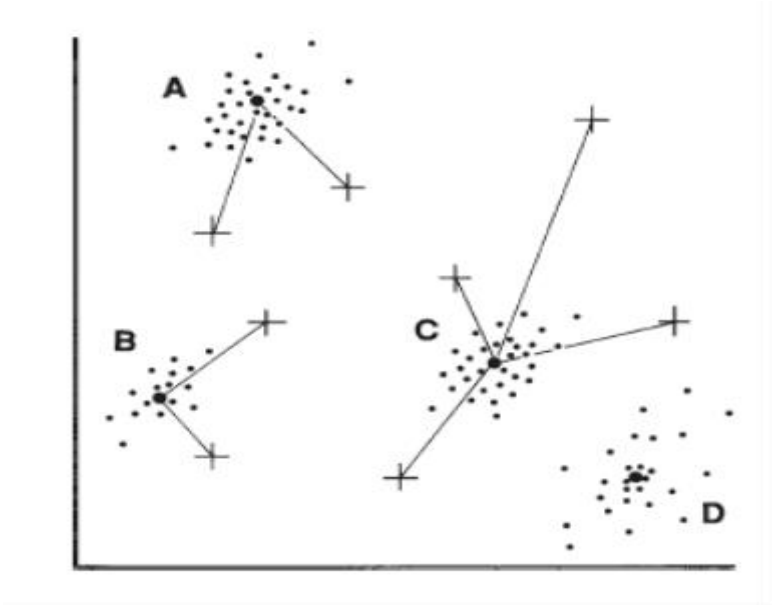
In nature the classes that we classify exhibit natural variation in their spectral patterns. Further variability is added by the effects of haze, topographic shadowing, system noise, and the effects of mixed pixels. As a result, remote sensing images seldom record spectrally pure classes; more typically, they display a range of brightness's in each band. The classification strategies considered thus far do not consider variation that may be present within spectral categories and do not address problems that arise when frequency distributions of spectral values from separate categories overlap. The maximum likelihood (ML) procedure is the most common supervised method used with remote sensing. It can be described as a statistical approach to pattern recognition where the probability of a pixel belonging to each of a predefined set of classes is calculated; hence the pixel is assigned to the class with the highest probability MLC is based on the Bayesian probability formula.

(ii) Minimum distance Classification

Minimum distance classifies image data on a database file using a set of 256 possible class signature segments as specified by signature parameter. Each segment specified in signature, for example, stores signature data pertaining to a particular class. Only the mean vector in each class signature segment is used. Other data, such as standard deviations and covariance matrices, are ignored (though the maximum likelihood classifier uses this). Uses the mean vectors for each class and calculates the

Euclidean distance from each unknown pixel to the mean vector for each class. The pixels are classified to the nearest class.

Minimum Distance Classification for supervised classification, these groups are formed by values of pixels within the training fields defined by the analyst.Each cluster can be represented by its centroid, often defined as its mean value. As unassigned pixels are considered for assignment to one of the several classes, the multidimensional distance to each cluster centroid is calculated, and the pixel is then assigned to the closest cluster. Thus the classification proceeds by always using the "minimum distance" from a given pixel to a cluster centroid defined by the training data as the spectral manifestation of an informational class. Minimum distance classifiers are direct in concept and in implementation but are not widely used in remote sensing work. In its simplest form, minimum distance classification is not always accurate; there is no provision for accommodating differences in variability of classes, and some classes may overlap at their edges. It is possible to devise more sophisticated versions of the basic approach just outlined by using different distance measures and different methods of defining cluster centroids.



Minimum distance classifier

The result of the classification is a theme map directed to a specified database image channel. A theme map encodes each class with a unique gray level. The gray-level value used to encode a class is specified when the class signature is created. If the theme map is later transferred to the display, then a pseudo-color table should be loaded so that each class is represented by a different color.

(iii) Mahalanobis Distance: A direction-sensitive distance classifier that uses statistics for each class. It is similar to maximum likelihood classification, but it assumes all class covariances are equal, and therefore is a faster method. All pixels are classified to the closest training data.

Mahalanobis Distance is similar to Minimum Distance, except that the covariance matrix is used in the equation. Mahalanobis distance is a well-known statistical distance function. Here, a measure of variability can be incorporated into the distance metric directly. Mahalanobis distance is a distance measure between two points in the space defined by two or more correlated variables. That is to say, Mahalanobis distance takes the correlations within a data set between the variable into consideration. If there are two non-correlated variables, the Mahalanobis distance between the points of the variable in a

2D scatter plot is same as Euclidean distance. In mathematical terms, the Mahalanobis distance is equal to the Euclidean distance when the covariance matrix is the unit matrix. This is exactly the case then if the two columns of the standardized data matrix are orthogonal. The Mahalanobis distance depends on the covariance matrix of the attribute and adequately accounts for the correlations. Here, the covariance matrix is utilized to correct the effects of cross-covariance between two components of random variable.

$D=(X-M_c)^T (COV_c)^{-1}(X-M_c)$ (2)

where

D = Mahalanobis Distance, c = a particular class, X = measurement vector of the candidate pixel Mc = mean vector of the signature of class c, Covc = covariance matrix of the pixels in the signature of class c, Covc-1 = inverse of Covc, T = transposition function.

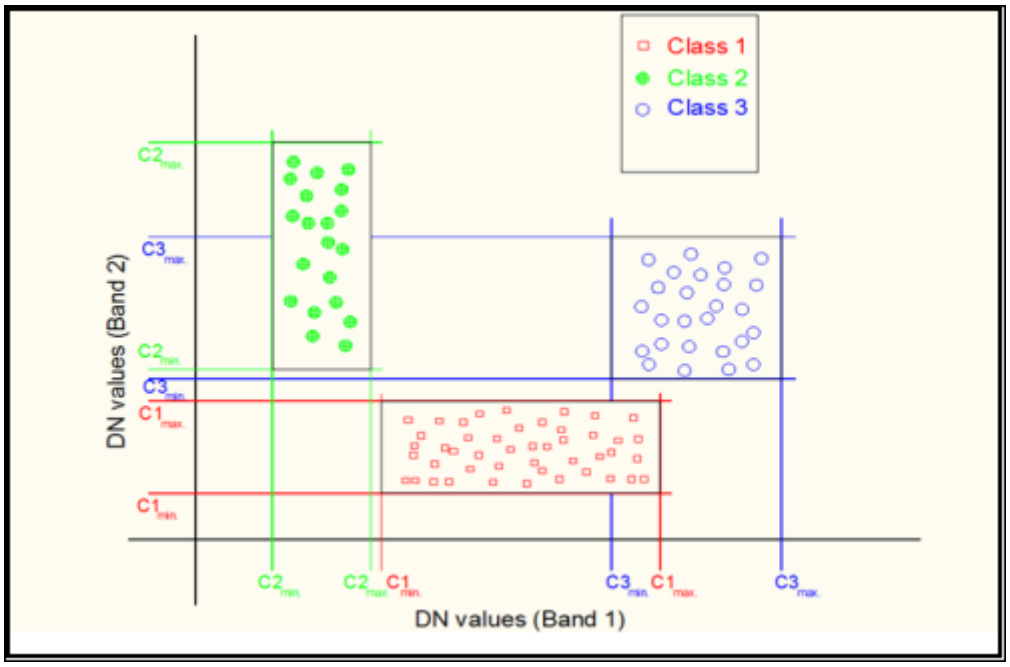
(iv) Parallelepiped Classification

The parallelepiped classifier uses the class limits and stored in each class signature to determine if a given pixel falls within the class or not. The class limits specify the dimensions (in standard deviation units) of each side of a parallelepiped surrounding the mean of the class in feature space.

If the pixel falls inside the parallelepiped, it is assigned to the class. However, if the pixel falls within more than one class, it is put in the overlap class (code 255). If the pixel does not fall inside any class, it is assigned to the null class (code 0).

The parallelepiped classifier is typically used when speed is required. The drawback is (in many cases) poor accuracy and a large number of pixels classified as ties (or overlap, class 255).

Parallelepiped classification, sometimes also known as box decision rule, or level-slice procedures, are based on the ranges of values within the training data to define regions within a multidimensional data space. The spectral values of unclassified pixels are projected into data space; those that fall within the regions defined by the training data are assigned to the appropriate categories. In this method a parallelepiped-like (i.e., hyper-rectangle) subspace is defined for each class. Using the training data for each class the limits of the parallelepiped subspace can be defined either by the minimum and maximum pixel values in the given class, or by a certain number of standard deviations on either side of the mean of the training data for the given class. The pixels lying inside the parallelepipeds are tagged to this class. Figure depicts this criterion in cases of two-dimensional feature space.



Implementation of the parallelepiped classification method for three classes using two spectral bands

Comparison of supervised classification techniques:

One of the most important keys to classify land use or land cover using suitable techniques the table showed advantages and disadvantages of each techniques

techniques	advantage	disadvantage
Parallelepiped	Fast and simple, calculations are made, thus cutting processing Not dependent on normal distributions.	Since parallelepipeds have corners, pixels that are actually quite far, spectrally, from the mean of the signature may be classified
Minimum Distance Classification	Since every pixel is spectrally closer to either one sample mean or another, there are no unclassified pixels. Fastest decision rule to compute, except for parallelepiped	Pixels that should be unclassified,, this problem is alleviated by thresholding out the pixels that are farthest from the means of their classes. Does not consider class variability
Mahalanobis Distance	Takes the variability of classes into account, unlike Minimum Distance or Parallelepiped	Tends to overclassify signatures with relatively large values in the covariance matrix. Slower to compute than Parallelepiped or Minimum Distance
Maximum Likelihood	Most accurate of the classifiers In classification. Takes the variability of classes into account by using the covariance matrix, as does Mahalanobis Distance	An extensive equation that takes a long time to compute Maximum Likelihood is parametric, meaning that it relies heavily on anormal distribution of the data in each input band

Advantages and Disadvantages of Supervised Classification:

In supervised classification the majority of the effort is done prior to the actual classification process. Once the classification is run the output is a thematic image with classes that are labeled and correspond to information classes or land cover types. Supervised classification can be much more accurate than unsupervised classification, but depends heavily on the training sites, the skill of the individual processing the image, and the spectral distinctness of the classes. If two or more classes are very similar to each other in terms of their spectral reflectance (e.g., annual-dominated grasslands vs. perennial grasslands), mis-classifications will tend to be high. Supervised classification requires close attention to the development of training data. If the training data is poor or not representative the classification results will also be poor. Therefore supervised classification generally requires more times and money compared to unsupervised.

