**MINOR PROJECT**

**"Predicting and Evaluating the Popularity of Online News"**

Report submitted in partial fulfillment of the requirements for the award of

**Degree of Bachelor of Technology**

**in**

**Software Engineering (SE)**

Under the supervision of

**Prof. Kusum Lata**

**(Assistant Professor, CSE)**

By:

**Arjun Rajpal (2K14/SE/021)**

**Arpit Jain (2K14/SE/022)**

**Avinav Goel (2K14/SE/024)**

To**:**



**Department of Computer Science and Engineering**

**Delhi Technological University**

**(Formerly Delhi College of Engineering)**

**DECLARATION**

I hereby certify that the work which is presented in the Minor Project entitled **“*Predicting and Evaluating the Popularity of Online News*”** in fulfillment of the requirement for the award of the Degree of Bachelor of Technology and submitted to the Department of Computer Engineering, Delhi Technological University (Formerly Delhi College Of Engineering), New Delhi is an authentic record of my own, carried out during a period from August 2016 to November 2016, under the supervision of **Prof. Kusum Lata, Assistant Professor, CSE Department.**

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University.

**Signature**

**ARJUN RAJPAL 2K14/SE/021**

**ARPIT JAIN 2K14/SE/022**

**AVINAV GOEL 2K14/SE/024**

**ACKNOWLEDGEMENT**

“The successful completion of any task would be incomplete without accomplishing the people who made it all possible and whose constant guidance and encouragement secured us the success.”

First of all, we are grateful to the Almighty for establishing us to complete this minor project. We are grateful to **Prof. Kusum Lata, Assistant Professor** (Department of Computer Science and Engineering), Delhi Technological University (Formerly Delhi College of Engineering), New Delhi and all other faculty members of our department, for their astute guidance, constant encouragement and sincere support for this project work.

We owe a debt of gratitude to our guide, **Prof. Kusum Lata, CSE Department** for incorporating in us the idea of a creative Minor Project, helping us in undertaking this project and also for being there whenever we needed her assistance.

I also place on record, my sense of gratitude to one and all, who directly or indirectly have lent their helping hand in this venture. We feel proud and privileged in expressing my deep sense of gratitude to all those who have helped me in presenting this project.

Last but never the least, we thank our parents for always being with us, in every sense.

**SUPERVISOR CERTIFICATE**

This is to certify that **ARJUN RAJPAL 2K14/SE/021, ARPIT JAIN 2K14/SE/022** and **AVINAV GOEL 2K14/SE/024,** the bonafide students of **Bachelor of Technology in Software Engineering** of **Delhi Technological University** (Formerly Delhi College Of Engineering), New Delhi of **2014–2018 batch** have completed their minor project entitled “***Predicting and Evaluating the Popularity of Online News***” under the supervision of **Prof. Kusum Lata, Assistant Professor, CSE DEPARTMENT**.

It is further certified that the work done in this dissertation is a result of candidate’s own efforts.

I wish his/her all success in her life.

Date:

**Prof. Kusum Lata**

Assistant Professor

Computer Science & Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Shahbad, Daulatpur, Bawana Road, Delhi – 110042

**ABSTRACT**

Consider the situation where an online news publishing agency is browsing submissions of news articles, but can only accept a few without going over budget or without overwhelming the audience. How does the agency determine which news articles will become popular or even viral, and which news articles will be mostly ignored by the general public? Are there any predictors that indicate how many times an article will be shared amongst audiences? Which articles should the agency purchase and publish?  
Implied in this question is the classification problem of binning. Our class variable, the number of shares, is a metric that defines how often an article is shared on social media, but it is a continuous variable and so its binning is not obvious. However, we are interested primarily in articles with high popularity as our positive class, more so than articles with low popularity. So how do we bin a numerical attribute into classes of ‘obscure’, ‘mediocre’, popular’, ‘viral’, etc.?

With the expansion of the Internet, more and more people enjoys reading and sharing online news articles. The number of shares under a news article indicates how popular the news is. In this project, we intend to find the best model and set of feature to predict the popularity of online news, using machine learning techniques. Our data comes from Mashable, a well-known online news website. We implemented various learning algorithms on the dataset, ranging from various regressions to SVM. Their performances are recorded and compared. Feature selection method has been used to improve performance and reduce features. SVM turns out to be the best model for prediction, and it can achieve an accuracy of 67% with optimal parameters. Our work can help online news companies to determine news popularity before publication.

**TABLE OF CONTENTS**

*Declaration…………………………………………………………………………………………………………..i*

*Acknowledgement……………………………………………………………………………………………… ii*

*Certificate………………………………………………………………………………………………………….. iii*

*Abstract……………………………………………………………………………………………………………..iv*

*List of Figures…………………………………………………………………………………………………….ix*

1. INTRODUCTION 1

1.1 OBJECTIVE 1

2. DATA MINING 3

2.1 STEPS IN DATA MINING 4

2.2 DATA MINING FUNCTIONALITIES 5

3. DATA PROCESSING 7

3.1 DATA EXTRACTION 7

3.2 DATA CLEANING 7

3.2.1 MISSING VALUES 8

3.2.2 NOISY VALUES 9

3.3 DATA TRANSFORMATION 9

3.4 DATA REDUCTION 10

3.5 DATA INTEGRATION 10

3.6 GRAPHICAL REDUCTION 12

4. MACHINE LEARNING ALGORITHMS SELECTION AND ASSESSMENT 12

4.1 CLASSIFICATION PROBLEM 12

4.1.1 BINARY AND MULTICLASS CLASSIFICATION 12

4.1.2 COMPARING CLASSIFICATION METHODS 12

4.2 CLASSIFICATION ALGORITHMS 13

4.2.1 K NEAREST NEIGHBOURS 13

4.2.1.1 ALGORITHMS AND PSEUDO-CODE 14

4.2.2 K-MEANS CLUSTERING 14

4.2.2.1 ALGORITHM AND PSEUDO-CODE 15

4.2.3 NAIVE BAYES CLASSIFIER 15

4.2.3.1 PROBABILISTIC MODEL 16

4.2.3.2 PSEUDO - CODE 18

4.2.4 LOGISTIC REGRESSION 19

4.2.4.1 HYPOTHESIS FUNCTION 19

4.2.4.2 PSEUDO - CODE 20

4.2.5 DECISION TREE CLASSIFIER 20

4.2.5.1 COMPLEXITY ANALYSIS 21

4.2.5.2 C4.5 DECISION TREE ALGORITHM 22

4.2.5.3 C4.5 DECISION TREE PSEUDO CODE 23

4.2.6 SUPPORT VECTOR MACHINE (SVM) 24

4.2.6.1 LINEAR SVM 25

4.2.6.2 NON LINEAR SVM 25

4.2.6 EVALUATION OF BINARY CLASSIFIERS 26

4.2.6.1 SENSITIVITY AND SPECIFICITY 26

4.2.6.2 POSITIVE AND NEGATIVE DECISION VALUES 26

4.2.6.3 SINGLE METRICS 27

5. MULTI INSTANCE LEARNING 28

5.1 INTRODUCTION 28

5.2 OVERVIEW OF PARADIGMN 29

5.2.1 INSTANCE SPACE PARADIGMN 30

5.2.2 BAG OF SPACE PARADIGMN 31

5.2.3 EMBEDDED SPACE PARADIGMN 33

6. HYBRID ALGORITHMS 35

6.1 K-Means algorithm ..35

6.2 Naïve Bayesian Classifier ..37

6.3 K-Means with Naïve Bayes ..38

* 1. Decision Tree……………………………………………………………………………………………..38
  2. Decision Tree with K-Means……………………………………………………………………….39
  3. Photo Classifier…………………………………………………………………………………………..39

7. MACHINE LEARNING TOOLS and IDE………………………..........................................42

7.1 Development Software…… 42

7.2 IDE. .43

7.3 Software Library ..43

8. PROPOSED METHODOLOGY……….……………………………........................................45

* 1. Feature Extraction….….……...........................................................................45
  2. Bag of Words representation………………………………………………………………… ..45
  3. Label Classification… ..46
  4. Multi Label Training……………….…….……………………………………………………………47
  5. Association Rules……………………………………………………………………………………..47

9. RESULTS, ACCURACY COMPARISON AND CONCLUSION…..……..............................49

* 1. Dataset……………………………………………………….................................................49

9.2 Implementation Details and Results…………………………………………………..………50

1. REFERENCES…………………………………………………………………………………………………53

**LIST OF FIGURES**

**Chapter 2 :**

1. Data Mining : http://www.saedsayad.com/images/DM.png
2. Data Mining Components : www.tutorialspoint.com/data\_mining/dm\_systems
3. Clustering Analysis : http://www.slideshare.net/NontawatB/08-clustering

**Chapter 3 :**

3.1 Histogram : www.statcrunch.com/5.0/viewresult.php?resid=1414715

**Chapter 4 :**

4.1 Plotting of Sigmoid Function : http://ufldl.stanford.edu/wiki/index.php/File:Sigmoid\_Function.png

**Chapter 6:**

5.1 Calculated Result ( Broken Assumption Non - Spherical Distribution of data) :

http://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means

5.2 Expected Result ( Broken Assumption Non - Spherical Distribution of data ) :

http://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means

5.3 Zero Probability Problem : http://www.slideshare.net/gladysCJ/lesson-71-naive-bayes-classifier?next\_slideshow=1

**Chapter 7 :**

6.1 Python(x,y) IDE : http://timothyandrewbarber.blogspot.in/2012/06/getting-python-to-work-in-windows.html

ix

**Chapter 1-INTRODUCTION**

**1.1** **OBJECTIVE**

The aim of the project is to ― ***Predict and Evaluate the Popularity of Online News*.**

**Description of the problem:**

The problem is to build a model that predicts the number of shares on Social Media and in turn help to determine the popularity of Online News articles. Currently media houses rely solely on human judgement to estimate the popularity of any article posted on the web.

Such a model is helpful for the media publishing houses to make important decisions like when to publish an article, if a given article will be popular, which category is famous, etc.

**File/Dataset:**

The data set consists of 39,644 instances of articles published on Mashable.com from 2013 to 2014. The dataset contains 58 total attributes, ranging from the URL of the article, the number of keywords of specific types, what day of the week the article was published, the results of a Latent Dirichlet Allocation algorithm, and the dependent variable: the number of shares of the article. The data set was accessed at the UCI dataset repository referenced in the appendix. There are no missing values in the data set and no significant noise

Fields:

0. url: URL of the article

1. timedelta: Days between the article publication and

the dataset acquisition

2. n\_tokens\_title: Number of words in the title

3. n\_tokens\_content: Number of words in the content

4. n\_unique\_tokens: Rate of unique words in the content

5. n\_non\_stop\_words: Rate of non-stop words in the content

6. n\_non\_stop\_unique\_tokens: Rate of unique non-stop words in the

content

7. num\_hrefs: Number of links

8. num\_self\_hrefs: Number of links to other articles

published by Mashable

9. num\_imgs: Number of images

10. num\_videos: Number of videos

11. average\_token\_length: Average length of the words in the

content

12. num\_keywords: Number of keywords in the metadata

13. data\_channel\_is\_lifestyle: Is data channel 'Lifestyle'?

14. data\_channel\_is\_entertainment: Is data channel 'Entertainment'?

15. data\_channel\_is\_bus: Is data channel 'Business'?

16. data\_channel\_is\_socmed: Is data channel 'Social Media'?

17. data\_channel\_is\_tech: Is data channel 'Tech'?

18. data\_channel\_is\_world: Is data channel 'World'?

