2011

Mining Associations Using Directed Hypergraphs

Ramanuja N. Simha
University of South Florida, rsimha@mail.usf.edu

Mining Associations Using Directed Hypergraphs

by

Ramanuja N. Simha

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Rahul Tripathi, Ph.D.
Yicheng Tu, Ph.D.
Xiaoning Qian, Ph.D.

Date of Approval:
March 9, 2011

Keywords: Association Rules, Multi-valued Attributes, Financial Time-series,
Discretization, Clustering, Similarity, Dominator

DEDICATION

to my parents, brother, and sister

ACKNOWLEDGEMENTS

This thesis has been written under the kind supervision of Dr. Rahul Tripathi, my thesis advisor. I am extremely grateful for his helpful guidance and feedback while reviewing the various versions of the thesis draft. His in-depth technical insights and comments have been immensely motivational to me in achieving a high quality of research.

I am extremely grateful to Dr. Mayur Thakur from Google Inc., for being a collaborator in this project and for providing me with ideas and in-depth technical insights during the discussions. We had a number of e-mail and phone conversations during which his comments have been highly motivational to me.

I wish to thank Dr. Rahul Tripathi for permitting me to include work done jointly with him. The work in Chapter 3 and Chapter 4 was jointly done with him.

My thesis committee members were Dr. Rahul Tripathi, Dr. Yicheng Tu, and Dr. Xiaoning Qian. I am extremely thankful to all of them for their helpful comments.

I have learnt immensely about achieving very high levels of academic quality, technical excellence, and aptitude to learn, by taking theory courses under Dr. Rahul Tripathi, and by working as a Teaching Assistant for theory courses and as a Research Assistant under Dr. Rahul Tripathi. I have been fortunate for having had such an opportunity. During this period, the experiences hugely contributed in shaping my research interests in algorithms and problem solving.

I wish to thank Srinivasan Ramkumar from Qualcomm/Riverbed Technology for providing support and motivation throughout my studies by understanding the need to conquer challenges for becoming successful in graduate research.

I would like to thank my parents, my brother, and my sister for giving me inspiration and support whenever I needed during my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

This thesis proposes a novel directed hypergraph based model for any database. We introduce the notion of association rules for multi-valued attributes, which is an adaptation of the definition of quantitative association rules known in the literature. The association rules for multi-valued attributes are integrated in building the directed hypergraph model. This model allows to capture attribute-level associations and their strength. Basing on this model, we provide association-based similarity notions between any two attributes and present a method for finding clusters of similar attributes. We then propose algorithms to identify a subset of attributes known as a *leading indicator* that influences the values of almost all other attributes. Finally, we present an association-based classifier that can be used to predict values of attributes. We demonstrate the effectiveness of our proposed model, notions, algorithms, and classifier through experiments on a financial time-series data set (S&P 500).

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Data Mining involves searching interesting patterns and/or classifying data. Association rules help to discover interesting patterns by identifying implication relationships among attribute-value pairs present in the data. Similarly, classification rules help to classify data by predicting values of specific attributes in the data. In general, any association or classification rule consists of two components: the antecedent and the consequent. Association rules may have more than one attribute in their consequent, whereas classification rules always have a single attribute in their consequent. An example of an association rule is: "If a customer buys milk and diapers, then the customer also buys beer and eggs." Here, milk, diapers, beer, and eggs are attributes, and they each take value '1' (where '1' denotes present and '0' denotes absent) in the association rule. An example of a classification rule is: "If weather is sunny and humidity is low, then play is true." Here, weather, humidity, and play are attributes, and the values 'sunny' and 'low' help to predict the value 'true' for the attribute play. Attributes can be either categorical, e.g., days of a week, or quantitative, e.g., stock price. Agrawal et al. [AIS93] presented association rules that target identifying implication relationships among categorical attributes that take only 0/1 values. Such association rules are called *boolean* association rules. Srikant et al. [SA96] introduced the more general *quantitative* association rules that accommodate both categorical and quantitative attributes. Henceforward, we will refer to quantitative association rules as association rules.

Association rules find their use in identifying implication relationships among attributes in market-basket type data. This data type is a transactional data where each observation in the database is a transaction consisting of items purchased by a customer. Much work [BAG99, SA95, AS94, SVA97, NLHP98] has been done in mining association rules

that satisfy constraints such as, minimum support (a measure of significance) and minimum confidence (a measure of predictive ability). Association rules can be used to solve the following problems: finding clusters of similar attributes, finding a small subset of attributes that influences a large section of other attributes, and finding classification rules to predict values of attributes. For instance, in market-basket type data, a practical application of association rules is to identify clusters of similar items based on the customer sales information. This helps to understand patterns in sales of items and to group items based on customer interests. Similarly, identifying a small subset of items that influence the sales of all other items in market-basket type data may help in recognizing major sales indicators. Also, using the classification rules, the purchase of particular items for customers could be predicted based on the prior purchases made by them.

In fact, applications of association rules go far beyond the realms of just the market-basket type domain. Some other domains where association rules have important applications are as follows: in medicine for identifying relationships among medical conditions and diseases [Ord06], in bioinformatics for identifying interrelationships among genes [CH03, CSCR$^+$06], in social networks for identifying social relationships, and in finance for identifying prediction relationships among stocks. We provide below some examples from these domains that will help elucidate the applications of association rules. In each of these examples, 35% is the confidence and 5% is the support of the stated rule.

1. "35% of the times when the age of a patient is in the range 40-49 years and the cholesterol of the same patient is in the range 220-229 mg/dL, the patient's blood-pressure is in the range 130-139 mmHg" and this rule occurs in 5% of the total observations. Here, age, cholesterol, and blood-pressure are attributes.

2. "35% of the times when gene 1 and gene 2 in a patient are under expressed, gene 3 is over expressed in the patient" and this rule occurs in 5% of the total observations. Here, gene 1, gene 2, and gene 3 are attributes.

3. "35% of the times when a person has high interest in reading and playing, the person has low interest in music" and this rule occurs in 5% of the total observations. Here, reading, playing, and music are attributes.

4. "35% of the times when the price of stock A and of stock B go up, the price of stock C goes down" and this rule occurs in 5% of the total observations. Here, stocks A, B, and C are attributes.

Our present work is motivated by the goal of building a model that inherently handles many-to-many relationships, thus enabling to capture these relationships among attributes of a database more accurately. Such a model is expected to be suited for handling problems such as, similarity and clustering because many of the relationships exhibited in real world are not restricted to be one-to-one.

Knobbe et al. [KH06] proposed an approach to mine a small set of binary attributes that help differentiate observations in a database. Siebes et al. [SVL06] and Bringmann et al. [BZ07] proposed techniques for compressing a database. The compressed set of binary attributes could then be used in a data mining classifier to avoid overfitting. The above methods mainly attempt to identify informative sets of *binary attributes*, whereas our approach attempts to build a generic model for any database containing *multi-valued attributes* and address a variety of problems such as, similarity, clustering, leading indicators, and classification.

In this thesis, we propose a novel model for any database using a directed hypergraph in which the nodes represent attributes and the directed hyperedges represent many-to-many relationships among the attributes. The weights on directed hyperedges capture the likeliness of association in a particular direction. We introduce the notion of association rules for multi-valued attributes, which is an adaptation of the definition of quantitative association rules known in the literature [SA96]. The association rules for multi-valued attributes are integrated in building the directed hypergraph model. Basing on this model, we provide association-based similarity notions between any two attributes, present a method for finding clusters of similar attributes, and propose an algorithm to identify a subset of attributes known as a *leading indicator* that influences the values of almost all other attributes. Finally, we present an association-based classifier that can be used to predict values of attributes. We demonstrate the effectiveness of our proposed model, notions, algorithm, and classifier through experiments on a financial time-series data set (S&P 500).

1.2 Thesis Overview

We present background on approximation algorithms and data mining in Chapter 2. The various approximation algorithms presented are: finding minimum cost set cover, finding minimum size graph dominator, and finding minimum diameter clustering. We then discuss some fundamental concepts and approaches in data mining such as linear regression, perceptron rule learning algorithm, and k-means clustering.

In Chapter 3, we introduce the notion of association rules for multi-valued attributes, which is an adaptation of quantitative association rules known in the literature. We then propose a novel directed hypergraph based modeling for any database that allows us to compute association-based similarity between multi-valued attributes and find clusters of similar multi-valued attributes.

Using the directed hypergraph model, we propose in Chapter 4 two greedy algorithms for identifying a subset of attributes known as a leading indicator that influences the values of almost all other attributes. The first greedy algorithm is based on an adaptation of the greedy $O(\log n)$-approximation algorithm for computing a minimum cardinality dominating set in graphs. The second greedy algorithm is based on an adaption of the greedy $O(\log n)$-approximation algorithm for computing a minimum cost set cover. Finally, we present an association-based classier that can be used to predict values of attributes.

In Chapter 5, we conduct experiments on financial time-series obtained from Yahoo Finance [Yah10] on the following problems: (a) finding clusters of similar attributes, (b) finding leading indicators, and (c) predicting values of financial time-series using the association-based classifier. We propose an equi-depth partitioning technique to discretize the financial time-series in the S&P 500. This transformation is used to construct a database that is suitable for the association hypergraph modeling. The directed hypergraph $\mathcal{H}$ required for the experiments is then constructed using the database. We show the association characteristics of financial time-series by presenting the degree distribution of nodes in the association hypergraph, by displaying the directed edges and directed hyperedges with highest association confidence value (ACV) for selected financial time-series, and by comparing the directed hyperedges with the highest ACV with those of the corresponding two directed edges. We

also display the clusters of financial time-series in the similarity graph, compute the leading indicators for a collection of financial time-series, and present the statistics of financial time-series predictions.

In this thesis we show that directed hypergraphs, due to their inherent structure, help us to capture interesting characteristics that exhibit many-to-one relationships among attributes in a database. The proposed model displays the versatile usage of directed hypergraphs in addressing various problems relevant to mining associations in databases. Our experiments on financial time-series demonstrate that directed hypergraphs capture more relationships than directed graphs and in the stock market domain, our modeling allows us to address problems such as computing financial time-series similarity, computing financial time-series leading indicators, and predicting financial time-series values, using a common framework/methodology.

CHAPTER 2

PRELIMINARIES

2.1 Approximation Algorithms

There are many problems whose decision version is NP-complete and optimization version is NP-hard in nature, i.e., obtaining an optimal solution to the problem in polynomial time is currently unknown. Due to their practical importance, it is extremely useful to have an efficient polynomial time solution to such problems by carrying out any of the following: (a) constraining the input, (b) introducing randomization in the solution approach, and (c) obtaining a near optimal solution. In this section, we shall focus on obtaining a near optimal solution to hard problems. Approaches that produce a near optimal solution are used in the design of polynomial time approximation algorithms. The quality measure of an approximation algorithm is given by its approximation factor.

Let us say that we have an optimization problem. Based on the problem, an optimal solution may be a solution with either the maximum value or the minimum value. In other words, the problem may be a maximization or a minimization.

Definition 2.1 *For an optimization problem A that takes input I, an algorithm ALG has an approximation ratio α if the cost $ALG_A(I)$ of the solution produced by the algorithm on input I is within a factor of α of the cost $OPT_A(I)$ of an optimal solution on the same input I. That is,*

$$\max\left(\frac{ALG_A(I)}{OPT_A(I)}, \frac{OPT_A(I)}{ALG_A(I)}\right) \leq \alpha.$$

For a minimization problem, the ratio $ALG_A(I)/OPT_A(I) \leq \alpha$ gives a factor by which the solution cost produced by the algorithm exceeds the solution cost of an optimal solution. For a maximization problem, the ratio $OPT_A(I)/ALG_A(I) \leq \alpha$ gives a factor by which the solution cost of an optimal solution exceeds the solution cost produced by the algorithm.

2.1.1 Set Cover

Definition 2.2 *Let U be a universe consisting of n elements and let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ be a collection of subsets of U. A set cover SC is a subcollection of $\mathcal{S}$ that covers all the elements in U.*

Algorithm 1 presented below computes a set cover SC of size $ALG_{SC}(I)$ given a universe U of size n and a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of U as input I. Let $Cover$ denote the set of elements that are currently covered by the algorithm. The greedy strategy of the algorithm is as follows: for every subset $S_i \in \mathcal{S}$ that is not part of the set cover yet, the algorithm computes its cost effectiveness $\alpha(S_i)$ that reflects S_i's covering ability, i.e, $\alpha(S_i)$ is the average cost paid by the greedy algorithm to cover the elements in S_i that are not already in $Cover$, i.e., $1/|S_i - Cover|$. During each iteration of the algorithm (until all the elements in U are covered), the subset S_i^* with the lowest average cost (highest cost effectiveness) is added to the set cover. In other words, a price of $1/|S_i^* - Cover|$ is paid to cover each element in $S_i^* - Cover$. Therefore, the cost of the set cover obtained is $\sum_{k=1}^{n} price(u_k)$.

The greedy set cover algorithm can be implemented in linear time.

Theorem 2.3 ([Joh74, Lov75, Chv79]) *Given a universe U of size n and a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of U as input I, the cost of the set cover $ALG_{SC}(I)$ computed by Algorithm 1 is at most a factor of $O(\log n)$ greater than the cost of an optimal set cover $OPT_{SC}(I)$ in $O(n \log m)$ time.*

Proof Let $u_1, u_2, \ldots, u_n \in U$ be the order in which the elements are added to $Cover$ by Algorithm 1. Since an optimal solution can cover all the elements in U with a cost of $OPT_{SC}(I)$, there must always exist a set having an average cost of at most $OPT_{SC}(I)/|U - Cover|$. We know that $|U - Cover|$ is at least $n - k + 1$ when an element u_k is about to be covered. Since the algorithm picks the lowest average cost subset during each iteration, we have

$$price(e_k) \leq \frac{OPT_{SC}(I)}{|U - Cover|} \leq \frac{OPT_{SC}(I)}{n - k + 1}.$$

Input : A universe U of size n and a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of U.
Output: A set cover SC.

```
1  begin
2      SC ← ∅;
3      Cover ← ∅;
4      while Cover ≠ U do
5          foreach S ∈ S do
6              count ← 0;
7              foreach u ∈ S do
8                  if u ∉ Cover then
9                      count ← count + 1;
10                 end
11             end
12             α(S) ← 1/count;
13         end
14         Let S_0 be such that α(S_0) ← min_{S∈S}(α(S));
15         Cover ← Cover ∪ S_0;
16         SC ← SC ∪ {S_0};
17     end
18     return SC;
19 end
```

Algorithm 1: A greedy algorithm for computing a set cover.

Therefore, the cost of the set cover $ALG_{SC}(I)$ is

$$\begin{aligned} ALG_{SC}(I) &= \sum_{k=1}^{n} price(u_k) \\ &\leq \sum_{k=1}^{n} \frac{OPT_{SC}(I)}{n-k+1} \\ &\leq \left(\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}\right) \times OPT_{SC}(I) \\ &\leq \log n \times OPT_{SC}(I). \end{aligned}$$

This completes the proof. ∎

2.1.2 Graph Dominating Set

Definition 2.4 *Let $G = (V, E)$ be a graph. A dominating set is a subset $DomSet \subseteq V$ of vertices such that, for each vertex $v \in V$, either there exists an edge $(u, v) \in E$ such that $u \in DomSet$ or $v \in DomSet$.*

Theorem 2.5 ([Joh74, Lov75, Chv79]) *There is a greedy $O(\log n)$-approximation algorithm for graph dominating set problem that, given any graph $G = (V, E)$ as input I, computes a dominating set of G whose size is within $O(\log n)$ of the optimal dominating set size, where n is the number of vertices and m is the number of edges, of G.*

The following construction can be used to transform an instance of graph dominating set into an instance of set cover. Let $U = V$ be the set of vertices that need to be covered and let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be subsets of vertices, where each subset S_i contains vertex v_i and its neighborhood set $N(v_i)$. This instance of set cover can be solved using the approximation result in Theorem 2.3. Since every set added by Algorithm 1 to the set cover always adds only a vertex to the graph dominating set, the approximation guarantee provided in the theorem for the size of the set cover holds for the size of the graph dominating set.