2. UNSUPERVISED CLASSIFICATION:

Unsupervised classification is where the outcomes (groupings of pixels with common characteristics) are based on the software analysis of an image without the user providing sample classes. The computer uses techniques to determine which pixels are related and groups them into classes. The user can specify which algorism the software will use and the desired number of output classes but otherwise does not aid in the classification process. However, the user must have knowledge of the area being classified when the groupings of pixels with common characteristics produced by the computer have to be related to actual features on the ground (such as wetlands, developed areas, coniferous forests, etc.).

The steps for running an unsupervised classification are:

1. Generate clusters
2. Assign classes

Generate clusters

In this step, the software clusters pixels into a set number of classes. So, the first step is to assign the number of classes you want it to generate. In addition, you have to identify which bands you want it to use.

If you're using Landsat, here is a list of **Landsat bands**. For Sentinel, here are **Sentinel-2 bands**. We also have a **handy guide on spectral signatures** which explains which spectral bands are useful for classifying different classes.

In ArcGIS, the steps for generating clusters are:

- First, you have to activate the spatial analyst extension (Customize ‣ Extensions ‣ Spatial Analyst).
- In this unsupervised classification example, we use Iso-clusters (Spatial Analysis Tools ‣ Multivariate ‣ Iso clusters).

INPUT: The image you want to classify.

NUMBER OF CLASSES: The number of classes you want to generate during the unsupervised classification. For example, if you are working with **multispectral imagery** (red, green, blue and NIR bands), then the number here will be 40 (4 classes x 10).

MINIMUM CLASS SIZE: This is the number of pixels to make a unique class.

When you click OK, it creates clusters based on your input parameters. But you still need identify which land cover classes each cluster belongs to.

Assign classes

Now that you have clusters, the last step is to identify each class from the iso-clusters output. Here are some tips to make this step easier:

- In general, it helps to select colours for each class. For example, set water as blue for each class.
- After setting each one of your classes, we can merge the classes by using the reclassify tool.

If land cover appears in 2 classes, you will need to make some manual edits. For example, if vegetation was mistakenly classified as water (perhaps algae in the water), you will have to manually edit the polygon.

In most cases, it helps to convert the raster to vector and use the editing toolbar. You can split polygons to help properly identify them.

The unsupervised clustering is a kind of clustering which takes place with minimum input from the operator; no training sample is available and part of the feature space is achieved by identifying natural groupings of the given data. In unsupervised clustering technique, an individual pixel is compared to each cluster to see the closest pixel. Finally, a map of all pixels in the image, classified as to different clusters, each pixel is most likely to belong, is produced. This then must be interpreted by the user as to what the colour patterns may mean in terms of classes, etc. that are actually present in the real world scene; this requires some knowledge of the scene's feature or cluster content from general experience or personal familiarity with the area imaged. The objective here is to group multi-band spectral response patterns into clusters that are statistically separable. The cluster numbers are initially set and the total number can be varied arbitrarily. Generally, in an area within a SAR image, multiple pixels in the same cluster correspond to some ground feature or cluster so that patterns of gray levels result in a new image depicting the spatial distribution of the clusters. These levels can then be assigned to produce a cluster map. This can be done by either being adequately familiar with the major classes expected in the scene, or, where feasible, by visiting ground truth and visually correlating map patterns to their ground counterparts. Since the classes are not selected beforehand, this method is termed as unsupervised classification.

Unsupervised Classification Techniques:

There are three unsupervised classification techniques namely K-means clustering, PCA based K-

means clustering and Fuzzy C-Means clustering (FCM).

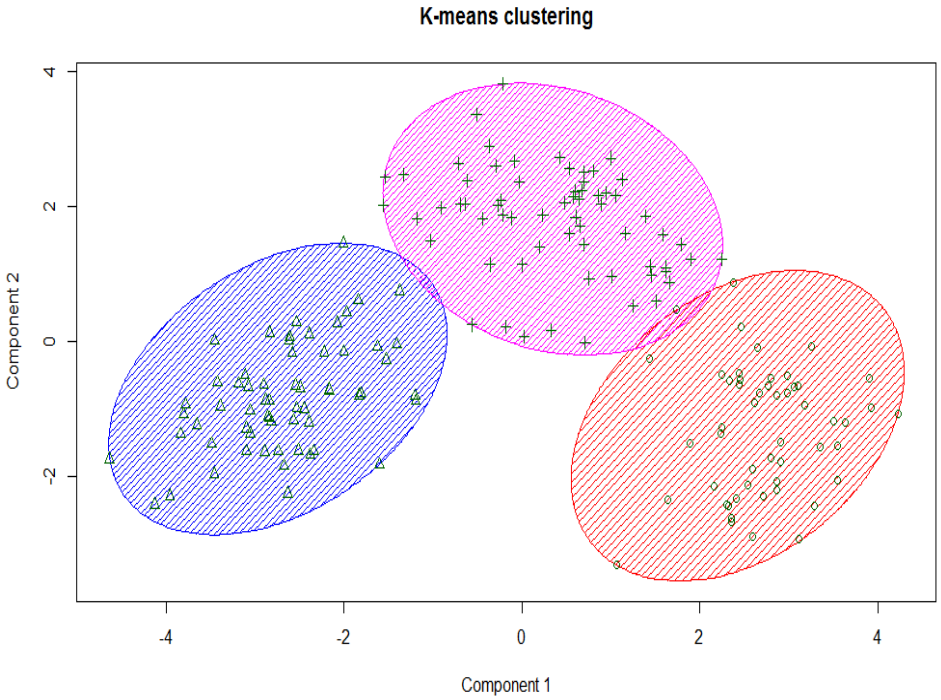
K- means clustering: K-means algorithm is an iterative algorithm that tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way kmeans algorithm works is as follows:

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The approach kmeans follows to solve the problem is called **Expectation-Maximization**. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. Below is a breakdown of how we can solve it mathematically



Unsupervised classification finds spectral classes (or clusters) in a multiband image without the analyst's intervention. The Image Classification toolbar aids in unsupervised classification by providing access to the tools to create the clusters, capability to analyze the quality of the clusters, and access to classification tools. Unsupervised classification is a method which examines a large number of unknown pixels and divides into a number of classed based on natural groupings present in the image values. Unlike supervised classification, unsupervised classification does not require analyst-specified training data. The basic premise is that values within a given cover type should be close together in the measurement space (i.e. have similar gray levels), whereas data in different classes should be comparatively well separated (i.e. have very different gray levels) (PCI, 1997; Lillesand and Kiefer, 1994; Eastman, 1995)

The classes that result from unsupervised classification are spectral classed which based on natural groupings of the image values, the identity of the spectral class will not be initially known, must compare classified data to some form of reference data (such as larger scale imagery, maps, or site visits) to determine the identity and informational values of the spectral classes. Thus, in the supervised approach, to define useful information categories and then examine their spectral separability; in the unsupervised approach the computer determines spectrally separable class, and then define their information value. (PCI, 1997; Lillesand and Kiefer, 1994)

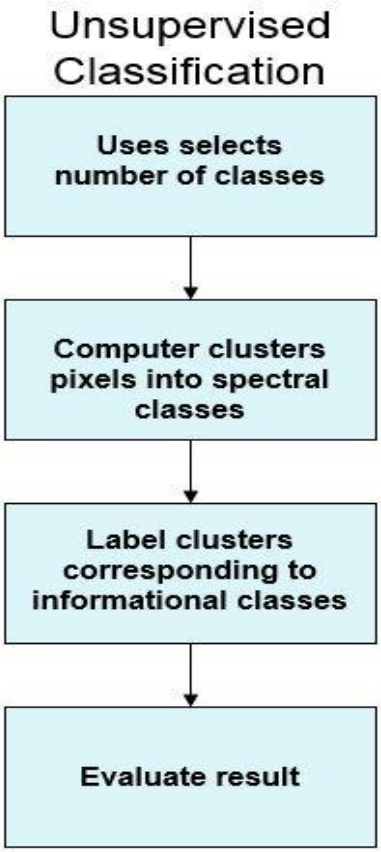
Unsupervised classification is becoming increasingly popular in agencies involved in long term GIS database maintenance. The reason is that there are now systems that use clustering procedures that are extremely fast and require little in the nature of operational parameters. Thus it is becoming possible to train GIS analysis with only a general familiarity with remote sensing to undertake classifications that meet typical map accuracy standards. With suitable ground truth accuracy assessment procedures, this tool can provide a remarkably rapid means of producing quality land cover data on a continuing basis.

When access to domain knowledge or the experience of an analyst is missing, the data can still be analyzed by numerical exploration, whereby the data are grouped into subsets or clusters based on

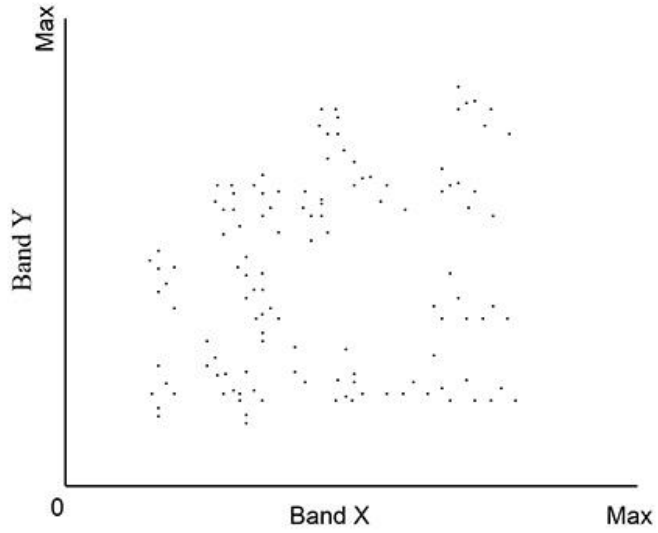
statistical similarity

Using the Image Classification toolbar and Training Sample Manager, it was determined the training samples were representative for the area and statistically separate. Therefore, a maximum likelihood classification was performed from the toolbar.