19. kw\_min\_min: Worst keyword (min. shares)

20. kw\_max\_min: Worst keyword (max. shares)

21. kw\_avg\_min: Worst keyword (avg. shares)

22. kw\_min\_max: Best keyword (min. shares)

23. kw\_max\_max: Best keyword (max. shares)

24. kw\_avg\_max: Best keyword (avg. shares)

25. kw\_min\_avg: Avg. keyword (min. shares)

26. kw\_max\_avg: Avg. keyword (max. shares)

27. kw\_avg\_avg: Avg. keyword (avg. shares)

28. self\_reference\_min\_shares: Min. shares of referenced articles in

Mashable

29. self\_reference\_max\_shares: Max. shares of referenced articles in

Mashable

30. self\_reference\_avg\_sharess: Avg. shares of referenced articles in

Mashable

31. weekday\_is\_monday: Was the article published on a Monday?

32. weekday\_is\_tuesday: Was the article published on a Tuesday?

33. weekday\_is\_wednesday: Was the article published on a Wednesday?

34. weekday\_is\_thursday: Was the article published on a Thursday?

35. weekday\_is\_friday: Was the article published on a Friday?

36. weekday\_is\_saturday: Was the article published on a Saturday?

37. weekday\_is\_sunday: Was the article published on a Sunday?

38. is\_weekend: Was the article published on the weekend?

39. LDA\_00: Closeness to LDA topic 0

40. LDA\_01: Closeness to LDA topic 1

41. LDA\_02: Closeness to LDA topic 2

42. LDA\_03: Closeness to LDA topic 3

43. LDA\_04: Closeness to LDA topic 4

44. global\_subjectivity: Text subjectivity

45. global\_sentiment\_polarity: Text sentiment polarity

46. global\_rate\_positive\_words: Rate of positive words in the content

47. global\_rate\_negative\_words: Rate of negative words in the content

48. rate\_positive\_words: Rate of positive words among non-neutral

tokens

49. rate\_negative\_words: Rate of negative words among non-neutral

tokens

50. avg\_positive\_polarity: Avg. polarity of positive words

51. min\_positive\_polarity: Min. polarity of positive words

52. max\_positive\_polarity: Max. polarity of positive words

53. avg\_negative\_polarity: Avg. polarity of negative words

54. min\_negative\_polarity: Min. polarity of negative words

55. max\_negative\_polarity: Max. polarity of negative words

56. title\_subjectivity: Title subjectivity

57. title\_sentiment\_polarity: Title polarity

58. abs\_title\_subjectivity: Absolute subjectivity level

59. abs\_title\_sentiment\_polarity: Absolute polarity level

60. shares: Number of shares (target)

**Chapter 2-DATA MINING**

**Replace chapter 2 &3 with PROJECT FILE.pdf chapter 2&3// TODO**

**Newocr.com**

Data Mining is about explaining the past and predicting the future by means of data analysis. Data mining is a multi-disciplinary field which combines statistics, machine learning, artificial intelligence and database technology. The value of data mining applications is often estimated to be very high. Many businesses have stored large amounts of data over years of operation, and data mining is able to extract very valuable knowledge from this data. The businesses are then able to leverage the extracted knowledge into more clients, more sales, and greater profits. This is also true in the engineering and medical fields.

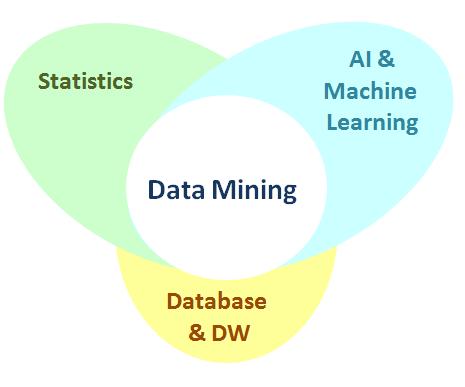


Figure 2.1 Data Mining

**Statistics**

The science of collecting, classifying, summarizing, organizing, analyzing, and interpreting data.

**Artificial Intelligence**

The study of computer algorithms dealing with the simulation of intelligent behaviors in order to perform those activities that are normally thought to require intelligence.

**Machine Learning**

The study of computer algorithms to learn in order to improve automatically through experience.

**Database**

The science and technology of collecting, storing and managing data so users can retrieve, add, update or remove such data.

**Data warehousing**

The science and technology of collecting, storing and managing data with advanced multi-dimensional reporting services in support of the decision making processes.

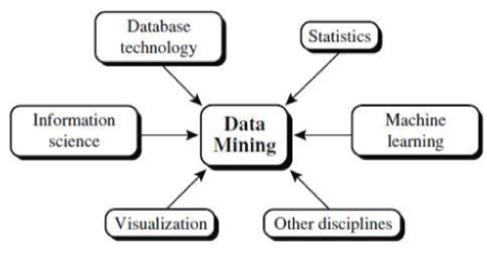


Figure 2.2 Data Mining Components

**2.1 Steps of Data Mining**

 **Data Integration:** First of all the data are collected and integrated from all thedifferent sources.

 **Data Selection:** We may not all the data we have collected in the first step. So in thisstep we select only those data which we think useful for data mining.

 **Data Cleaning:** The data we have collected are not clean and may contain errors,missing values, noisy or inconsistent data. So we need to apply different techniques to get rid of such anomalies.

 **Pattern** **Evaluation** **and** **Knowledge** **Presentation:** This step involves visualization, transformation, removing redundant patterns etc from the patterns we generated.

 **Data Transformation:** The data even after cleaning are not ready for mining as weneed to transform them into forms appropriate for mining. The techniques used to accomplish this are smoothing, aggregation, normalization etc.

 **Data Mining:** Now we are ready to apply data mining techniques on the data todiscover the interesting patterns. Techniques like clustering and association analysis are among the many different techniques used for data mining.

 **Decisions / Use of Discovered Knowledge:** This step helps user to make use of

the knowledge acquired to take better decisions.

**2.2 DATA MINING FUNCTIONALITIES**

Data mining systems should also allow users to specify hints to guide or focus thesearch for interesting patterns. Since some patterns may not hold for all of the data the database,

a measure of certainty or "trustworthiness" is usually associated with each discovered pattern.

Major Data mining functionalities, and the kinds of patterns are:

**1. Classification and prediction:**

**Classification** is a data mining function that assigns items in a collection to target categories orclasses. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to classify patients as heart patient and non - heart patients.

**Prediction (forecasting) / Estimate:** prediction analysis is related to regression techniques. Themain idea of predictive analysis is to discover the relationships between dependent and independent variables and the relationships between independent variables. For example, if sales is an independent variable, the beneficiation can be a dependent variable.

**2. Clustering analysis:**

**Clustering** is a process of partitioning a set of data (or objects) into a set of meaningful sub-classes, called **clusters**.

Help users understand the natural grouping or structure in a data set. Used either as a stand-alone tool to get insight into data distribution or as a preprocessing step for other algorithms.

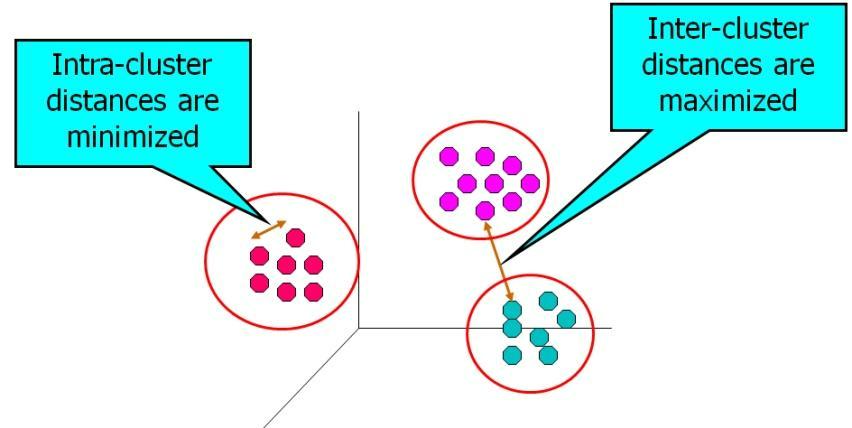


Figure 2.3 Clustering Analysis

**Chapter-3 DATA PROCESSING**

**Proj\_File**

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

**3.1 DATA EXTRACTION:**

**Data extraction** is the act or process of retrievingdataout of (usuallyunstructuredor poorly structured) data sources for further data processing or data storage (data migration). The import into the intermediate extracting system is thus usually followed by data transformation and possibly the addition of metadata prior to export to another stage in the data workflow.

Typical unstructured data sources include web pages, emails, documents, PDFs, scanned text, mainframe reports, spool files etc. Extracting data from these unstructured sources has grown into a considerable technical challenge where as historically data extraction has had to deal with changes in physical hardware formats, the majority of current data extraction deals with extracting data from these unstructured data sources, and from different software formats. This growing process of data extraction from the web is referred to as **Web scraping**.

**3.2 DATA CLEANING:**

Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.

**3.2.1 Missing Values**

**1) Ignore the tuple**:

This is usually done when the class label is missing (assuming the mining task involves classification). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably. By ignoring the tuple, we do not make use of the remaining attribute’s values in the tuple. Such data could have been useful to the task at hand.

**2) Fill in the missing values manually**:

In general, this approach is time consuming and may not be feasible given a large data set with many missing values.

**3) Use a global constant to fill in the missing values:**

Replace all missing attribute values by the same constant such as a label like ―Unknown‖ or −∞. If missing values are replaced by, say, ―Unknown, then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of

―Unknown. Hence, although this method is simple, it is not foolproof.

**4) Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value:**

For normal data distributions, the mean can be used, while skewed data distribution should employ the median.

**5) Use the attribute mean or median for all samples belonging to the same class as the given tuple:**

For example, if predicting heart disease, we may replace the missing value with the mean age value for the patients. If the data distribution for a given class is skewed, the median value is a better choice.

**6) Use the most probable value to fill in the missing value:**

This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree

Methods 3 through 6 bias the data—the filled-in value may not be correct. Method 6, however, is a popular strategy. In comparison to the other methods, it uses the most information from the present data to predict missing values. By considering the other attribute’s values in its estimation of the missing value for income, there is a greater chance that the relationships between income and the other attributes are preserved.

**3.2.2 Noisy Data**

Noise is a random error or variance in a measured variable. Some of the data smoothing techniques are:

1. **Binning:** Binning methods smooth a sorted data value by consulting its ―neighborhood,‖that is, the values around it. The sorted values are distributed into a number of ―buckets,‖ or bins. Because binning methods consult the neighborhood of values, they perform local smoothing. Some binning strategies are smoothing by bin means, smoothing by bin medians or smoothing by bin boundaries.
2. **Regression:** Data smoothing can also be done by regression, a technique that conformsdata values to a function. Linear regression involves finding the ―best‖ line to fit two attributes (or variables) so that one attribute can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.
3. **Outlier analysis:** Outliers may be detected by clustering, for example, where similarvalues are organized into groups, or ―clusters.‖ Intuitively, values that fall outside of the set of clusters may be considered outliers

**3.3 DATA TRANSFORMATION**

In data transformation, the data are transformed or consolidated into forms appropriate for mining. Strategies for data transformation include the following:

1. **Smoothing**: It works to remove noise from the data. Techniques include binning,regression, and clustering.
2. **Attribute construction**: It is where new attributes are constructed and added from the givenset of attributes to help the mining process.
3. **Aggregation**: It is where summary or aggregation operations are applied to the data.
4. **Normalization**: It is where the attribute data are scaled so as to fall within a smaller range,such as −1.0 to 1.0, or 0.0 to 1.0.
5. **Discretization**: It is where the raw values of a numeric attribute (e.g. age) are replaced byinterval labels (e.g. 0–10, 11–20, etc.) or conceptual labels (e.g. youth, adult, senior). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a concept hierarchy for the numeric attribute.

**3.4 DATA REDUCTION**

Data reduction strategies include dimensionality reduction, numerosity reduction, and data compression

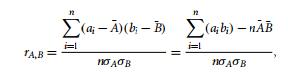
1. **Dimensionality reduction :** It is the process of reducing the number of random variablesor attributes under consideration. Dimensionality reduction methods include wavelet transforms and principal components analysis.
2. **Numerosity reduction techniques**: These techniques replace the original data volumeby alternative, smaller forms of data representation.
3. **Data compression**: These transformations are applied so as to obtain a reduced or

―compressed‖ representation of the original data. If the original data can be reconstructed from the compressed data without any information loss, the data reduction is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data reduction is called lossy.