2.1.3 The t-clustering

Let us assume we have a set of attributes, e.g., financial time-series, genes, or images, and we are required to find independent groups of attributes where attributes belonging to the same group are similar to each other. The general clustering problem addresses this by defining an objective function based on which the attributes are grouped. A variant of the general clustering problem is the problem t-clustering, which groups the attributes based on how close they are to each other. In other words, the objective of t-clustering is to group attributes into t clusters so that the maximum distance between any two attributes within the same cluster (also called the diameter of the grouping a.k.a. clustering) is minimized. This problem also assumes that the Euclidean distance function displays metric properties. A distance function $d(\cdot, \cdot)$ on a point set $X = \{x_1, x_2, \ldots, x_n\}$ is said to have metric properties if and only if it satisfies:

1. $d(x_1, x_2) \geq 0$ for all $x_i \in X$, and
2. $d(x_1, x_2) = 0$ if and only if $x_1 = x_2$, and
3. $d(x_1, x_2) = d(x_2, x_1)$, and
4. $d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2)$ (Triangle Inequality).

Definition 2.6 *Let $X = (x_1, x_2, \ldots, x_n)$ be a set of points with a distance function between any pair of points $d(\cdot, \cdot)$ that satisfies the metric properties and let t be an integer. A t-clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ is a partition of X into t clusters $C_1, C_2, \ldots, C_t$ by designating t points of X as centers, such that $\mathcal{C}$ minimizes the dimeter $Diam(\cdot)$ over all possible clusterings. That is $\mathcal{C}$ minimizes*

$$Diam(\mathcal{C}) = \max_i \max_{x_a, x_b \in C_i} d(x_a, x_b).$$

Algorithm 2 presented below finds such a partition of X by designating some t points as cluster centers. Initially, any point $\mu_1 \in V$ is picked as the first cluster center. During each iteration of the algorithm, it finds a point $\mu_i \in X$ that is the farthest from the existing centers $\mu_1, \mu_2, \ldots, \mu_{i-1}$, i.e., a point $\mu_i \in X$ that maximizes $\min_{j<i} d(\cdot, \mu_j)$. This iterative process is carried out until there are t cluster centers. Now, t clusters are created such that the i'th cluster C_i contains all points $x \in X$ whose closest center is μ_i.

Input : $X = (x_1, x_2, \ldots, x_n)$ be a set of points with a distance function between any pair of points $d(x_a, x_b)$ and the number of clusters t.
Output: A clustering C where each C_i is a cluster whose closest center is μ_i.

```
1 begin
2     Pick any μ_1 ∈ X as the first cluster center;
3     for i = 2 to t do
4         Let μ_i be the point in X that is farthest from μ_1, ..., μ_{i-1}; (i.e., that
          maximizes min_{j<i} d(·, μ_j));
5     end
6     Create t clusters such that the i'th cluster is C_i = {all x ∈ X whose closest center
      is μ_i};
7 end
```

Algorithm 2: The t-clustering algorithm.

Theorem 2.7 ([Gon85]) *Given a set of points $X = \{x_1, x_2, \ldots, x_n\}$ and a distance function between any pair of points $d(\cdot, \cdot)$ that satisfies metric properties and an integer t as input I, the diameter $ALG_{TC}(I)$ computed by Algorithm 2 is at most a factor of 2 greater than the cost of an optimal diameter $OPT_{TC}(I)$.*

Proof Let us consider the farthest point $u \in X$ from the existing centers $\mu_1, \mu_2, \ldots, \mu_t$. If r is the distance between u and the center closest to it, then every point in X has a distance

less than r to its closest center. From triangle inequality, the diameter of the t clustering is at most $2r$. Also, from the way the centers are chosen in every iteration of the Algorithm 2, we know that all the centers $\mu_1, \mu_2, \ldots, \mu_t$ and the point u are placed at least r distance apart from each other. If a t-clustering of these points is found, then the diameter of the clustering is at least r (i.e., $OPT_{TC}(I)$). From above we have

$$ALG_{TC}(I) \leq 2 \times OPT_{TC}(I).$$

This completes the proof. ∎

2.2 Directed Hypergraphs

Directed hypergraphs are a generalization of directed graphs in which each directed hyperedge has one or more source (tail) vertices and has one or more destination (head) vertices. They have found a variety of applications in Computer Science, e.g., in propositional logic [AG97], databases [ADS83, ADS85], scheduling [LS95, GS02], routing in dynamic networks [Pre00], bioinformatics [KNO$^+$03], and data mining [CDP04]. Theoretical problems related to directed hypergraphs have been studied in [Vie04, TT09].

A related notion is undirected hypergraphs, which generalize undirected graphs.

Definition 2.8 [Ber73] *An* undirected hypergraph $\mathcal{G}$ *is a pair* $(V_{\mathcal{G}}, E_{\mathcal{G}})$*, where* $V_{\mathcal{G}}$ *is a finite set of vertices and* $E_{\mathcal{G}} \subseteq 2^{V_{\mathcal{G}}}$ *is a finite set of hyperedges such that, for every hyperedge* $e_i \in E_{\mathcal{G}}$*,* $e_i \neq \emptyset$*.*

Definition 2.9 [GLPN93] *A* directed hypergraph $\mathcal{H}$ *is a pair* (V, E)*, where* V *is a finite set of vertices and* $E \subseteq 2^V \times 2^V$ *is a finite set of directed hyperedges such that, for every directed hyperedge* $e = (T, H) \in E$*,* $T \neq \emptyset$*,* $H \neq \emptyset$*, and* $T \cap H = \emptyset$*. (Here,* T *is called the tail set and* H *is called the head set of* e*.)*

Notice that directed hypergraphs are different from undirected hypergraphs [Ber73]. While directed hypergraphs generalize directed graphs, undirected hypergraphs generalize undirected graphs.

2.3 Data Mining

Data mining involves learning information or concepts from databases and using them to solve problems. Based on the structure of the database, there are different problems that can be addressed. For example, suppose our database contains attributes such as humidity, weather, temperature, and type of play, and contains observations that in fact are records of values that these attributes take on different days. In order to learn about the type of play, one can fix type of play as the *classification attribute* and look into the values of other attributes. A classifier is an algorithm that learns rules about the classification attribute based on other attribute values. This approach can be used to predict the class attribute value of unseen observations. Since the classifier is built under the supervision of a set of observations, i.e., training dataset, this method is known as supervised classification. The class value may also be non-discrete in nature.

In certain circumstances, a database may not have any information to distinguish a particular attribute as the classification attribute. In such a scenario, learning information about the database may not aid in the prediction of a class value. However, learning information not only corresponds to the prediction of some attribute value in the database, but also can be visualized as inferring relationships among attributes. For example, in the sample database discussed earlier, an interesting association can be as follows: "The pattern of humidity having the value 80%, temperature having the value 90F, and weather having the value Rainy is very frequent in the database." Such an inference is known as an association rule. Here, there is no particular attribute whose value is predicted. But, as we have seen in the example, learning information corresponds to identifying interesting and useful patterns among attributes.

In the previous example, we saw that learning information can be visualized in terms of inferring relationships among attributes. If this concept is generalized, then it can be termed as identifying groups of attributes with similar characteristics. Here, characteristics may relate to the patterns among attributes identified earlier. This problem is known as clustering, as the objective is to find attributes and classify them into groups based on their characteristics. For example, let the set of attributes be a set of financial time-series

and the set of observations be the stock prices recorded for the financial time-series on various days. An interesting question here is to group the financial time-series into clusters containing similar financial time-series. The quality of the clustering obtained is generally verified based on the information available to the user. In this case, one way to verify the quality of the financial time-series clusters obtained is by looking at the sectors of the time-series. Here, the quality of the clustering may be defined good if a high percentage of the time-series that belong to the same cluster are from the same sector.

2.3.1 Classification Rule Mining

We review the linear regression classifier used to construct classification rules. Linear regression can be used to predict the class attribute value when there are non-discrete attributes. Let $A_1, A_2, \ldots, A_n$ denote the attributes and let $O_1, O_2, \ldots, O_m$ denote the observations corresponding to the attributes. Let there be a database that is visualized as a $m \times n$ table containing m rows and n columns, where rows correspond to observations and columns correspond to attributes. Let o_{ij} denote the value of the j'th attribute for the i'th observation where $1 \leq i \leq m$ and $1 \leq j \leq n$. For convenience, let us say that the last attribute, A_n, be the class attribute whose value needs to be predicted. In order to predict A_n's value, linear regression builds an attribute weight assignment model where $w_1, w_2, \ldots, w_{n-1}$ are the weights assigned to the first $n-1$ attributes. The weight assignment model (classifier) is iteratively built by computing the predicted value for A_n for each observation. For the i'th observation, A_n's predicted value is given by:

$$w_1 o_{i1} + w_2 o_{i2} + \cdots + w_{n-1} o_{in-1} = \sum_{j=1}^{n-1} w_j o_{ij}.$$

Intuitively, in order to have an accurate prediction of A_n for the i'th observation, the difference between A_n's predicted value $\sum_{j=1}^{n-1} w_j o_{ij}$ and A_n's actual value o_{in} has to be minimized. Thus, for the i'th observation, where $1 \leq i \leq m$, the classifier is constructed by reducing the sum of squares of difference $(o_{in} - \sum_{j=1}^{n-1} w_j o_{ij})^2$. Overall, linear regression minimizes $\sum_{i=1}^{m} \left(o_{in} - \sum_{j=1}^{n-1} w_j o_{ij}\right)^2$. Using the attribute weights in the constructed classifier, the class attribute value of an unseen observation can be predicted.

The basic idea used in linear regression is to express the class attribute value as a linear combination of other attributes. When the attribute values are discrete, expressing the class attribute as a linear combination of other discrete attributes fails to predict the value of the class attribute. This is due to the fact that the error computed between the predicted class value and the actual class value has little meaning as this error does not indicate how far the predicted value is from the actual value. Also, the predictions may lie outside the set of discrete values allowed in the dataset.

```
1  for i ← 0 to n − 1 do
2      w_i ← 0;
3  end
4  while there exists an incorrectly classified observation in the training data do
5      for j ← 1 to m do
6          if O_j is currently incorrectly classified then
7              if O_j belongs to the first class then
8                  Add values of attributes A_0, A_1, ..., A_{n−1} in this observation to
                   weights w_0, w_1, ..., w_{n−1} in the equation;
9              end
10             else
11                 Subtract values of attributes A_0, A_1, ..., A_{n−1} in this observation from
                   weights w_0, w_1, ..., w_{n−1} in the equation;
12             end
13         end
14     end
15 end
```

Algorithm 3: Perceptron learning rule.

We now look at another approach to predict the class attribute value. Let $A_1, A_2, \ldots, A_n$ denote the attributes and let $O_1, O_2, \ldots, O_m$ denote the observations corresponding to the attributes as before. Let us assume the class attribute A_n is 0/1-valued. This approach tries to separate the observations into either 0-valued or 1-valued by using a hyperplane. The equation of the hyperplane is as follows:

$$w_0 A_0 + w_1 A_1 + w_2 A_2 + \cdots + w_{n-1} A_{n-1} = 0.$$

During the construction of the hyperplane, the weight w_0 of attribute A_0, called *bias*, is used to contribute a constant value in the equation of the hyperplane and hence $A_0 = 1$. The following algorithm in Algorithm 3, called *perceptron learning rule*, is used to construct

the hyperplane. A *perceptron* [Ros58] is a binary classifier that classifies an observation into the first class if the sum from the equation is greater than 0 and classifies an observation into the second class otherwise.

We see that the algorithm iteratively either increments or decrements the attribute weights by the attribute values of the observation, if the class attribute value of that observation is incorrectly classified in the equation of the hyperplane. The goal is to predict class values of all the observations correctly. If the data set is not linearly separable, the above algorithm would not terminate. In such cases, the algorithm can be terminated forcefully after the execution of a certain number of iterations.

2.3.2 Clustering

Definition 2.10 ([Llo82]) *Let $X = (x_1, x_2, \ldots, x_n)$ be a set of points with a distance function between any pair of points $d(\cdot, \cdot)$ and let k be an integer. A* k-means-clustering *$\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ is a partition of X into k clusters $C_1, C_2, \ldots, C_k$ such that $\mathcal{C}$ minimizes the sum of squares of distances of points from the centroid $\mu_i = \sum_{x_j \in C_i} x_j / |C_i|$ for each cluster $C_i \in \mathcal{C}$. That is, $\mathcal{C}$ minimizes*

$$\sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2.$$

The k-means clustering algorithm, provided in Algorithm 4, is a well known technique to find a k-means-clustering given points in Euclidean space. Initially k random cluster centers are picked. The k-means algorithm then works iteratively. In each iteration, the algorithm assigns the remaining points to the closest cluster center and computes the centroid for each of the k clusters. These centroids are the new cluster centers for the next iteration. The algorithm continues until the clusters centers between two consecutive iterations remain unchanged.

The output of k-means algorithm depends on the initial cluster centers that are picked. The k-means algorithm in each iteration picks the centroids of the points in the current clusters as new cluster centers $NewCC$. The algorithm terminates when the current cluster centers $CurrentCC$ and new cluster centers $NewCC$ are the same. The k-means clustering

Input : The number of clusters k and a set of points $X = (x_1, x_2, \ldots, x_n)$ with a distance function between any pair of points $d(x_a, x_b)$.
Output: A clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

begin
  $NewCC \leftarrow$ Pick any k points, say, $\mu_1, \mu_2, \ldots, \mu_k$;
  $CurrentCC \leftarrow \emptyset$;
  while $CurrentCC \neq NewCC$ **do**
    $CurrentCC \leftarrow NewCC$;
    Create k clusters such that the i'th cluster is $C_i = \{$all $x \in X$ whose closest center is $\mu_i \in CurrentCC\}$;
    for $i \leftarrow 1$ *to* k **do**
      $\mu_i \leftarrow \frac{\sum_{x_j \in C_i} x_j}{|C_i|}$;
    end
    $NewCC \leftarrow \{\mu_1, \mu_2, \ldots, \mu_k\}$;
  end
end

Algorithm 4: The k-means algorithm.

algorithm is suitable for finding clusters in a set of points that contain subgroups of points that have symmetric shapes such as, circular, since these shapes allow the points to be assigned to unique centroids. This makes the algorithm iteratively move towards the unique centroids of the individual subgroups of points in order to obtain a stable clustering. When the algorithm is input with subgroups of points that have asymmetric shapes with irregular boundaries, it moves the centroids without being able to obtain a stable clustering, thus constantly modifying the clustering configuration. The worst case running time for the k-means algorithm is known to be $2^{\Omega(\sqrt{n})}$ [AV06].

CHAPTER 3

MODELING ASSOCIATIONS IN DATABASES USING DIRECTED HYPERGRAPHS

3.1 Associations Between Multi-Valued Attributes

Let $\mathcal{D}$ be a database in the form of a $m \times n$ table, where the rows correspond to observations and the columns correspond to multi-valued attributes. Let $\mathcal{O} = \{O_1, O_2, \ldots, O_m\}$ be the set of observations and $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be the set of attributes. The table entry for each attribute A_i and each observation O_j is a value from a fixed finite set $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$. We denote such a database $\mathcal{D}$ in the form of $\mathcal{D}(\mathcal{A}, \mathcal{O}, \mathcal{V})$. For any $X \subseteq \mathcal{A} \times \mathcal{V}$, let $\pi_1(X)$ denote $\{A_i \mid \exists v_j (A_i, v_j) \in X\}$.