Steps of Unsupervised Classification:



Unsupervised classification is a form of pixel based classification and is essentially computer automated classification. The user specifies the number of classes and the spectral classes are created solely based on the numerical information in the data (i.e. the pixel values for each of the bands or indices). Clustering algorithms are used to determine the natural, statistical grouping of the data. The pixels are grouped together into based on their spectral similarity. The computer uses feature space to analyze and group the data into classes. Roll over the below image to see how the computer might use feature space to group the data into ten classes.



While the process is basically automated, the user has control over certain inputs. This includes the Number of Classes, the Maximum Iterations, (which is how many times the classification algorithm runs) and the Change Threshold %, which specifies when to end the classification procedure. After the data has been classified the user has to interpret, label and color code the classes accordingly.

Advantages of Unsupervised Classification:

Unsupervised classification is fairly quick and easy to run. There is no extensive prior knowledge of area required, but you must be able to identify and label classes after the classification. The classes are created purely based on spectral information; therefore they are not as subjective as manual visual interpretation.

Disadvantages of Unsupervised Classification:

One of the disadvantages is that the spectral classes do not always correspond to informational classes. The user also has to spend time interpreting and label the classes following the classification. Spectral properties of classes can also change over time, so you can't always use the same class information when moving from one image to another.

Supervised classification generally performs better than unsupervised classification IF good quality training data is available unsupervised classifiers are used to carry out preliminary analysis of data prior to supervised classification

frontiers in Big Data

REVIEW
published: 03 April 2019
doi: 10.3389/fdata.2019.00002

Deep Representation Learning for Social Network Analysis

*Qiaoyu Tan, Ninghao Liu and Xia Hu**

Department of Computer Science and Engineering, Texas A&M University, College Station, TX, United States

Social network analysis is an important problem in data mining. A fundamental step for analyzing social networks is to encode network data into low-dimensional representations, i.e., network embeddings, so that the network topology structure and other attribute information can be effectively preserved. Network representation leaning facilitates further applications such as classification, link prediction, anomaly detection, and clustering. In addition, techniques based on deep neural networks have attracted great interests over the past a few years. In this survey, we conduct a comprehensive review of the current literature in network representation learning, utilizing neural network models. First, we introduce the basic models for learning node representations in homogeneous networks. We will also introduce some extensions of the base models, tackling more complex scenarios such as analyzing attributed networks, heterogeneous networks, and dynamic networks. We then introduce techniques for embedding subgraphs and also present the applications of network representation learning. Finally, we discuss some promising research directions for future work.

Keywords: deep learning, social networks, deep social network analysis, representation learning, network embedding

OPEN ACCESS

Edited by:
Yuxiao Dong,
Microsoft Research, United States

Reviewed by:
Chuan-Ju Wang,
Academia Sinica, Taiwan
Donghui Hu,
Hefei University of Technology, China

****Correspondence:***
Xia Hu
xiahu@tamu.edu

Specialty section:
This article was submitted to Data Mining and Management, a section of the journal Frontiers in Big Data

Received: *03 December 2018*
Accepted: *12 March 2019*
Published: *03 April 2019*

Citation:
Tan Q, Liu N and Hu X (2019) Deep Representation Learning for Social Network Analysis. Front. Big Data 2:2. doi: 10.3389/fdata.2019.00002

1. INTRODUCTION

Social networks, such as Facebook, Twitter, and LinkedIn, have greatly facilitated communication between web users around the world. The analysis of social networks helps summarizing the interests and opinions of users (nodes), discovering patterns from the interactions (links) between users, and mining the events that take place in online platforms. The information obtained by analyzing social networks could be especially valuable for many applications. Some typical examples include online advertisement targeting (Li et al., 2015), personalized recommendation (Song et al., 2006), viral marketing (Leskovec et al., 2007; Chen et al., 2010), social healthcare (Tang and Yang, 2012), social influence analysis (Peng et al., 2017), and academic network analysis (Dietz et al., 2007; Guo et al., 2014).

One central problem in social network analysis is how to extract useful features from non-Euclidean structured networks, to enable the deployment of downstream machine learning prediction models for specific analysis. For example, in the case of recommending new friends to a user in a social network, the key challenge might be how to embed network users into a low-dimensional space so that the closeness between users could be easily measured with distance metrics. To process structure information in networks, most previous efforts mainly rely on hand-crafted features, such as kernel functions (Vishwanathan et al., 2010), graph statistics (i.e., degrees or clustering coefficients) (Bhagat et al., 2011), or other carefully engineered features (Liben-Nowell and Kleinberg, 2007). However, such feature engineering processes could be very time-consuming and expensive, rendering it ineffective for many real-world applications. An alternative way to avoid

this limitation, is to automatically learn feature representations that capture various information sources in networks (Bengio et al., 2013; Liao et al., 2018). The goal is to learn a transformation function that maps nodes, subgraphs, or even the whole network as vectors to a low-dimensional feature space, where the spatial relations between the vectors reflect the structures or contents in the original network. Given these feature vectors, subsequent machine learning models such as classification models, clustering models and outlier detection models could be directly used toward target applications.

Along with the substantial performance improvement gained by deep learning on image recognition, text mining, and natural language processing tasks (Bengio, 2009), developing network representation methods using neural network models have received increased attention in recent years. In this review, we provide a comprehensive overview of recent advancements in network representation learning, using neural network models. After introducing the notations and problem definitions, we first review the basic representation learning models for node embedding in homogeneous networks. Specifically, based on the type of representation generation modules, we divide the existing approaches into three categories: embedding look-up based, autoencoder based and graph convolution based. We then provide an overview of the approaches that learn representations for subgraphs in networks, which to some extent rely on the techniques of node representation learning. After that, we list some applications of network representation models. Finally, we discuss some promising research directions for future work.

2. NOTATIONS AND PROBLEM DEFINITIONS

In this section, we define some important terminologies that will be used in later sections, and then provide the formal definition of the network representation learning problem. In general, we use boldface uppercase letters (e.g., $\mathbf{A}$) to denote matrices, boldface lowercase letters (e.g., $\mathbf{a}$) to denote vectors, and lowercase letters (e.g., a) to denote scalars. The (i,j) entry, the i-th row and the j-th column of a matrix $\mathbf{A}$ is denoted as $\mathbf{A}_{ij}$, $\mathbf{A}_{i*}$, and $\mathbf{A}_{*j}$, respectively.

Definition 1 (Network). Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{Y}\}$ be a network, where the i-th node (or vertex) is denoted as $v_i \in \mathcal{V}$ and $e_{i,j} \in \mathcal{E}$ denotes the edge between node v_i and v_j. $\mathbf{X}$ and $\mathbf{Y}$ are node attributes and labels, if available. Besides, we let $\mathbf{A} \in \mathbb{R}^{N\times N}$ denote the associated adjacency matrix of $\mathcal{G}$. $\mathbf{A}_{ij}$ is the weight of $e_{i,j}$, where $\mathbf{A}_{ij} > 0$ indicates that the two nodes are connected, and otherwise $\mathbf{A}_{ij} = 0$. For undirected graphs, $\mathbf{A}_{ij} = \mathbf{A}_{ji}$.

In many scenarios, the nodes and edges in $\mathcal{G}$ can also be associated with the type information. Let $\tau_v : \mathcal{V} \to T^v$ be a node-type mapping function and $\tau_e : \mathcal{E} \to T^e$ be an edge-type mapping function, where T^v and T^e denote the set of node and edge types, respectively. Here, each node $v_i \in \mathcal{V}$ has one specific type, e.g., $\tau_v(v_i) \in T^v$. Similarly, for each edge e_{ij}, $\tau_e(e_{ij}) \in T^e$.

Definition 2 (Homogeneous Network). A homogeneous network is a network in which $|T^v| = |T^e| = 1$. All nodes and edges in G belong to one single type.

Definition 3 (Heterogeneous Network). A heterogeneous network is a network with $|T^v| + |T^e| > 2$. There are at least two different types of nodes or edges in heterogeneous networks.

Given a network $\mathcal{G}$, the task of network representation learning is to train a mapping function f that maps certain components in $\mathcal{G}$, such as nodes or subgraphs, into a latent space. Let D be the dimension of the latent space and usually $D \ll |\mathcal{V}|$. In this work, we focus on the problem of node representation learning and subgraph representation learning.

Definition 4 (Node Representation Learning). Suppose $\mathbf{z} \in \mathbb{R}^D$ denotes the latent vector of node v, node representation learning aims to build a mapping function f so that $\mathbf{z} = f(v)$. It is expected that nodes with similar roles or characteristics, which are defined according to specific application domains, are mapped close to each other in the latent space.

Definition 5 (Subgraph Representation Learning). Let g denote a subgraph of $\mathcal{G}$. The nodes and edges in g are denoted as $\mathcal{V}_S$ and $\mathcal{E}_S$, respectively, and we have $\mathcal{V}_S \subset \mathcal{V}$ and $\mathcal{E}_S \subset \mathcal{E}$. The subgraph representation learning aims to learn a mapping function f so that $\mathbf{z} = f(g)$, where in this case $\mathbf{z} \in \mathbb{R}^D$ corresponds to the latent vector of g.

Figure 1 shows a toy example of network embedding. There are three subgraphs in this network distinguished with different colors: $\mathcal{V}_{S_1} = \{v_1, v_2, v_3\}$, $\mathcal{V}_{S_2} = \{v_4\}$, and $\mathcal{V}_{S_3} = \{v_5, v_6, v_7\}$. Given a network as input, the example below generates one representation for each node, as well as for each of the three subgraphs.