**3.5 DATA INTEGRATION**

Data mining often requires data integration—the merging of data from multiple data stores. Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent data mining process. Some data integration strategies are:

1. **Entity Identification Problem:** There are a number of issues to consider during dataintegration. Schema integration and object matching can be tricky. How can equivalent real-world entities from multiple data sources be matched up? This is referred to as the entity identification problem. When matching attributes from one database to another during integration, special attention must be paid to the structure of the data. This is to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system.
2. **Redundancy and Correlation Analysis:** Redundancy is another important issue in dataintegration. An attribute may be redundant if it can be ―derived‖ from another attribute or set of attributes. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting d
3. ata set. Some redundancies can be detected by correlation analysis. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. For numeric attributes, correlation coefficient and covariance, both of which access how one attribute‘s values vary from those of another, can be used. For numeric attributes, we can evaluate the correlation between two attributes, A and B, by computing the correlation coefficient (also known as Pearson‘s product moment coefficient, named after its inventer, Karl Pearson). This is



where n is the number of tuples, ai and bi are the respective values of A and B in tuple i,

A¯ and B¯ are the respective mean values of A and B, σA and σB are the respective standard deviations of A and B, and sigma(ai bi) is the sum of the AB cross-product (i.e., for each tuple, the value for A is multiplied by the value for B in that tuple). Note that −1

≤ r(A,B) ≤ +1. If rA,B is greater than 0, then A and B are positively correlated, meaning that the values of A increase as the values of B increase. The higher the value, the stronger the. Hence, a higher value may indicate that A (or B) may be removed as a redundancy. If the resulting value is equal to 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A and B are negatively correlated, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other.

1. **Tuple Duplication**: In addition to detecting redundancies between attributes, duplicationshould also be detected at the tuple level. The use of denormalized tables (often done to improve performance by avoiding joins) is another source of data redundancy. Inconsistencies often arise between various duplicates, due to inaccurate data entry or updating some but not all data occurrences.

**Chapter-4 MACHINE LEARNING ALGORITHMS**

**SELECTION AND ASSESSMENT**

//TODO

Usage index from PROJECT FILE.pdf:

Chapter 4: Intro

4.1

4.2

4.3

4.4 Describe Linear Regression (SELF)

4.5 XXX(Leave)

4.6

4.7SVM(4.2.6 and 4.2.6.1, 4.2.6.2 from self docx)

4.8

**4.1 CLASSIFICATION PROBLEM**

In  [machine learning,](https://en.wikipedia.org/wiki/Machine_learning) classification is the problem of identifying to which of a set of  [categories](https://en.wikipedia.org/wiki/Categorical_data) a new  [observation](https://en.wikipedia.org/wiki/Observation) belongs, on the basis of a  [training set](https://en.wikipedia.org/wiki/Training_set) of data containing observations whose category membership is known.

Classification is a supervised learning problem i.e. problems in which correctly classified tuples are given and algorithm learns from these tuples. An algorithm that implements classification, especially in a concrete implementation, is known as a  [classifier.](https://en.wikipedia.org/wiki/Pattern_recognition) The term "classifier" sometimes also refers to the mathematical  [function,](https://en.wikipedia.org/wiki/Function_(mathematics)) implemented by a classification algorithm, that maps input data to a category.

**4.1.1 Binary and multiclass classification**

Classification can be thought of as two separate problems –  [binary classification](https://en.wikipedia.org/wiki/Binary_classification) and  [multiclas](https://en.wikipedia.org/wiki/Multiclass_classification)s  [classification.](https://en.wikipedia.org/wiki/Multiclass_classification) In binary classification, a better understood task, only two classes are involved, whereas multiclass classification involves assigning an object to one of several classes. Since many classification methods have been developed specifically for binary classification, multiclass classification often requires the combined use of multiple binary classifiers.

**4.1.2 Comparing classification methods**

Classification methods can be evaluated and compared according to the following criteria:

1. **Accuracy**: The accuracy of a classifier refers to its ability to correctly predict the lable ofunlabelled or previously unseen data or tuple. Higher the accuracy better will be the algorithm.
2. **Speed:** Speed of a classifier refers to how much classification time does the classificationalgorithm requires to complete the classification procedure.
3. **Robustness:** Robustness of a classifier is its ability to make correct classifications whenthe given data is noisy and has outliers.
4. **Scalability:** Scalability of a classifier is its ability to classify efficiently when largeamounts of data is given
5. **Interpretability:** This refers to the level of understanding and insight that is provided bythe classifier.

**4.2 CLASSIFICATION ALGORITHMS**

**4.2.1 K nearest neighbors**

The k-Nearest Neighbors algorithm is a  [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for  [classificatio](https://en.wikipedia.org/wiki/Statistical_classification)n and  [regression.](https://en.wikipedia.org/wiki/Regression_analysis) In both cases, the input consists of the k closest training examples in the  [featur](https://en.wikipedia.org/wiki/Feature_space)e  [space.](https://en.wikipedia.org/wiki/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,](https://en.wikipedia.org/wiki/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,](https://en.wikipedia.org/wiki/Instance-based_learning) or  [lazy learning,](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

Both for classification and regression, it can be useful to assign weight to 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,](https://en.wikipedia.org/wiki/K-means) another popular  [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique.

**4.2.1.1 Algorithm and pseudo code**

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](https://en.wikipedia.org/wiki/Feature_vector) 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](https://en.wikipedia.org/wiki/Continuous_variable) is  [Euclidean distance.](https://en.wikipedia.org/wiki/Euclidean_distance) For discrete

|  |  |
| --- | --- |
| variables, such as for text classification, another metric | can be used, such as the overlap |
| metric (or  [Hamming distance](https://en.wikipedia.org/wiki/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 Neighbo](https://en.wikipedia.org/wiki/Large_Margin_Nearest_Neighbor)r analysis.

Pseudo code:

1. Classify (X,Y,x) // X: training data, Y: class labels of X, x: unknown sample
2. for i=1 to m do
3. Compute distance d(Xi,x)
4. end for
5. Compute set I containing indices for the k smallest distance d(Xi,x).
6. Return majority label for { Yi where i belongs to I (integer set)}

**4.2.2 K-means Clustering**

K-means clustering is a method of  [vector quantization,](https://en.wikipedia.org/wiki/Vector_quantization) originally from signal processing, that is popular for  [cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis) in  [data mining.](https://en.wikipedia.org/wiki/Data_mining) k-means clustering aims to  [partition](https://en.wikipedia.org/wiki/Partition_of_a_set) n observations into k clusters in which each observation belongs to the cluster with the nearest  [mean,](https://en.wikipedia.org/wiki/Mean) serving as a  [prototype](https://en.wikipedia.org/wiki/Prototype) of the cluster.

The problem is computationally difficult  [(NP-hard);](https://en.wikipedia.org/wiki/NP-hard) however, there are efficient  [heuristi](https://en.wikipedia.org/wiki/Heuristic_algorithm)c  [algorithms](https://en.wikipedia.org/wiki/Heuristic_algorithm) that are commonly employed and converge quickly to a  [local optimum.](https://en.wikipedia.org/wiki/Local_optimum) These are usually similar to the  [expectation-maximization algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm) for  [mixtures](https://en.wikipedia.org/wiki/Mixture_model) of  [Gaussia](https://en.wikipedia.org/wiki/Gaussian_distribution)n  [distributions](https://en.wikipedia.org/wiki/Gaussian_distribution) via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the  [k-nearest neighbor classifier,](https://en.wikipedia.org/wiki/K-nearest_neighbor) a popular  [machin](https://en.wikipedia.org/wiki/Machine_learning)e  [learning](https://en.wikipedia.org/wiki/Machine_learning) technique for classification that is often confused with k-means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k-means to classify new data into the existing clusters. This is known as  [nearest centroid classifier](https://en.wikipedia.org/wiki/Nearest_centroid_classifier) or Rocchio algorithm.

Given a set of observations (x1, x2, …, xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {S1, S2, …, Sk} so as to minimize the within-cluster sum of squares (WCSS) (sum of distance functions of each point in the cluster to the K center). In other words, its objective is to find:



where ***μ****i* is the mean of points in *Si*.

**4.2.2.1 Algorithm and pseudo code**

The most common algorithm uses an iterative refinement technique. Given an initial set of k means m1(1),…,mk(1) , the algorithm proceeds by alternating between two steps:

**Assignment step**: Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared  [Euclidean distance,](https://en.wikipedia.org/wiki/Euclidean_distance) this is intuitively the "nearest" mean.



where each  is assigned to exactly one , even if it could be assigned to two or more of them.

**Update step**: Calculate the new means to be the  [centroids](https://en.wikipedia.org/wiki/Centroids) ofthe observations in the newclusters.



Pseudo code:

1. Select K points as the initial centroids.
2. Repeat
3. From K clusters by assigning all points to the closest centroid.
4. Recompute the centroid of each cluster.
5. until The centroids don‘t change

**4.2.3 Naive bayes classifier**

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of  [feature](https://en.wikipedia.org/wiki/Feature_vector) values, where the class labels are drawn from some finite set. It is not a single  [algorithm](https://en.wikipedia.org/wiki/Algorithm) for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is  [independent](https://en.wikipedia.org/wiki/Independence_(probability_theory)) of the value of any other feature, given the class variable.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a  [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) setting. In many practical applications, parameter estimation for naive Bayes models uses the method of  [maximum likelihood;](https://en.wikipedia.org/wiki/Maximum_likelihood) in other words, one can work with the naive Bayes model without accepting  [Bayesian probability](https://en.wikipedia.org/wiki/Bayesian_probability) or using any Bayesian methods.

**4.2.3.1 Probabilistic model**

Naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector  representing some n features (independent variables), it assigns to this instance probabilities



for each of K possible outcomes or classes

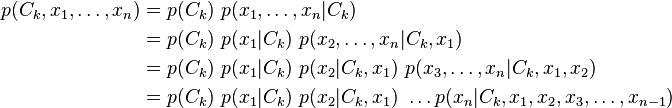
Using  [Bayes theorem,](https://en.wikipedia.org/wiki/Bayes%27_theorem) the conditional probability can be decomposed as



In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on  and the values of the features  are given, so that the denominator is effectively constant. The numerator is equivalent to the  [joint probability](https://en.wikipedia.org/wiki/Joint_probability) model



which can be rewritten as follows, using the  [chain rule](https://en.wikipedia.org/wiki/Chain_rule_(probability)) for repeated applications of the definition of  [conditional probability](https://en.wikipedia.org/wiki/Conditional_probability):



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Now the | "naive"  [conditional independence](https://en.wikipedia.org/wiki/Conditional_independence) assumptions | come into | play: | assume | that | each |
| feature | is conditionally  [independent](https://en.wikipedia.org/wiki/Statistical_independence) of every other | feature | for | , | given | the |



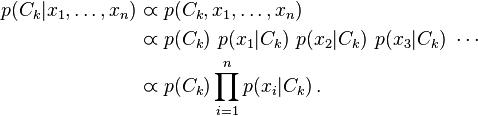
category . This means that

,

,

,

and so on, for . Thus, the joint model can be expressed as



This means that under the above independence assumptions, the conditional distribution over the class variable  is:



where the evidence  is a scaling factor dependent only on , that is, a constant if the values of the feature variables are known.

