UNIVERSITY OF CALIFORNIA, SAN DIEGO

Hierarchical Clustering and Interaction

A Thesis submitted in partial satisfaction of the requirements for the degree of Master of Science

in

Computer Science

by

Madhavi Yenugula

Committee in charge:

Professor Sanjoy Dasgupta, Chair
Professor Julian McAuley
Professor Lawrence K. Saul

2016

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor Prof.Sanjoy Dasgupta for his guidance, mentorship and for being an amazing teacher. I would like to thank Prof. Lawrence Saul and Prof. Julian McAuley for reviewing my thesis and for being a part of my committee.

I am grateful to my family for their unconditional love and support through the toughest of times. I would also like to thank my friends Himaja, Maneesha and Meghana for their patience, support and friendship. I'd also like to acknowledge the crucial role UCSD played in the two important years of my life, both professionally and personally.

ABSTRACT OF THE THESIS

Hierarchical Clustering and Interaction

by

Madhavi Yenugula

Master of Science in Computer Science

University of California, San Diego, 2016

Professor Sanjoy Dasgupta, Chair

Hierarchical Clustering is a clustering method which defines clusters on the data at various granularities - starting with a single cluster with all input data points to clusters with just individual points. Any desired number of clusters can be obtained by breaking off the hierarchy at some level and nodes of the pruned branches can be merged to form clusters.

Clusters at any given level of hierarchy depend on clusters formed in the previous level. Hierarchical clustering approaches operate greedily without backtracking. The final hierarchy is often not what the user expects, it can be improved by providing feedback.

This work studies various ways of interacting with the hierarchy - providing feedback to and incorporating feedback into the hierarchy. We discuss metrics to quantify quality of a hierarchy. We apply the designed feedback mechanism on datasets with different attribute types. We report results of application of these methods on datasets and improvements in the hierarchies as per defined metrics.

Introduction

Clustering is one of the most common forms of unsupervised learning with a wide range of applications in machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression and computer graphics. Clustering refers to the task of grouping objects together such that objects in the same group are more similar to each other than those in other groups. The groups thus formed are referred to as *clusters*. Examples of clustering algorithms include K-means[17], Fuzzy C-means[3], Hierarchical clustering[12] and Mixture of gaussians using EM algorithm [8].

Naturally, there can be more than one criterion to cluster data. Given the unsupervised nature of clustering algorithms, the criterion specific to user's needs is often not reflected in the clusters produced by these algorithms. Constrained clustering, discussed in [25] elaborates on *must-link* and *cannot-link* constraints. These constraints can be added by the user, which are in turn used to produce desirable clusters. These are pair-wise constraints which allow or disallow points to be in the same cluster. These type of constraints are used in [26], [7], [28] for various applications.

Hierarchical Clustering is a tree representation of *clusters* of a given dataset. Leaves of the tree represent individual data points and each internal node defines a cluster on leaf nodes of its descendants. Hierarchical Clustering algorithms fall into two classes - agglomerative hierarchical clustering (bottom-up approach) and partitional hierarchical clustering (top-down approach).

In the bottom-up approach we start with individual clusters and merge nodes recursively to produce one single cluster with all the data points. In top-down approach we start with a single cluster and split recursively till we have single node clusters. These are greedy approaches with no backtracking. They lead to rigid hierarchies, sometimes resulting in undesirable clusters as in the case of flat clustering techniques described above. Hence, we believe that there is a scope for improvement in the hierarchy through user interaction. In this work, we primarily explore different ways to interact with the hierarchy. Feedback provided by the user is assimilated into the hierarchy using *triplet constraints* (analogous form of pair-wise constraints) and class information.

We experiment with 3 different types of datasets and analyze the results. In particular, we compare the resulting hierarchy with the ideal hierarchy. Ideal hierarchy is a tree with a root node and each of its children representing one of the underlying classes. An example is provided in Figure 2.5. *Triplet distance* metric quantifies the distance between a given hierarchy and the ideal hierarchy. From our experiments, we observe that in most cases, interaction helps in bringing a hierarchy closer to the ideal hierarchy. The objective of this work is to address following questions:

1. In what ways can we interact with the hierarchy of clusters of a dataset?

2. How does the hierarchy improve as we interact ?

The thesis is organized into 6 chapters. Chapter 1 introduces Hierarchical Clustering, algorithms used to construct the hierarchy, data representations to express similarity/distance between the data points and various datasets we used throughout the work. Chapter 2 introduces triplet distance, triplet constraints and methods to visualize the hierarchy. Chapter 3 discusses triplet constraints and the results obtained on different datasets through triplet constraint interaction. Chapter 4 explores other methods of interacting with hierarchy: by adding constraints using labels and a combination of triplets with labels. Chapter 6 summarizes our results and discusses future work.