3. NEURAL NETWORK BASED MODELS

It has been demonstrated that neural networks have powerful capabilities in capturing complex patterns in data, and have achieved substantial success in the fields of computer vision, audio recognition, and natural language processing, etc. Recently, some efforts have been made to extend neural network models to learn representations from network data. Based on the type of base neural networks that are applied, we categorize them into three subgroups: look-up table based models, autoencoder based models, and GCN based models. In this section, we first give an overview of network representation learning from the perspective of *encoding* and *decoding*. We then discuss the details of some well-known network embedding models and how they fulfill the two steps. In this section, we only discuss representation learning for nodes. The models dealing with subgraphs will be introduced in later sections.

3.1. Framework Overview From the Encoder-Decoder Perspective

In order to elaborate the diversity of various neural network architectures, we argue that different techniques can be derived from the aspect of *encoding* and *decoding* schema, as well as their *target network structure* constrained for low dimensional feature space. Specifically, existing methods can be reduced to solving the following optimization problem:

$$\min_{\Psi} \sum_{\phi \in \Phi_{tar}} \mathcal{L}(\psi_{dec}(\psi_{enc}(\mathcal{V}_\phi)), \phi | \Psi), \tag{1}$$

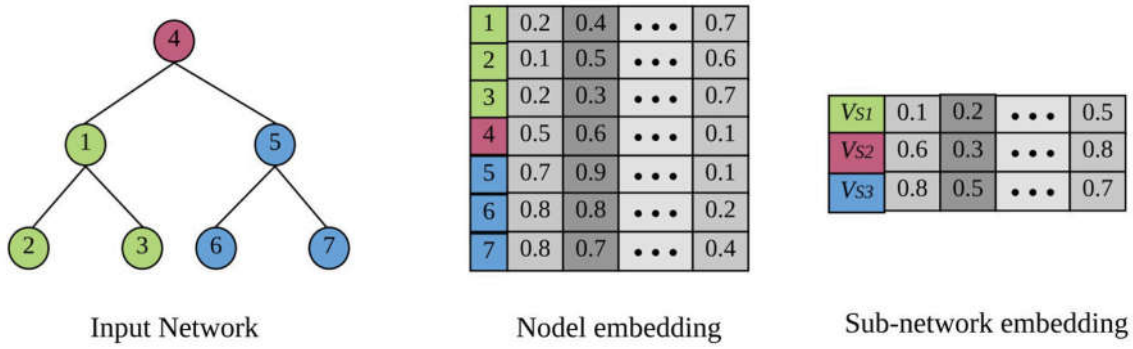


FIGURE 1 | A toy example of node representation learning and subgraph representation learning (best viewed in color). There are three subgraphs in the input network denoted by different colors. The target of node embedding is to generate one representation for each individual node, while subgraph embedding is to learn one representation for an entire subgraph.

where Φ_{tar} is the target relations that the embedding algorithm expects to preserve, and $\mathcal{V}_\phi$ denotes the nodes involved in ϕ. $\psi_{enc}: \mathcal{V} \rightarrow \mathcal{R}^D$ is the *encoding* function that maps nodes into representation vectors, and ψ_{dec} is a decoding function that reconstructs the original network structure from the representation space. Ψ denotes the trainable parameters in encoders and decoders. By minimizing the loss function above, model parameters are trained so that the desired network structures Ψ_{tar} are preserved. As we will show in subsequent sections, from the overview framework aspect, the primary distinctions between various network representation methods rely on how they define the three components.

3.2. Models With Embedding Look-Up Tables

Instead of using multiple layers of nonlinear transformations, network representation learning could be achieved simply by using look-up tables which directly map a node index into its corresponding representation vector. Specifically, a look-up table could be implemented using a matrix, where each row corresponds to the representation of one node. The diversity of different models mainly lies in the definition of target relations in the network data that we hope to preserve. In the rest of this subsection, we will first introduce DeepWalk (Perozzi et al., 2014) to discuss the basic concepts and techniques in network embedding, and then extend the discussion to more complex and practical scenarios.

3.2.1. Skip-Gram Based Models

As a pioneering network representation model, DeepWalk treats nodes as words, samples random walks as sentences and utilizes the skip-gram model (Mikolov et al., 2013) to learn the representations of nodes as shown in **Figure 2**. In this case, the encoder ψ_{enc} is implemented as two embedding look-up tables $\mathbf{Z} \in \mathbb{R}^{N \times D}$ and $\mathbf{Z}^c \in \mathbb{R}^{N \times D}$, respectively for target embeddings and context embeddings. The network information $\phi \in \Phi_{tar}$ that we try to preserve is defined as the node-context pairs $[v_i, \mathcal{N}(v_i)]$ observed in the random walks, where $\mathcal{N}(v_i)$ denotes the context nodes (or neighborhood) of v_i. The objective is to maximize the probability of observing a node's neighborhood conditioned on embeddings:

$$\mathcal{L} = -\sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} \log p(\mathbf{e}_j \mathbf{Z}^c | \mathbf{e}_i \mathbf{Z}), \tag{2}$$

where $\mathbf{e}_i$ is a one-hot row vector of length N that picks the i-th row of $\mathbf{Z}$. Let $\mathbf{z}_i = \mathbf{e}_i \mathbf{Z}$ and $\mathbf{z}_j^c = \mathbf{e}_j \mathbf{Z}^c$, the conditional probability above is formulated as

$$p(\mathbf{z}_j^c | \mathbf{z}_i) = \frac{\exp(\mathbf{z}_j^c \mathbf{z}_i^T)}{\sum_{k=1}^{|\mathcal{V}|} \exp(\mathbf{z}_k^c \mathbf{z}_i^T)}, \tag{3}$$

so that ψ_{dec} could be regarded as link reconstruction based on the normalized proximity between different nodes. In practice, the computation of the probability is expensive due to the summation over every node in the network, but hierarchical softmax or negative sampling can be applied to reduce time complexity.

There are also some approaches that are developed based on similar ideas. LINE (Tang et al., 2015) defines the first-order and second-order proximity for learning node embedding, where the latter can be seen as a special case of DeepWalk with context window length set as 1. Meanwhile, node2vec (Grover and Leskovec, 2016) applies different random walk strategies, which provides a trade-off between breadth-first search (BFS) and depth-first search (DFS) in networks search strategies. Planetoid (Yang et al., 2016) extends skip-gram models for semi-supervised learning, which predicts the class label of nodes along with the context in the input network data. In addition, it has been shown that there exists a close relationship between skip-gram models and matrix factorization algorithms (Levy and Goldberg, 2014; Qiu et al., 2018). Therefore, network embedding models that utilize matrix factorization techniques, such as LE (Belkin and Niyogi, 2002), Grarep (Cao et al., 2015), and HOPE (Ou et al., 2016), may also be implemented in the similar manner. Random sampling-based approaches have the capacity to allow a flexible and stochastic measure of node similarity, making them not only achieve higher performance in many

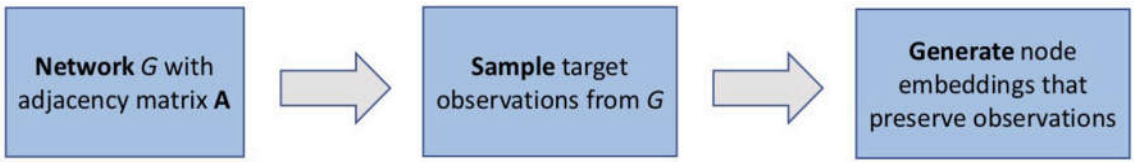


FIGURE 2 | Building blocks of models with embedding look-up tables. There are two key components of these works: *sampling* and *modeling*. The primary distinction between different methods under this line relies on how to define the two components.

applications, but also become more scalable toward large-scale datasets.

3.2.2. Attributed Network Embedding Models

Social networks are rich in side information, where nodes could be associated with various attributes that characterize their properties. Inspired by the idea of inductive matrix completion (Natarajan and Dhillon, 2014), TADW (Yang et al., 2015) extends the framework of DeepWalk by incorporating features of vertices into network representation learning. Besides sampling from plain networks, FeatWalk (Huang et al., 2019) proposes a novel feature-based random walk strategy to generate node sequences by considering node similarity on attributes. With the random walks based on both topological and attribute information, the skip-gram model is then applied to learn node representations.

3.2.3. Heterogeneous Network Embedding Models

Nodes in networks could be of different types, which poses the challenge of how to preserve relations among them. HERec (Shi et al., 2019) and metapath2vec++ (Dong et al., 2017) propose meta-path based random walk schema to discover the context across different types of nodes. The skip-gram architecture in metapath2vec++ is also modified, so that the normalization term in softmax only considers nodes of the same type. In a more complex scenario where we have both nodes and attributes of different types, HNE (Chang et al., 2015) combines feed-forward neural networks and embedding models toward a unified framework. Suppose $\mathbf{z}^a$ and $\mathbf{z}^b$ denote the latent vectors of two different types of nodes, HNE defines two additional transformation matrices $\mathbf{U}$ and $\mathbf{V}$ to respectively map $\mathbf{z}^a$ and $\mathbf{z}^b$ to the joint space. Let $v_i, v_j \in \mathcal{V}_a$ and $v_k, v_l \in \mathcal{V}_b$, intra-type node similarity and inter-type node similarity are defined as

$$s(v_i, v_j) = \mathbf{z}_i^a \mathbf{U} (\mathbf{z}_j^a \mathbf{U})^T, \quad s(v_i, v_k) = \mathbf{z}_i^a \mathbf{U} (\mathbf{z}_k^b \mathbf{V})^T, \quad s(v_k, v_l) = \mathbf{z}_k^b \mathbf{V} (\mathbf{z}_l^b \mathbf{V})^T, \tag{4}$$

where we hope to preserve various types of similarities during training. As for obtaining $\mathbf{z}^a$ and $\mathbf{z}^b$, HNE applies different feed-forward neural networks to map raw input (e.g., images and texts) to latent spaces, thus enabling an end-to-end training framework. Specifically, the authors use a CNN to process images and a fully-connected neural network to process texts.