Constructing a classifier from the probability model**[** [edi](https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&action=edit&section=3)t**]**

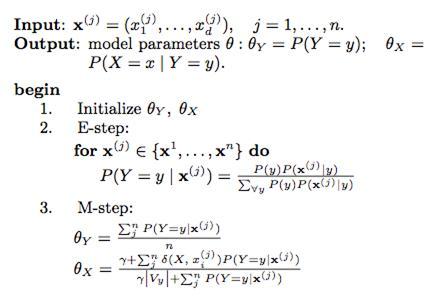
The discussion so far has derived the independent feature model, that is, the naive Bayes  [probability model.](https://en.wikipedia.org/wiki/Probability_model) The naive Bayes  [classifier](https://en.wikipedia.org/wiki/Statistical_classification) combines this model with a  [decision rule](https://en.wikipedia.org/wiki/Decision_rule). One common rule is to pick the hypothesis that is most probable; this is known as the  [maximu](https://en.wikipedia.org/wiki/Maximum_a_posteriori)m

[a posteriori](https://en.wikipedia.org/wiki/Maximum_a_posteriori) or MAP decision rule. The corresponding classifier, a  [Bayes classifier,](https://en.wikipedia.org/wiki/Bayes_classifier)  is the

function that assigns a class label  for some k as follows:



**4.2.3.2 Pseudo code**



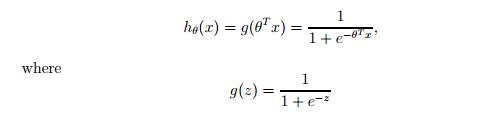
**4.2.4 Logistic regression classifier**

Logistic regression is a  [regression](https://en.wikipedia.org/wiki/Regression_analysis) model where the  [dependent variable (DV)](https://en.wikipedia.org/wiki/Dependent_and_independent_variables) is  [categorical](https://en.wikipedia.org/wiki/Categorical_variable). Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a  [logistic function,](https://en.wikipedia.org/wiki/Logistic_function) which is the cumulative logistic distribution.

Logistic regression can be seen as a special case of  [generalized linear model](https://en.wikipedia.org/wiki/Generalized_linear_model) and thus analogous to  [linear regression.](https://en.wikipedia.org/wiki/Linear_regression) The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression. In particular the key differences of these two models can be seen in the following two features of logistic regression. First, the conditional distribution  is a  [Bernoulli distribution](https://en.wikipedia.org/wiki/Bernoulli_distribution) rather than a  [Gaussian distribution,](https://en.wikipedia.org/wiki/Gaussian_distribution) because the dependent variable is binary. Second, the predicted values are probabilities and are therefore restricted to (0,1) through the  [logistic distribution function](https://en.wikipedia.org/wiki/Logistic_function) because logistic regression predicts the probability of particular outcomes.

**4.2.4.1 Hypothesis function**

In logistic regression hypothesis function is given by



is called logistic or sigmoid function.

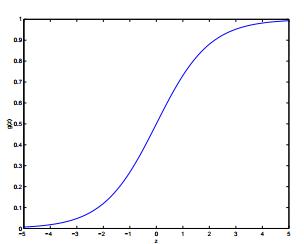
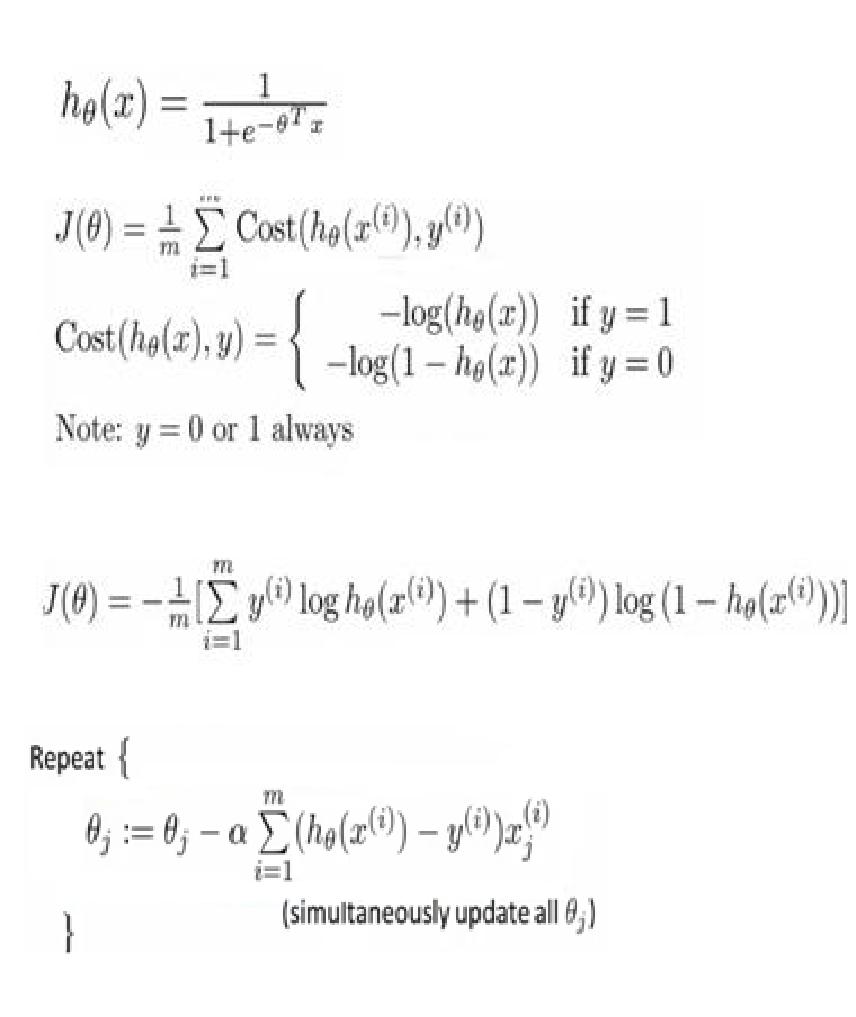


Figure 4.1 Plot of sigmoid function

The stochastic gradient ascent rule for logistic regression is:



**4.2.4.2 Pseudocode**



**4.2.5 Decision tree classifier**

Decision tree learning uses a  [decision tree](https://en.wikipedia.org/wiki/Decision_tree) as a  [predictive model](https://en.wikipedia.org/wiki/Predictive_modelling) which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in  [statistics,](https://en.wikipedia.org/wiki/Statistics)  [data mining](https://en.wikipedia.org/wiki/Data_mining) and  [machine learning.](https://en.wikipedia.org/wiki/Machine_learning) Tree models where the target variable can take a finite set of values are called classification trees. In these tree structures,  [leaves](https://en.wikipedia.org/wiki/Leaf_node) represent class labels and branches represent  [conjunctions](https://en.wikipedia.org/wiki/Logical_conjunction) of features that lead to those class labels.

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. Each  [interio](https://en.wikipedia.org/wiki/Interior_node)r  [node](https://en.wikipedia.org/wiki/Interior_node) corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A tree can be "learned" by splitting the source  [set](https://en.wikipedia.org/wiki/Set_(mathematics)) into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called  [recursiv](https://en.wikipedia.org/wiki/Recursive_partitioning)e  [partitioning.](https://en.wikipedia.org/wiki/Recursive_partitioning) The  [recursion](https://en.wikipedia.org/wiki/Recursion) is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (TDIDT) is an example of a  [greedy algorithm,](https://en.wikipedia.org/wiki/Greedy_algorithm) and it is by far the most common strategy for learning decision trees from data. Entropy as well as gini impurity can be used as learning rules in decision trees.

Some advantages of decision trees are:

* Simple to understand and to interpret. Trees can be visualized.
* Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
* The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
* Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable. See  [algorithms](http://scikit-learn.org/stable/modules/tree.html#tree-algorithms) for more information.
* Able to handle multi-output problems.
* Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
* Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
* Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

* Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.

* + The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.

1. **Complexity analysis**

The run time cost to construct a balanced binary tree is 

and query time . Although the tree construction algorithm attempts to generate balanced trees, they will not always be balanced. Assuming that the subtrees remain



|  |  |
| --- | --- |
| approximately balanced, the cost at each node consists of searching through | to |
| find the feature that offers the largest reduction in entropy. This has a | cost |

of  at each node, leading to a total cost over the entire trees (by summing the cost at each node) of .

**4.2.5.2 C4.5 Decision tree algorithm**

C4.5 is an extension of Quinlan's earlier  [ID3 algorithm.](https://en.wikipedia.org/wiki/ID3_algorithm) The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a  [statistical classifier](https://en.wikipedia.org/wiki/Statistical_classification). C4.5 builds decision trees from a set of training data in the same way as  [ID3,](https://en.wikipedia.org/wiki/ID3_algorithm) using the concept of  [information entropy.](https://en.wikipedia.org/wiki/Entropy_(information_theory)) The training data is a set  of already classified samples. Each sample  consists of a p-dimensional vector , where the  represent attribute values or features of the sample, as well as the class in which  falls.

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized  [information gain](https://en.wikipedia.org/wiki/Information_gain) (difference in  [entropy](https://en.wikipedia.org/wiki/Entropy_(information_theory))). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

This algorithm has a few base cases.

* All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
* None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.

* Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

**4.2.5.3 C4.5 Decision tree pseudo code**

The general algorithm for building decision trees is:

1. Check for base cases
2. For each attribute *a*
   1. Find the normalized information gain ratio from splitting on *a*
3. Let *a\_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a\_best*

5. Recur on the sublists obtained by splitting on *a\_best*, and add those nodes as children of *node*

**4.2.6 Support Vector Machine (SVM)**

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the illustration below. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).



The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below. Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

**4.2.6.1 Linear SVM**

We are given a training dataset of n points of the form

 (\vec{x}_1, y_1),\, \ldots ,\, (\vec{x}_n, y_n)

y_i=-1y_i=1p \vec{x}_i y_iwhere are either 1 or −1,each indicating the class to which the point \vec{x}_i  belongs. Each is dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points \vec{x}_i for which group of points for which , which is defined so that the distance between the hyperplane and the nearest point \vec{x}_i from either group is maximized.

Any hyperplane can be written as the set of points \vec{x} satisfying

\vec{w}\cdot\vec{x} - b=0,\,

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

where {\vec{w}} is the (not necessarily normalized) normal vector to the hyperplane. The parameter \tfrac{b}{\|\vec{w}\|} determines the offset of the hyperplane from the origin along the normal vector {\vec{w}}

**4.2.6.2 Non Linear Classification**

The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier. However, in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space.

It is noteworthy that working in a higher-dimensional feature space increases the generalization error of support vector machines, although given enough samples the algorithm still performs well.

k(\vec{x_i},\vec{x_j})=(\vec{x_i} \cdot \vec{x_j})^dSome common kernels include:

* Polynomial (homogeneous):
* k(\vec{x_i},\vec{x_j})=(\vec{x_i} \cdot \vec{x_j} + 1)^dPolynomial (inhomogeneous):
* \gamma > 0k(\vec{x_i},\vec{x_j})=\exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)Gaussian radial basis function: , for . Sometimes parametrized using \gamma=1/{2 \sigma^2}
* Hyperbolic tangent: k(\vec{x_i},\vec{x_j})=\tanh(\kappa \vec{x_i} \cdot \vec{x_j}+c), for some (not every) \kappa > 0  and  c < 0 

**4.2.6.3 Complexity analysis**

Support Vector Machines are powerful tools, but their compute and storage requirements increase rapidly with the number of training vectors. The core of an SVM is a quadratic programming problem (QP), separating support vectors from the rest of the training data. The QP solver used by this libsvm-based implementation scales between O(n_{features} \times n_{samples}^2)and O(n_{features} \times n_{samples}^3) depending on how efficiently the libsvm cache is used in practice (dataset dependent). If the data is very sparse n_{features} should be replaced by the average number of non-zero features in a sample vector.