Chapter 1

Background

Clustering refers to the process of grouping similar data into "clusters" and separating dissimilar data into different clusters. Flat clustering techniques involve a parameter, k, number of clusters the data should be grouped into. Often times, it is hard to determine a specific k that results in a clustering which fits expectations of the user. Additionally clustering produces a single grouping of data points. A hierarchical view of clusters in the data is more informative than a single grouping returned by traditional flat clustering methods. Hierarchical Clustering addresses these issues but with an additional computational overhead.

Hierarchical Clustering[18] is a tree of clusters of data. Each level in the hierarchy (or tree) defines a clustering of the given data at different granularity. Scanning the tree from top to bottom, the root cluster of the hierarchy constitutes a cluster of all data points in the given dataset. The data points corresponding to a node/cluster in the tree get split among the node's children in next level.

Figure 1.1 shows a possible hierarchical clustering for the dataset with data points $\{1,2,3,4\}$. The root node of the hierarchy has all 4 nodes. At level 1, the root node gets split into 2 child nodes, $C_1 = \{1,2,4\}$ and $C_2 = \{3\}$. C_1 is then split into two other nodes, and this process continues till all nodes contain a single data point.

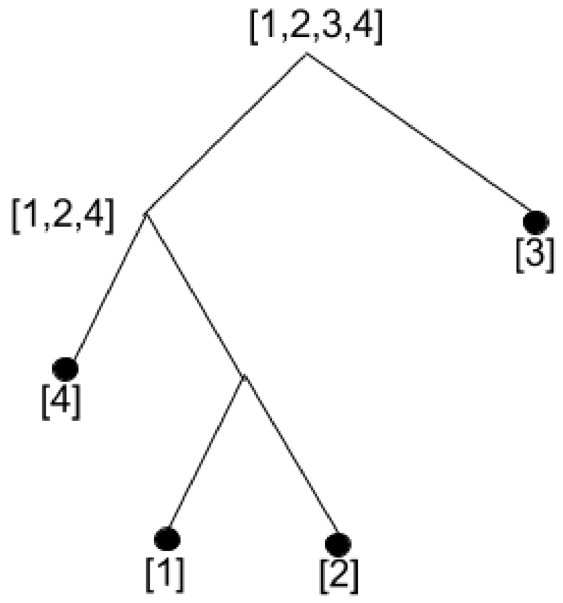


Figure 1.1. Hierarchical Clustering Example

The hierarchy of clusters for a given dataset can be constructed in a top-down or bottom-up approach. In top-down approaches we start with a single cluster with all data points and recursively split the clusters at each level to build the hierarchy. In bottom-up approaches, we start with each of data points in the dataset as individual nodes and merge nodes at each level till we form a single cluster node containing all data points.

1.1 Agglomerative hierarchical clustering

Agglomerative Hierarchical clustering [19] constructs the hierarchy of clusters for a given dataset in a bottom-up fashion. At each stage of the bottom-up approach, distance(similarity) matrix between each of the clusters is computed and clusters *closest(most similar)* to each other are merged. Popular agglomerative approaches include single-linkage, complete-linkage and average-linkage clustering.
Let $d(x,y)$ be the distance between two data points x and y.

- *single-linkage clustering:* Distance *D(X,Y)* between two clusters X and Y is the minimum distance between members of each cluster. It is defined as

  $$D(X,Y) = \min_{x \in X, y \in Y} d(x,y).$$

  In other words, similarity between two clusters is similarity of the most similar members.

- *complete-linkage clustering:* Distance *D(X,Y)* between two clusters X and Y is the maximum distance between members of each cluster. It is defined as

  $$D(X,Y) = \max_{x \in X, y \in Y} d(x,y).$$

  In other words, similarity between two clusters is similarity of the most dissimilar members.

- *average-linkage clustering:* Distance *D(X,Y)* between two clusters X and Y is the mean distance between members of each cluster. It is defined as

  $$D(X,Y) = \frac{1}{|X||Y|} \max_{x \in X, y \in Y} d(x,y).$$

  This method is a compromise between the two extremes listed above (single-linkage and complete-linkage).

  The *closest(most similar)* clusters according to the distance metric are merged at every level, resulting in the hierarchy.

1.2 Partitional hierarchical clustering

Partitional hierarchical clustering constructs the hierarchy of clusters for a given dataset in a top-down fashion. Each stage in the top-down approach includes splitting the nodes(clusters) using similarity metrics and the resulting clusters are further split to construct the hierarchy. Splitting the nodes at a level using spectral clustering [21] is an example for top-down approaches. This method is discussed in detail in Section 1.4.

1.3 Similarity

Clustering is grouping *similar* data into clusters. Usually, data is provided as a list of feature vectors. One way to represent the *similarity* is by computing a similarity matrix of the given data [24].