3.2.4. Dynamic Embedding Models

Real world social networks are not static and will evolve over time with addition/deletion of nodes and links. To deal with this challenge, DNE (Du L. et al., 2018) presents a decomposable objective to learn the representation of each node separately, where the impact of network changes on existing nodes is measurable and greatly affected nodes will be chosen for updates as the learning process proceeds. In addition, DANE (Li J. et al., 2017) leverages matrix perturbation theory to tackle online embedding updates.

3.3. Autoencoder Techniques

In this section, we discuss network representation models based on the autoencoder architecture (Hinton and Salakhutdinov, 2006; Bengio et al., 2013). As shown in **Figure 3**, an autoencoder consists of two neural network modules: encoder and decoder. The encoder ψ_{enc} maps the features of each node into a latent space, and the decoder ψ_{doc} reconstructs the information about the network from the latent space. Usually the hidden representation layer has a smaller size than that of the input/output layer, forcing it to create a compressed representation that captures the non-linear structure of network. Formally, following Equation (1), the objective function of autoencoder is to minimize the reconstruction error between the input and the output decoded from low-dimensional representations.

3.3.1. Deep Neural Graph Representation (DNGR)

DNGR (Cao et al., 2016) attempts to preserve a node's local neighborhood information using a stacked denoising autoencoder. Specifically, assume $\mathbf{S}$ is the PPMI matrix (Bullinaria and Levy, 2007) constructed from $\mathbf{A}$, then DNGR minimizes the following loss:

$$\mathcal{L} = \sum_{v_i \in \mathcal{V}} ||\psi_{dec}(\mathbf{z}_i) - \mathbf{S}_{i*}||_2^2 \quad s.t. \ \mathbf{z}_i = \psi_{enc}(\mathbf{S}_{i*}), \tag{5}$$

where $\mathbf{S}_{i*} \in \mathbb{R}^{|\mathcal{V}|}$ denotes the associated neighborhood information of v_i. In this case, $\Phi_{tar} = \{\mathbf{S}_{i*}\}_{v_i \in V}$ and DNSR targets to reconstruct the PPMI matrix. $\mathbf{z}_i$ is the embedding of node v_i in the hidden layer.

3.3.2. Structural Deep Network Embedding (SDNE)

SDNE (Wang et al., 2016) is another autoencoder-based model for network representation learning. The objective function

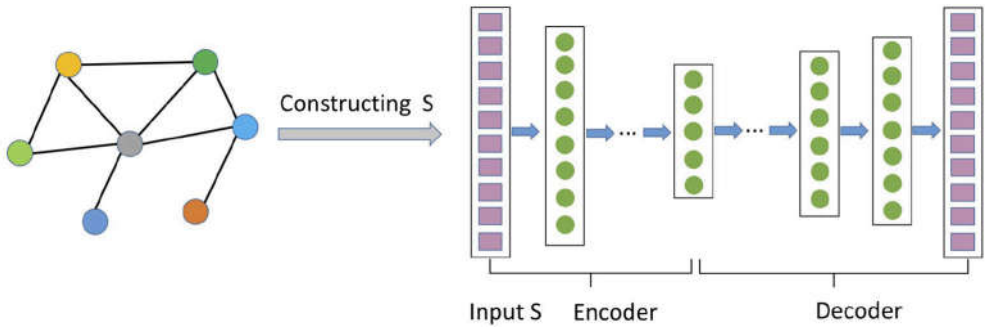


FIGURE 3 | An example of autoencoder-based network representation algorithms. Rows of the proximity matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ are fed into the autoencoder to learn and generate embeddings $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}|\times D}$ at the hidden layer.

of SDNE is:

$$\mathcal{L} = \sum_{v_i \in V} ||(\psi_{dec}(\mathbf{z}_i) - \mathbf{S}_{i*}) \odot \mathbf{b}_i||_2^2 + \sum_{i,j=1}^{|V|} \mathbf{S}_{ij} ||\mathbf{z}_i - \mathbf{z}_j||_2^2, \quad \Psi_{tar} = \{\mathbf{S}_{i*}, \mathbf{S}_{ij}\}. \tag{6}$$

The first term is an autoencoder as in Equation (5), except that the reconstruction error is weighted, so that more emphasis is put on recovering non-zero entries in $\mathbf{S}_{i*}$. The second part is motivated by Laplacian Eigenmaps that imposes nearby nodes to have similar embeddings. Besides, SDNE differs from DNGR in the definition of $\mathbf{S}$, where DNGR defines $\mathbf{S}$ as the PPMI matrix while SDNE sets $\mathbf{S}$ as the adjacency matrix.

It is worth noting that, unlike Equation (2) which uses one-hot indicator vector for embedding look-up, DNGR and SDNE transform each node's information to an embedding by training neural network modules. Such distinction allows autoencoder-based methods to directly model on a node's neighborhood structure and features, which is not straightforward for random walk approaches. Therefore, it is straightforward to incorporate richer information sources (e.g., node attributes) into representation learning, as will be introduced below. However, autoencoder-based methods may suffer from scalability issues as the input dimension is $|\mathcal{V}|$, which may result in significant time costs in real massive datasets.

3.3.3. Autoencoder-Based Attributed Network Embedding

The structure of autoencoders facilitates the incorporation of multiple information sources toward joint representation learning. Instead of only mapping nodes to the latent space, CAN (Meng et al., 2019) proposes to learn the representation of nodes and attributes in the same latent space by using variational autoencoders (VAEs) (Doersch, 2016), in order to capture the affinities between nodes and attributes. DANE (Gao and Huang, 2018) utilizes the correlation between topological and attribute information of nodes by building two autoencoders for each information source, and then encourages the two sets of latent representations to be consistent and complementary. Li H. et al. (2017) adopts another strategy, where topological feature vector and content information vector (learned by doc2vec Le and Mikolov, 2014) are directly concatenated and put into a VAE to capture the nonlinear relationship between them.

3.4. Graph Convolutional Approaches

Inspired by the significant performance improvement of convolutional neural networks (CNN) in image recognition, recent years have witnessed a surge in adapting convolutional modules to learn representations of network data. The intuition behind it is to generate node embedding by aggregating information from its local neighborhood as shown in **Figure 4**. Different from autoencoder-based approaches, the encoding function of graph convolutional approaches leverages a node's local neighborhood as well as attribute information. Some efforts (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Hamilton W. et al., 2017) have been made to extend traditional convolutional networks for network data to generate network embedding in the past few years. The convolutional filters of these approaches are either spatial filters or spectral filters. Spatial filters operate directly on the adjacency matrix whereas spectral filters operate on the spectrum of graph Laplacian (Defferrard et al., 2016).

3.4.1. Graph Convolutional Networks (GCN)

GCN (Bronstein et al., 2017) is a well-known semi-supervised graph convolutional networks. It defines a convolutional operator on network, and iteratively aggregates embeddings of neighbors of a node and uses the aggregated embedding as well as its own embedding at previous iteration to generate the node's new representation. The layer-wise propagation rule of encoding function ψ_{enc} is defined as:

$$\mathbf{H}^k = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{k-1}\mathbf{W}^{k-1}), \tag{7}$$

where $\mathbf{H}^{k-1}$ denotes the learned embeddings in layer $k-1$, and $\mathbf{H}^0 = \mathbf{X}$. $\hat{\mathbf{A}} = (\mathbf{I}_G + \mathbf{A})$ is the adjacency matrix with added self-connections. $\mathbf{I}_G$ is the identity matrix, $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$. $\mathbf{W}^{k-1}$ is a layer-wise trainable weight matrix. $\sigma(\cdot)$ denotes an activation function such as ReLU. The loss function for supervised training

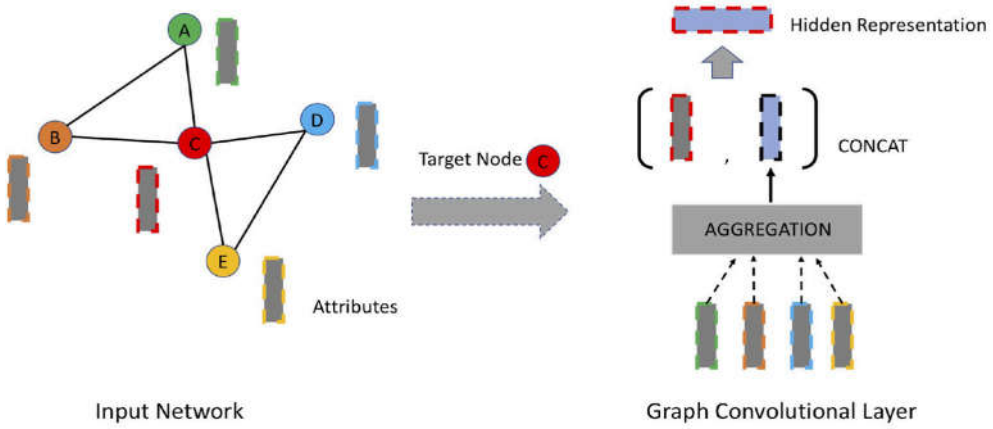


FIGURE 4 | An overview of graph convolutional networks. The dashed rectangles denote node attributes. The representation of each individual node (e.g., node C) is aggregated from its immediate neighbors (e.g., node A, B, D, E), concatenated with the lower-layer representation of itself.

is to evaluate the cross-entropy error over all labeled nodes:

$$\mathcal{L} = -\sum_{v_i \in \mathcal{V}} \sum_{f=1}^{F} \mathbf{Y}_{if} \ln \hat{\mathbf{Y}}_{if}, \;\; s.t. \;\; \hat{\mathbf{Y}} = \psi_{dec}(\mathbf{Z}), \;\; \mathbf{Z} = \psi_{enc}(\mathbf{X}, \mathbf{A}), \tag{8}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times F}$ is the predictive matrix with F candidate labels. $\psi_{dec}(\cdot)$ can be viewed as a fully-connected network with the softmax activation function to map representations to predicted labels. Note that unlike autoencoders that explicitly treat each node's neighborhood as features or reconstruction goals as in Equations (5) or (6), GCN implicitly applies the local neighborhood links on each encoding layer as pathways to aggregate embeddings from neighbors, so that higher order network structures are utilized. Since Equation (8) is a supervised loss function, Φ_{tar} is not applicable here. However, the loss function can also be formulated in unsupervised manners, similar to the skip-gram model (Kipf and Welling, 2016; Hamilton W. et al., 2017). GCN may suffer from the scalability problem when the size of $\mathbf{A}$ is large. The corresponding training algorithms have been proposed to tackle this challenge (Ying et al., 2018a), where the network data is processed in small batches and we can sample a node's local neighbors instead of using all of them.