**4.2.7 Evaluation of binary classifiers**

**4.2.7.1 Sensitivity and Specificity:**

1. [Sensitivity](https://en.wikipedia.org/wiki/Sensitivity_(tests)) or  [True Positive Rate](https://en.wikipedia.org/wiki/True_Positive_Rate) (TPR), also known as  [recall,](https://en.wikipedia.org/wiki/Recall_(information_retrieval)) is the proportion of people that tested positive and are positive (True Positive, TP) of all the people that actually are positive (Condition Positive, CP = TP + FN).
2. [Specificity](https://en.wikipedia.org/wiki/Specificity_(tests)) (SPC) or  [True Negative Rate](https://en.wikipedia.org/wiki/True_Negative_Rate) (TNR) is the proportion of people that tested negative and are negative (True Negative, TN) of all the people that actually are negative (Condition Negative, CN = TN + FP).

**4.2.7.2 Positive and negative predictive values:**

1) Positive prediction value answers the question "If the test result is *positive*, how well does that *predict* an actual presence of disease?". It is calculated as TP/(TP + FP); that is, it is the proportion of true positives out of all positive results.

2) Negative prediction value is the same, but for negatives.

**4.2.7.3 Single metrics:**

1) Fraction Correct (FC) measures the fraction of all instances that are correctly categorized; it is the ratio of the number of correct classifications to the total number of correct or incorrect classifications.

2)  [F-score](https://en.wikipedia.org/wiki/F-score) is a combination of the  [precision](https://en.wikipedia.org/wiki/Precision_(information_retrieval)) and the  [recall,](https://en.wikipedia.org/wiki/Recall_(information_retrieval)) providing a single score.



**XXX(Leave)Chapter-5 Multi Instance Learning**

**5.1 Introduction**

In the standard supervised learning task, we learn a classifier based on a training set of feature vectors, where each feature vector has an associated class label. In the Multiple Instance Learning (MIL) task we learn a classifier based on a training set of bags, where each bag contains multiple feature vectors (called instances in the MIL terminology). In this setting, each bag has an associated label, but we do not know the labels of the individual instances that conform the bag. Furthermore, not all the instances are necessarily relevant, i.e., there might be instances inside one bag that do not convey any information about its class, or that are more related to other classes of bags, providing confusing information.

In many fields, we find problems that are most naturally formulated using the multiple instance learning setting. This is the case of drug discovery (pharmacy), classification of text documents (information retrieval), classification of images (computer vision), speaker identification (signal processing) and bankruptcy prediction (economy), to mention a few fields that make use of this framework (see section 2 for a more detailed discussion about real examples). This makes the MIL problem an important topic in the machine learning community, where many methods have been published in the last years. Despite this fact, there is a lack of surveys or analytical studies that compare the performance of the different families of MIL algorithms

The overview of some of the approaches to deal with the advent growth of multi-image multi-label methods are given in this section. Dietterich et al [4] proposed multi instance learning (MIL) for drug prediction problem in which the objects to be classified were chemical molecules. The system had to classify that whether given chemical molecule is a good drug or not. Andrews et al [5] demonstrated two approaches to modify Support Vector Machines (SVM), mi-SVM and MI-SVM for instance-level classification and bag-level classification respectively. In order to solve multi instance learning, traditional neural networks were used in [3,6]. However they did not apply deep learning methodology nor did they use it for computer vision applications.

Maron and Lozano-Pérez [7] presented the Diverse density algorithm for solving MIL problem. In this approach a point is found in feature space such that it is near to instances of positive bags and at the same time is at a greater distance from instances of negative bags. Hence, the learning phase involves searching for an optimal point which is the one that has the maximum diverse density which is measured in terms of how many different instances of positive bags are located near that point, and how many negative instances are far away from that point. The algorithm has been applied to stock selection and content based image retrieval [8].

Deep learning architectures comprises various layers consisting of feature detectors. Simple and local features are detected by lower layers which are then fed to higher layers that calculate more complex features [9, 10, 11, 12]. Convolution neural networks (CNN) have been applied for a variety of computer vision applications including object recognition [10] and video classification [12]. Xu et al. [13] presented deep learning method to compute features for multi-instance learning in medical imaging. Song et al. [14] applied CNN features for weakly supervised object localization.

Wang et al [15] employed k-nearest neighbour (kNN) algorithm for MIL framework by utilizing a bag-level distance metric, the minimum Hausdorff distance that is defined as the shortest distance between any two bag instances. Two multi instance learning algorithms were presented: citation-kNN which predicts the label of a bag based not only on the nearest neighbours (references) but also on the bags that consider the concerned bas as a neighbour (citers), which proved to be more efficient than the kNN based only on references and the Bayesian method, which estimates the labels of a bag based only on the abels of its neighbours. Chevaleyre et al [16] applied decision trees and decision rules in the MIL framework. A multi instance version of decision tree algorithm ID3 called ID3-MI and that of rule learning algorithm RIPPER called RIPPER-MI was introduced. The growth of the decision tree was based on the information gain of a feature with respect to set of instances that was given by a multiple instance coverage function, which was related to the entropy of the instances, where the entropy was given by multiple instance entropy function.

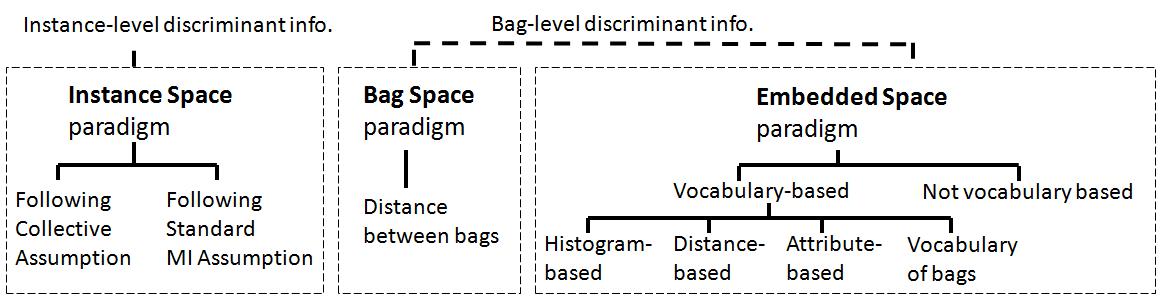
Amores [1] presented an analysis of different multiple instance learning methods that comprised of both review and comparison of different paradigms which included instance space paradigm, bag space paradigm and embedded space paradigm. In the embedded space paradigm we utilize a vocabulary-based mapping, in bag space paradigm we consider the similarity between all instances and in instance space paradigm we infer an instance based classifier from the training data. Under particular constraints, the bag space paradigm becomes computationally more expensive and therefore vocabulary based ES methods are better suited for certain type of data.

**5.2 Overview of Paradigms**

A bag is a set X = {~x1, . . . , ~xN}, where the elements ~xi are feature vectors called instances in the MIC terminology, and the cardinality N can vary across the bags. All the instances ~xi live in 3 a d-dimensional feature space, ~xi ∈ R d , called instance space.

The objective of the MIC problem is to learn a model, at training time, that can be used to predict the class labels of unseen bags. In this work, we only consider the binary classification problem, where a bag X can be either positive or negative. Our objective is to estimate a classification function F(X) ∈ [0, 1] that provides the likelihood that X is positive. In order to learn such a function, we are given a training set with M bags and their corresponding labels, T = {(X1, y1), . . . , (XM, yM)}, where yi ∈ {0, 1} is the label of Xi (yi = 0 if Xi is negative, and yi = 1 if it is positive).

In addition to the bag-level classification function F(X), many methods try to learn an instance level classification function f(~xi) that operates directly on the instances ~xi . Throughout this work we will use uppercase to refer to bags X and to the bag-level classifier F, and we will use lowercase to refer to instances ~x and to the instance-level classifier f.

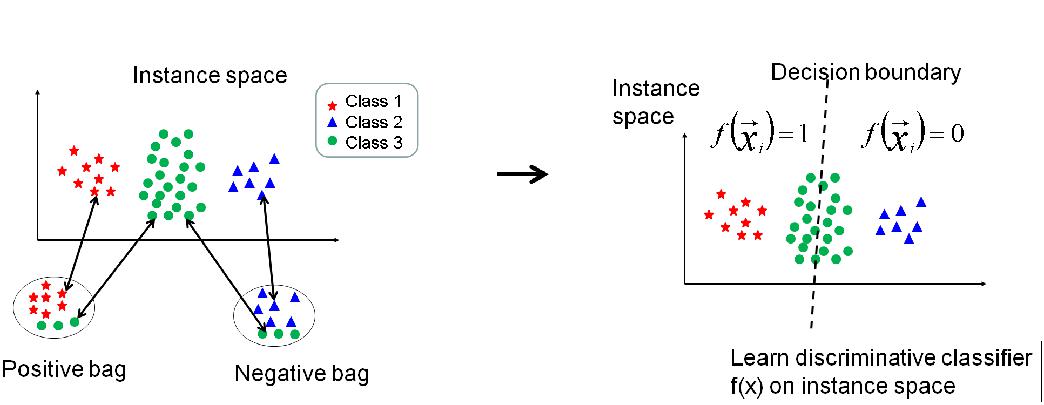


**5.2.1 Instance Space Paradigm**

In the Instance Space (IS) paradigm, the discriminative information is considered to lie at the instance-level. Therefore, the discriminative learning process occurs at this level: a discriminative instance-level classifier f(~x) is trained to separate the instances in positive bags from those in negative ones (see Fig. 1). Based on it, given a new bag X the bag-level classifier F(X) is obtained by simply aggregating instance-level scores f(~x), ∀~x ∈ X. We say that this type of paradigm is based on local, instance-level information, in the sense that the learning process considers the characteristics of individual instances, without looking at more global characteristics of the whole bag.

The methods falling in this category must address the question of how to infer an instancelevel classifier f(~x) without having access to a training set of labelled instances. In order to solve this issue, some assumption must be made about the relationship between the labels of the bags in the training set and the labels of the instances contained in these bags. In this sense, two sub-categories of IS methods emerge clearly in the literature: the ones following the Standard MI (SMI) assumption and the ones following the Collective assumption.

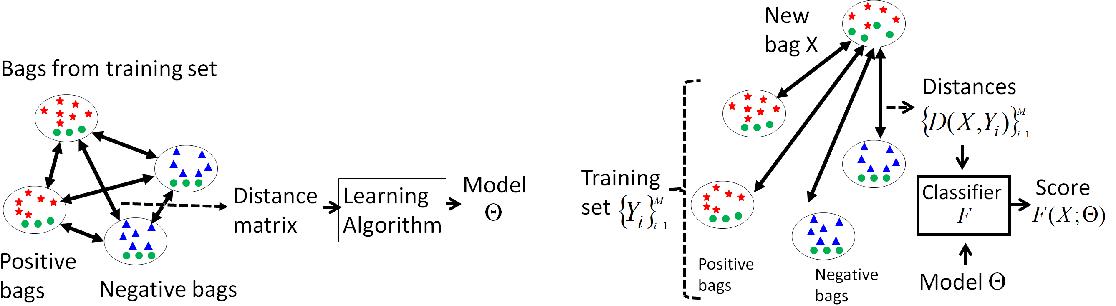
The SMI assumption states that every positive bag contains at least one positive instance (i.e. an instance belonging to some target positive class), while in every negative bag all of the instances are negative. This is an asymmetrical assumption which is used in many MIC problems such as the traditional one of drug discovery. Note that this assumption is that one of the instances has some desirable properties that make the bag positive. Therefore, the methods following this assumption try to identify the type of instance that makes the bag positive



**5.2.2 Bag Space Paradigm**

In the Bag Space (BS) paradigm, the discriminative information is considered to lie at the bag-level. In this paradigm each bag X is treated as a whole entity, and the learning process discriminates between entire bags. As a result, it obtains a discriminative bag-level classifier F(X) which makes use of the information from the whole bag X in order to take a discriminative decision about the class of X. We say that this type of paradigm is based on global, bag-level information, because the discriminative decision is taken by looking at the whole bag, instead of aggregating local instance-level decisions.