We next present the definition of an association rule for multi-valued attributes and the support and confidence measures for such an association rule.

Definition 3.1 *An association rule for multi-valued attributes (in short,* mva-type association rule*) in a database $\mathcal{D}(\mathcal{A}, \mathcal{O}, \mathcal{V})$ is an implication relationship of the form $X \overset{mva}{\Longrightarrow} Y$, where $X, Y \subseteq \mathcal{A} \times \mathcal{V}$ and $\pi_1(X)$ and $\pi_1(Y)$ are disjoint subsets of $\mathcal{A}$.*

Definition 3.2 *The support and confidence measures are generalized for multi-valued attributes in a database $\mathcal{D}(\mathcal{A}, \mathcal{O}, \mathcal{V})$ as follows:*

1. *Let $X = \{(A_{i1}, v_{j1}), (A_{i2}, v_{j2}), \ldots, (A_{ir}, v_{jr})\}$ be any subset of $\mathcal{A} \times \mathcal{V}$. The* support *of X, denoted by* $\mathrm{Supp}(X)$*, is defined as the fraction of observations in $\mathcal{D}$ for which A_{i1} takes value v_{j1}, A_{i2} takes value v_{j2}, ..., and A_{ir} takes value v_{jr}.*

2. *Let $X \overset{mva}{\Longrightarrow} Y$ be an mva-type association rule. Then the* confidence *of this rule, denoted by* $\mathrm{Conf}(X \overset{mva}{\Longrightarrow} Y)$*, is defined as follows:*

$$\mathrm{Conf}(X \overset{mva}{\Longrightarrow} Y) = \frac{\mathrm{Supp}(X \cup Y)}{\mathrm{Supp}(X)}.$$

The above definition of an mva-type association rule has been adapted from the definition of a quantitative association rule [SA96] with a minor change to simplify the definition. In an mva-type association rule, attributes are associated with values from a fixed finite set, whereas in a quantitative association rule, attributes are associated with either categorical values (e.g. zip code, make of car) or intervals (e.g. age, income). During the process of discovering quantitative association rules, the attribute values are then mapped to discrete values. On the other hand, our definition of database $\mathcal{D}$ (in particular, the set of values $\mathcal{V}$) assumes that the attribute values are already mapped to discrete values. In this sense, our definition of mva-type association rules simplifies the definition of quantitative association rules given in [SA96].

Note that the definitions of support and confidence in the market-basket type database can be viewed as a special case of Definition 3.2. For instance, let A_1, A_2, and A_3 be 0/1-valued (i.e., *binary*) attributes. Then, the measure "support of $\{A_1, A_2\}$" in the market-basket type data can be seen as equivalent to $\text{Supp}(\{(A_1, 1), (A_2, 1)\})$ and the measure "confidence of $\{A_1, A_2\} \Longrightarrow \{A_3\}$" can be seen as equivalent to $\text{Conf}(\{(A_1, 1), (A_2, 1)\} \stackrel{mva}{\Longrightarrow} \{(A_3, 1)\})$, given by Definition 3.2.

We now give some examples of databases containing multi-valued attributes, observations, and the values that attributes take for each observations. These examples are from various domains, such as, medicine, bioinformatics, and social networks.

Our first example is a Patient database.

Example 3.3 *Consider a Patient database in Table 3.1 where each observation consists of attribute values such as age, cholesterol, blood-pressure, and heart-rate of different patients. Here, patient* 1 *has age* 25 *years, cholesterol* 135 *mg/dL, blood pressure* 135 *mmHg, and heart-rate* 75 *beats per minute. Similarly, records for other patients can be read from this table.*

In order to improve the usability of a database that contains attributes each of which can take real values in an arbitrary range, it is a general practice to discretize the attribute values. In the patient database in Table 3.1, for each attribute value a_i, we consider the discretized value $\lfloor a_i/10 \rfloor$. Table 3.2 displays the discretized database.

Table 3.1 Patient database.

Observations	Attributes			
Patient Id Id	Age A	Cholesterol C	Blood-Pressure B	Heart-Rate H
1	25	105	135	75
2	62	160	165	85
3	32	125	139	71
4	12	95	105	67
5	38	129	135	75
6	39	121	117	71
7	41	134	145	73
8	85	125	155	78

Table 3.2 Patient database (after discretization).

Observations	Attributes			
Patient Id Id	Age A	Cholesterol C	Blood-Pressure B	Heart-Rate H
1	2	10	13	7
2	6	16	16	8
3	3	12	13	7
4	1	9	10	6
5	3	12	13	7
6	3	12	11	7
7	4	13	14	7
8	8	12	15	7

Let us consider the mva-type association rule $X \overset{mva}{\Longrightarrow} Y$, where $X = \{(A, 3), (C, 12)\}$ and $Y = \{(B, 13)\}$. This is an implication relationship that states: "If the age of a patient is in the range 30-39 and the cholesterol of the same patient is in the range 120-129 mg/dL, then it is likely that the patient's blood-pressure is in the range 130-139 mmHg. Here, $\text{Supp}(X) = 3/8 = 0.375$ and $\text{Conf}(X \overset{mva}{\Longrightarrow} Y) = \text{Supp}(X \cup Y)/\text{Supp}(X) = 2/3 = 0.667$.

Our next example is a Gene database.

Example 3.4 *Consider a Gene database in Table 3.3 where each observation consists of attribute values such as gene 1 expression value, gene 2 expression value, gene 3 expression value, and gene 4 expression value of different patients. Here, patient* 1 *has gene 1 expression value* 54.23*, gene 2 expression value* 66.22*, gene 3 expression value* 342.32*, and gene 4 expression value* 422.21*. Similarly, records for other patients can be read from this table.*

Table 3.3 Gene database.

Observations	Attributes			
Patient Id Id	Gene 1 G1	Gene 2 G2	Gene 3 G3	Gene 4 G4
1	54.23	66.22	342.32	422.21
2	541.21	324.21	165.21	852.21
3	321.67	125.98	139.43	71.11
4	123.87	95.54	105.88	678.65
5	388.44	129.33	135.65	754.32
6	399.98	121.54	117.55	719.33
7	414.33	134.73	145.32	733.22
8	855.78	125.93	155.76	789.43

Table 3.4 Gene database (after discretization).

Observations	Attributes			
Patient Id Id	Gene1 G1	Gene 2 G2	Gene 3 G3	Gene 4 G4
1	$\downarrow$	$\downarrow$	$\leftrightarrow$	$\leftrightarrow$
2	$\leftrightarrow$	$\downarrow$	$\downarrow$	$\uparrow$
3	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
4	$\downarrow$	$\downarrow$	$\downarrow$	$\uparrow$
5	$\leftrightarrow$	$\downarrow$	$\downarrow$	$\uparrow$
6	$\leftrightarrow$	$\downarrow$	$\downarrow$	$\uparrow$
7	$\leftrightarrow$	$\downarrow$	$\downarrow$	$\uparrow$
8	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$

In the gene database in Table 3.3, for each attribute value a_i, we consider the discretized value $\downarrow$ if $0 \leq a_i \leq 333$, the discretized value $\leftrightarrow$ if $334 \leq a_i \leq 666$, and the discretized value $\uparrow$ if $667 \leq a_i \leq 999$. Table 3.4 displays the discretized database.

Let us consider the mva-type association rule $X \overset{mva}{\Longrightarrow} Y$, where $X = \{(G2, \downarrow), (G3, \downarrow)\}$ and $Y = \{(G4, \uparrow)\}$. This is an implication relationship that states: "If gene 2 and gene 3 in a patient are under expressed, then it is likely that gene 4 is over expressed in the patient. Here, $\text{Supp}(X) = 7/8 = 0.875$ and $\text{Conf}(X \overset{mva}{\Longrightarrow} Y) = \text{Supp}(X \cup Y)/\text{Supp}(X) = 6/7 = 0.857$.

Our next example is a Personal Interest database.

Example 3.5 *Consider a Personal Interest database from a social network in Table 3.5, where each observation consists of a rating for attributes such as 'read', 'play', 'music', and 'eat' of different people (where 0 denotes the lowest interest and 10 denotes the highest*

interest). Here, person 1 *has a rating of* 10 *for read,* 10 *for play,* 3 *for music, and* 5 *for eat. Similarly, records for other people can be read from this table.*

Table 3.5 Personal interest database.

Observations	Attributes			
Person Id Id	Read R	Play P	Music M	Eat E
1	10	10	3	5
2	7	9	4	6
3	3	1	9	10
4	5	1	10	7
5	9	8	2	6
6	8	10	7	6
7	5	4	6	5
8	8	10	1	8

Table 3.6 Personal interest database (after discretization).

Observations	Attributes			
Person Id Id	Read R	Play P	Music M	Eat E
1	h	h	l	m
2	m	h	m	m
3	l	l	h	h
4	m	l	h	m
5	h	h	l	m
6	h	h	m	m
7	m	m	m	m
8	h	h	l	h

In the personal interest database in Table 3.5, for each attribute value a_i, we consider the discretized value l (low) if $0 \leq a_i \leq 3$, m (moderate) if $4 \leq a_i \leq 7$, and h (high) if $8 \leq a_i \leq 10$. Table 3.6 displays the discretized database.

Let us consider the mva-type association rule $X \overset{mva}{\Longrightarrow} Y$, where $X = \{(R, h), (P, h)\}$ and $Y = \{(M, l)\}$. This is an implication relationship that states: "If a person has high interest in reading and playing, then it is likely that the person has low interest in music. Here, $\text{Supp}(X) = 4/8 = 0.5$ and $\text{Conf}(X \overset{mva}{\Longrightarrow} Y) = \text{Supp}(X \cup Y)/\text{Supp}(X) = 3/4 = 0.75$.

Thus, with the above examples, we have seen examples of databases containing multi-valued attributes.

3.1.1 Discretization

We provide a methodology to discretize attribute values in financial time-series databases in Section 5.1.1. In these databases, every observation corresponds to a reading taken at a particular time and the order of observations is important. Our discretization methodology captures the relationship between any two consecutive observations in a financial time-series database. Thus, in the resulting database with discretized attribute values, the order of observations is irrelevant.

3.2 Association Hypergraphs

Definition 3.6 *An* association hypergraph $\mathcal{H}$ *for a database* $\mathcal{D}(\mathcal{A}, \mathcal{O}, \mathcal{V})$ *is a directed hypergraph whose vertex set* V *is* $\mathcal{A}$ *and hyperedge set* E *consists of directed hyperedges* (T, H)*, where* T *and* H *are disjoint subsets of* $\mathcal{A}$*. Each directed hyperedge* $e = (T, H)$ *has an* association confidence value *in the range* $[0, 1]$*, denoted* $ACV(e)$ *or* $ACV(T, H)$*, and an* association table *(as shown in Table 3.7), denoted* $AT(e)$ *or* $AT(T, H)$*, that are defined as follows:*

1. *The association confidence value of a directed hyperedge* $(\{t_1, \ldots, t_r\}, \{h_1, \ldots, h_s\})$ *equals*

$$\sum_{v_1,\ldots,v_r \in \mathcal{V}} \text{Supp}(\{(t_1, v_1), \ldots, (t_r, v_r)\}) \times \text{Conf}(\{(t_1, v_1), \ldots, (t_r, v_r)\} \stackrel{mva}{\Longrightarrow} H^*),$$

   where $H^* = \{(h_1, v_1^*), \ldots, (h_s, v_s^*)\}$ *and* v_1^**,* ...*,* v_s^* *depend on* v_1*,* ...*,* v_r *such that they maximize the confidence of the mva-type association rule*

$$\{(t_1, v_1), \ldots, (t_r, v_r)\} \stackrel{mva}{\Longrightarrow} \{(h_1, v_1'), \ldots, (h_s, v_s')\}$$

   over all choices of v_1'*,* ...*,* $v_s' \in \mathcal{V}$*. In other words, it equals* $\sum_{v_1,\ldots,v_r \in \mathcal{V}} \text{Supp}(\{(t_1, v_1),$ ..., $(t_r, v_r)\} \cup H^*)$*, where* H^* *is as defined above.*

2. *The association table of a directed hyperedge* $(\{t_1, t_2, \ldots, t_r\}, \{h_1, h_2, \ldots, h_s\})$ *has rows corresponding to the set of all possible values that* t_1*,* ...*,* t_r *can take from* $\mathcal{V}$*. The row corresponding to* $t_1 = v_1$*,* ...*,* $t_r = v_r$*, where* v_1*,* ...*,* $v_r \in \mathcal{V}$*, is a list that contains*

(a) $\text{Supp}(\{(t_1, v_1), \ldots, (t_r, v_r)\})$,

(b) *the values* $v_1^*, \ldots, v_s^* \in \mathcal{V}$ *defined in part (1) above, and*

(c) *the confidence of the mva-type association rule*

$$\{(t_1, v_1), \ldots, (t_r, v_r)\} \stackrel{mva}{\Longrightarrow} \{(h_1, v_1^*), \ldots, (h_s, v_s^*)\}.$$

In other words, every row corresponds to an mva-type association rule.

The motivation for representing a database $\mathcal{D}$ using a directed hypergraph is to capture a more general implication relationship between attributes than the one identified by mva-type association rules. For example, we now want to answer the following question: "Regardless of the values the attributes in set T take, what is the likeliness of predicting the values of the attributes in set H?". The above intuition helps us to model such a relationship between attributes in T and attributes in H using a directed hyperedge (T, H) and to capture the likeliness as the association confidence value $ACV(T, H)$ of this directed hyperedge.

In this thesis, we consider only associations of the form (T, H), where T and H are disjoint subsets of attributes, $|T| \leq 2$, and $|H| \leq 1$. Having no constraint on $|T|$ and $|H|$ adds complexity in the model because of the numerous possibilities involved. Extending our methodology to model associations in databases to the general case is a subject of future work. Henceforth, we use the term *association hypergraph* to refer to this restricted case, and call any directed hyperedge (T, H) in which $|T| = 1$ a directed edge and one in which $|T| = 2$ a 2-to-1 directed hyperedge.

3.2.1 Constructing Association Hypergraphs

The association hypergraph $\mathcal{H}$ for a database $\mathcal{D}(\mathcal{A}, \mathcal{O}, \mathcal{V})$ has node set $\mathcal{A}$ and the hyperedge set consists of directed hyperedges of the form (T, H), where T and H are disjoint subsets of $\mathcal{A}$. We construct directed hyperedges in the order of their head set. For a fixed combination of two or fewer attributes, say $\{A_1, A_2\}$, and any other attribute, say A_3, we determine whether $(\{A_1, A_2\}, \{A_3\})$ could be included as a directed hyperedge of $\mathcal{H}$ by checking whether the combination is γ-*significant* according to Definition 3.7.

Table 3.7 An example association table (AT) for the combination ($\{A_1, A_2\}$, $\{A_3\}$).

Index	Values of A_1 and A_2	Support Supp($\{(A_1, v_1), (A_2, v_2)\}$)	Most frequent value of A_3 (v_3^*)	Confidence Conf($\{(A_1, v_1), (A_2, v_2)\} \stackrel{mva}{\Longrightarrow} \{(A_3, v_3^*)\}$)
1	$\langle 1, 1\rangle$	0.14	2	0.43
2	$\langle 1, 2\rangle$	0.03	1	0.62
3	$\langle 1, 3\rangle$	0.06	1	0.75
4	$\langle 2, 1\rangle$	0.21	3	0.45
5	$\langle 2, 2\rangle$	0.11	2	0.38
6	$\langle 2, 3\rangle$	0.17	1	0.66
7	$\langle 3, 1\rangle$	0.01	2	0.49
8	$\langle 3, 2\rangle$	0.04	3	0.81
9	$\langle 3, 3\rangle$	0.23	2	0.73

Definition 3.7 *Consider a combination (T, H) for inclusion as a directed hyperedge of the association hypergraph $\mathcal{H}$, where $|T| \geq 1$. For $\gamma \geq 1$, we say that (T, H) is* γ-significant *if* $ACV(T, H) \geq \gamma \cdot \max_{v \in T}\{ACV(T - \{v\}, H)\}$.