3.4.2. Inductive Training With GCN

So far many basic models we have reviewed mainly generate network representations in a transductive manner. GraphSAGE (Hamilton W. et al., 2017) emphasized the inductive capability of GCN. Inductive learning is essential for high-throughput machine learning systems, especially when operating on evolving networks that constantly encounter unseen nodes (Yang et al., 2016; Guo et al., 2018). The core representation update scheme of GraphSAGE is similar to that of traditional GCN, except that the operation on the whole network is replaced by sample-based representation aggregators:

$$\mathbf{h}_i^k = \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_i^{k-1}, \text{AGGREGATE}_k(\{\mathbf{h}_j^{k-1}, \forall j \in \mathcal{N}(v_i)\}))), \tag{9}$$

where $\mathbf{h}_i^k$ is the hidden representation of node v_i in the k-th layer. CONCAT denotes concatenation operator and AGGREGATE_k represents neighborhood aggregation function of the k-th layer (e.g., element-wise mean or max operator). $\mathcal{N}(v_i)$ denotes the neighbors of v_i. Compared with Equation (7), GraphSAGE only needs to aggregate feature vectors from the partial set of neighbors, making it scalable for large-scale data. Given the attribute features and neighborhood relations of an unseen node, GraphSAGE can generate the embedding of this node by leveraging its local neighbors as well as attributes via forward propagation.

3.4.3. Graph Attention Mechanisms

Attention mechanisms have become the standard technique in many sequence-based tasks, in order to make models focus on the most relevant parts of the input in making decisions. We could also utilize attention mechanisms to aggregate the most important features from nodes' local neighbors. GAT (Velickovic et al., 2017) extends the framework of GCN by replacing the standard aggregation function with an attention layer to aggregate messages from most important neighbors. Thekumparampil et al. (2018) also proposes to remove all intermediate fully-connected layers in conventional GCN and to replace the propagation layers with attention layers. It thus allows the model to learn a dynamic and adaptive local summary of neighborhoods, greatly reduces the parameters, and also achieves more accurate predictions.

4. SUBGRAPH EMBEDDING

Besides learning representations for nodes, recent years have also witnessed an increasing branch of research efforts that try to learn representations for a set of nodes and edges as an integral. Thus, the goal is to represent a subgraph with a low-dimensional vector. Many traditional methods that operate on subgraphs rely on graph kernels (Haussler, 1999), which decompose a network into some atomic substructures such as graphlets, subtree patterns, and paths, and treat these substructures as features to obtain an embedding through further transformation. In this section, however, we focus on reviewing methods that seek to automatically learn embeddings of subgraphs using deep models. For those who are interested in graph kernels, we refer the readers to Vishwanathan et al. (2010).

According to the literature, most existing methods are built on the techniques used for node embedding, as introduced in section 3. However, in graph representation problems, the label information is associated with particular subgraphs instead of individual nodes or links. In this review, we divide the approaches of subgraph representation learning into two categories based on how they aggregate node-level embeddings in each subgraph. The detailed discussion for each category is as below.

4.1. Flat Aggregation

Assume $\mathcal{V}_S$ denotes the set of nodes in a particular subgraph and $\mathbf{z}_S$ represents the subgraph's embedding, $\mathbf{z}_S$ could be obtained by aggregating the embeddings of all individual nodes in the subgraph:

$$\mathbf{z}_S = \psi_{aggr}(\{\mathbf{z}_i, v_i \in \mathcal{V}_S\}), \tag{10}$$

where ψ_{aggr} denotes the aggregation function. Methods based on such flat aggregation usually define ψ_{aggr} that captures simple correlations among nodes. For example, Niepert et al. (2016) directly concatenates node embeddings together and utilize standard convolutional neural networks as an aggregation function to generate graph representation. Dai et al. (2016) employs a simple element-wise summation operation to define ψ_{aggr}, and learns graph embedding by summing all embeddings of individual nodes.

In addition, some methods apply recurrent neural networks (RNNs) for representing graphs. Some typical methods first sample a number of graph sequences from the input network, and then apply RNN-based autoencoders to generate an embedding for each graph sequence. The final graph representation is obtained by either averaging (Jin et al., 2018) or concatenating (Taheri et al., 2018) these graph sequence embeddings.

4.2. Hierarchical Aggregation

In contrast to flat aggregation, the motivation behind *hierarchical* aggregation is to preserve the hierarchical structure that might be presented in the subgraph by aggregating neighborhood information via a hierarchical way. Bruna et al. (2013) and Defferrard et al. (2016) attempt to utilize such a hierarchical structure of networks by combining convolutional neural networks with graph coarsening. The main idea behind them is to stack multiple graph coarsening and convolutional layers. In each layer, they first apply graph cluster algorithms to group nodes, and then merge node embeddings within each cluster using element-wise max-pooling. After clustering, they generate a new coarse network by stacking embeddings of clusters together, which is again fed into convolutional layers and the same process repeats. Clusters in each layer can be viewed as subgraphs, and cluster algorithms are used to learn the assignment matrix of subgraphs, so that the hierarchical structure of the network is also propagated through the layers. Although these methods work well in certain applications, they actually follow a two-stage fashion, where the stages of clustering and embedding may not reinforce each other.

To avoid this limitation, DiffPool (Ying et al., 2018b) proposes an end-to-end model that does not depend on a deterministic clustering subroutine. The layer-wise propagation rule is formulated as below:

$$\mathbf{M}^{(k+1)} = \mathbf{C}^{(k)^T}\mathbf{Z}^{(k)}, \quad \mathbf{A}^{(k+1)} = \mathbf{C}^{(k)^T}\mathbf{A}^{(k)}\mathbf{C}^{(k)}, \tag{11}$$

where $\mathbf{Z}^{(k)} \in \mathbb{R}^{N_k \times D}$ denotes node embeddings, $\mathbf{C}^{(k)} \in \mathbb{R}^{N_{N_k} \times N_{k+1}}$ is the cluster assignment matrix learned from the previous layer. The goal of the left equation is to generate the $(k+1)$-th coarser network embedding $\mathbf{M}^{(k+1)}$ by aggregating node embeddings according to cluster assignment $\mathbf{C}^{(k)}$; while the right equation is to learn a new coarsened adjacency matrix $\mathbf{A}^{(k+1)} \in \mathbb{R}^{N_{k+1} \times N_{k+1}}$ from the previous adjacency matrix $\mathbf{A}^{(k)}$, which stores the similarity between each pair of clusters. Here, instead of applying deterministic clustering algorithm to learn $\mathbf{C}^{(k)}$, they adopt graph neural networks (GNNs) to learn it. Specifically, they use two separate GNNs on the input embedding matrix $\mathbf{M}^{(k)}$ and coarsened adjacency matrix $\mathbf{A}^{(k)}$ to generate assignment matrix $\mathbf{C}^{(k)}$ and embedding matrix $\mathbf{Z}^{(k)}$, respectively. Formally, $\mathbf{Z}^{(k)} = \text{GNN}_{k,embed}(\mathbf{A}^{(k)}, \mathbf{M}^{(k)})$, and $\mathbf{C}^{(k)} = softmax[\text{GNN}_{k,pool}(\mathbf{A}^{(k)}, \mathbf{M}^{(k)})]$. The two steps could reinforce each other to improve the performance. DiffPool may suffer from computational issues brought by the computation of soft clustering assignment, which is further addressed in Cangea et al. (2018).

5. APPLICATIONS

The representations learned from networks can be easily applied to downstream machine learning models for further analysis on social networks. Some common applications include node classification, link prediction, anomaly detection, and clustering.

5.1. Node Classification

In social networks, people are often associated with semantic labels with respect to certain aspects about them, such as affiliations, interests, or beliefs. However, in real-world scenarios, people are usually partially or sparsely labeled, since labeling is expensive and time consuming. The goal of node classification is to predict labels of unlabeled nodes in networks by leveraging their connections with the labeled ones considering the network structure. According to Bhagat et al. (2011), existing methods can be classified into two categories, e.g., random walk based, and feature extraction-based methods. The former aims to propagate labels with random walks (Baluja et al., 2008), while the latter targets to extract features from a node's surrounding information and network statistics.

In general, the network representation approach follows the second principle. A number of existing network representation models, like Yang et al. (2015), Wang et al. (2016), and Liao et al. (2018), focus on extracting node features from the network using representation learning techniques, and then apply machine learning classifiers like support vector machine, naive Bayes classifiers, and logistic regression for prediction. In contrast to separating the steps of node embedding and node classification, some recent work (Dai et al., 2016; Hamilton W. et al., 2017; Monti et al., 2017) designs an end-to-end framework to combine the two tasks, so that the discriminative information inferred from labels can directly benefit the learning of network embedding.

5.2. Link Prediction

Social networks are not necessarily complete as some links might be missing. For example, friendship links between two users in a social network can be missing even if they actually know each other in real world. The goal of link prediction is to infer the existence of new interactions or emerging links between users in the future, based on the observed links and the network evolution mechanism (Liben-Nowell and Kleinberg, 2007; Al Hasan and Zaki, 2011; Lü and Zhou, 2011). In network embedding, an effective model is expected to preserve both network structure and inherent dynamics of the network in the low-dimensional space. In general, the majority of previous work focuses on predicting missing links between users under homogeneous network settings (Grover and Leskovec, 2016; Ou et al., 2016; Zhou et al., 2017), and some efforts also attempt to predict missing links in heterogeneous networks (Liu Z. et al., 2017, 2018). Although, beyond

the scope of this survey, applying network embedding for building recommender systems (Ying et al., 2018a) may also be a direction that is worth exploring.