Given the fact that the bag space is a non-vector space, the BS methods make use of nonvectorial learning techniques. As far as we know, all the existent non-vectorial techniques work through the definition of a distance function D(X, Y) that provides a way of comparing any two non-vectorial entities X and Y (where these entities are bags in our problem). Once this distance function has been defined, it can be used into any standard distance-based classifier such as KNearest Neighbor (K-NN), or similarly into any kernel-based classifier such as SVM. Fig. 2 illustrates the idea under this paradigm. Although we use the term “distance” in Fig. 2, the BS paradigm also includes methods that use other types of pairwise comparisons between bags, such as kernel-based comparisons K(X, Y) in SVM-based methods.



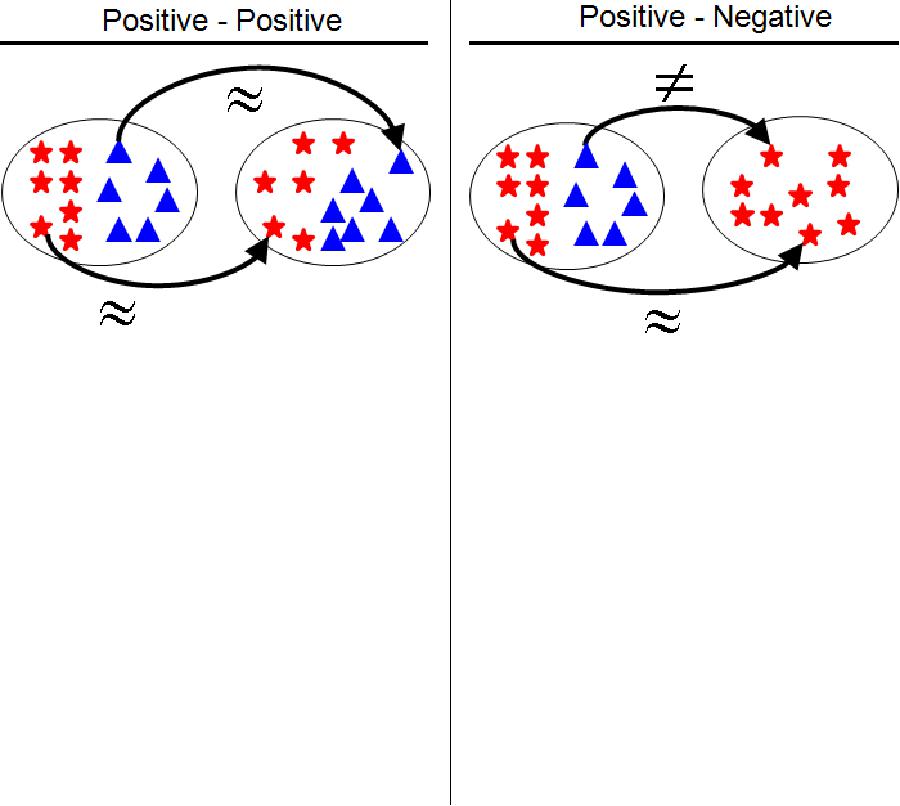
The idea of the IS paradigm just reviewed is to estimate a model that summarizes the properties of the single instances, by discriminating those typically found in positive bags versus those found in negative ones. This makes this type of methods consider local information, in the sense that the obtained model is about instances and not about bags as a whole. At classification time, the classifier F(X) is obtained as an aggregation of local responses f(~x), where each of them consider only one instance ~x at a time.

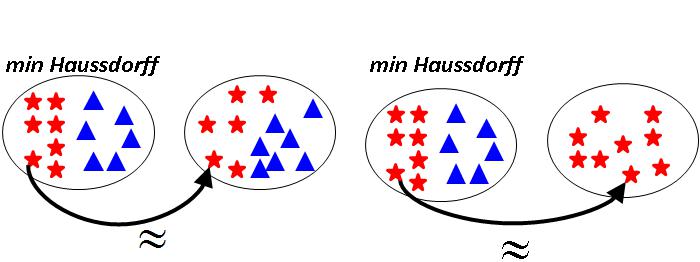
In contrast, the methods of the BS paradigm treat the bags X as a whole, and the discriminant learning process is performed in the space of bags. This allows the algorithm to take into account more information while performing the inference of F(X). In order to learn a non-vectorial entity such as a bag, we can define a distance function D(X, Y) that compares any two bags X and Y, and plug this distance function into a standard distance-based classifier such as K-NN or SVM

Note that a bag X is nothing else than a set of points in a d-dimensional space. Therefore, any distance function D(X, Y) that compares two sets of points X and Y can be used in this context. In this work we study the minimal Hausdorff distance used in [24], the Earth Movers Distance (EMD) [25], the Chamfer distance [26], and the kernel by Gartner et al. [14]. Let us first see the definition of these functions and in section 5.2 we discuss the intuition behind them. The minimal Hausdorff distance is defined as:

This is the distance between the closest points of X and Y. The EMD distance, on the other hand, is the result of an optimization process. Let X = {~x1, . . . , ~xN}, and Y = {~y1, . . . ,~yM}. The EMD distance is defined as:

where the weights wi j are obtained through an optimization process that globally minimizes D(X, Y) subject to some constraints



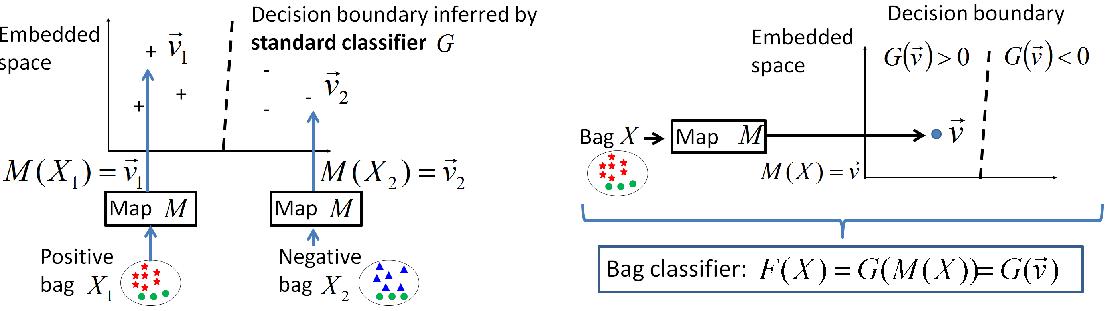


**5.2.3 Embedded Space Paradigm**

Both the last paradigm and the one presented in this section are based on extracting global information about the bag. In the BS paradigm this is done in an implicit way through the definition of the distance function D(X, Y) or kernel function K(X, Y). This function defines how bags are compared, and therefore, how the information about them is considered in the matching.

In the ES paradigm, this is done in an explicit way, by defining a mapping M : X 7→ ~v from the bag X to a feature vector ~v which summarizes the characteristics of the whole bag. Different definitions of this mapping function put emphasis on different types of information, and have a high impact on the performance of the method.

In this sense, we can split the existing ES methods in roughly two sub-categories. In the first one the methods simply aggregate the statistics of all the instances inside the bag, without making any type of differentiation among instances. In contrast, in the vocabulary-based paradigm the mapping is constructed by analyzing how the instances of the bag match certain prototypes that have been previously discovered in the data.



**XXX(Leave)Chapter 6-HYBRID ALGORITHM**

In this section, the description of approaches to machine learning that combine and thus club the advantageous features of several previously used algorithmic ideas along with some smart human decisions pertaining to the specificity of the problem such as the handling of two highly correlated features etc. These approaches resulted in accuracy gain at the cost of a negligible increase in both the learning and testing time. It is important to note that the gain from the hybrid approach is bound to increase while no expected noticeable change to the processing time, when it comes to application on the real world data, which might vary so much that using a single fixed approach might not result into good enough results.

Before proceeding to our Hybrid algorithms, the general as well as significant drawbacks from the previous approaches which motivated our approach are given below.

**6.1 K-Means Algorithm**:

The assumptions on which this algorithm works are as follows

* k-means assumes the variance of the distribution of each attribute (variable) is spherical;
* all variables have the same variance;
* the prior probability for all k clusters is the same, i.e., each cluster has roughly equal number of observations;

If any one of these 3 assumptions are violated, then k-means will fail to give the expected result accuracy.

This has been illustrated on the next page.

Broken Assumption: Non - Spherical Data

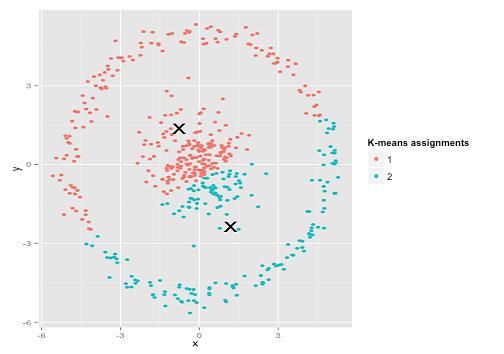


Figure 5.1 Calculated Result

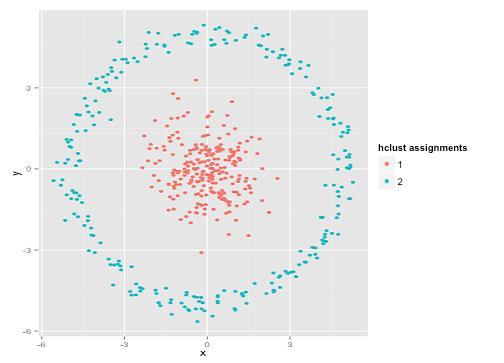


Figure 5.2 Expected Result

There can be several such examples where the K-Means clustering alone may result out to be an inappropriate approach. It is clear that the failure is a possibility due to the assumptions turning out to be wrong and not the improper working of the algorithm.

**Complexity Analysis of k-means Clustering**

The k-means problem is solved using Lloyd‘s algorithm. The **average complexity** is given by

**O(k n T)**, were n is the number of samples and T is the number of iteration. The **worst case complexity** is given by **O(n^(k+2/p))** with n = n\_samples, p = n\_features. In practice, the k-means algorithm is very fast (one of the fastest clustering algorithms available), but it falls in local minima. That‘s why it can be useful to restart it several times.

**6.2 Naïve Bayesian Classifier**

This algorithm is based on the following equation.



1. It assumes the probability distribution of the attributes is independent of each other.
2. Zero Frequency Problem: To make sure that zero frequency problem does not cause issues, the implementation used Laplacian Smoothing, thus making the result an approximation, especially for the classes with smaller total frequencies.

So if you have no occurrences of a class label and a certain attribute value together then the frequency-based probability estimate will be zero.

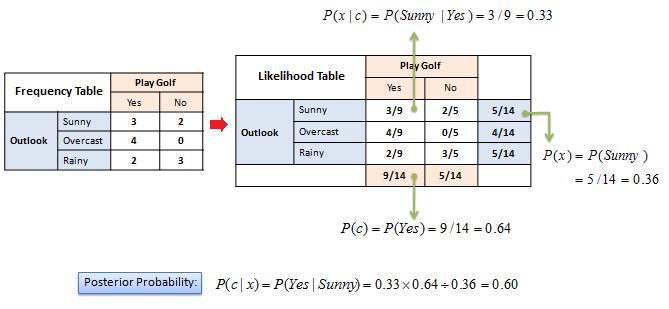


Figure 5.3 Zero Probability Problem

So to assure the independence between the various features, we decided to apply the clustering algorithm first, then followed by the Bayesian Classification, as this helps us eliminate the Bayesian drawbacks, while minimising the K-means error probability.

**6.3 K-means with naïve bayes**

We initially implemented K-means individually. Then we worked on hybrid by combining K-means with naïve bayes. K-means is used to group together similar data. Then naïve bayes was implemented on each cluster and model was made. For each new testcase, it was first determined to which cluster it belongs. Then the naïve bayes model for that particular cluster was used to make prediction for the given testcase. K-means was used with the hope that grouping together similar data will help in increasing accuracy of naïve bayes algorithm. Here we had a tradeoff between time of computation and accuracy but additional gained accuracy was preferred.