Fully Connected Graph:

We first compute the Distance matrix D where each entry is the distance between two data points.

$$D_{ij} = |(x_i - x_j)|$$

Calculate σ as follows -

$$\sigma = \frac{\sqrt{\sum_{i=1..N} \sum_{j=i..N} D_{ij} \times D_{ij}}}{N}$$

We then apply gaussian similarity function to restrict similarity values to *0-1* range.

$$S_{ij} = e^{-\frac{D_{ij}^2}{2 \times \sigma^2}}$$

***k*-nearest neighbor Graph:**

We can also construct the similarity matrix S using nearest neighbors. An entry S_{ij} in the matrix is non-zero if i is one of the k-nearest neighbors of j or j is one of the k-nearest neighbors of i.

$$S_{ij} = \begin{cases} e^{-\frac{D_{ij}^2}{2\times\sigma^2}} & \text{if } i \text{ and } j \text{ are } k\text{-nearest neighbors to each other} \\ 0 & \text{otherwise} \end{cases}$$

1.4 Spectral Clustering Algorithm

[6] defines a cost function for the hierarchy, proves NP-hardness of constructing the optimal hierarchy, and discusses a greedy approximation for the problem. For a binary split $S \to (S_1, S_2)$, the split cost is $|S|w(S_1, S_2)$. The weight of the cut $w(S_1, S_2)$ is given by

$$w(S_1, S_2) = \sum_{\{i,j\}\in E: i\in S_1, j\in S_2} w_{ij}.$$

w_{ij} is the weight/similarity between the data points x_i and x_j. The cost of a k-ary split is given by,

$$w(S_1, \ldots, S_k) = \sum_{1\le i\le j\le k} w(S_i, S_j).$$

The cost of the hierarchy T is the sum of split costs of all the internal nodes,

$$cost(T) = \sum_{\text{splits } S\to(S_1,\ldots,S_k) \text{ in } T} |S|w(S_1, \ldots, S_k).$$

The paper states that a greedy criterion is to choose the split $V \to (S, V \setminus S)$ that yields the maximum shrinkage per unit cost

$$\frac{w(S, V\setminus S)}{|S|.|V\setminus S|}.$$

This is the *sparsest cut* problem. various approximation algorithms are available for solving this problem ([13], [2], [24]).

We construct a binary hierarchy, each cluster in the hierarchy is split into two child clusters at each level. We find the sparsest cut using Spectral Clustering ([24], [21]). Spectral Clustering algorithms use distance/similarity matrix of the data points and involve computation of eigen vectors to split the cluster into child clusters.

The algorithm for spectral clustering from [21] is detailed in Algorithm 1. We start with a set of n data points $\{d_1, d_2, ..., d_n\}$ which need to be clustered into k clusters.

Algorithm 1

```
1: Calculate the similarity matrix $S_{n\times n}$ using methods described in Section 1.3.
2: Construct the Laplacian $L = D^{-1/2} S D^{-1/2}$.
3: $\{x_1, x_2, .., x_k\}$ = $k$ largest eigen vectors of $L$.
4: $X_{n\times k} = [x_1, x_2, .., x_k]$ with eigenvectors as columns.
5: Construct $Y_{n\times k}$ by normalizing rows of X
       $Y_{ij} = \frac{X_{ij}}{\sqrt{\sum_j X_{ij}^2}}$
   .
6: Considering each of the $k$ rows in $Y$ as a point in $\mathbb{R}^k$, cluster the each data point $d_i$ into $k$ clusters (using eg., $k$-means).
```

Other references: [23], [27].

1.5 DataSets

We use *Animals with Attributes* dataset and *MNIST* datasets primarily for our analysis in this work. We chose datasets whose data can be grouped into distinct clusters.

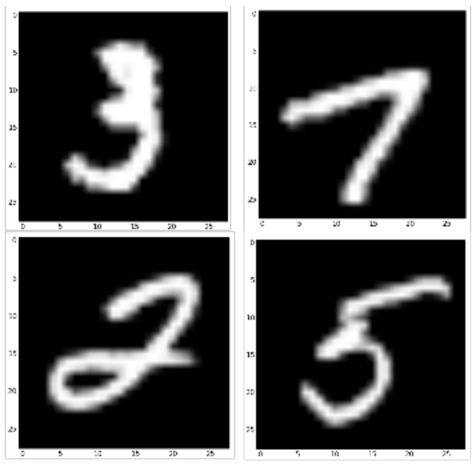


Figure 1.2. MNIST examples

MNIST Dataset:

MNIST [16] is a large database of images of hand written digits centered in a 28×28 pixel image (a total of 784 features). In addition to the feature vectors, we use the class information (also available with the dataset) in our experiments. Some images from the dataset are included in 1.2.

We work with random samples of different sizes ranging from 100 to 10000 of the available MNIST dataset of 60000 images. Each image falls into one of the 10 classes corresponding to the digit the image represents.