5.3. Anomaly Detection

Another challenging task in social network analysis is anomaly detection. Malicious activities in social networks, such as spamming, fraud, and phishing, can be interpreted as rare or unexpected behaviors that deviate from the majority of normal users. While numerous algorithms have been proposed for spotting anomalies and outliers in networks (Savage et al., 2014; Akoglu et al., 2015; Liu N. et al., 2017), anomaly detection methods, based on network embedding techniques, have recently received increased attention (Hu et al., 2016; Liang et al., 2018; Peng et al., 2018). The discrete and structural information in networks are merged and projected into the continuous latent space, which facilitates the application of various statistical or geometrical algorithms in measuring the degree of isolation or outlierness of network components. In addition, in contrast to detect malicious activities in a static way, Sricharan and Das (2014) and Yu et al. (2018) also attempted to study the problem in dynamic networks.

5.4. Node Clustering

In addition to the above applications, node clustering is another important network analysis problem. The target of node clustering is to partition a network into a set of clusters (or subgraphs), so that nodes in the same cluster are more similar to each other than those from other clusters. In social networks, such clusters are widely spread in terms of communities, such as groups of people that belong to similar affiliations or have similar interests. Most previous work focuses on clustering networks with various metrics of proximity or connection strength between nodes. For example, Shi and Malik (2000) and Ding et al. (2001) seek to maximize the number of connections within clusters while minimizing the connections between clusters. Recently, many efforts have resort to network representation techniques for node clustering. Some methods treat embedding and clustering as disjointed tasks, where they first embed nodes to low-dimensional vectors, and then apply traditional clustering algorithms to produce clusters (Tian et al., 2014; Cao et al., 2015; Wang et al., 2017). Other methods such as Tang et al. (2016) and Wei et al. (2017) consider the optimization problem of clustering and network embedding in a unified objective function and generate cluster-induced node embeddings.

6. CONCLUSION AND FUTURE DIRECTIONS

In recent years there has been a surge in leveraging representation learning techniques for network analysis. In this review, we have provided an overview of the recent efforts on this topic. Specifically, we summarize existing techniques into three subgroups based on the type of the core learning modules: representation look-up tables, autoencoders, and graph convolutional networks. Although many techniques have been developed for a wide spectrum of social networks analysis problems in the past few years, we believe there still remains many promising directions that are worth further exploring.

6.1. Dynamic Networks

Social networks are inherently highly dynamic in real-life scenarios. The overall set of nodes, the underlying network structure, as well as attribute information, might evolve over time. As an example, these elements in real world social networks such as Facebook could correspond to users, connections, and personal profiles. This property makes existing static learning techniques fail in working properly. Although several methods have been proposed to tackle dynamic networks, they often rely on certain assumptions, such as assuming that the node set is fixed and only deals with dynamics caused by edge deletion and addition (Li J. et al., 2017). Furthermore, the changes in attribute information are rarely considered in existing works. Therefore, how to design effective and efficient network embedding techniques for truly dynamic networks remains an open question.

6.2. Hierarchical Network Structure

Most of the existing techniques mainly focus on designing advanced encoding or decoding functions trying to capture node pairwise relationships. Nevertheless, pairwise relations can only provide insights about local neighborhoods, and might not infer global hierarchical network structures, which is crucial for more complex networks (Benson et al., 2016). How to design effective network embedding methods that are capable of preserving hierarchical structures of networks is a promising direction for further work.

6.3. Heterogeneous Networks

Existing network embedding methods mainly deal with homogeneous networks. However, many relational systems in real-life scenarios can be abstracted as heterogeneous networks with multiple types of nodes or edges. In this case, it is hard to evaluate semantic proximity between different network elements in the low-dimensional space. While some work has investigated the use of metapaths (Dong et al., 2017; Huang and Mamoulis, 2017) to approximate semantic similarity for heterogeneous network embedding, many tasks on heterogeneous networks have not been fully evaluated. Learning embeddings for heterogeneous networks is still at the early stage, and more comprehensive techniques are required to fully capture the relations between different types of network elements, toward modeling more complex real systems.

6.4. Scalability

Although deep learning based network embedding methods have achieved substantial performances due to their great capacities, they still suffer from the problem of efficiency. This problem will become more severe when dealing with real-life massive datasets with billions of nodes and edges. Designing deep representation learning frameworks that are scalable for real network datasets is another driving factor to advance the research in this domain. Additionally, similar to using GPUs for traditional deep models built on grid structured data, developing computational paradigms for large-scale network processing could be an alternative way toward efficiency improvement (Bronstein et al., 2017).

6.5. Interpretability

Despite the superior performances achieved by deep models, one fundamental limitation of them is the lack of interpretability (Liu N. et al., 2018). Different dimensions in the embedding space usually have no specific meaning, thus it is difficult to comprehend the underlying factors that have been preserved in the latent space. Since the interpretability aspect of machine learning models is currently receiving increased attention (Du M. et al., 2018; Montavon et al., 2018), it might also be important to explore how to understand the representation learning outcome, how to develop interpretable network representation learning models, as well as how to utilize interpretation to improve the representation models. Answering these questions is helpful to learn more meaningful and task-specific embeddings toward various social network analysis problems.

Speech Recognition

Speech recognition, sometimes referred to as automated or automatic speech recognition (ASR) or speech-to-text (STT), is a technology that enables computers to transcribe human speech into written text.

Speech recognition algorithms have evolved tounderstand natural speech in different languages, dialects, accents, and speech patterns.

The term automatic speech recognition was coined by engineers in the early 1990s to emphasize that speech recognition is a machine-processed technology. However, ASR and speech recognition are now used interchangeably.

How Speech Recognition Works

When we speak into a personal device's built-in microphone, the speech-to-text technology breaksdown the recording, adjusts for background noise, pitch, volume, and tempo, and converts the digital information into frequencies that can be analyzed.

To translate human speech accurately, speech recognition software relies on machine learning and natural language processing (NLP). Once the software receives the input speech signal, it generates word sequences that best match it and produces a readable transcription that the user can further process or correct.

However, as simple as this process may sound, speech recognition technology is incredibly complex, involving signal processing, machine learning, and natural language processing.

Additionally, the output accuracy depends on various factors, such as the quality of the original recording, the complexity of the language, and the system application.

To interpret human speech, computers must follow a series of steps, including:

1. Translating sound vibrations into electrical signals,

2. Digitizing the signals,
3. Analyzing the digital signals,
4. Matching the signals with suitable text representing the sounds.

With AI applications and machine learning, speech recognition technology improves performance and precision over time.

Speech recognition algorithms are trained on different speech patterns, languages, dialects, and accents to adapt to human speech's highly variable and context-specific nature.

Speech vs. Voice Recognition

Although "voice" and "speech" recognition are frequently interchanged terms, they actually refer to two distinct processes with different results. Understanding the differences between the two is crucial for businesses seeking to leverage these technologies for growth and communication.

While speech recognition translates anyone's voice, voice recognition is a biometric system that recognizes and authenticates a specific user's voice.

It analyzes the unique features of a person's voice, including pitch, tone, and rhythm, to create a unique voiceprint for identification.

This technology is often used for security purposes, such as unlocking mobile devices or accessing systems.

Types of Speech Recognition Technology

Speech recognition technology is broadly categorized into speaker-dependent and speaker-independent systems.

- Speaker-dependent systems are trained by the individual who will be using the system, resulting in high accuracy for word recognition but only for the specific individual who trained the system. This is the most common approach for personal computers.

- Speaker-independent systems are designed to respond to a specific word or phrase without being dependent on the speaker's identity. This requires the system to be able to recognize a wide variety of speech patterns, inflections, and enunciations of the target word.

While the command word count may be lower than that of speaker-dependent systems, high accuracy rates for speech recognition systems can still be achieved within processing limits.

As a result, these systems are often used in industrial applications, such as the AT&T system utilized in telephone systems.

What are the Features of Speech Recognition?

In addition to speaker-dependent and speaker-independent systems, speech recognition technology has evolved to include different types of modeling to meet diverse needs. The features of speech recognition are:

1. Statistical Speech Recognition

Statistical speech recognition (SSR) is a type of speech recognition technology that uses statistical models to analyze and understand spoken language.

It involves breaking down speech into individual phonetic units and analyzing their frequency and patterns to recognize words and phrases.

This approach is based on probability and uses machine learning algorithms to improve accuracy continuously.

2. Acoustic Modeling

Acoustic modeling (AM) is a process in speech recognition technology that involves creating a statistical representation of the sound patterns associated with individual phonetic units in a language, such as vowels and consonants.

It focuses on capturing the acoustic characteristics of speech, such as pitch, tone, and pronunciation to improve speech recognition systems' accuracy.

Essentially, acoustic modeling helps the computer "learn" how to recognize different sounds and words spoken by a variety of people, even in noisy environments.

3. Language Modeling

Language modeling (LM) is a technique used in NLP and speech recognition that involves predicting the likelihood of a particular sequence of words based on their frequency and context.

A language model attempts to learn the patterns and structure of language by analyzing large amounts of text data, allowing it to generate coherent sentences and understand the meaning behind the human language.

This technique helps improve speech recognition accuracy and enables chatbots and virtual assistants to respond to user requests more naturally and conversationally.

4. Keyword Spotting

Keyword spotting is a type of audio processing that focuses on identifying specific words or phrases within a more extensive audio stream.

This allows for more efficient and targeted speech recognition, as the system only needs to process the relevant portions of the audio.

Keyword spotting is particularly useful in applications where quick identification of specific information is required, such as in voice-controlled search engines or virtual assistants.

Applications of Speech Recognition Technology

Speech recognition technology has many applications, from virtual assistants to translation and custom voice commands.

Google Translate is a perfect example of how speech recognition technology can be used for translation. With over 100 languages supported, users can easily get translations by speaking into their device's microphone.

In addition, custom voice commands have been added to speech recognition software in recent years, enabling end-users to perform a series of actions with a single voice command.

These features have made speech recognition technology valuable in various industries, including healthcare, law enforcement, business, legal, entertainment, and education.

Speech Recognition Use in Healthcare

Speech recognition technology has become a crucial tool for streamlining documentation processes in the healthcare industry.

