ISSN: 2321-7782 (Online)
Volume 4, Issue 4, April 2016
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

# Text Categorization by Distributional Features

**B. Vasundhara Devi**[1]
Asst. Prof, Computer Science and Engineering
SNIST (Autonomous)
Ghatkesar, Hyderbad – India

**A. Yaganteeswarudu**[2]
Asst. Prof, Computer Science and Engineering
SNIST (Autonomous)
Ghatkesar, Hyderbad – India

***Abstract:*** ***Text categorization (also known as text classification or topic spotting) is the task of automatically sorting a set of documents into categories from a predefined set. Automated text classification is attractive because it frees organizations from the need of manually organizing document bases, which can be too expensive, or simply not feasible given the time constraints of the application or the number of documents involved.***

***Earlier research in text categorization is concern about word frequency in given document which is not fully expressed the abundant information contained in the document. With the widely used "bag-of-word" representation, previous researches usually assign a word with values that express whether this word appears in the document concerned or how frequently this word appears. Although these values are useful for text categorization, they have not fully expressed the abundant information contained in the document.***

***Document classification/categorization is a problem in information science. The task is to assign an electronic document to one or more categories, based on its contents. Document classification tasks can be divided into two sorts: supervised document classification where some external mechanism (such as human feedback) provides information on the correct classification for documents, and unsupervised document classification, where the classification must be done entirely without reference to external information. There is also a semi-supervised document classification, where parts of the documents are labeled by the external mechanism.***

***Text mining, sometimes alternately referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the divining of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities).***

***Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too complex to describe generally in programming languages, so that in effect programs must automatically describe programs. Artificial intelligence is a closely related field, as also probability theory and statistics, data mining, pattern recognition, adaptive control, and theoretical computer science.***

 

***This paper explores the effect of other types of values, which express the distribution of a word in the document. While the frequency of a word expresses the intuition that the more frequent, the more important, the compactness of the appearances of a word shows that the less compact, the more important and the position of the first appearance of a word shows that the earlier, the more important. In order to test the effect of these distributional features, kNN and SVM are used. The tf–idf weight (term frequency–inverse document frequency) is a weight often used in information retrieval and text mining. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. We are extending these existing models, where we perform text categorization based on term frequency, document frequency and word distribution features and distribution frequency. This approach would find relation between content to categorize.***

***Keywords: k-N;tf-d;SVM;distribution frequency; Machine learning.***

## I. INTRODUCTION

### Document classification

Document classification or document categorization is a problem in library science, information science and computer science. The task is to assign a document to one or more classes or categories. This may be done "manually" (or "intellectually") or algorithmically. The intellectual classification of documents has mostly been the province of library science, while the algorithmic classification of documents is mainly in information science and computer science. The problems are overlapping, however, and there is therefore interdisciplinary research on document classification. The documents to be classified may be texts, images, music, etc. Each kind of document possesses its special classification problems. When not otherwise specified, text classification is implied.

Documents may be classified according to their subjects or according to other attributes (such as document type, author, printing year etc.). In the rest of this article only subject classification is considered. There are two main philosophies of subject classification of documents: The content based approach and the request based approach.

"Content based" versus "request based" classification

Content based classification is classification in which the weight given to particular subjects in a document determines the class to which the document is assigned. It is, for example, a rule in much library classification that at least 20% of the content of a book should be about the class to which the book is assigned. In automatic classification it could be the number of times given words appears in a document.

Request oriented classification (or -indexing) is classification in which the anticipated request from users is influencing how documents are being classified. The classifier asks himself: “Under which descriptors should this entity be found?” and “think of all the possible queries and decide for which ones the entity at hand is relevant”. Only if empirical data about use or users are applied should request oriented classification be regarded as a user-based approach.

### Classification versus indexing

Sometimes a distinction is made between assigning documents to classes ("classification") versus assigning subjects to documents ("subject indexing") but as Frederick Wilfrid Lancaster has argued, this distinction is not fruitful. "These terminological distinctions,” he writes, “are quite meaningless and only serve to cause confusion” .The view that this distinction is purely superficial is also supported by the fact that a classification system may be transformed into a thesaurus and vice versa. Therefore is the act of labeling a document (say by assigning a term from a controlled vocabulary to a document) at the same time to assign that document to the class of documents indexed by that term (all documents indexed or classified as X belong to the same class of documents).