If (T, H) is γ-significant, then we include this directed hyperedge in $\mathcal{H}$. Otherwise we skip this combination and proceed to the next one. The weight of a directed hyperedge (T, H) is set to $ACV(T, H)$.

The association table for the directed hyperedge $(\{A_1, A_2\}, \{A_3\})$ is constructed as follows. $\text{Supp}(\{(A_1, v_1), (A_2, v_2)\})$ is computed by counting observations in the database for which A_1's value is v_1 and A_2's value is v_2. $\text{Conf}(\{(A_1, v_1), (A_2, v_2)\} \stackrel{mva}{\Longrightarrow} \{(A_3, v_3^*)\})$ is then computed by counting such observations in the database for which A_3's value is v_3^*, where v_3^* is the most frequent value for A_3. This process is repeated for all possible values that A_1 and A_2 can take from $\mathcal{V}$.

Theorem 3.8 *Let A_1, A_2, and A_3 be attributes over common observations. Then the following holds:*

1. $ACV(\{A\}, \{X\}) \geq ACV(\emptyset, \{X\})$.
2. $ACV(\{A, B\}, \{X\}) \geq \max\{ACV(\{A\}, \{X\}), ACV(\{B\}, \{X\})\}$.

Proof We prove part (1) only as the proof of part (2) is similar. Assume that the symbolic representations of A_1 and A_3 are over the alphabet $\{1, 2, \ldots, k\}$. Since A_1 and A_3 have common observations, all the observations of A_1 and A_3 contribute towards building the association table AT of the combination $(\{A_1\}, \{A_3\})$. Suppose that there are d rows in AT. Let $\text{Maj}(d)$ denote the number of times the most frequent value of A_3 occurs in these rows. Then, the fraction of the most frequent value of A_3 is $\text{Maj}(d)/d$. W.l.o.g., assume that the most frequent value of A_3 is 1.

Out of d rows in AT, let d_i of them have value i for A_1. Thus, we have $\sum_{i=1}^{k} d_i = d$. Assume that, for each $1 \leq i \leq k$, the number of rows in AT such that A_1 takes value i and A_3 takes its most frequent value, 1, is a_i. Then, we have $\sum_{i=1}^{k} a_i = \text{Maj}(d)$. For every $1 \leq i \leq k$, restricting AT to all rows in which A_1 takes value i, let $\text{Maj}(d_i)$ denote the number of rows the most frequent value of A_3 belongs to in this restricted AT.

By definition, $ACV(\{A_1\},\{A_3\})$ is given by $\sum_{i=1}^{k} d_i/d \cdot \text{Maj}(d_i)/d_i$, which simplifies to $\sum_{i=1}^{k} \text{Maj}(d_i)/d$. Next, notice that for every $1 \leq i \leq k$, $\text{Maj}(d_i) \geq a_i$, since the most frequent value of A_3 out of all rows of AT that have $A_1 = i$ must occur at least a_i times. It follows that $\sum_{i=1}^{k} \text{Maj}(d_i) \geq \sum_{i=1}^{k} a_i = \text{Maj}(d)$. Hence, we get that

$$\begin{aligned} ACV(\{A_1\},\{A_3\}) &= \sum_{i=1}^{k} \text{Maj}(d_i)/d \\ &\geq \text{Maj}(d)/d \\ &= ACV(\emptyset,\{X\}). \end{aligned}$$

This completes the proof. ∎

3.3 Association-Based Similarity Between Multi-Valued Attributes

Undirected hypergraphs have been used to cluster binary attributes and subsets of binary attributes. Han et al. [HKKM98] used undirected hypergraphs where nodes represent binary attributes and hyperedges represent subsets of attributes. They applied an undirected hypergraph clustering algorithm to identify clusters of similar attributes. Ozdal et al. [OA04] also used undirected hypergraphs where nodes represent patterns (i.e., subsets of attributes) and hyperedges represent relationships among patterns, and proposed a clustering approach for these hypergraph models. Lent et al. [LSW97] used clustering to group association rules involving multi-valued attributes. This grouping was done based on certain attribute conditions that segment the observations in the database.

We propose similarity notions that measure association-based similarity between any two nodes of the *association hypergraph*. We can imagine two nodes A and B to be in-similar if significantly many directed hyperedges that contain A in their headset lead to valid directed hyperedges that contain B in their headset when A is replaced by B. Here, A and B are in-similar since they both share similar incoming directed hyperedges. Likewise, A and B can be regarded as out-similar by relating to the tailset instead of the headset of directed hyperedges. These similarity notions are then used to define clustering of multi-valued attributes.

Notation 3.9 *Let $\mathcal{H} = (V, E)$ be an association hypergraph as defined in Section 3.1.*

1. *For any $A \in V$, $\text{out}_{\mathcal{H}}(A)$ denotes the set of all directed hyperedges of $\mathcal{H}$ whose tail set contains A.*

2. *For any $A \in V$, $\text{in}_{\mathcal{H}}(A)$ denotes the set of all directed hyperedges of $\mathcal{H}$ whose head set contains A.*

3. *For any $A_1, A_2 \in V$ and $e = (T, H) \in \text{out}_{\mathcal{H}}(A_1)$, $e|_{\text{T:}A_1 \to A_2}$ denotes the directed hyperedge (T', H') whose head set H' is H and whose tail set T' is formed from T by replacing node A_1 by node A_2 (i.e., $H' = H$ and $T' = (T - \{A_1\}) \cup \{A_2\}$). For any set of directed hyperedges S and nodes A_1 and A_2, $S|_{\text{T:}A_1 \to A_2}$ denotes the set $\bigcup_{e \in S}\{e|_{\text{T:}A_1 \to A_2}\}$.*

4. *For any $A_1, A_2 \in V$ and $e = (T, H) \in \text{in}_{\mathcal{H}}(A_1)$, $e|_{\text{H:}A_1 \to A_2}$ denotes the directed hyperedge (T', H') whose tail set T' is T and whose head set H' is formed from H by replacing node A_1 by node A_2 (i.e., $T' = T$ and $H' = (H - \{A_1\}) \cup \{A_2\}$). For any set of directed hyperedges S and nodes A_1 and A_2, $S|_{\text{H:}A_1 \to A_2}$ denotes the set $\bigcup_{e \in S}\{e|_{\text{H:}A_1 \to A_2}\}$.*

Notation 3.10 *Let A_1 and A_2 be attributes and $\mathcal{H} = (V, E)$ be an association hypergraph as defined in Section 3.1. Let $\emptyset$ denote the empty directed hyperedge.*

1. *$\text{out}_{\mathcal{H}}(A_1) \otimes \text{out}_{\mathcal{H}}(A_2)$ is the set of directed hyperedge pairs (e, f) s.t. $e \in \text{out}_{\mathcal{H}}(A_1)$, $f \in \text{out}_{\mathcal{H}}(A_2)$, and $e = f|_{\text{T:}A_2 \to A_1}$. The definition of $\text{in}_{\mathcal{H}}(A_1) \otimes \text{in}_{\mathcal{H}}(A_2)$ is similar.*

2. *$\text{out}_{\mathcal{H}}(A_1) \oplus \text{out}_{\mathcal{H}}(A_2)$ is the union of the following sets of directed hyperedge pairs: (1) $\text{out}_{\mathcal{H}}(A_1) \otimes \text{out}_{\mathcal{H}}(A_2)$, (2) $(e, \emptyset)$ s.t. $e \in \text{out}_{\mathcal{H}}(A_1)$ and $e \neq f|_{\text{T:}A_2 \to A_1}$ for each $f \in \text{out}_{\mathcal{H}}(A_2)$, and (3) $(\emptyset, f)$ s.t. $f \in \text{out}_{\mathcal{H}}(A_2)$ and $e|_{\text{T:}A_1 \to A_2} \neq f$ for each $e \in \text{out}_{\mathcal{H}}(A_1)$. The definition of $\text{in}_{\mathcal{H}}(A_1) \oplus \text{in}_{\mathcal{H}}(A_2)$ is similar.*

3.3.1 In-Similarity and Out-Similarity

In Definition 3.11, we define the similarity notions *in-similarity* and *out-similarity* for any pair (A_1, A_2) of attributes. The in-similarity of attributes A_1 and A_2, denoted by

in-sim$_{\mathcal{H}}(A_1, A_2)$, is the weighted fraction of directed hyperedges $e \in \text{in}_{\mathcal{H}}(A_1) \cup \text{in}_{\mathcal{H}}(A_2)$ such that switching A_1 to A_2 in the head set of e results in another directed hyperedge; the weights are the association confidence values of directed hyperedges. Similarly, the out-similarity of attributes A_1 and A_2, denoted by out-sim$_{\mathcal{H}}(A_1, A_2)$, is the weighted fraction of directed hyperedges $e \in \text{out}_{\mathcal{H}}(A_1) \cup \text{out}_{\mathcal{H}}(A_2)$ such that switching A_1 to A_2 in the tail set of e results in another directed hyperedge.

Definition 3.11 *Let A_1 and A_2 be attributes and $\mathcal{H} = (V, E)$ be an association hypergraph as defined in Section 3.1. The following similarity notions are defined for A_1 and A_2:*

1. out-sim$_{\mathcal{H}}(A_1, A_2) = \frac{\sum_{(e,f) \in \text{out}_{\mathcal{H}}(A_1) \otimes \text{out}_{\mathcal{H}}(A_2)} \min\{ACV(e),\ ACV(f)\}}{\sum_{(e,f) \in \text{out}_{\mathcal{H}}(A_1) \oplus \text{out}_{\mathcal{H}}(A_2)} \max\{ACV(e),\ ACV(f)\}}$.
2. in-sim$_{\mathcal{H}}(A_1, A_2) = \frac{\sum_{(e,f) \in \text{in}_{\mathcal{H}}(A_1) \otimes \text{in}_{\mathcal{H}}(A_2)} \min\{ACV(e),\ ACV(f)\}}{\sum_{(e,f) \in \text{in}_{\mathcal{H}}(A_1) \oplus \text{in}_{\mathcal{H}}(A_2)} \max\{ACV(e),\ ACV(f)\}}$.

Example 3.12 *Suppose $\mathcal{H}$ has directed hyperedges $a = (\{A_1, A_3\}, \{A_6\})$, $b = (\{A_1, A_4\}, \{A_6\})$, $c = (\{A_2, A_3\}, \{A_6\})$, $d = (\{A_2, A_4, A_5\}, \{A_6\})$, and $e = (\{A_4, A_5\}, \{A_6\})$, where A_1, A_2, A_3, A_4, A_5, and A_6 are attributes. Let the ACVs of a, b, c, d, and e be* 0.4, 0.5, 0.6, 0.7, *and* 0.8, *respectively. Then, we have* $\text{out}_{\mathcal{H}}(A_1) \otimes \text{out}_{\mathcal{H}}(A_2) = \{(a, c)\}$, $\text{out}_{\mathcal{H}}(A_1) \oplus \text{out}_{\mathcal{H}}(A_2) = \{(a, c), (b, \emptyset), (\emptyset, d)\}$, *and so* weighted-out-sim$_{\mathcal{H}}(A_1, A_2) = \frac{0.4}{0.6+0.5+0.7} = 0.22$.

3.3.2 Clusters of Similar Attributes

We define below the notion of a similarity graph induced by any subset $\mathcal{S}$ of attributes that assigns every attribute pair $\{A_1, A_2\}$ in $\mathcal{S}$ an undirected edge whose weight depends on the in-similarity and the out-similarity of the pair.

Definition 3.13 *Let $\mathcal{H} = (V, E)$ be an association hypergraph as defined in Section 3.1. Given any collection $\mathcal{S}$ of attributes, a* similarity graph $SG_{\mathcal{S}} = (V', E')$ *induced by $\mathcal{S}$ in $\mathcal{H}$ is an undirected, weighted, complete graph whose node set V' is $\mathcal{S}$ and edge set E' contains all attribute pairs in $\mathcal{S}$ such that, for every edge $\{A_1, A_2\} \in E'$, its weight $d(A_1, A_2)$ is defined as* 1 − (weighted-in-sim$_{\mathcal{H}}(A1, A2)$ + weighted-out-sim$_{\mathcal{H}}(A1, A2)$) / 2.

Our objective is to determine a partition of $\mathcal{S}$ into subsets of attributes such that attributes within each subset are highly similar in their associative characteristics. The t-clustering

algorithm by Gonzalez [Gon85], presented in Algorithm 2 (Chapter 2), finds such a partition of $\mathcal{S}$ by designating some t attributes as cluster centers. The algorithm takes the parameter t in the input and assigns each attribute to its closest cluster center. This is a factor 2-approximation algorithm for minimizing the diameter of the t-clustering, assuming that the distances (i.e., weights) satisfy the *metric properties*. (Here, the diameter of a t-clustering is the maximum distance between any two data points that are within the same cluster.)

If our original association hypergraph $\mathcal{H}$ has n vertices and m directed hyperedges, then the construction of similarity graph $SG_{\mathcal{S}}$ takes time $O(m^2)$, and the computation of the t-clustering of vertices in $SG_{\mathcal{S}}$ takes additional time $O(|t| \cdot |\mathcal{S}|)$.

CHAPTER 4

COMPUTATIONAL PROBLEMS

An interesting question that arises in the context of association rules is about devising a methodology to build classification rules. A classification rule suffers from *overfitting* when the rule closely models a particular characteristic of the training data set and, due to this specificity, the rule fails to predict the value of the attribute in an unseen test data set. A classification rule suffers from *underfitting* when the rule models no particular characteristic of the training data set and, due to this generality, the rule fails to predict the value of the attribute both in the training data set and in an unseen test data set. Liu et al. [LHM98] addressed the above problems by mining association rules that have only one attribute in their consequent, proposing a pruning based algorithm to generate classification rules, and testing on an unseen data set. Bayardo [Bay97] proposed various methods that again use pruning to reduce the number of association rules discovered by standard association rule mining algorithms. But the quality of the rules is unclear as they have not been tested on an unseen data set. Liu et al. [LHM99] addressed the above problems by introducing multiple minimum supports during mining of association rules.

In this section, we use the directed hypergraph based model to first propose algorithms to identify a small subset of attributes that influence the characteristics of almost all other attributes and then present the association-based classifier that can be used to predict the values of attributes.

4.1 Leading Indicators

A leading indicator $\mathcal{X}$ for any set $\mathcal{S}$ of attributes is a subset of $\mathcal{S}$ such that knowing only the values for the attributes in $\mathcal{X}$ allows us to infer the value for all attributes in $\mathcal{S} - \mathcal{X}$. Motivated by the notion of a dominating set for any graph that essentially captures the

property of a subset of nodes covering all the nodes of the graph by at most one edge, we define below the notion of a dominator for a set of vertices of any association hypergraph. Our hypothesis is that a dominator for nodes corresponding to the set $\mathcal{S}$ of attributes in the association hypergraph modeling attribute relationships gives a leading indicator for $\mathcal{S}$.