Physicians who interact with patients must record notes of the visit, providing a status update and guiding patients towards the next steps. Similarly, healthcare specialties that don't involve direct patient interaction also require reporting.

The introduction of electronic health records made the documentation process more involved. In the past, physicians either wrote notes or dictated them directly into a voice recorder, which was then converted into a more legible version by a third party.

While transcriptionists are still relatively common in medical practice, speech recognition has proven to be a more efficient approach. With voice recognition software, physicians can quickly and accurately transcribe their notes, saving time and reducing the risk of errors.

Speech Recognition Use in Law Enforcement

Recording encounters is vital to law enforcement, much like in the healthcare industry. As a standard procedure, incident forms and police reports must be completed.

However, the task of completing paperwork can be overwhelming and time-consuming. Fortunately, speech recognition technology has made this task more manageable for law enforcement professionals.

With speech recognition, notes can be quickly recorded from a mobile device, at home, or in the office, freeing up valuable time for other work responsibilities or personal time.

This technology allows for a more efficient and accurate recording of encounters, leading to more comprehensive and reliable reports.

Speech Recognition Use in Business

One of the most important speech recognition for business applications is customer service. Here digital operators can understand and interpret callers' voice commands, reducing the need for human staff and improving customer satisfaction.

Speech recognition can also transcribe entire meetings, automatically distinguishing between different speakers. This makes note-taking easier and more accurate.

Finally, speech recognition can make data entry faster and more efficient by allowing users to use voice commands to run functions or macros in databases and data processing tools.

This can save time and reduce errors, especially when dealing with large numerical values.

Speech Recognition Use in Legal

The legal field is benefiting from speech recognition technology in two key areas. Firstly, technology reduces the time legal professionals spend preparing court documents such as memos and briefs.

This can eliminate the need to allocate the work to paralegals or legal scribes.

Secondly, technology is transforming the way court transcripts are prepared. As a result, court reporters must have sharp attention to detail, a specialized legal vocabulary, and fast typing skills.

However, a shortage of qualified court reporters and demand for their services continues to rise. To address this issue, some courts have adopted speech typing, which involves court reporters speaking dialogue into a speech device in real time for accurate transcription.

Speech Recognition Use in Education

With the vast amount of educational content available on YouTube, speech recognition has become a heavily used tool in education, allowing for automated closed captioning that continues to improve.

Educators and professionals in various fields have used this technology by uploading videos for quick and free transcription.

Furthermore, speech recognition technology can transcribe lectures, providing students with an efficient way to keep notes.

With the ability to transcribe spoken words in real-time, students can focus on understanding the material rather than worrying about taking notes.

What are the Advantages of Speech Recognition?

There is a number of speech recognition advantages, which are a driving force behind the growing interest in this field.

The benefits of speech recognition technology include:

Machine-to-Human Communication

Speaking instead of typing every letter can significantly speed up the process, making it ideal for interpersonal and human-computer interaction and communication.

We all rely on virtual assistants on our phones to send texts or make calls with just a few simple voice commands.

Hands-Free Technology

Completing tasks without using our hands has become increasingly important in today's fast-paced world, where multitasking is essential. This is where hands-free technology comes in handy.

For instance, voice search enables us to access information on the go and even have it read out loud by a digital assistant. This not only saves us time but also makes life more convenient.

Accessibility

Text-to-speech systems allow visually impaired users to have text read out loud. In contrast, speech-to-text systems make it possible for individuals with hearing difficulties to read transcriptions of spoken words.

Advanced audio transcription software like Google Meet can even provide real-time captions in multiple languages by translating speech on the fly.

What are the Challenges of Speech Recognition Technology?

Despite the ever-expanding list of the benefits and applications of speech recognition technology, its complexity also presents several challenges.

The challenges in speech recognition include:

Accuracy and Precision

Speech recognition faces challenges in both accuracy and precision. Accuracy refers to how well the software recognizes spoken words and transcribes them correctly. In contrast, precision refers to how well the software can distinguish between similar-sounding words or phrases.

For example, if someone says "there" instead of "their," the software must be able to recognize the correct word based on the context of the sentence. This requires a high level of precision.

Noise and Disturbances

Background noise, such as traffic, construction work, or conversations in the vicinity, can interfere with the user's voice signal, making it difficult for the software to distinguish the spoken words.

Similarly, disturbances in the environment, such as a sudden loud noise, can cause errors in the speech recognition process.

To overcome these challenges, speech recognition software uses various techniques, such as noise cancellation algorithms, to filter out background noise and enhance the accuracy of the user's voice signal.

However, these methods are not fool-proof and may only work effectively in some situations. Therefore, it is essential to use speech recognition technology in a controlled and quiet environment to ensure optimal performance.

Language and Accent Barriers

While speech recognition systems have come a long way in accurately recognizing spoken language, they still need help understanding accents and dialects that deviate significantly from the standard language models they were trained on.

This can be particularly problematic in multicultural or multilingual environments where different accents and dialects are prevalent.

For example, an English-speaking speech recognition system trained in American English may have difficulty accurately recognizing the accents of speakers from other English-speaking countries, such as the United Kingdom, Australia, or India.

In addition, speech recognition systems may also struggle with languages that have unique phonetic features or use tonal distinctions, such as Mandarin or Cantonese.

These languages require more advanced language models and algorithms to recognize spoken words and phrases accurately.

Privacy and Security

Speech recognition systems often process sensitive and personal information, such as passwords, credit card numbers, and private conversations. Therefore, protecting users' data privacy and preventing unauthorized access is crucial.

One of the primary privacy concerns with speech recognition is data collection and storage. Voice recordings may contain sensitive information, and the storage and use of these recordings can pose a risk to user privacy if not handled correctly.

Moreover, speech recognition technology may also face security challenges related to malicious attacks or breaches that could compromise sensitive data.

For instance, a hacker could gain access to a voice-controlled device or system and use it to gather information, such as login credentials or financial information.

To address these challenges, developers of speech recognition technology must incorporate privacy and security features in their products, such as encryption, secure data storage, and user control over data collection and deletion.

Future of Speech Recognition Technology

Speech recognition technology will continue to improve and become available to everyone. Humans and machines will work together to learn new words and styles of speaking.

Speech recognition systems will also follow responsible AI principles, which include:

- Fairness is crucial for recognizing speech regardless of someone's background or status. Therefore, reducing bias is important, and governments, businesses, and organizations are making efforts to identify and mitigate this.
- Explainability is also essential, and future systems will be transparent about collecting and analyzing data and their performance.
- Respecting privacy is essential, as voice is considered personal data. Measures exist to protect data, and new technology is being developed to safeguard privacy.

Companies deploying ASR systems will be accountable for using the technology responsibly and adhering to responsible AI principles.

Collaboration between humans and machines will be necessary to create a fair and liable future for speech recognition technology.

Speech Recognition: Key Takeaways

Speech recognition technology has evolved from a limited tool to an advanced algorithm that accurately transcribes natural language, making it a vital technology in today's fast-paced world. It allows for machine-to-human communication, hands-free technology, and audio transcription for accessibility.

However, it still faces challenges such as:

- Accuracy and precision,
- Noise and disturbances,
- Language and accent barriers,
- Privacy and security.

The future of speech recognition technology will involve collaboration between humans and machines and will follow responsible AI principles.

If you are interested in speech recognition technology, we encourage you to stay informed about its advancements, potential applications, and ethical implications.

Additionally, consider trying out different speech recognition software and exploring how it can make your life easier and more convenient.

Deep Learning

DR. PRAVEEN BLESSINGTON T
Professor
Email: praveentblessington@gmail.com
Mobile: 9730562120

Applications of Deep Learning

- **Computer Vision**:
  - Object Detection and Recognition
  - Image Classification
  - Facial Recognition
  - Autonomous Vehicles
- **Natural Language Processing (NLP)**:
  - Sentiment Analysis
  - Language Translation
  - Chatbots and Virtual Assistants
  - Text Summarization
- **Speech Recognition**:
  - Voice Assistants (e.g., Siri, Alexa)
  - Transcription Services
  - Speaker Identification

Continues....

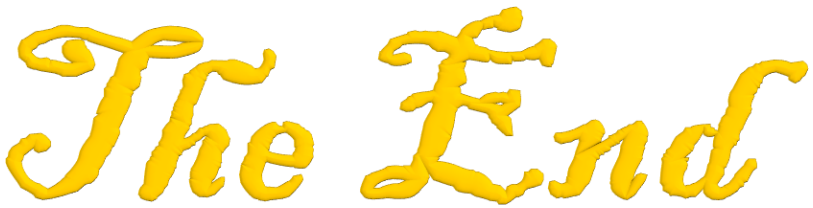
- **Healthcare**:
  - Medical Image Analysis
  - Disease Diagnosis
  - Drug Discovery
  - Predictive Analytics
- **Finance**:
  - Algorithmic Trading
  - Credit Scoring
  - Fraud Detection
  - Risk Assessment
- **Recommendation Systems**:
  - Movie and Product Recommendations (e.g., Netflix, Amazon)
  - Content Personalization
  - Music and News Recommendations
- **Gaming**:
  - Character Animation
  - Game Testing and Quality Assurance
  - Procedural Content Generation

Continues....

- **Autonomous Systems**:
  - Autonomous Drones
  - Robotics
  - Self-driving Cars
- **Manufacturing**:
  - Quality Control and Defect Detection
  - Predictive Maintenance
  - Supply Chain Optimization
- **Agriculture**:
  - Crop Monitoring
  - Pest Detection
  - Precision Agriculture
- **Environmental Science**:
  - Climate Modelling
  - Natural Disaster Prediction
  - Ecosystem Monitoring

Continues….

- **Marketing and Advertising:**
  - Customer Segmentation
  - Click-Through Rate Prediction
  - Ad Targeting
- **Security:**
  - Intrusion Detection
  - Video Surveillance
  - Biometric Authentication
- **Art and Creativity:**
  - Generative Adversarial Networks (GANs) for Art Generation
  - Style Transfer
  - Music Composition
- **Education:**
  - Personalized Learning
  - Automated Grading
  - Intelligent Tutoring Systems


The End