 

**Automatic document classification (ADC)**

Automatic document classification tasks can be divided into three sorts: supervised document classification where some external mechanism (such as human feedback) provides information on the correct classification for documents, unsupervised document classification (also known as document clustering), where the classification must be done entirely without reference to external information, and semi-supervised document classification, where parts of the documents are labeled by the external mechanism. There are several software products under various license models available.

## II. K-NN (K-NEAREST NEIGHBOURS ALGORITHM)

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor. The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data. The algorithm has nothing to do with and is not to be confused with k-means, another popular machine learning technique.

**Algorithm**

Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). In the context of gene expression microarray data, for example, k-NN has also been employed with correlation coefficients such as Pearson and Spearman.Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest

 

neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. K-NN can then be applied to the SOM.

**Parameter selection**

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques (see hyper parameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when k = 1) is called the nearest neighbor algorithm.

The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular[citation needed] approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes

In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal k in this setting is via bootstrap method.

**Properties**

k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel. The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets. Using an appropriate nearest neighbor search algorithm makes k-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed.k-NN has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). k-NN is guaranteed to approach the Bayes error rate for some value of k (where k increases as a function of the number of data points). Various improvements to k-NN are possible by using proximity graphs.

**Metric Learning**

The K-nearest neighbor classification performance can often be significantly improved through (supervised) metric learning. Popular algorithms are Neighbourhood components analysis and Large margin nearest neighbor. Supervised metric learning algorithms use the label information to learn a new metric or pseudo-metric.

**Feature extraction**

When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying k-NN algorithm on the transformed data in feature space.

 

An example of a typical computer vision computation pipeline for face recognition using k-NN including feature extraction and dimension reduction pre-processing steps (usually implemented with OpenCV):

Haar face detection

Mean-shift tracking analysis

PCA or Fisher LDA projection into feature space, followed by k-NN classification

**Dimension reduction**

For high-dimensional data (e.g., with number of dimensions more than 10) dimension reduction is usually performed prior to applying the k-NN algorithm in order to avoid the effects of the curse of dimensionality. The curse of dimensionality in the k-NN context basically means that Euclidean distance is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector (imagine multiple points lying more or less on a circle of with the query point at the center; the distance from the query to all data points in the search space is almost the same).Feature extraction and dimension reduction can be combined in one step using principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) techniques as a pre-processing step, followed by clustering by k-NN on feature vectors in reduced-dimension space. In machine learning this process is also called low-dimensional embedding.For very-high-dimensional datasets (e.g. when performing a similarity search on live video streams, DNA data or high-dimensional time series) running a fast approximate k-NN search using locality sensitive hashing, "random projections","sketches" or other high-dimensional similarity search techniques from VLDB toolbox might be the only feasible option.

**Decision boundary**

Nearest neighbor rules in effect implicitly compute the decision boundary. It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity.

**Data reduction**

Data reduction is one of the most important problems for work with huge data sets. Usually, only some of the data points are needed for accurate classification. Those data are called the prototypes and can be found as follows:

Select the class-outliers, that is, training data that are classified incorrectly by k-NN (for a given k)

Separate the rest of the data into two sets: (i) the prototypes that are used for the classification decisions and (ii) the absorbed points that can be correctly classified by k-NN using prototypes. The absorbed points can then be removed from the training set.

**Selection of class-outliers**

A training example surrounded by examples of other classes is called a class outlier. Causes of class outliers include:

Random error

Insufficient training examples of this class (an isolated example appears instead of a cluster)

Missing important features (the classes are separated in other dimensions which we do not know)

Too many training examples of other classes (unbalanced classes) that create a "hostile" background for the given small

**Class**

Class outliers with k-NN produce noise. They can be detected and separated for future analysis. Given two natural numbers, k>r>0, a training example is called a (k,r)NN class-outlier if its k nearest neighbors include more than r examples of other classes.

 

**CNN for data reduction**

Condensed nearest neighbor (CNN, the Hart algorithm) is an algorithm designed to reduce the data set for k-NN classification. It selects the set of prototypes U from the training data, such that 1NN with U can classify the examples almost as accurately as 1NN does with the whole data set.

Calculation of the border ratio.

Three types of points: prototypes, class-outliers, and absorbed points.

Given a training set X, CNN works iteratively:

Scan all elements of X, looking for an element x whose nearest prototype from U has a different label than x.

Remove x from X and add it to U

Repeat the scan until no more prototypes are added to U.

Use U instead of X for classification. The examples that are not prototypes are called "absorbed" points.