Definition 4.1 *A dominator for a set $\mathcal{S}$ of vertices in an association hypergraph $\mathcal{H} = (V, E)$ is a set $\mathcal{X} \subseteq V$ such that, for every $u \in \mathcal{S} - \mathcal{X}$, there is a directed hyperedge $e = (T, H) \in E$ such that $T \subseteq \mathcal{X}$ and $u \in H$. That is, each node $u \in \mathcal{S} - \mathcal{X}$ is covered using only directed hyperedges whose tail set is from the set $\mathcal{X}$.*

We next provide two greedy algorithms for identifying a subset of attributes known as a leading indicator that influences the values of almost all other attributes. The first greedy algorithm is based on an adaptation of the greedy $O(\log n)$-approximation algorithm for computing a minimum cardinality dominating set in graphs. The second greedy algorithm is based on an adaption of the greedy $O(\log n)$-approximation algorithm for computing a minimum cost set cover.

4.1.1 An Adaptation of Graph Dominating Set Approximation Algorithm

Algorithm 5 is a greedy algorithm for computing a dominator for any set $\mathcal{S}$ of vertices in an association hypergraph $\mathcal{H} = (V, E)$. It is an adaptation of the greedy $O(\log n)$-approximation algorithm for computing a minimum cardinality dominating set in graphs. In order to minimize the size of the dominator, the following greedy heuristic is applied: for every node u that is not part of the dominating set yet, the algorithm computes the node effectiveness $\alpha(u)$ that reflects u's covering ability. During each iteration of the algorithm (until all the nodes in the graph are covered), the node with the highest effectiveness value is added to the dominator set. The algorithm runs in time $O(|\mathcal{S}| \cdot |E|)$.

4.1.2 An Adaptation of Set Cover Approximation Algorithm

Algorithm 6 computes a dominator for any set $\mathcal{S}$ of vertices of any association hypergraph $\mathcal{H} = (V, E)$. It is an adaptation of the greedy $O(\log n)$-approximation algorithm for Set Cover. The algorithm maintains a variable *DomSet* to store the (partially constructed)

dominator set of $\mathcal{H}$ and a variable *CoveredSet* to store the set of vertices covered by those in *DomSet*. This is an iterative algorithm such that in each iteration some subset of vertices t^* is greedily chosen and made part of *DomSet*. The algorithm maintains a variable T^* that stores subsets of vertices for possible inclusion in *DomSet*. Initially, T^* is the collection of all tailsets of directed hyperedges in $\mathcal{H}$, i.e., $T^* = \{T(e) \mid e \in E\}$.

Input : A set $\mathcal{S}$ of vertices and an association hypergraph $\mathcal{H} = (V, E)$.
Output: A dominator *DomSet* for the set $\mathcal{S}$ of vertices.

1 **begin**
2 DomSet $\leftarrow \emptyset$;
3 CoveredSet $\leftarrow \emptyset$;
4 **while** CoveredSet $\neq \mathcal{S}$ **do**
5 **foreach** vertex $u \in V -$ DomSet **do**
6 **if** $u \notin$ CoveredSet and $u \in \mathcal{S}$ **then**
7 $\alpha(u) \leftarrow 1$;
8 **end**
9 **else**
10 $\alpha(u) \leftarrow 0$;
11 **end**
12 $\alpha(u) \leftarrow \alpha(u) + \sum_{v \notin \text{CoveredSet} \wedge v \in S} L(u, v)$,
13 where $L(u, v) \leftarrow \max_{e: u \in T(e) \wedge v \in H(e)} \frac{w(e)}{|T(e) - \text{DomSet}|}$
14 **end**
15 Let $u_0 \in V$ be such that $\alpha(u_0) = \max_{u \notin \text{DomSet}} \alpha(u)$;
16 DomSet $\leftarrow$ DomSet $\cup \{u_0\}$;
17 CoveredSet $\leftarrow$ CoveredSet $\cup \{u_0\} \cup \{v \in \mathcal{S} \mid \exists e \in E$ s.t. $v \in H(e)$
18 and $T(e) \subseteq$ DomSet$\}$;
19 **end**
20 return *DomSet*;
21 **end**

Algorithm 5: A greedy algorithm for computing dominators in directed hypergraphs which is an adaptation of graph dominating set approximation.

The algorithm iterates until all the vertices in the set $\mathcal{S}$ are covered. For every subset $t^* \in T^*$, a measure $\alpha(t^*)$ is defined that reflects t^*'s covering ability in the following sense: $\alpha(t^*)$ contains all new vertices that can be covered by including t^* in *DomSet*. In each iteration in Line 5, $\alpha(t^*)$ is computed for every $t^* \in T^*$ as follows. First, $\alpha(t^*)$ is assigned to the number of vertices that are in t^* as well as $\mathcal{S}$, but not yet in *CoveredSet* (Lines 6–12). Next, $\alpha(t^*)$ adds up the number of vertices in $\mathcal{S} - \mathit{CoveredSet}$ that t^* covers via some directed hyperedge e whose tailset $T(e)$ is part of t^* (Lines 13–17). Specifically, for the latter

contribution in $\alpha(t^*)$, every directed hyperedge e is traversed and if both $T(e) \subseteq t^*$ and $H(e) \in \mathcal{S} - CoveredSet$, then $H(e)$ is counted in the computation of $\alpha(t^*)$. Clearly, all sets $t^* \in T^*$ whose effectiveness $\alpha(t^*)$ is zero are insignificant; Line 15 discards such sets from T^* so that they are not considered in later iterations. Towards the end of every iteration, the set t^* of maximum effectiveness is included in $DomSet$ and the variable $CoveredSet$ is updated to account for the changed $DomSet$ (Lines 20–22).

Since each iteration takes $O(|E|^2)$ time and there are $|\mathcal{S}|$ iterations, the computation time of the algorithm is $O(|\mathcal{S}| \cdot |E|^2)$.

We now provide two enhancements with the goal of improving the computation time of Algorithm 6 and also of reducing the size of the dominator set output by this algorithm. Enhancement 1, presented in Algorithm 7, should be added between Lines 20–21, and Enhancement 2, presented in Algorithm 8, should be added between Lines 22–24 of Algorithm 6.

Enhancement 1. During each iteration of the while loop in Line 5, Algorithm 6 includes in $DomSet$ a subset t_0^* whose effectiveness $\alpha(t_0^*)$ is the highest among all $t^* \in T^*$. This enhancement considers the scenario when there are more than one possible candidate t_0^* in some iteration. In such a case, the candidate subset t_0^* that contributes the least number of elements to $DomSet$ (i.e., that minimizes $|t_0^* - DomSet|$) might be better than any other candidate $t_0^{*'}$.

Enhancement 2. We know that Algorithm 6 updates the coverage of $DomSet$ in every iteration by going over each directed hyperedge whose tailset is a part of $DomSet$ and adding vertices in the headset that are in $\mathcal{S}$ to $CoveredSet$. By removing a subset t^* that is already a part of the $DomSet$ from T^*, the enhancement saves the computation time required to go over all the directed hyperedges in the subsequent iterations of the algorithm to compute $\alpha(t^*)$.

4.2 Association-Based Classifier

Let $\mathcal{S}$ be the set of attributes $A_1, A_2, \ldots, A_t$ and let these attributes take values $v_1, v_2, \ldots, v_t$, respectively. Let $\mathcal{T}$ be another set of attributes, disjoint from $\mathcal{S}$. The association-based

Input : A set $\mathcal{S}$ of vertices and an association hypergraph $\mathcal{H} = (V, E)$.
Output: A dominator $DomSet$ for the set $\mathcal{S}$ of vertices.

begin
  $DomSet \leftarrow \emptyset$;
  $CoveredSet \leftarrow \emptyset$;
  $T^* \leftarrow \{T(e) \mid e \in E\}$;
  while $CoveredSet \neq \mathcal{S}$ **do**
    foreach *set* $t^* \in T^*$ **do**
      $\alpha(t^*) \leftarrow 0$;
      foreach *vertex* $u \in t^*$ **do**
        if $u \notin CoveredSet$ and $u \in \mathcal{S}$ **then**
          $\alpha(t^*) \leftarrow \alpha(t^*) + 1$;
        end
      end
      foreach directed hyperedge e such that $T(e) \subseteq t^*$ **do**
        if $H(e) \notin CoveredSet$ and $H(e) \in \mathcal{S}$ **then**
          $\alpha(t^*) \leftarrow \alpha(t^*) + 1$;
        end
      end
      if $\alpha(t^*) == 0$ **then** $T^* \leftarrow T^* - \{t^*\}$;
    end
    Let t_0^* be such that $\alpha(t_0^*) \leftarrow \max_{t^* \in T^*}(\alpha(t^*))$;
    $DomSet \leftarrow DomSet \cup t_0^*$;
    $CoveredSet \leftarrow CoveredSet \cup \{t_0^*\} \cup \{v \in \mathcal{S} \mid \exists e \in E \text{ s.t. } v \in H(e) \text{ and } T(e) \subseteq DomSet\}$;
  end
  return $DomSet$;
end

Algorithm 6: A greedy algorithm for computing dominators in directed hypergraphs which is an adaptation of set cover approximation.

foreach *set* $t^* \in T^*$ **do**
  if $\alpha(t_0^*) == \alpha(t^*)$ **then**
    if $|t_0^* - DomSet| > |t^* - Domset|$ **then**
      $t_0^* \leftarrow t^*$;
    end
  end
end

Algorithm 7: Enhancement 1.

foreach *set* $t^* \in T^*$ **do**
  if $t^* \subseteq DomSet$ **then**
    $T^* \leftarrow T^* - t^*$;
  end
end

Algorithm 8: Enhancement 2.

classifier determines the values of all attributes in $\mathcal{T}$ given the values of attributes in $\mathcal{S}$. For this problem, we will assume that $\mathcal{S}$ is a dominator for $\mathcal{T}$ in the association hypergraph $\mathcal{H}$ and so $\mathcal{S}$ can be computed using the algorithm presented in Section 4.1. This assumption stands on our earlier stated hypothesis that a dominator for any set of attributes is also a leading indicator for the set.

Input : An association hypergraph $\mathcal{H} = (V, E)$ modeling attribute relationships, a set $\mathcal{T}$ of attributes, and a set $\mathcal{S} = \{(A_1, v_1), (A_2, v_2), \ldots, (A_t, v_t)\}$, where $A_1, A_2, \ldots, A_t$ are attributes and $v_1, v_2, \ldots, v_t \in \mathcal{V}$ are their respective values.

Output: An assignment of values that assigns each attribute $Y \in \mathcal{T}$ its best classified value y^* and the classification confidence val$[y^*]$ associated with every such assignment y^* to Y.

begin
  foreach attribute $Y \in \mathcal{T}$ **do**
    for $y \leftarrow 1$ *to* k **do**
      val$[y] \leftarrow 0$;
    end
    foreach directed hyperedge $e = (T, H) \in E$ with $H = \{Y\}$ and $T \subseteq \{A_1, A_2, \ldots, A_t\}$ **do**
      Let T be $\{A_1, A_2\}$ and let y be the most frequent value of Y given "$A_1 = v_1$" and "$A_2 = v_2$";
      val$[y] \leftarrow$ val$[y] +$ Supp$(\{(A_1, v_1), (A_2, v_2)\}) \times$ Conf$(\{(A_1, v_1), (A_2, v_2)\} \overset{mva}{\Longrightarrow} \{(Y, y)\})$;
    end
    Let $y^* \in \{1, \ldots, k\}$ be such that val$[y^*] = \max_{y \in \{1,\ldots,k\}}$ val$[y]$;
    val$[y^*] \leftarrow$ val$[y^*] / \sum_{y \in \{1,\ldots,k\}}$ val$[y]$;
    Output "$(Y, y^*,$ val$[y^*])$";
  end
end

Algorithm 9: Association-based classifier.

To compute the value for any attribute $Y \in \mathcal{T}$ given only the values $v_1, v_2, \ldots, v_t$ of a set $\mathcal{S}$ of the attributes, we iterate over all directed hyperedges of $\mathcal{H}$. For each directed hyperedge e whose tailset is a subset of the attributes in $\mathcal{S}$ and whose headset is $\{Y\}$, we examine the AT associated with e. Specifically, assume that $e = (\{A_1, A_2\}, \{Y\})$. Using the AT of e, we find Supp$(\{(A_1, v_1), (A_2, v_2)\})$ and Conf$(\{(A_1, v_1), (A_2, v_2)\} \overset{mva}{\Longrightarrow} \{(Y, y)\})$, where $y \in \mathcal{V}$ is the most frequent value of Y given that A_1 takes value v_1 and A_2 takes value v_2. The contribution of the directed hyperedge e in the value assignment y of Y is Supp$(\{(A_1, v_1), (A_2, v_2)\}) \times$ Conf$(\{(A_1, v_1), (A_2, v_2)\} \overset{mva}{\Longrightarrow} \{(Y, y)\})$. The total contribution

of all directed hyperedges in the value assignment y of Y is denoted by val$[y]$. At the end of all iterations, we choose the value y^* of Y for which val$[y^*]$ is maximum. The *classification confidence* associated with the value assignment y^* to Y is then the normalized val$[y^*] \in [0, 1]$.

We see that the contributions from all mva-type association rules of both relevant directed edges and relevant directed hyperedges are taken into consideration during the process of determination of the value of an attribute. This approach avoids overfitting by not fixing the value assignment for an attribute by looking only into a particular mva-type association rule that closely models some characteristic of the data set. Also, this approach avoids underfitting as it considers the appropriate weights for all mva-type association rules before fixing the value assignment for an attribute.

Algorithm 9 describes this solution approach. The input is an association hypergraph $\mathcal{H} = (V, E)$ modeling attribute relationships, a set $\mathcal{T}$ of attributes, and a set $\mathcal{S} = \{(A_1, v_1), (A_2, v_2), \ldots, (A_t, v_t)\}$, where $A_1, A_2, \ldots, A_t$ are attributes and $v_1, v_2, \ldots, v_t \in \mathcal{V}$ are their respective values. The algorithm returns an assignment of values that assigns each attribute $Y \in \mathcal{T}$ its best classified value y^* and also returns the classification confidence val$[y^*]$ associated with every such assignment y^* to Y. Its running time is $O(k^2 \cdot |\mathcal{T}| \cdot |E|)$.

CHAPTER 5

EXPERIMENTATION

The Standard & Poor's (S&P) 500 is a market-value-weighted index of the stock prices of some 500 large publicly held companies. We used Yahoo Finance [Yah10] to obtain stock information for all financial time-series in S&P 500. In this section, all analysis is based on the daily closing stock price information from Jan 1, 1995 to Dec 21, 2009. We had to restrict start date to Jan 1, 1995 since a number of financial time-series in the current S&P 500 started trading only from the mid 90s. As a result of this clean up of our data, the number of financial time-series in our analysis is 346. These time-series belong to the following industrial sectors: Basic Materials ($\mathcal{BM}$), Capital Goods ($\mathcal{CG}$), Conglomerates ($\mathcal{C}$), Consumer Cyclical ($\mathcal{CC}$), Consumer Noncyclical ($\mathcal{CN}$), Energy ($\mathcal{E}$), Financial ($\mathcal{F}$), Healthcare ($\mathcal{H}$), Services ($\mathcal{SV}$), Technology ($\mathcal{T}$), Transportation ($\mathcal{TP}$), and Utilities ($\mathcal{U}$). Each industrial sector is subdivided into sub-sectors (e.g., Technology ($\mathcal{T}$) has 11 sub-sectors including Communications Equipment, Computer Hardware, Computer Networks, Computer Peripherals, Computer Services, Computer Storage Devices, Electronic Instr. and Controls, Office Equipment, Scientific and Technical Instr., Semiconductors, and Software and Programming.) The total number of sub-sectors over the entire sectors is 104.