Animals With Attributes:

Animals with Attributes (AWA) dataset [15] contains 50 animals each containing 85 attributes. Each of the attributes have a semantic meaning associated with it. Attributes of the dataset include quadrapedal, water, weak etc. Binary form of the attributes represent the presence or absence of the feature and the numerical form of attributes represent the degree of presence of attribute in the animal. A complete list of animals can be seen in Figure 1.3.

The class information for animals in the AWA dataset are taken from semantic hierarchy defined in [9]. Each leaf animal falls into the class defined by the class names at the second level of the tree: rodent, primate, aquatic mammal, carnivore, ungulate. Mole, elephant, rabbit, bat are in a separate class by themselves. For eg., blue whale is an aquatic mammal, otter is a carnivore. This is a high level classification of animals as we are not classifying elements into classes from lower levels of Figure 1.3.

We also experiment with Mushrooms dataset for the interpretable hierarchical clustering case, the details of this dataset are described in Section 5.1.

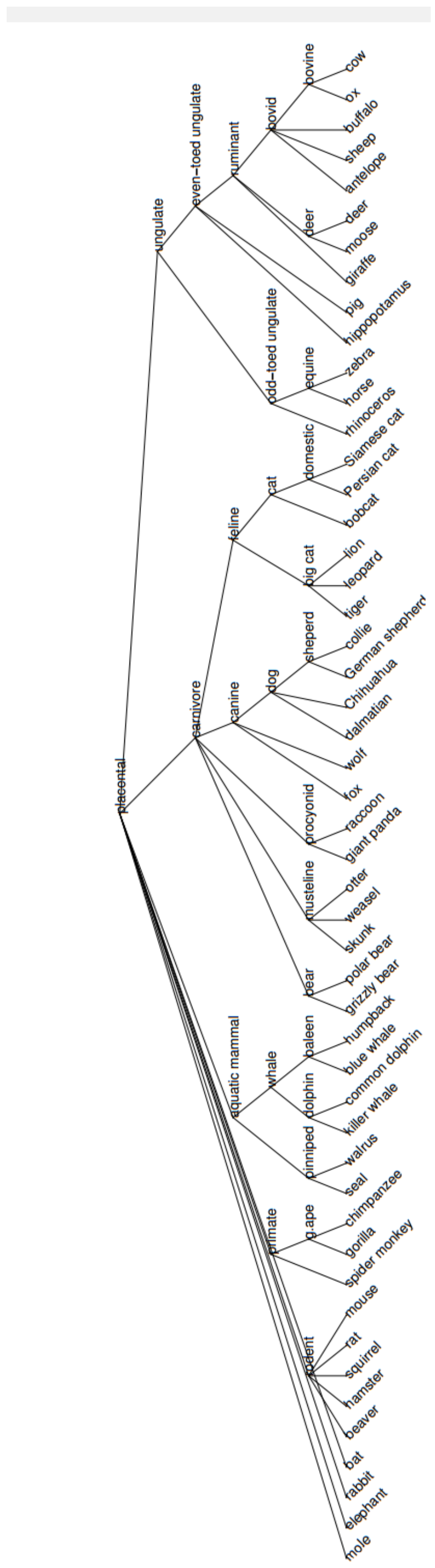


Figure 1.3. AWA - Semantic Hierarchy

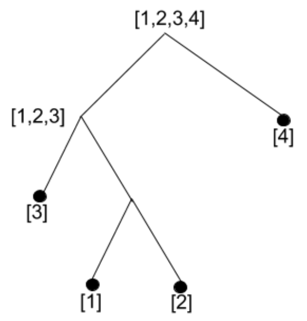
Chapter 2

Triplet constraints, Triplet distance and Visualization

2.1 Triplet Constraints

For flat clustering techniques, we discussed *must-link* and *cannot-link* constraints. The analogous form of these constraints for hierarchical clustering case are the *triplet constraints*. The main form of interaction with the hierarchy in this work is through the addition of triplet constraints. A triplet constraint is of the form $[(d_1,d_2),d_3]$ where each of d_1,d_2,d_3 represent 3 different data points. By enforcing the constraint, we constrain the hierarchy such that the data points (d_1,d_2) are split from d_3 before they are split from each other. In other words, (d_1,d_2) are split from d_3 higher up in the hierarchy compared to the node where (d_1,d_2) are split. The notion of triplets in trees is discussed in detail in [1].

Consider an example dataset of size 4 - $(1,2,3,4)$. By adding the constraint $[(1,2),3]$, we only accept hierarchies which follow the afore mentioned rules. Figure 2.1-2.3 are the only valid hierarchies for the given constraint $[(1,2),3]$.