It is efficient to scan the training examples in order of decreasing border ratio. The border ratio of a training example x is defined as

$$a(x) = ||x'-y|| / ||x-y||$$

where ||x-y|| is the distance to the closest example y having a different color than x, and ||x'-y|| is the distance from y to its closest example x' with the same label as x.

The border ratio is in the interval [0,1] because ||x'-y|| never exceeds ||x-y||. This ordering gives preference to the borders of the classes for inclusion in the set of prototypes'. A point of a different label than x is called external to x. The calculation of the border ratio is illustrated by the figure on the right. The data points are labeled by colors: the initial point is x and its label is red. External points are blue and green. The closest to x external point is y. The closest to y red point is x' .

The border ratio a(x)=||x'-y||/||x-y|| is the attribute of the initial point x.

**k-NN regression**

In k-NN regression, the k-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the k nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

Compute the Euclidean or Mahalanobis distance from the query example to the labeled examples.

Order the labeled examples by increasing distance.

Find a heuristically optimal number k of nearest neighbors, based on RMSE. This is done using cross validation.

Calculate an inverse distance weighted average with the k-nearest multivariate neighbors.

**Validation of results**

A confusion matrix or "matching matrix" is often used as a tool to validate the accuracy of k-NN classification. More robust statistical methods such as likelihood-ratio test can also be applied.

## III. TF-IDF

tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus:8 It is often used as a weighting factor in information retrieval and

 

text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

**Term frequency**

Suppose we have a set of English text documents and wish to determine which document is most relevant to the query "the brown cow". A simple way to start out is by eliminating documents that do not contain all three words "the", "brown", and "cow", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document and sum them all together; the number of times a term occurs in a document is called its term frequency.

The first form of term weighting is due to Hans Peter Luhn (1957) and is based on the Luhn Assumption:

The weight of a term that occurs in a document is simply proportional to the term frequency.

Inverse document frequency

However, because the term "the" is so common, this will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "brown" and "cow". The term "the" is not a good keyword to distinguish relevant and non-relevant documents and terms, unlike the less common words "brown" and "cow". Hence an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Karen Spärck Jones (1972) conceived a statistical interpretation of term specificity called IDF, which became a cornerstone of term weighting:

The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.

**Definition**

tf–idf is the product of two statistics, term frequency and inverse document frequency. Various ways for determining the exact values of both statistics exist.

**Term frequency**

Variants of TF weight

| weighting scheme | TF weight |
|---|---|
| binary | {0,1} |
| raw frequency | $f_{t,d}$ |
| log normalization | $1 + \log f_{t,d}$ |
| double normalization 0.5 | $0.5 + 0.5 * (f_{t,d}) / \max(f_{t,d})$ |
| double normalization K | $K + (1 - K) *(f_{t,d}) / \max(f_{t,d})$ |

In the case of the term frequency tf(t,d), the simplest choice is to use the raw frequency of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw frequency of t by f(t,d), then the simple tf scheme is tf(t,d) = f(t,d). Other possibilities include: 128

Boolean "frequencies": tf(t,d) = 1 if t occurs in d and 0 otherwise;

logarithmically scaled frequency: tf(t,d) = 1 + log f(t,d), or zero if f(t, d) is zero;

 

augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

tf (t,d)= 0.5 +( 0.5 *f(t,d))/(max(f(w,d))

**Inverse document frequency**

Variants of IDF

| weight weighting scheme | IDF weight |
|---|---|
| unary | 1 |
| inverse frequency | $\log^{N/nt}$ |
| inverse frequency smooth | log (1 + N /nt) |
| inverse frequency max | log $(1 + \max_{t*} n_t / n_t)$ |
| probabilistic inverse frequency | $\log^{(N-nt)/nt}$ |

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

## IV. LINEAR CLASSIFIERS: WHICH HYPER PLANE?

Lots of possible solutions for *a, b, c.*

Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]

E.g., perceptron

Support Vector Machine (SVM) finds an optimal* solution.

Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary

One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Support Vector Machine (SVM) was first heard in 1992, introduced by Boser, Guyon, and Vapnik in COLT-92. Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support Vector machines can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Support vector machine was initially popular with the NIPS community and now is an active part of the machine learning research around the world. SVM becomes famous when, using pixel maps as input; it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task. It is also being used for many applications, such as hand writing

 

analysis, face analysis and so forth, especially for pattern classification and regression based applications. The foundations of Support Vector Machines (SVM) have been developed by Vapnik and gained popularity due to many promising features such as better empirical performance. The formulation uses the Structural Risk Minimization (SRM) principle, which has been shown to be superior, to traditional Empirical Risk Minimization (ERM) principle, used by conventional neural networks. SRM minimizes an upper bound on the expected risk, where as ERM minimizes the error on the training data. It is this difference which equips SVM with a greater ability to generalize, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to solve regression problems.