5.1 Association Hypergraph Modeling

5.1.1 Discretization

We now describe how to transform financial time-series data set into a database $\mathcal{D}(\mathcal{A}, \mathcal{O}, \mathcal{V})$ suitable for the association hypergraph modeling. For each financial time-series in the data set, we create a delta time-series, which is a list of real numbers whose i'th entry is the fractional change in the closing stock price of the $(i+1)$'th day relative to the closing stock price of the i'th day. We then compute a k-threshold vector, for some integer $k \geq 2$, for each

financial time-series. A k-threshold vector for a financial time-series A is a $(k-1)$-tuple $\langle a_1, a_2, \ldots, a_{k-1} \rangle$ such that, for every $1 \leq i \leq k$, we have $a_{i-1} < a_i$ and the number of entries of the delta time-series for A that lie in the range $[a_{i-1}, a_i)$ is roughly $1/k$. The k-threshold vector is computed as follows: Let the number of entries in the delta time-series be N. Sort the delta time-series in non-decreasing order and then, for each $1 \leq i \leq k-1$, assign a_i to the $\lfloor (i/k) * N \rfloor$'th entry in the sorted list. The k-threshold vectors are used to perform an equi-depth partitioning of the original data set and discretize it over the value set $\mathcal{V} = \{1, 2, \ldots, k\}$.

Our discretization procedure makes a single pass over each delta time-series and compares its entries with the components of the threshold vector. If an entry lies in the range $[a_{i-1}, a_i)$ for some $1 \leq i \leq k$, then the entry is assigned to the value i. In this way, the original financial time-series data set is transformed into the database $\mathcal{D}$. Here, each financial time-series is an attribute. The set of values $\mathcal{V}$ is $\{1, \ldots, k\}$, and each observation is a list of values assigned to the financial time-series on a particular day. Finally we construct an association hypergraph $\mathcal{H}$ from the database $\mathcal{D}$ using the methodology described in Section 3.2.

5.1.2 Choice of Parameters

For the construction of the association hypergraph $\mathcal{H}$, we try two configurations of parameters as follows: configuration $C1$ sets k to 3, γ (in Definition 3.7) for directed edges (say $\gamma_{1\to1}$) to 1.15, and for 2-to-1 directed hyperedges (say $\gamma_{2\to1}$) to 1.05 and configuration $C2$ sets k to 5, $\gamma_{1\to1}$ for directed edges to 1.20, and $\gamma_{2\to1}$ for directed hyperedges to 1.12. The choice for these values in both $C1$ and $C2$ are based on the following reasoning: (a) these values of $\gamma_{2\to1}$ lead to higher ACV values of 2-to-1 directed hyperedges when compared to the constituent directed edges and a similar remark can be made for the values of $\gamma_{1\to1}$, (b) these are the stable values of $\gamma_{1\to1}$ and $\gamma_{2\to1}$; that is, slight perturbations to these values do not result in significant changes to the numbers of directed edges and 2-to-1 directed hyperedges in $\mathcal{H}$, and (c) the values in $C1$ and the values in $C2$ result in comparable numbers of directed edges and 2-to-1 directed hyperedges in $\mathcal{H}$. The configuration $C1$ leads to 106,475 directed edges with a mean ACV of 0.436 and 157,412 2-to-1 directed hyperedges

with a mean ACV of 0.437 and the configuration $C2$ leads to 109,810 directed edges with a mean ACV of 0.288 and 274,048 2-to-1 directed hyperedges with a mean ACV of 0.288.

5.2 Association Characteristics of Financial Time-Series

Figure 5.1(a) shows the weighted in-degree distribution of the nodes of the association hypergraph for configuration $C1$. Here the weighted in-degree of a node v is $\sum_{e:\{v\}=H(e)} w(e)$, i.e., the sum of weights of all hyperedges entering v. Figure 5.1(b) shows the weighted out-degree distribution of the nodes of the association hypergraph for configuration $C1$. Here the weighted out-degree of a node v is $\sum_{e:v\in T(e)} \frac{w(e)}{|T(e)|}$, i.e., the sum of normalized weights of all hyperedges leaving v. The mean ACV of directed edges is 0.43 and of 2-to-1 directed hyperedges is 0.44.

Using our directed hypergraph based modeling, we are able to deduce some interesting facts about the current stock market that would have been difficult to validate by other means. Our findings concern relationship between producers and consumers in the stock market.

Roughly speaking, a producer is any entity (company) that has very few dependency on others for its resource requirements. A producer thrives mostly on its own or has little resource requirements compared to the rest. Some notable sectors that are likely to have entities in producer category are $\mathcal{BM}$, $\mathcal{CG}$, $\mathcal{E}$, and $\mathcal{SV}$ sectors. On the other hand, a consumer is an entity that is highly dependent on other entities or end-users for its own functioning. Sectors such as $\mathcal{CC}$, $\mathcal{CN}$, $\mathcal{H}$, $\mathcal{SV}$, and $\mathcal{T}$ are likely to have entities in consumer category. A particular exception is the $\mathcal{SV}$ sector in which there are both producers and consumers depending on the particular industry they belong to. For example, entities in $\mathcal{SV}$ sector that deal with real estate operations (e.g., Kimco Realty Corporation) are mostly producers whereas entities in $\mathcal{SV}$ sector that provide basic services to the end user (e.g., Yahoo! Inc.) mostly satisfy the consumer category.

Time-series that are less dependent on other time-series for their resources (e.g., Producers) are likely to be more predictable in comparison to others. Why? Since these time-series do not depend on other time-series for raw materials and other basic requirements, their

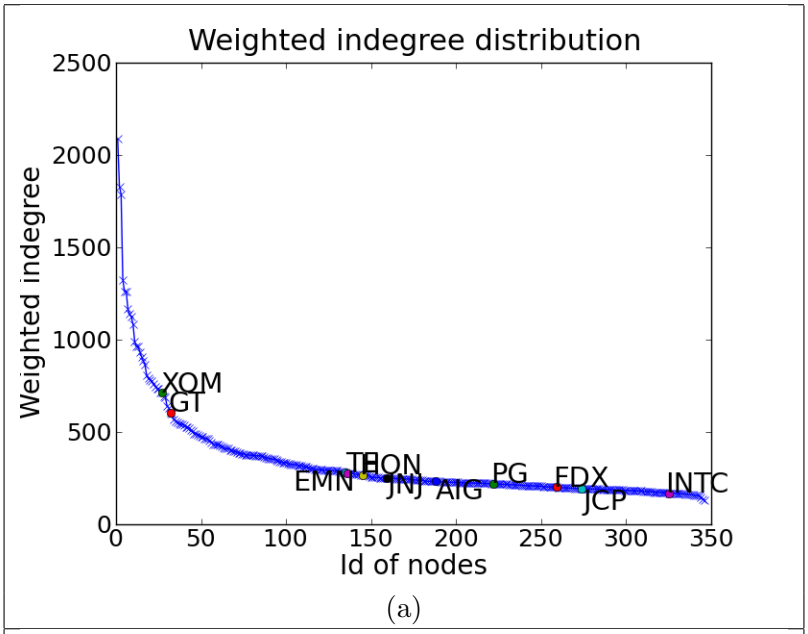

Figure 5.1 Weighted degree distribution. (a) Weighted in-degree distribution and (b) Weighted out degree distribution. The ids of nodes are on the x-axis and their weighted in-degree (out-degree) values are on the y-axis.

value in the market can be estimated by analyzing the demand of the sources of raw materials on which they rely on. The weighted in-degree of a node in our directed hypergraph model indicates the level of predictability of the corresponding time-series. The greater the weighted in-degree of a node is, the more predictable the corresponding time-series is. We see from Figure 5.1(a) that XOM ($\mathcal{E}$ sector) and GT ($\mathcal{CC}$ sector) have high weighted in-degree values in comparison to others from our chosen list. In the case of GT (The Goodyear Tire & Rubber company), although the sector it belongs to is different from the ones we listed for producer category, one of the basic materials this time-series depends on is rubber, which is a naturally occurring compound. We also experimentally found out that of the time-series with the top 25 weighted in-degree values, 72% of them belong to sectors $\mathcal{BM}$ (iron, gold, silver, and metal mining industries), $\mathcal{E}$ (oil, gas, and coal industries), and $\mathcal{SV}$ (real estate industries).

Time-series that have direct interactions with end-users for their products (e.g., Consumers) are more likely to be good predictors. Why? These time-series happen to be good predictors as they get affected by their immediate actions. The economy is driven by the end-user's requirements. By being in direct contact with the end-user, the performance of the consumer time-series in the market follows direct patterns of the behavior of the end-user.

The weighted out-degree of a node in our modeling indicates how much that node (time-series) can predict other time-series. The greater the weighted out-degree of a node is, the higher its ability to predict other time-series is. We see from Figure 5.1(b) that PG ($\mathcal{CN}$ sector) and JNJ ($\mathcal{H}$ sector) have high out-degree values in comparison to others from our chosen list. We also experimentally found out that of the time-series with the top 25 weighted out-degree values, 84% of the time-series belong to sectors $\mathcal{H}$ (facilities, biotechnology, and drugs), $\mathcal{SV}$ (particularly, business, real estate, and restaurant services), and $\mathcal{T}$ (software, hardware, and semiconductors).

Table 5.1 shows, for each configuration, the directed edge and the 2-to-1 directed hyperedge with the highest ACV for each selected financial time-series from different sectors. For example, the best prediction directed edge for GT (The Goodyear Tire & Rubber Company) shown in Row 3 for $C1$ and $C2$ represents a relationship between GT and PPG (PPG

Table 5.1 The directed edge and the 2-to-1 directed hyperedge with the highest ACV for each selected financial time-series from different sectors and for each configuration choice are shown. The sector information of each financial time-series is indicated in parenthesis (google finance).

Row	**Time-series**	**Configuration**	**Top directed edge**	**Top 2-to-1 directed hyperedge**
1	EMN ($\mathcal{BM}$)	$C1$	PPG ($\mathcal{BM}$) $\rightarrow$ EMN ($\mathcal{BM}$)	AVY ($\mathcal{BM}$), GT ($\mathcal{CC}$) $\rightarrow$ EMN ($\mathcal{BM}$)
		$C2$	PPG ($\mathcal{BM}$) $\rightarrow$ EMN ($\mathcal{BM}$)	BLL ($\mathcal{BM}$), IFF ($\mathcal{BM}$) $\rightarrow$ EMN ($\mathcal{BM}$)
2	HON ($\mathcal{CG}$)	$C1$	TXT ($\mathcal{C}$) $\rightarrow$ HON ($\mathcal{CG}$)	CAT ($\mathcal{CG}$), ITT ($\mathcal{T}$) $\rightarrow$ HON ($\mathcal{CG}$)
		$C2$	UTX ($\mathcal{CG}$) $\rightarrow$ HON ($\mathcal{CG}$)	BA ($\mathcal{CG}$), ROK ($\mathcal{T}$) $\rightarrow$ HON ($\mathcal{CG}$)
3	GT ($\mathcal{CC}$)	$C1$	PPG ($\mathcal{BM}$) $\rightarrow$ GT ($\mathcal{CC}$)	DOW ($\mathcal{BM}$), F ($\mathcal{CC}$) $\rightarrow$ GT ($\mathcal{CC}$)
		$C2$	PPG ($\mathcal{BM}$) $\rightarrow$ GT ($\mathcal{CC}$)	ETN ($\mathcal{T}$), FMC ($\mathcal{BM}$) $\rightarrow$ GT ($\mathcal{CC}$)
4	PG ($\mathcal{CN}$)	$C1$	CL($\mathcal{CN}$) $\rightarrow$ PG ($\mathcal{CN}$)	CLX($\mathcal{CN}$), K ($\mathcal{CN}$) $\rightarrow$ PG ($\mathcal{CN}$)
		$C2$	CL($\mathcal{CN}$) $\rightarrow$ PG ($\mathcal{CN}$)	ABT($\mathcal{H}$), CPB ($\mathcal{CN}$) $\rightarrow$ PG ($\mathcal{CN}$)
5	XOM ($\mathcal{E}$)	$C1$	CVX ($\mathcal{E}$) $\rightarrow$ XOM ($\mathcal{E}$)	HES ($\mathcal{E}$), SLB ($\mathcal{E}$) $\rightarrow$ XOM ($\mathcal{E}$)
		$C2$	CVX ($\mathcal{E}$) $\rightarrow$ XOM ($\mathcal{E}$)	COG ($\mathcal{E}$), PEG ($\mathcal{U}$) $\rightarrow$ XOM ($\mathcal{E}$)
6	AIG ($\mathcal{F}$)	$C1$	C ($\mathcal{F}$) $\rightarrow$ AIG ($\mathcal{F}$)	BEN($\mathcal{F}$), PGR ($\mathcal{F}$) $\rightarrow$ AIG ($\mathcal{F}$)
		$C2$	C ($\mathcal{F}$) $\rightarrow$ AIG ($\mathcal{F}$)	AON ($\mathcal{F}$), CI ($\mathcal{F}$) $\rightarrow$ AIG ($\mathcal{F}$)
7	JNJ ($\mathcal{H}$)	$C1$	MRK ($\mathcal{H}$) $\rightarrow$ JNJ ($\mathcal{H}$)	IFF ($\mathcal{BM}$), SYY ($\mathcal{SV}$) $\rightarrow$ JNJ ($\mathcal{H}$)
		$C2$	MRK ($\mathcal{H}$) $\rightarrow$ JNJ ($\mathcal{H}$)	CL ($\mathcal{CN}$), PEP ($\mathcal{CN}$) $\rightarrow$ JNJ ($\mathcal{H}$)
8	JCP ($\mathcal{SV}$)	$C1$	M ($\mathcal{SV}$) $\rightarrow$ JCP ($\mathcal{SV}$)	FDO ($\mathcal{SV}$), GPS ($\mathcal{SV}$) $\rightarrow$ JCP ($\mathcal{SV}$)
		$C2$	M ($\mathcal{SV}$) $\rightarrow$ JCP ($\mathcal{SV}$)	COST ($\mathcal{SV}$), HD ($\mathcal{SV}$) $\rightarrow$ JCP ($\mathcal{SV}$)
9	INTC $\mathcal{T}$)	$C1$	LLTC ($\mathcal{T}$) $\rightarrow$ INTC ($\mathcal{T}$)	EMC ($\mathcal{T}$), QCOM ($\mathcal{T}$) $\rightarrow$ INTC ($\mathcal{T}$)
		$C2$	XLNX ($\mathcal{T}$) $\rightarrow$ INTC ($\mathcal{T}$)	CTXS ($\mathcal{T}$), QCOM ($\mathcal{T}$) $\rightarrow$ INTC ($\mathcal{T}$)
10	FDX ($\mathcal{TP}$)	$C1$	AXP ($\mathcal{F}$) $\rightarrow$ FDX ($\mathcal{TP}$)	EXPD ($\mathcal{TP}$), ITT ($\mathcal{T}$) $\rightarrow$ FDX ($\mathcal{TP}$)
		$C2$	AXP ($\mathcal{F}$) $\rightarrow$ FDX ($\mathcal{TP}$)	EXPD ($\mathcal{TP}$), BAC ($\mathcal{F}$) $\rightarrow$ FDX ($\mathcal{TP}$)
11	TE ($\mathcal{U}$)	$C1$	PGN ($\mathcal{U}$) $\rightarrow$ TE ($\mathcal{U}$)	PEG ($\mathcal{U}$), SO ($\mathcal{U}$) $\rightarrow$ TE ($\mathcal{U}$)
		$C2$	AEP ($\mathcal{U}$) $\rightarrow$ TE ($\mathcal{U}$)	SO ($\mathcal{U}$), TEG ($\mathcal{U}$) $\rightarrow$ TE ($\mathcal{U}$)

Industries, Inc.). This relationship may be interpreted in terms of GT procuring raw materials (e.g., precipitated silicas) from PPG for the manufacturing or processing of rubber. A more interesting many-to-one relationship is represented by the best prediction 2-to-1 directed hyperedge for GT shown in Row 3 for $C1$. Here, the 2-to-1 directed hyperedge for GT represents a relationship of GT with DOW (The Dow Chemical Company) and F (Ford Motor Company). This relationship may be interpreted in terms of GT procuring raw materials (e.g., polyurethane polymer) from DOW, whereas the relationship with F may be attributed towards F utilizing the products (e.g., tires) from GT. Thus, we see that 2-to-1 directed hyperedges provide meaningful and more interesting information than directed edges do.