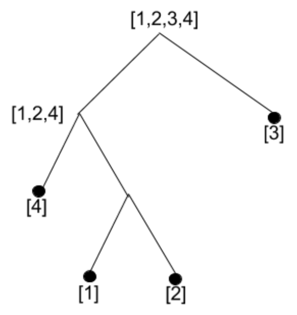
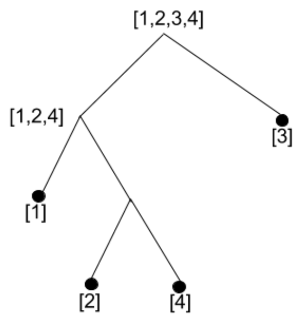


Figure 2.1. Valid Hierarchies with constraint $[(1,2),3]$



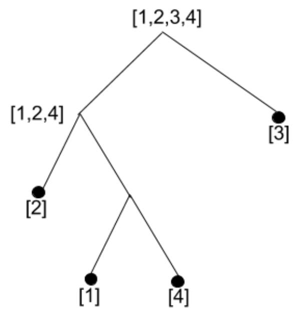


Figure 2.2. Valid Hierarchies with constraint $[(1,2),3]$

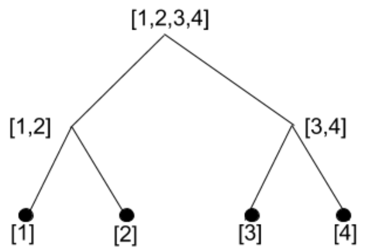


Figure 2.3. Valid Hierarchy with constraint $[(1,2),3]$

An example of invalid hierarchy is shown in Figure 2.4. This hierarchy is invalid as the constraint asks for data points $1,2$ to be split from each other only after they are split away from 3. But as seen in the figure, $1,2$ are split from each other before 3 is split away from 1.

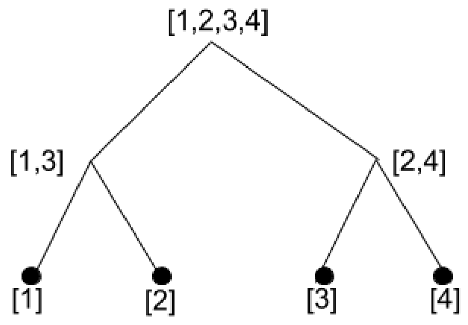


Figure 2.4. Invalid Hierarchy with constraint $[(1,2),3]$

2.2 Triplet Distance

We define Triplet distance for data which falls into different classes. For eg., In MNIST data, each image belongs to one of the 10 classes corresponding to the number displayed by image. So MNIST data falls into 10 categories. In case of Animals dataset, the classes are shown in Figure 1.3.

Each of these datasets (under consideration) have a true hierarchy associated with them. Figure 2.5 shows one such hierarchy. The ideal hierarchy splits all the data points into appropriate classes. Given a hierarchy we need to compare it with the ideal hierarchy and determine how *far* it is from the ideal hierarchy. As we interact with the hierarchy, we expect the *triplet distance* measure to capture the increasing similarity with the optimal hierarchy.

2.2.1 Setup

Let P be set of possible labels for the data. In the case of MNIST, it is $\{0, 1, 2, \ldots, 8, 9\}$. Let L be the actual labels for the sample data points we are considering. If the dataset size is 500, then $|L| = 500$ and every $L_i \in P$, $\forall i \leq 500$. Say, H is the hierarchy defined on the data points. Every node in the hierarchy contains the following information.

```
ParentNode {
  data points in this node
  childNode1
  childNode2
}
```

The hierarchy H can now be expressed as a root cluster with all input data points, and children. The data points are divided among the two child nodes. For a dataset of 5 points, an example root node can be

```
[1,2,3,4,5]
childNode1 - [1,2,3]
childNode2 - [4,5]
```

2.2.2 Triplet distance

Optimal Tree

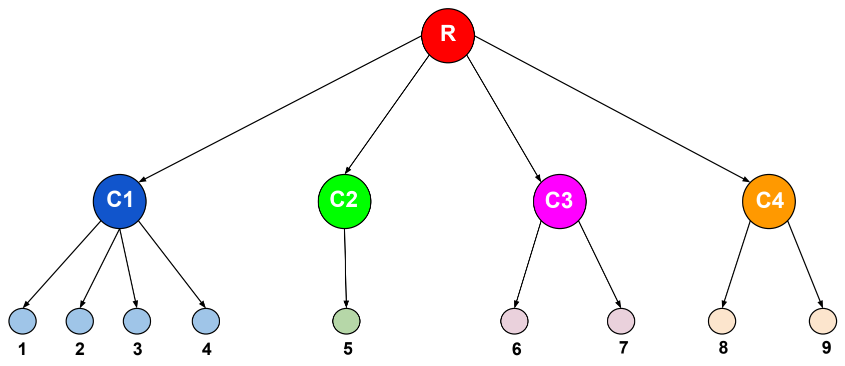


Figure 2.5. Optimal hierarchy

Given any sample of data points, an optimal hierarchy (H^*) is defined. It is a tree with a single root (R) and with different possible classes as children (C1, C2, . . . in Figure

2.5). We calculate the triplet distance for any hierarchy with respect to this optimal hierarchy.