**Statistical Learning Theory**

The statistical learning theory provides a framework for studying the problem of gaining knowledge, making predictions, making decisions from a set of data. In simple terms, it enables the choosing of the hyper plane space such a way that it closely represents the underlying function in the target space.

In statistical learning theory the problem of supervised learning is formulated as follows. We are given a set of training data $\{(\mathbf{x}_1,y_1)\ldots(\mathbf{x}_l,y_l)\}$ in $R^n$ ´ R sampled according to unknown probability distribution P(**x**,y), and a loss function V(y,f(**x**)) that measures the error, for a given **x**, f(**x**) is "predicted" instead of the actual value yIn statistical modeling we would choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space. More on statistical learning theory can be found on introduction to statistical learning theory.

**Learning and Generalization**

Early machine learning algorithms aimed to learn representations of simple functions. Hence, the goal of learning was to output a hypothesis that performed the correct classification of the training data and early learning algorithms were designed to find such an accurate fit to the data. The ability of a hypothesis to correctly classify data not in the training set is known as its generalization. SVM performs better in term of not over generalization when the neural networks might end up over generalizing easily. Another thing to observe is to find where to make the best trade-off in trading complexity with the number of epochs; the illustration brings to light more information about this. The below illustration is made from the class notes.

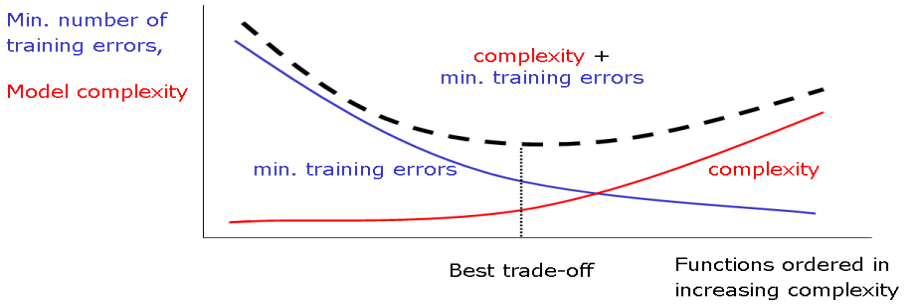


Figure 1: Number of Epochs Vs Complexity.

## V. INTRODUCTION TO SVM: WHY SVM?

Firstly working with neural networks for supervised and unsupervised learning showed good results while used for such learning applications. MLP's uses feed forward and recurrent networks. Multilayer perceptron (MLP) properties include universal approximation of continuous nonlinear functions and include learning with input-output patterns and also involve advanced network architectures with multiple inputs and outputs.

 

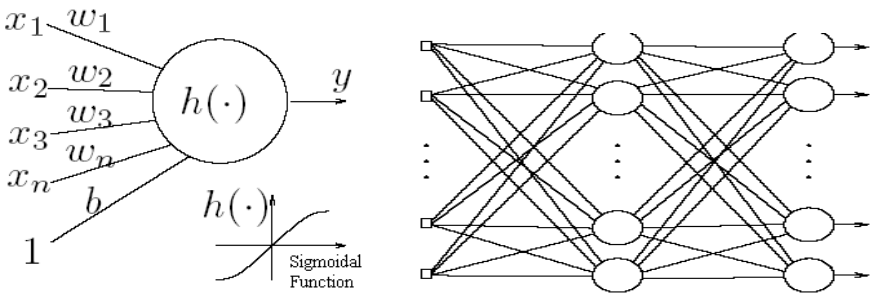


Figure 2: a] Simple Neural Network b] Multilayer Perceptron. These are simple visualizations just to have a overview as how neural network looks like.

There can be some issues noticed. Some of them are having many local minima and also finding how many neurons might be needed for a task is another issue which determines whether optimality of that NN is reached. Another thing to note is that even if the neural network solutions used tends to converge, this may not result in a unique solution. Now let us look at another example where we plot the data and try to classify it and we see that there are many hyper planes which can classify it. But which one is better?