Table 5.2 shows, for each configuration, the 2-to-1 directed hyperedge with the highest ACV and the constituent directed edges for each selected financial time-series from different sectors. For example, in Row 5, for $C1$ the accuracy of HES (Hess Corporation) predicting XOM (Exxon Mobil Corporation) is 0.55 and SLB (Schlumberger Limited) predicting XOM is 0.54, but both of them together predicting XOM is 0.58. This shows that the combination of two financial time-series leads to a better predicting 2-to-1 directed hyperedge.

5.3 Association-Based Similarity

5.3.1 Comparison with Euclidean Similarity

Figure 5.2 compares in-similarity and out-similarity values with Euclidean similarity values for configuration $C1$. Here, the Euclidean similarity between any two financial time-series A and B is computed as follows. Let $\Delta(A) = (a_1, a_2, \ldots, a_n)$ and $\Delta(B) = (b_1, b_2, \ldots, b_n)$, where a_i and b_i are the values that A and B take in the i'th observation. The Euclidean distance between A and B is defined as $\text{ED}(A, B) = ||\text{normalized}(\Delta(A)) - \text{normalized}(\Delta(B))||$, where for any vector $V = (v_1, v_2, \ldots, v_n)$, $\text{normalized}(V) = (v_1/||V||, v_2/||V||, \ldots, v_n/||V||)$ and $||V|| = (\sum_{i=1}^{n} v_i^2)^{\frac{1}{2}}$. Now, the Euclidean similarity $\text{ES}(A, B)$ between A and B is defined as $\text{ES}(A, B) = 1 - \frac{1}{2} \times \text{ED}(A, B)$. Note that $\text{ES}(A, B)$ is a real value in the range $[0, 1]$ such that a higher value indicates a greater similarity. Figure 5.2 shows that Euclidean similarity does not differentiate between financial time-series pairs as distinctly as our similarity mea-

Table 5.2 The 2-to-1 directed hyperedge with the highest ACV and the constituent directed edges for each selected financial time-series from different sectors and for each configuration choice are shown.

Row	**Time-series**	**Configuration**	**Top 2-to-1 directed hyperedge**	**Directed edge 1**	**Directed edge 2**
1	EMN ($\mathcal{BM}$)	$C1$	AVY, GT → EMN (0.52)	AVY → EMN (0.49)	GT → EMN (0.49)
		$C2$	BLL, IFF → EMN (0.37)	BLL → EMN (0.32)	IFF → EMN (0.33)
2	HON ($\mathcal{CG}$)	$C1$	CAT, ITT → HON (0.53)	CAT → HON (0.5)	ITT → HON (0.49)
		$C2$	BA, ROK → HON (0.38)	BA → HON (0.33)	ROK → HON (0.33)
3	GT ($\mathcal{CC}$)	$C1$	DOW, F → GT (0.51)	DOW → GT (0.48)	F → GT (0.47)
		$C2$	ETN, FMC → GT (0.37)	ETN → GT (0.33)	FMC → GT (0.33)
4	PG ($\mathcal{CN}$)	$C1$	CLX, K → PG (0.53)	CLX → PG (0.5)	K → PG (0.49)
		$C2$	ABT, CPB → (0.36)	ABT → PG (0.32)	CPB → PG (0.32)
5	XOM ($\mathcal{E}$)	$C1$	HES, SLB → XOM (0.58)	HES → XOM (0.55)	SLB → XOM (0.54)
		$C2$	COG, PEG → XOM (0.37)	COG → XOM (0.33)	PEG → XOM (0.31)
6	AIG ($\mathcal{F}$)	$C1$	BEN, PGR → AIG (0.54)	BEN → AIG (0.51)	PGR → AIG (0.51)
		$C2$	AON, CI → AIG (0.37)	AON → AIG (0.33)	CI → AIG (0.33)
7	JNJ ($\mathcal{H}$)	$C1$	IFF, SYY → JNJ (0.48)	IFF → JNJ (0.45)	SYY → JNJ (0.45)
		$C2$	CL, PEP → JNJ (0.36)	CL → JNJ (0.32)	PEP → JNJ (0.31)
8	JCP ($\mathcal{SV}$)	$C1$	FDO, GPS → JCP (0.51)	FDO → JCP (0.48)	GPS → JCP (0.48)
		$C2$	COST, HD → JCP (0.37)	COST → JCP (0.32)	HD → JCP (0.33)
9	INTC $\mathcal{T}$)	$C1$	EMC, QCOM → INTC (0.55)	EMC → INTC (0.52)	QCOM → INTC (0.52)
		$C2$	CTXS, QCOM → INTC (0.4)	CTXS → INTC (0.35)	QCOM → INTC (0.35)
10	FDX ($\mathcal{TP}$)	$C1$	EXPD, ITT → FDX (0.52)	EXPD → FDX (0.49)	ITT → FDX (0.46)
		$C2$	EXPD, BAC → FDX (0.37)	EXPD → FDX (0.33)	BAC → FDX (0.33)
11	TE ($\mathcal{U}$)	$C1$	PEG, SO → TE (0.55)	PEG → TE (0.52)	SO → TE (0.52)
		$C2$	SO, TEG → TE (0.4)	SO → TE (0.35)	TEG → TE (0.35)

sures do. This could be because of the fact that Euclidean similarity accounts for pair-wise differences in price variations on a day-to-day basis whereas the similarity measures account for the closeness in being associated with common sets of financial time-series on an average basis.

5.3.2 Clusters of Financial Time-Series

Figure 5.3 shows a clustering of financial time-series for configuration $C1$. The clusters are obtained using the approach explained in Section 3.3.2. Here, the collection $\mathcal{S}$ is the set of all financial time-series in our data set. The value of parameter t in the t-clustering algorithm is set to 104, which is the total number of sub-sectors over the entire sectors (as pointed out at the beginning of Chapter 5). The first cluster center is picked from the sector $\mathcal{T}$ as this sector has the maximum number of financial time-series in our data set.

We experimentally verified that the weight function in Definition 3.13 satisfies the triangle inequality property, and hence the factor 2-approximation of the optimal diameter of the t-clustering is in fact achieved by the algorithm. For clarity of display, Figure 5.3 shows only clusters of size greater than 6. The edges that connect all cluster centers to other nodes in their clusters and the edges that interconnect the cluster centers are also shown in the figure. This partial similarity graph consists of 256 nodes and 298 edges. To show the quality of the clustering obtained, the following information is relevant: (i) the mean diameter over all clusters obtained is 0.83 and the overall mean distance in $SG_{\mathcal{S}}$ is 0.89 and (ii) the largest cluster of size 29 contains all financial time-series from the sector $\mathcal{T}$.

5.4 Leading Indicators of Financial Time-Series

In this experiment, we find the leading indicators for the collection $\mathcal{S}$ of all financial time-series using the approach explained in Section 4.1. In order to obtain a dominator set that covers the rest of the financial time-series via directed edges and 2-to-1 directed hyperedges of high ACV, we set a threshold for ACV (ACV-threshold) and discard all directed edges and 2-to-1 directed hyperedges below this threshold during the computation of the dominator. Now, for the computation of dominators for configurations $C1$ and $C2$, we consider

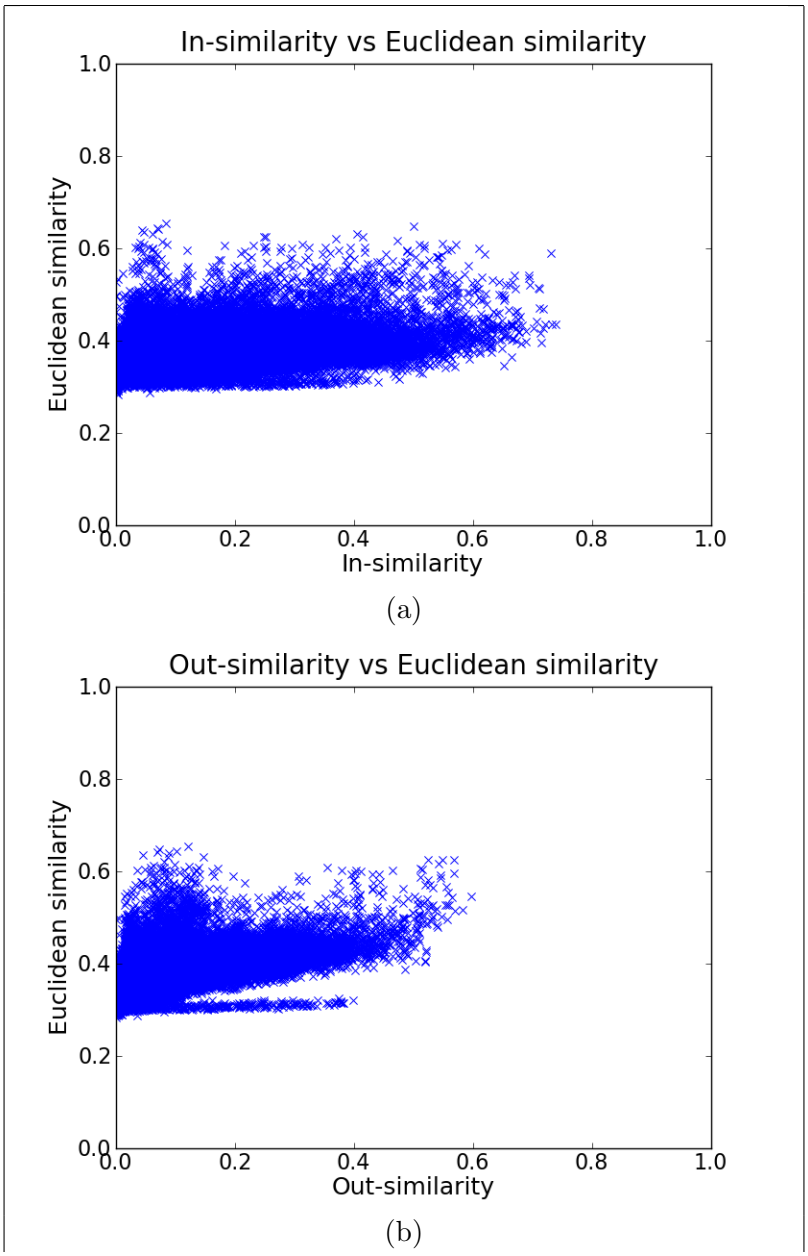

Figure 5.2 Euclidean similarity comparision. (a) In-similarity (for configuration $C1$) vs Euclidean similarity and (b) Out-similarity (for configuration $C1$) vs Euclidean similarity. The in-similarity (out-similarity) values are on the x-axis and the corresponding Euclidean similarity values are on the y-axis.

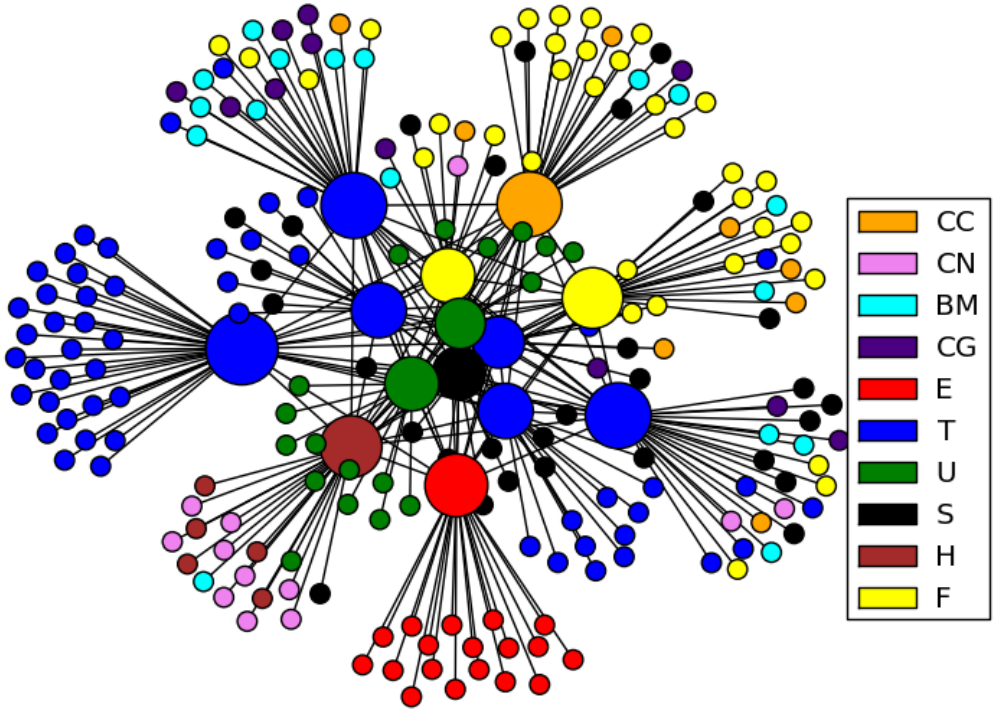

Figure 5.3 Clusters of financial time-series for configuration $C1$. Note that this figure is made up of multiple colors and the colors correspond to sectors in financial domain. The big circles represent cluster centers and the small ones represent other nodes. The size of the big circles is directly proportional to the number of nodes assigned to them. The small circles are attached to their respective cluster centers.

the following choices for thresholds: (i) top 40% directed hyperedges w.r.t. ACVs—this sets ACV-threshold to 0.45 for $C1$ and sets ACV-threshold to 0.32 for $C2$, (ii) top 30% directed hyperedges w.r.t. ACVs—this sets ACV-threshold to 0.46 for $C1$ and sets ACV-threshold to 0.33 for $C2$, and (iii) top 20% directed hyperedges w.r.t. ACVs—this sets ACV-threshold to 0.47 for $C1$ and sets ACV-threshold to 0.34 for $C2$. Tables 5.4 and 5.3 shows the size of dominator for almost all the financial time-series in our data set. Here, in row 1 of Table 5.4, for the case when ACV-threshold is set to 0.45, our greedy approximation algorithm (Section 4.1) finds a dominator of size 16 that covers 96% of all financial time-series in our data set.

5.5 Association-Based Classifier

In this experiment, we evaluate the accuracy of the assignments given by the Asociation-Based Classifier on several test data sets. For each training data set, we construct an association hypergraph $\mathcal{H} = (V, E)$ using the procedure described in Section 3.2. Next, in the corresponding test data set, all the financial time-series are converted to their respective delta time-series and then discretized using the methodology described in Section 5.1.1. We choose a small collection $\mathcal{S}$ of financial time-series, usually a dominator for all financial time-series in our data sets. The values of every financial time-series A in the dominator are already known from the discretized representation of A. The best predicted value of all other financial time-series in our data sets is computed using the association-based classifier presented in Section 4.2. We define the *classification confidence* for any financial time-series Y on a particular test data set as the fraction of days on which the value assigned by our classifier matches the value in Y's discretized representation, obtained from the same data set.

We also compute the accuracy of value assignments given by other data mining classifiers such as the support vector machine (SVM), multilayer perceptron, and logistic regression. For experiments here, the classifiers provided by Weka [HFH+09] are used. The following methodology is used to predict the values of any financial time-series Y by constructing a training data set whose feature set is the set $\mathcal{S}$ of financial time-series: Consider a directed

Table 5.3 The size of a dominator for all financial time-series and the mean classification confidence of different classifiers for each configuration are shown obtained using Algorithm 5.