In Figure 2.5, the leaves are data points. $\{1,2,3,4\} \in$ C1, $\{5\} \in$ C2, $\{6,7\} \in$ C3, $\{8,9\} \in$ C4. The triples defined on the hierarchy in Figure 2.5 are listed in Table 2.1.

Table 2.1. List of Triples

[(1,2),5]	[(1,2),6]	[(1,2),7]	[(1,2),8]
[(1,2),9]	[(1,3),5]	[(1,3),6]	[(1,3),7]
[(1,3),8]	[(1,3),9]	[(1,4),5]	[(1,4),6]
[(1,4),7]	[(1,4),8]	[(1,4),9]	[(2,3),5]
[(2,3),6]	[(2,3),7]	[(2,3),8]	[(2,3),9]
[(2,4),5]	[(2,4),6]	[(2,4),7]	[(2,4),8]
[(2,4),9]	[(3,4),5]	[(3,4),6]	[(3,4),7]
[(3,4),8]	[(3,4),9]	[(6,7),1]	[(6,7),2]
[(6,7),3]	[(6,7),4]	[(6,7),5]	[(6,7),8]
[(6,7),9]	[(8,9),1]	[(8,9),2]	[(8,9),3]
[(8,9),4]	[(8,9),5]	[(8,9),6]	[(8,9),7]

Let $|C_i| = n_i$ and $\sum_i n_i = |datapoints| = |D|$. The optimal hierarchy H^* has a total of $H^*_{triples}$ triples.

$$H^*_{triples} = \sum_i (|D| - n_i) \times \binom{n_i}{2}$$

All these triples are of the form $[(d_1,d_2),d_3]$ such that $d_1,d_2 \in C_i,\ d_3 \in C_j,\ i \neq j$. Triplet

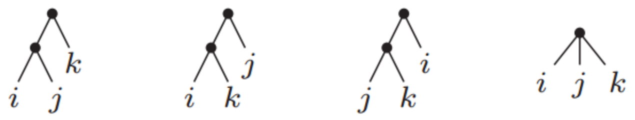


Figure 2.6. Possible topologies on $\{i, j, k\}$

distance ratio between any given hierarchy H and the optimal hierarchy is given by

$$\frac{\sum_{t \in H^*} 1(t \notin H)}{H^*_{triples}}$$

A slightly different notion of triplet distance is discussed in [4]. A total of $\binom{n}{3}$ triples are defined on a set of n data points. The triplet distance is the number of triples whose topology differs in the two trees. Possible topologies on $\{i, j, k\}$ is shown in Figure 2.6.

2.2.3 Algorithm

Every time a parent node is split into two child nodes, depending on the split, the hierarchy may move away from the optimal hierarchy. We first calculate the triplet distance created through dividing the parent cluster data points among the two children (since we have a binary hierarchical tree), then recursively compute and add the triplet distance among each of the children.

Algorithm 2

```
1: procedure CalculateDistance($P$)
2:     if P has no children then
3:         return 0
4:     $C_1, C_2$ = children of parent $P$
5:     dist = 0
6:     dist += CalculateDistanceBetweenChildren($C_1, C_2$)
7:     dist += CalculateDistance($C_1$)
8:     dist += CalculateDistance($C_2$)
9:     return dist
```

The parent node's data points are split among 2 children. Constraints violated can be of two types :

1. $d_1, d_2 \in$ child1 and label(d_1) != label (d_2) and $\exists\ d_3 \in$ child2 such that label(d_3) == label(d_1).

2. $d_1, d_2 \in$ child2 and label(d_1) != label (d_2) and $\exists\ d_3 \in$ child1 such that label(d_3) == label(d_1).

Constraint $[(d_1, d_3), d_2]$ is violated in both cases.

Algorithm 3. Calculate Distance between children

```
1: procedure CalculateDistanceBetweenChildren(child1, child2)
2:     dist = 0
3:     $L_1$ = child1's labels
4:     $L_2$ = child2's labels
5:     for label in $L_1$ do                                  ▷ identifies case1
6:         $c_1$ = count of label in $L_1$
7:         $c_2$ = count of label in $L_2$
8:         $d_1$ = count of all other labels in $L_1$
9:         dist += $c_1 \times c_2 \times d_1$
10:    for label in $L_2$ do                                  ▷ identifies case2
11:        $c_1$ = count of label in $L_1$
12:        $c_2$ = count of label in $L_2$
13:        $d_2$ = count of all other labels in $L_2$
14:        dist += $c_1 \times c_2 \times d_2$
15:    return dist
```

Complexity:

Algorithm 2 can run for upto $O(n)$ iterations, n is the input dataset size. Complexity of Algorithm 3 is O(c^2) where c is number of classes in the data. So, the run time of triplet distance calculation is $O(nc^2)$.