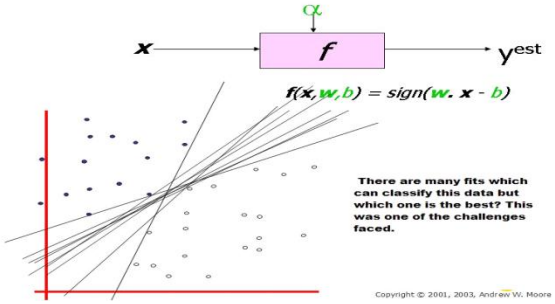


Figure 3: Here we see that there are many hyper planes which can be fit in to classify the data but which one is the best is the right or correct solution. The need for SVM arises. (Taken Andrew W. Moore 2003). Note the legend is not described as they are sample plotting to make understand the concepts involved.

From above illustration, there are many linear classifiers (hyper planes) that separate the data. However only one of these achieves maximum separation. The reason we need it is because if we use a hyper plane to classify, it might end up closer to one set of datasets compared to others and we do not want this to happen and thus we see that the concept of maximum margin classifier or hyper plane as an apparent solution. The next illustration gives the maximum margin classifier example which provides a solution to the above mentioned problem.

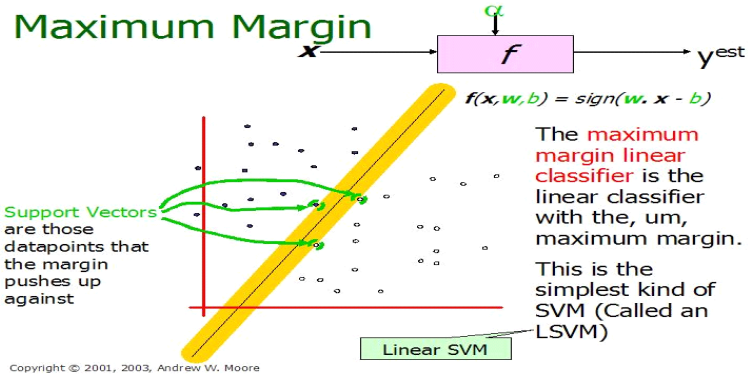


Figure 4: Illustration of Linear SVM.

The above illustration is the maximum linear classifier with the maximum range. In this context it is an example of a simple linear SVM classifier. Another interesting question is why maximum margin? There are some good explanations which include better empirical performance. Another reason is that even if we've made a small error in the location of the boundary this gives us least chance of causing a misclassification. The other advantage would be avoiding local minima and better classification.

 

Now we try to express the SVM mathematically and for this tutorial we try to present a linear SVM. The goals of SVM are separating the data with hyper plane and extend this to non-linear boundaries using kernel trick.

## References

1. http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf
2. Naik K (2010) A Survey of Software Based Energy Saving Methodologies for Handheld Wireless Communication Devices. Tech. Report No. 2010-13. Dept. of ECE, University of Waterloo viii. V. Venkatachalam, M. Franz. "Power reduction techniques for microprocessor systems". ACM Computer Survey, Sempember 2005.
3. http://nlp.stanford.edu/IR-book/html/htmledition/linear-versus-nonlinear-classifiers-1.html
4. U S Department of Energy March 2011 "Best Practices Guide for Energy Efficient Data Center Design" by National Renewable Energy Laboratory.
5. Mehtal H, Owens RM, Irwin MJ, Chen R, Ghosh D (1997) Techniques for Low Energy Software. In: Proceedings of the 1997 international symposium on Low power electronics and design (ISLPED '97). ACM, New York, NY, USA, pp 72–75.
6. B.Zhai et al. "Energy efficient near-threshold chip multiprocessing". Proceedings of the 2007 Int'l Symp on Low Power Electronics and Design Series, ISLPED2007.
7. "Quickly build and deploy Software as a Service applications", Ironspeed Inc. white paper
8. "Tutorials Windows Azure" http://www.windowsazure.com/en-us/develop/net/tutorials/getstarted

## AUTHOR(S) PROFILE

**B. Vasundhara Devi** , Working as assistant professor in CSE department in Sreenidhi institute science and technology (Autonomous), Ghatkesar, Hyderbad.Having 4 Years of experience in teaching. Completed M.Tech in Avanthi institute of engineering and technology, Visakapatnam in 2010.

**A. Yaganteeswarudu,** Working as assistant professor in CSE department in Sreenidhi institute of science and technology (Autonomous), Ghatkesar, Hyderabad. Having 6 Years of experience in teaching. Completed M.Tech in SJCET, Kurnool in 2012.

 