For this algorithm, we first discretized the data as naïve bayes requires data which is in discrete form. We couldn‘t implement Gaussian naïve bayes as data distribution was not Gaussian and still using the algo would have resulted in poor accuracy. We had two choices for data discretization, equal width discretization and equal frequency discretization. Equal width discretization resulted in better performance.

We ran our algorithm on different number of clusters but found that 5 number of clusters resulted in highest accuracy. Hence 5 clusters have been used for this algorithm.

On running this hybrid algorithm on Cleveland heart disease dataset we got an accuracy of 78.4% which was an improvement over naïve bayes implementation without k-means.

**6.4 Decision Tree**

Decision trees have been used to obtain some of the most accurate solutions by clubbing them with other statistical learning techniques. Still we highlight some of the drawbacks of Decision tree Algorithms

* Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high

dimensional space is very likely to overfit.

 In general, the run time cost to construct a balanced binary tree is  and query time . Although the tree construction algorithm attempts to generate balanced trees, they will not always be balanced.

Assuming that the subtrees remain approximately balanced, the cost at each node consists of searching through  to find the feature that offers the largest reduction in entropy. This has a cost of  at each node, leading to a total cost over the entire trees (by summing the cost at each node) of .

So we first use K-Means to form the clusters, and then apply the Decision tree algorithms, so that the overfitting can be minimized as well as the smaller and more closely related cluster sizes enable us to improvise both pre-pruning and post-pruning of the tree for the same given size.

**6.5 Decision tree with Kmeans**

We initially implemented decision tree individually. Then we worked on hybrid by combining Kmeans with decision tree. In random forest, data is randomly divided in different groups and decision tree is made on each group and result is determined. However in this algorithm, Kmeans is used to group together similar data. Then decision tree was implemented on each cluster and model was made. For each new testcase, it was first determined to which cluster it belongs. Then the decision tree model for that particular cluster was used to make prediction for the given testcase. Kmeans was used with the hope that grouping together similar data will help in increasing accuracy of decision tree algorithm. Decision tree will have fewer branches due to grouping of similar data and will not overfit. The hybrid algorithm was run on both continuous dataset and dataset after discretization. Continuous dataset resulted in higher accuracy. We ran our algorithm on different number of clusters but found that 5 number of clusters resulted in highest accuracy. Hence 5 clusters have been used for this algorithm.

On running this hybrid algorithm on Cleveland heart disease dataset we got an accuracy of 79.1% which was an improvement decision tree implementation without kmeans and random forest implementation.

**6.6 Photo Classifier**

Yelp hosts tens of millions of photos uploaded by Yelpers from all around the world. The wide variety of these photos provides a rich window into local businesses, a window we’re only just peeking through today.

One way we’re trying to open that window is by developing a photo understanding system which allows us to create semantic data about individual photographs. The data generated by the system has been powering our recent launch of tabbed photo browsing as well as our first attempts at content-based photo diversification.

One can imagine a variety of ways to tackle the ambitious goal of holistically understanding pictures. To help simplify our problem, we focused initially on only sorting photos into a handful of predefined classes. Further, we focused only on categories of photos directly relevant to restaurants as shown below

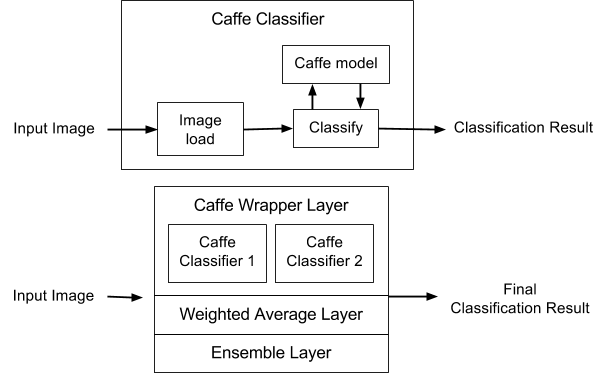
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| http://engineeringblog.yelp.com/images/posts/2015-10-14-photo-classification/image03.png | http://engineeringblog.yelp.com/images/posts/2015-10-14-photo-classification/image01.png | http://engineeringblog.yelp.com/images/posts/2015-10-14-photo-classification/image09.png | http://engineeringblog.yelp.com/images/posts/2015-10-14-photo-classification/image05.png | http://engineeringblog.yelp.com/images/posts/2015-10-14-photo-classification/image08.png |
| **food** | **drink** | **inside** | **outside** | **menu** |

To develop a classifier that can put a photo into one of these groups, we need to first collect many photos with known labels. We collected this information through a few different ways:

* **Photo captions**: A good number of “menu” photos have the word “menu” in their captions. Similarly, we can find photos titled “sushi” or “burger” that are likely to be food. For the former, we had to worry about false positives because it’s not uncommon to see “food” or “drink” photos whose captions are of the pattern “Best on Their Menu!”, and as a result some cleanup was necessary. To aid us in identifying food items, we relied on Yelp’s menu structures which maintain each business’ list of food items. We found that matching food items from the list to the captions of photos yielded a dataset of high precision.
* **Photo attributes**: When uploading photos to Yelp, users are allowed to mark a few attributes about the photo, such as *“Is it the storefront?”*They are not always accurate, but still serve as a good source of candidate photos.
* **Crowdsourcing**: We ran additional tasks through a crowdsourcing partner to correct our guesses for what label should be applied to each photo and to collect more “inside” and “outside” photos. We have found that this led to generally good quality labels at a reasonable cost (both in time and money).

Once we had our labeled data, we employed deep convolutional neural networks (CNNs) in the form of “AlexNet” to recognize those classes. CNNs usually consist of a deep stack of multiple**convolutional layers** (for extracting spatially local and translation-invariant features), **ReLU (Rectified Linear Units) layers** (for non-saturating activations), **pooling layers** (for down-sampling and translation-invariance), **local response normalization layers** (for better generalization) and **fully-connected layers** as in conventional feedforward neural networks. Softmax outputs and regularization methods such as dropout are also commonly used. Our CNN was built on AWS EC2 GPU instances based on the Caffe framework. We like Caffe because it’s easy to use, performant, open source (BSD 2-clause), and under active development. To address Caffe’s software dependencies, we wrapped our CNN using Docker so that it could be more easily deployed.

We also created abstractions to ensure that our CNN could be easily integrated with other possible forms of classifiers, including different instances of CNNs. As illustrated below, our baseline is a “Caffe Classifier” that runs the CNN by means of Caffe; it’s a special form of an abstract classifier that can take different signals and perform different classification algorithms. Our current “facade” classifier is an ensemble that takes the weight average of classification results from two independently trained Caffe Classifiers. It would be quite straightforward if we decide to further incorporate new classifiers relying on other signals, such as photo captions.



On an evenly split gold test set of 2,500 photos, our current classifier shown above has an overall precision of 94% of precision and recall of 70%. While these numbers can definitely be improved, we found them reasonably good for the applications described below.

**Chapter 7-MACHINE LEARNING TOOLS and IDE**

**7.1 DEVELOPMENT SOFTWARE**

 Why **Python**?

**Python** is far more powerful and superior to many other languages(scientific): it's ageneral-purpose language that include a well suited GUI along with powerful libraries. It is free licensed and open-source, fundamentally object-oriented, highly portable, extensible and embeddable and easily maintainable.

**7.2 IDE**

* **PyCharm (IntelliJ)**
* It is primarily used for scientific programming in python language.
* It incorporates NumPy, SciPy and matplotlib
* Includes support for interactive features along with well supported plugins.
* It is available through cross platform on Windows with Python(x,y) and using PyQt on Linux Distributions.
* Syntax highlighting along with high introspection for code completion.
* Supports multiple Python consoles including IPython.

**7.3 SOFTWARE LIBRARY**

 **Scikit-learn**

* It is an open source and free licensed machine learning library for the Python programming language.
* It includes various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and is designed to interoperate with the Python numerical and scientific libraries  [NumPy](https://en.wikipedia.org/wiki/NumPy) and  [SciPy](https://en.wikipedia.org/wiki/SciPy) along with matplotlib to incorporate tools for creating interactive results that include graphs, pie charts and histograms.
* It is largely written in Python.
* It was created as an open source project in GSOC (Google Summer of Code).
* Simple and efficient tools for data mining and data analysis
* Accessible to everybody, and reusable in various contexts.
* **Numpy**

It is the one of the most fundamental package for scientific computing with Python and is incorporated in scikit-learn. It consists of some other things including:

* A powerful N-dimensional array object.
* Sophisticated functions.
* Useful for linear algebra and random number capabilities.
* Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined.

 **Matplotlib**

Matplotlib is a python 2D plotting library that contains a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in python scripts with ease to provide interactive features.

**Chapter 7-Proposed Methodology**

The proposed algorithm consists of four main steps. The features are extracted from images using a seven layer CNN architecture which are then represented using the BoW model with earth mover’s distance as a metric each bag. A custom kernel binary SVM classifier along with association rules are then used to predict labels for each restaurant.

**7.1 Feature Extraction**

For feature extraction we use the pre-trained model of Caffe [17] open source deep learning framework to extract features from images. A 4096-dimensional feature vector, fc7 feature is extracted from images through five convolution layers and two fully connected layers using the CNN architecture given by Krizhevsky et al. [18]. In order to compute fc7 features for each image, the images are converted into a form that is compatible with the CNN as its architecture requires a fixed input size of 227x227 pixels.

**7.1 Bag-of-Words Representation**

In order to classify images, the Bag-of-Words model is employed which treats image features as words. It represents the data items (images) as a histogram over features (words). The BoW algorithm was initially presented for text extraction domain in word document analysis, and was later modified further for computer vision applications [19]. The BoW classification model analogous to word dictionary is based on the process of vector quantization image features extracted from local regions or points, like texture, color etc. The BoW model can be described as follows. Given a dataset containing m images characterized by where is the features extracted using the above method,a unsupervised learning algorithm k-means, is used to cluster based on the number of clusters which depicts the categories for classification. Thus, each image in is represented as a vector v of counts over each feature.

) ) … (1)

The standard BoW pipeline is implemented. Firstly, the features are extracted using the method defined above. The feature vectors are quantized into words by mapping them to the nearest codevector in the codebook by using Earth Mover’s Distance (EMD).

where are the weights obtained through a global optimisation process that globally minimizes [20]. Given the bag of features representation of images from different classes, a classifier is trained to classify the images.

**7.3 Label Classification**

We train a SVM for each attribute to obtain binary SVM classifiers which can then be used to predict the labels for an unknown class. We transform the previously computed EMD, between the bags into a kernel using extended Gaussian kernel [21].

For each binary SVM classifier, Let where denotes an n-dimensional bag of features vector for the class, label where 1 denotes that the current attribute is associated with this class and -1 indicates that the attribute is not associated with the given class. Hence a total of binary SVM classifiers are obtained where gives the number of attributes. For kth classifier model, we can represent it by decision function. Using the representor theorem [] we have,

The decision function can then be given as

For the chosen symmetric kernel (), using mercers theorem.

The decision function is represented in the dual form

The variable vector can be determined by minimizing the following standard support vector formulation

s.t.

The parameters and are found by cross validation during the training phase.

**7.4 Multi Label Training**

For each binary SVM classifier we do a 10 fold cross validation to find parameters and and in each fold, the ratio of positive samples to negative samples is kept the same. We use LibSVM [ ] for training the SVM with a precomputed Gram matrix [ ]. For each iteration of the binary SVM classifier, an average F1 score is computed and if it is better than the previous iteration then the new values of and are kept. Thus for each of the binary classifiers we get pairs of and values which are used in the predicting the labels for whole of the training set to determine the optimal threshold value. The obtained, and values are then used to predict the labels/attributes for the test set.