Example :

Consider the split $C_1 = \{$1a, 1b, 2a$\}$ with labels 1,1,2 respectively. Similarly $C_2 = \{$1c, 2b, 2c}. The correct split is {1a, 1b, 1c} and {2a, 2b, 2c}. Constraints violated are listed below.

Constraints identified by case 1:

```
1a, 1c, 2a

1b, 1c, 2a

2a, 2b, 1a

2a, 2c, 1a

2a, 2b, 1b

2a, 2c, 1b
```

Constraints identified by case 2:

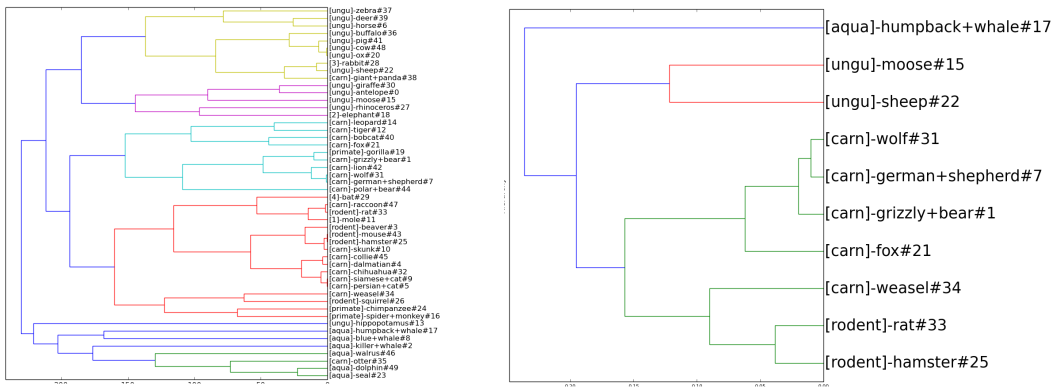
```
1a, 1c, 2b

1b, 1c, 2b

1a, 1c, 2c

1b, 1c, 2c

2a, 2b, 1c

2a, 2c, 1c
```

The above algorithm *calculate distance between children* gives 12 correctly.

2.3 Visualization

Dendrogram is a tool frequently used to visualize the clusters and the tree produced by hierarchical clustering algorithm. They are represented in the form of a tree. An example hierarchy is shown in Figure 2.8.

Dengrogram helps user to observe the hierarchy and provide feedback. But, as the dataset size increases, it becomes infeasible to observe the hierarchy. So, the hierarchy



(a) Full Hierarchy **(b)** Partial Hierarchy

Figure 2.7. Visualization of full and partial hierarchies on Animals dataset

induced on a subset of data points is shown to the user and user provides feedback. Example of a full hierarchy is shown in Figure 2.7a and the partial induced hierarchy is shown in Figure 2.7b. The feedback can be collected by showing the partial induced hierarcy over multiple samples of the dataset.

The horizontal axis of the figure has labels of data points. At each level in the hierarchy, a node (representing a cluster of data points) is split into 2 constituent nodes till the last level where each cluster has just one data point. Chapter 5 explains interpretable hierarchical clustering where the dendrogram is annotated with the feature responsible for the split. An example is shown in Figure 2.9. The text underneath each split represents the feature the data points in the cluster node are split upon.