Row	***Configuration***	***ACV-threshold*** (top % hyper-edges)	**Dominator Size**	**Percent Covered**	**Mean Classification Confidence**				
					In-sample	**Out-sample**			
					Association-Based Classifier	**Association-Based Classifier**	**SVM**	**Multilayer Perceptron**	**Logistic Regression**
1	$C1$	0.45 (40%)	13	99	0.643	0.719	0.546	0.716	0.541
		0.46 (30%)	15	95	0.646	0.723	0.509	0.718	0.508
		0.47 (20%)	22	94	0.65	0.724	0.494	0.719	0.492
2	$C2$	0.32 (40%)	20	96	0.646	0.716	0.429	0.627	0.231
		0.33 (30%)	30	96	0.649	0.719	0.433	0.638	0.238
		0.34 (20%)	31	91	0.65	0.722	0.403	0.633	0.224

Table 5.4 The size of a dominator for all financial time-series and the mean classification confidence of different classifiers for each configuration are shown obtained using Algorithm 6.

Row	***Configuration***	***ACV-threshold*** (top % hyper-edges)	**Dominator Size**	**Percent Covered**	**Mean Classification Confidence**				
					In-sample	**Out-sample**			
					Association-Based Classifier	**Association-Based Classifier**	**SVM**	**Multilayer Perceptron**	**Logistic Regression**
1	$C1$	0.45 (40%)	16	96	0.651	0.723	0.526	0.717	0.519
		0.46 (30%)	22	93	0.653	0.723	0.514	0.718	0.510
		0.47 (20%)	26	91	0.656	0.728	0.515	0.725	0.512
2	$C2$	0.32 (40%)	28	96	0.65	0.721	0.429	0.627	0.231
		0.33 (30%)	40	90	0.652	0.722	0.433	0.638	0.238
		0.34 (20%)	36	78	0.652	0.72	0.403	0.633	0.224

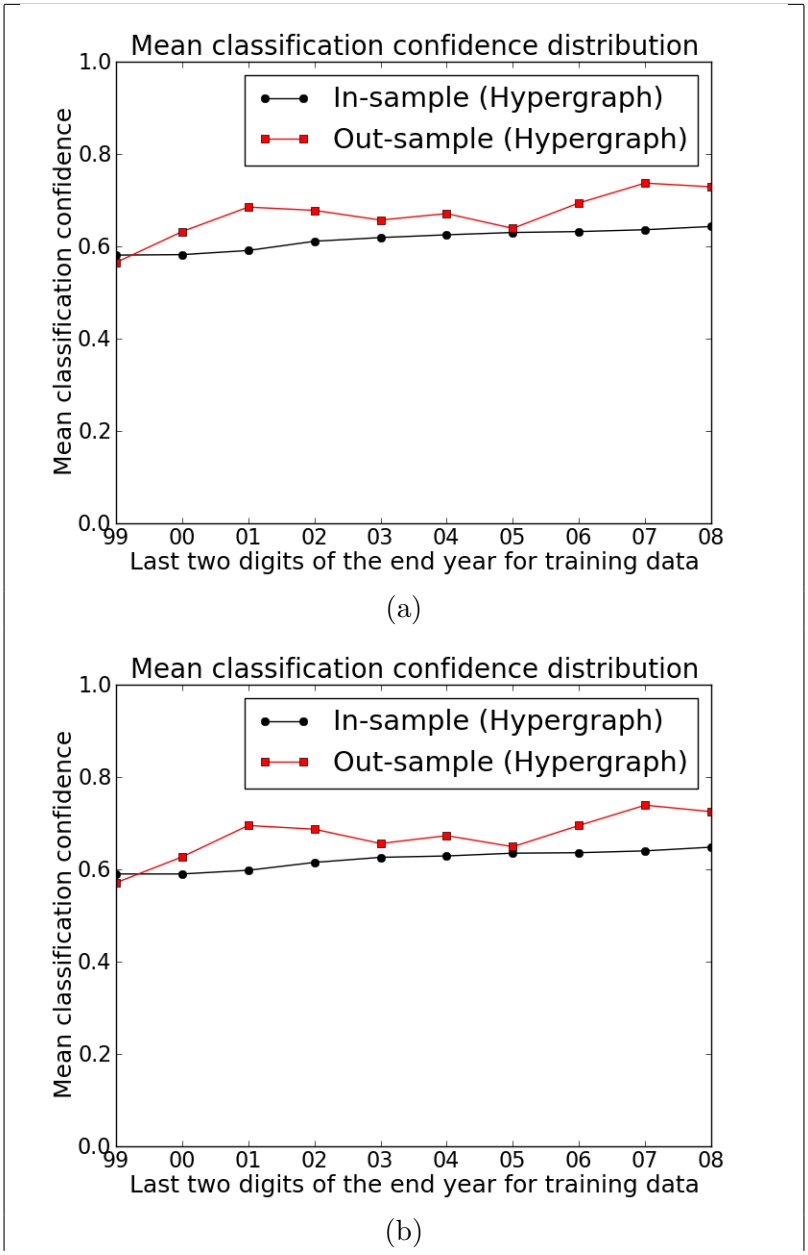

Figure 5.4 Classification confidence distribution of the association-based classifier for in-sample and out-sample data for configuration $C1$. The start year for training data set is 1996. Figure (a) uses the dominator obtained from Algorithm 5 and Figure (b) uses the dominator obtained from Algorithm 6.

hyperedge e in $\mathcal{H}$ such that $e = (\{A_1, A_2\}, \{Y\})$ and $A_1, A_2 \in \mathcal{S}$. The training data set is built by using each row in $AT(e)$ as a data point. Here, the particular value assignment $A_1 = v_1$ and $A_2 = v_2$ is the feature value, and the corresponding value y^* of Y (defined in Definition 3.6(1)) is the class value.

5.5.1 Evaluation

Tables 5.4 and 5.3 list, for each dominator, the mean classification confidence over all financial time-series for configurations $C1$ and $C2$. Here, *in-sample* indicates the training data set for constructing the directed hypergraph model and *out-sample* indicates the test data set for evaluating the value assignments made by the classifiers. The in-sample contains financial time-series data from Jan 1, 1996 to Dec 31, 2008 and the out-sample contains financial time-series data from Jan 1, 2009 to Dec 31, 2009. From this table, it is clear that the association-based classifier outperforms SVM, Logistic Regression, and Multilayer Perceptron for both configurations $C1$ and $C2$. Also, its mean classification confidence is consistent regardless of k, whereas the mean classification confidence of other classifiers decreases as k increases.

Figures 5.4(a) and 5.4(b) shows the distribution of the mean classification confidence over various in-sample and out-sample data for configuration $C1$ where the dominator having ACV-threshold of 0.45 is chosen to be $\mathcal{S}$. In these figures, the mean classification confidence for each in-sample data has been computed by increasing the training data set incrementally one year at a time, starting from Jan 1, 1996 and ending at Dec 31, 2008. The corresponding out-sample contains financial time-series data for one year immediately following the last day in the training data set. For instance, if the training data set is from Jan 1, 1996 to Dec 31, 2001, then the corresponding test data set contains financial time-series from Jan 1, 2002 to Dec 31, 2002. From these figures, it is evident that the association-based classifier achieves mean classification confidence in the range 0.60 to 0.75 on both in-sample and out-sample data. The higher classification confidence values for out-sample data compared to in-sample data may be attributed to the fact that the out-sample data is substantially smaller than the in-sample data.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

We proposed a directed hypergraph based model to capture attribute-level associations and their strength in any database. We tested this model on a financial time-series data set (S&P 500). The clustering method based on our similarity notions allowed to find clusters of financial time-series. The association-based classifier coupled with the leading indicator of all the financial time-series exhibited a methodology to use mva-type association rules and predict values of financial time-series. We also demonstrated the consistency of our model by varying k throughout our experiments.

Our work raises interesting questions on the applications of association rule mining. It might be fruitful to explore associations by applying the directed hypergraph model on data sets such as gene databases, social network data sets, and medical databases. It would be useful to understand how the different parameters (k, γ, and the sizes of head and tail sets) affect the model. Examples of applications of association rule mining in the context of social network data sets and medical database have been presented in Section 3.1.

In genetic research, a common problem is to effectively model the interrelationship among multiple genes. By recording the gene expression values of a set of genes, researchers work towards obtaining the gene expression values of others. In general, knowledge about the genes help researchers and physicists to understand the genetic state of patients and the genetic conditions for diseases.

Anastassiou [Ana07] provided a synergy-based method to analyze the interactions among multiple interacting genes. Such a framework can be used to predict the presence of a disease or the absence of a disease, based on given gene expression values. Although this is recent work in disease prediction, there been continuous interest in locating genes with similar characteristics. For example, Eisen et al. [ESBB98] provided a variation of clustering to

find clusters of similar genes. Finding a group of genes that instigate a certain disease is very important in recognizing a medical condition in patients.

The directed hypergraph based model, i.e. association hypergraph, presented in this thesis can be used to model gene interactions. By modeling the gene interactions using an association hypergraph, we can address the following problems: (1) identify clusters of similar genes and predict gene expression values of a set of genes, and (2) identify disease causing conditions present among a set of genes and predict the presence of a disease or the absence of a disease. Let a gene database consist of gene expression values recorded from patients. Also, let the database contain information on the status of patients being affected by a certain disease. Here genes and diseases are the multi-valued attributes, and each observation consists of the gene expression values and the disease status, of a particular patient.

The problem stated in (1) can be addressed by considering the part of the gene database that only consists of the multi-valued attributes that correspond to the gene expression values and constructing an association hypergraph using the technique described earlier in Section 3.1. By considering the multi-valued attributes that correspond to the gene expression values, interactions among the genes can be modeled using directed hyperedges. Then, groups of similar genes can be identified by finding clusters of similar multi-valued attributes, as given in Section 3.3.2. Also, by knowing the gene expression values of a subset of genes, the gene expression values of the remaining genes can be predicted using the association-based classifier given in Section 4.2.

The problem stated in (2) can be addressed by considering the entire gene database and constructing an association hypergraph using the technique described earlier in Section 3.1. In this problem, we are interested in obtaining a prediction for the presence of a disease or the absence of a disease in patients. Therefore, during the construction of the association hypergraph, the directed hyperedges whose headset consist a disease are the only ones that get included in the association hypergraph. The remaining directed hyperedges are not considered for inclusion in the association hypergraph. Now, by knowing the gene expression values of a subset of genes for a patient, a disease prediction can be obtained by using the association-based classifier given in Section 4.2.

REFERENCES

[ADS83] G. Ausiello, A. D'Atri, and D. Saccá. Graph algorithms for functional dependency manipulation. *Journal of the ACM*, 30(4):752–766, 1983.

[ADS85] G. Ausiello, A. D'Atri, and D. Saccá. Strongly equivalent directed hypergraphs. *Analysis and Design of Algorithms for Combinatorial Problems*, 25:1–25, 1985.

[AG97] G. Ausiello and R. Giaccio. On-line algorithms for satisfiability formulae with uncertainty. *Theoretical Computer Science*, 171(1–2):3–24, 1997.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD 93*, pages 207–216. ACM, 1993.

[Ana07] D. Anastassiou. Computational analysis of the synergy among multiple interacting genes. *Molecular Systems Biology*, 3:1–8, 2007.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB 94*, pages 487–499. Morgan Kaufmann, 1994.

[AV06] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *ACM Symposium on Computational Geometry*. ACM, 2006.

[BAG99] R. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *ICDE 99*, pages 188–197. IEEE, 1999.

[Bay97] R. Bayardo. Brute-force mining of high-confidence classification rules. In *KDD 97*, pages 123–126. ACM, 1997.

[Ber73] C. Berge. *Graphs and Hypergraphs*. North-Holland, 1973.

[BZ07] B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *ICDM 07*, pages 63–72. IEEE, 2007.

[CDP04] S. Chawla, J. Davis, and G. Pandey. On local pruning of association rules using directed hypergraphs. In *ICDE 04*, page 832. IEEE, 2004.

[CH03] C. Creighton and S. Hanash. Mining gene expression databases for association rules. *Bioinformatics*, 19:79–86, 2003.

[Chv79] V. Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4:233235, 1979.

[CSCR+06] P. Carmona-Saez, M. Chagoyen, A. Rodriguez, O. Trelles, J. Carazo, and A. Pascual-Montano. Integrated analysis of gene expression by association rules discovery. *BMC Bioinformatics*, 19:79–86, 2006.

[ESBB98] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25):14863–14868, December 1998.

[GLPN93] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2–3):177–201, 1993.

[Gon85] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[GS02] G. Gallo and M. Scutella. A note on minimum makespan assembly plans. *European Journal of Operational Research*, 142:309–320, 2002.

[HFH+09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[HKKM98] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *IEEE Data Eng. Bull.*, 21(1):15–22, 1998.

[Joh74] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256278, 1974.

[KH06] A. Knobbe and E. Ho. Maximally informative k-itemsets and their efficient discovery. In *KDD 06*, pages 237–244. ACM, 2006.

[KNO+03] L. Krishnamurthy, J. Nadeau, G. Ozsoyoglu, M. Ozsoyoglu, G. Schaeffer, M. Tasan, and W. Xu. Pathways database system: an integrated system for biological pathways. *Bioinformatics*, 19(8):930–937, 2003.

[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD 98*, pages 80–86. ACM, 1998.

[LHM99] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *KDD 99*, pages 337–341. ACM, 1999.

[Llo82] S. Lloyd. Least square quantization in pcm. *IEEE Transactions on Information Theory*, 28:129137, 1982.

[Lov75] L. Lovasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383390, 1975.

[LS95] W. Lin and M. Sarrafzadeh. A linear arrangement problem with applications. In *ISCAS 95*, pages 57–60, 1995.

[LSW97] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *ICDE 97*, pages 220–231. IEEE, 1997.

[NLHP98] R. Ng, L. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD 98*, pages 13–24. ACM, 1998.

[OA04] M. Ozdal and C. Aykanat. Hypergraph models and algorithms for data-pattern-based clustering. *Data Mining and Knowledge Discovery*, 9(1):29–57, 2004.

[Ord06] C. Ordonez. Comparing association rules and decision trees for disease prediction. In *HIKM 06*, pages 17–24. ACM, 2006.

[Pre00] D. Pretolani. A directed hypergraph model for random time dependent shortest paths. *European Journal of Operational Research*, 123:315–324, 2000.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386408, 1958.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB 95*, pages 407–419. Morgan Kaufmann, 1995.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD 96*, pages 1–12. ACM, 1996.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *KDD 97*, pages 67–73. ACM, 1997.

[SVL06] A. Siebes, J. Vreeken, and M. Leeuwen. Item sets that compress. In *SDM 06*. SIAM, 2006.

[TT09] M. Thakur and R. Tripathi. Linear connectivity problems in directed hypergraphs. *Theoretical Computer Science*, 410(27–29):2592–2618, 2009.

[Vie04] A. Vietri. The complexity of arc-colorings for directed hypergraphs. *Discrete Applied Mathematics*, 143(1-3):266–271, 2004.

[Yah10] Yahoo.com. Yahoo finance. http://finance.yahoo.com/, 2010.

ABOUT THE AUTHOR

Ramanuja Simha studied information science at The National Institute of Engineering affiliated to the Visvesvaraya Technological University from 2001 to 2005. He graduated with a Bachelor of Engineering in Information Science. Then, he worked at Tesco Hindustan Service Center as a Software Engineer from 2005 to 2008, and had the title of a Senior Software Engineer when he left the firm. He began his graduate studies in computer science at the University of South Florida in 2008. There, he pursued research in Algorithms and Data Mining under the supervision of Prof. Rahul Tripathi. During this period, he also completed a summer internship at the National Center for Atmospheric Research in the Computational and Information Systems Laboratory.