**7.5 Association Rules**

We mine association rules between the labels for further increasing the accuracy of the proposed system. Two types of rules have been considered, positive association rules and negative association rules. We take positive association rules to be those which predict the presence of a label for a class given the presence (absence) of certain other labels. Conversely, negative association rules are those which predict the absence of a label for a class given the presence (absence) of other associated labels. We have used PrefixSpan[ ] algorithm to mine data patterns in the training data with support value greater than , where is the minimum support value. For the resulting data patterns, those with a single item are not considered. For each label occurring in the remaining data patterns, confidence is computed for positive and negative sequences where confidence for a sequence is given as

here is the label under consideration, is the sequence containing for which confidence is being computed. Sequences with confidence values above minimum confidence value are added to the set of positive or negative association rule depending on the sequence. Minimum confidence values for individual labels are decided by their relative F1 scores. Labels with high F1 score have higher minimum confidence values and those with a low F1 score have relatively lower confidence values such that the difference between the minimum confidence value and F1 score is kept constant. Table 1 shows the positive and negative rules mined between different labels along with their corresponding support and confidence.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pattern | 0 | 0 8 | 0 ~4 8 | 0 ~4 ~7 8 | 1 | 1 2 | 1 2 4 | 1 2 5 | 1 2 5 6 |
| Support | 130 | 109 | 104 | 101 | 190 | 164 | 99 | 156 | 156 |

(a)

|  |  |
| --- | --- |
| **Rule** | **Confidence** |
| 2 4 🡪 6 | 1.0 |
| 1 2 🡪 5 | 0.951 |
| 1 2 5 🡪 6 | 1.0 |

|  |  |
| --- | --- |
| **Rule** | **Confidence** |
| 0 8 🡪 ~4 | 0.954 |
| 0 ~4 8 🡪 ~7 | 0.971 |

(b) (c)

*Table 1 : a) subset of patterns mined from the input label sequence with support value=0.25. b) Positive association rules for given subset. c) Negative association rules for the given subset. ~ symbol before a label number represents the absence of the label.*

**Chapter-9 RESULT and CONCLUSION**

We evaluate the efficiency of the proposed algorithm on the basis of F1 score. We tested the approach on Yelp Kaggle database [25] in order to include realistic images of a variety of resolution and sizes. The experiments are conducted using Caffe deep learning framework.

**9.1 Dataset**

Yelp Kaggle database contains user uploaded restaurant images. The problem is of multiimage multi label classification with each restaurant having multiple labels that need to be automatically tagged to each business/restaurant. The database contains around 230,000 images in the training dataset with 1996 businesses and around 240,000 images in the test dataset with 10,000 businesses. The label set is limited to size of 9 with each label depicting different meaning.

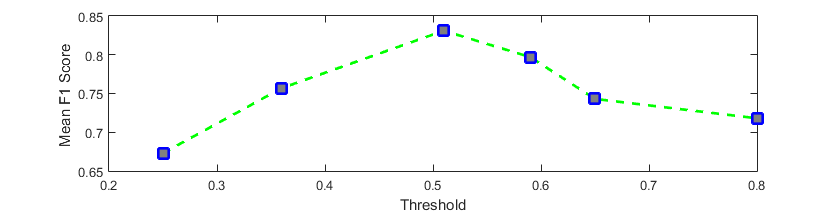
|  |  |
| --- | --- |
| Label # | Label Name |
| 0 | Good for lunch |
| 1 | Good for dinner |
| 2 | Takes reservations |
| 3 | Outdoor seating |
| 4 | Restaurant is expensive |
| 5 | Has alcohol |
| 6 | Has table service |
| 7 | Ambience is classy |
| 8 | Good for kids |

*Table 2. Labels for businesses*

The images shown in Fig 1. correspond to the images of particular business. The images not only contain food, but also the restaurant name, menu cards and drinks. As the images are user-uploaded photos, they do not have consistency. Some images are in portrait mode, some in landscape mode, some are in square shape etc. To make the images consistent and compatible with CNN they are reduced to fixed size of 227x227 pixels. The images in the training dataset are split into 70-30 training validation set. The proposed algorithm is trained on the training set, validated and the F1 score evaluated on the test set is reported as the performance measure.

**9.2 Implementation Details and Results**

We employed Caffe deep learning framework to compute the feature vector for all the training images. The standard BoW model is used to construct bag of features for each business. For vector quantization of features we use the EMD to calculate the distance of bag from every other bag in the training set. We use LibSVM and employ a custom kernel SVM for classification. Finally, we utilized association rule mining to extract rules between labels to further increase the accuracy of the proposed system. The SPMF open source java library is used for mining rules One of the critical parameters of our system is the threshold value ′ ′ defined above, we experimented for various values of ′′ and found 0.47 to be the optimal value. It is noticed that on increasing the threshold value there is a significant drop in correct labelling (true positives) for the system while decreasing the threshold results in degradation of performance due to an increase in mislabeling(false positives) as shown in Fig 2. It is further noticed that a slight increase in threshold value leads to a small improvement in the prediction of training dataset but the system performance degrades for test dataset. This can be justified as there is some noise in the training dataset and such fine-tuning increases the probability of overfitting.



*Fig 2. F1 score vs Threshold*

From Table 3, we observe that our system performs better for certain labels in comparison to other labels. Maximum F1 score is achieved for label 6:has\_table\_service while the minimum F1 score is obtained for label 3:outdoor\_seating. This can be attributed to the fact that majority of the images in the dataset of businesses with label 3 are in an indoor setting. It is also observed that the maximum gain in accuracy from association rules is obtained for label 3. For other labels there is a marginal to moderate increase in accuracy.

From Table 4, we observe that our system suffers from a higher chance of misclassification when the number of attributes associated with a business are low and there is strong variation in ambient illumination. This is due to limited applicability of association rules under the given conditions coupled with sensitivity of certain labels namely 0,1,3 and 7 on lighting conditions.

|  |  |  |
| --- | --- | --- |
| **Label** | **F1 score w/o Association Rules** | **F1 score with Association Rules** |
| 0:good\_for\_lunch | 0.75660 | 0.75682 |
| 1:good\_for\_dinner | 0.85372 | 0.85372 |
| 2:takes\_resrvations | 0.89410 | 0.89410 |
| 3:outdoor\_seating | 0.70193 | 0.74726 |
| 4:restaurant\_is\_expensive | 0.80407 | 0.80715 |
| 5:has\_alcohol | 0.87823 | 0.87893 |
| 6:has\_table\_service | 0.94421 | 0.94844 |
| 7:ambience\_is\_classy | 0.78620 | 0.79104 |
| 8:good\_for\_kids | 0.88831 | 0.88831 |

*Table 3: mean F1 score of each label for our model without Association rules and our model with Association rules. The mean F1 score is computed for 30% of the training data reserved for validating the model*

From Table 4, we observe that our system suffers from a higher chance of misclassification when the number of attributes associated with a business are low and there is strong variation in ambient illumination. This is due to limited applicability of association rules under the given conditions coupled with sensitivity of certain labels namely 0,1,3 and 7 on lighting conditions.

|  |  |  |
| --- | --- | --- |
| **Business Photos** | **Correct Labels** | **Predicted Labels** |
| C:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz9\55405.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz9\98432.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz9\124691.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz9\410723.jpg  C:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz199\27577.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz199\231917.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz199\191505.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz199\445720.jpg  C:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz227\87644.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz227\298690.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz227\119962.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz227\439951.jpg  C:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz228\18027.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz228\63731.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz228\320460.jpgC:\Users\vaibhav\Documents\Visual Studio 2015\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\biz228\336021.jpg | 1,2,4,5,6,7 | 1,2,4,5,6,7 |
| 0,1,2,3,5,6,7 | 0,1,2,3,5,6,7 |
| 3 | 0 |
| 1 3 8 | 0 3 8 |

It can be observed from Table 5 that our model outperforms the listed rival models in terms of mean F1 score. Also, there is a significant increase in accuracy over rival models. The 2 Layer model uses fully connected neural network (FCNN) to extract features. The VGG model is trained on a relatively small dataset and uses GoogleNet framework to extract features. This is mainly due to the fact that we use separate binary SVM classifiers for prediction of each label.

***Comparison Table***

|  |  |
| --- | --- |
| **Model** | **Mean F1 Score** |
| Random Guesser [ ] | 0.41337 |
| 2 Layer FC NN [ ] | 0.49 |
| VGG CNN-S + Fine-tuning [ ] | 0.60881 |
| Benchmark with Color Features [ ] | 0.64598 |
| Our model | 0.82245 |

Table 5: mean F1 score for various models along with the provided benchmark

Moreover the relationships between different labels are utilized in the form of association rules to further enhance the system performance by predicting missing labels (positive association rules) and removing misclassified labels (negative association rules) . Also, tenfold cross validation for parameters increases the robustness of the system.

**Chapter-11 REFERENCES**

[1] Jaume Amores. Multiple instance classification: Review, taxonomy and comparative study. Artificial Intelligence, 201:81–105, 2013.

[2] P. Auer, R. Ortner, A boosting approach to multiple instance learning, in: Proc. of European Conference on Computer Vision, 2004, pp. 63–74.

[3] Z.-H. Zhou and M.-L. Zhang. Neural networks for multi instance learning. In ICIIT, 2002

[4] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. "Solving the multiple-instance problem with axis-parallel rectangles," Artificial Intelligence, vol.89, no.1-2, pp.31-71, 1997.

[5] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In NIPS, 2002.

[6] J. Ramon and L. De Raedt. Multi instance neural networks. In ICML workshop on attribute-value and relational learning, 2000.

[7] O. Maron and T. Lozano-Pérez. "A framework for multiple-instance learning," in: Advances in Neural Information Processing Systems 10, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds. Cambridge, MA: MIT Press, pp.570-576, 1998.

[8] Yang, C. and Lozano-Pérez, T.: Image database retrieval with multiple-instance learning techniques, In: Proceedings of the 16th International Conference on Data Engineering, pp. 233-243, San Diego, CA, 2000

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. Signal Processing Magazine, IEEE, 29(6):82–97, 2012.

[10] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In NIPS, 2010

[11] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In ICML, 2009.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR, 2014

[13] Y. Xu, T. Mo, Q. Feng, P. Zhong, M. Lai, E. I. Chang, et al. Deep learning of feature representation with multiple instance learning for medical image analysis. In ICASSP, 2014

[14] H. O. Song, Y. J. Lee, S. Jegelka, and T. Darrell. Weaklysupervised discovery of visual pattern configurations. In NIPS, 2014

[15] J. Wang and J.-D. Zucker. Solving the multiple-instance problem: a lazy learning approach, Proceedings 17th International Conference on Machine Learning, pp. 1119-1125, 2000

[16] Y. Chevaleyre, J. D. Zucker, Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. Application to the mutagenesis problem. In: Stroulia, E., Matwin, S. (eds.): Lecture Notes in Artificial Intelligence, Vol. 2056. Springer, Berlin (2001) 204-214.

[17] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012

[19] A. Bosch, X. Muñoz, and R. Martí, “Which is the best way to organize/classify images by content?” Image and Vision Computing, vol. 25, no. 6, pp. 778–791, 2007

[20] Y. Rubner, C. Tomasi, and L. Guibas. The Earth Mover’s distance as a metric for image retrieval. International Journal of Computer Vision, 40(2):99–121, 2000.

[21] O. Chapelle, P. Haffner, V. Vapnik, Support vector machines for histogram-based image classification, IEEE Transactions on Neural Networks 10 (5) (1999) 1055–1064.

[22] Mercer, J. (1909), "Functions of positive and negative type and their connection with the theory of integral equations", Philosophical Transactions of the Royal Society A 209(441–458):415– 446, doi:10.1098/rsta.1909.0016

[23] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[24] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M-C. Hsu, PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proc. 2001 Int. Conf. Data Engineering (ICDE'01), Heidelberg, Germany, pp. 215–224, 2001

[25] Yelp Kaggle Database. https://www.kaggle.com/c/yelp-restaurant-photo-classification