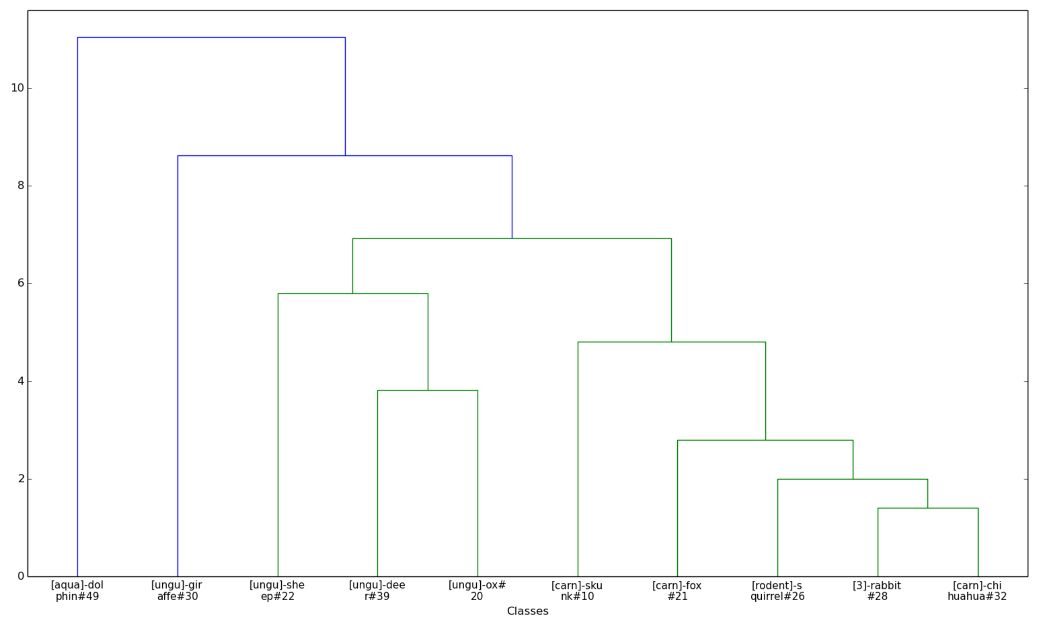


Figure 2.8. Example dendrogram for a hierarchy

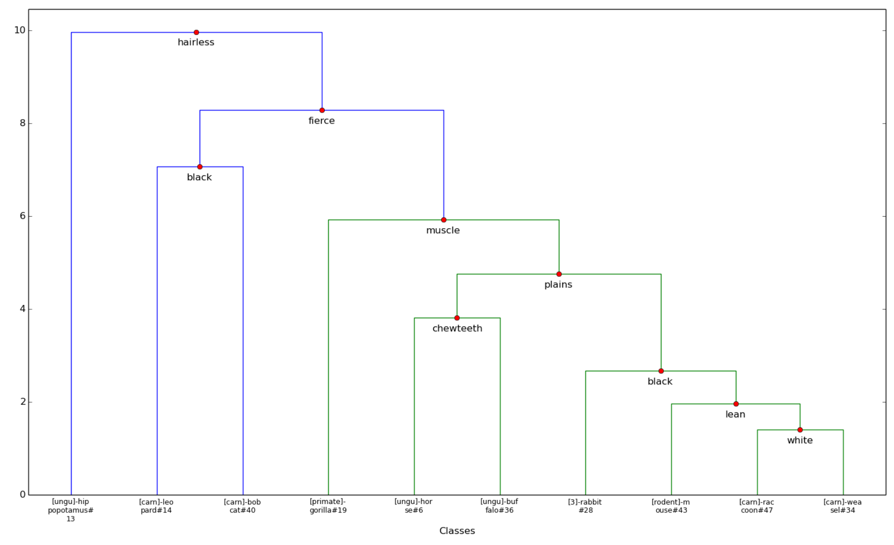


Figure 2.9. Example dendrogram for a Interpretable hierarchy

Chapter 3

Interaction with Triplet Constraints

We obtain the initial hierarchy using spectral clustering as described in Section 1.4. We try to improve the hierarchy by providing feedback. In this chapter, we look at providing feedback in the form of triplet constraints.

Triplet constraints are a natural way to interact with the hierarchy. Data points from same class are brought together and split away from those belonging to different classes. Since constraints are applied at every level, the hierarchy is corrected right from the topmost level.

3.1 Merging nodes

The constraints added to the hierarchy are to be strictly enforced (i.e., hard constraints). We merge nodes based on a simple criterion to enforce the constraints added through interactions. By merging a set of data points into one node, the merged node is treated as a single data point. Data points in the merged node are not be split from each other in that iteration.

By merging relevant data points into one node at every level of hierarchy building, we ensure that the constraints are not violated at each level. Figure 3.1 shows examples of merging required for different sets of constraints.

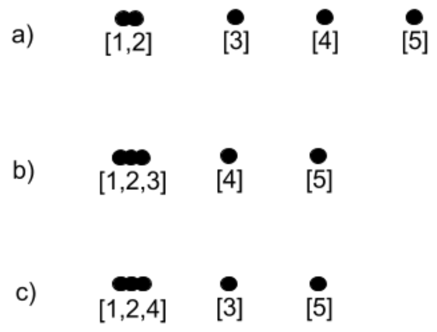


Figure 3.1. a) Merging nodes to enforce constraint [(1,2),3] b) Merging nodes to enforce constraints [(1,2),3] and [(2,3),4]. c) Merging nodes to enforce constraints [(1,2),3] and [(2,4),5]

3.1.1 Algorithm

Let $M = \{m_1, m_2, \ldots, m_n\}$ represent a node in the tree. Each m_i is a set containing one single data point. M should now be split into child clusters. Before splitting M we merge data points in M to reflect the constraints added to the hierarchy. We start the merge algorithm with M and the constraint set C.

Algorithm 4. Merge Algorithm

```
1: for constraint $[c_1, c_2, c_3]$ in C do
2:     if $c_1 \notin M$ or $c_2 \notin M$ or $c_3 \notin M$ then
3:         continue
4:     $A = \{m_i \mid c_1 \in m_i, m_i \in M\}$
5:     $B = \{m_i \mid c_2 \in m_i, m_i \in M\}$
6:     $A = A \cup B$
7:     $M = M - B$
8: return $M$
```

Figure 3.2 has visualization of the *merge algorithm* on a small set of nodes $\{1,2,3,4,5\}$

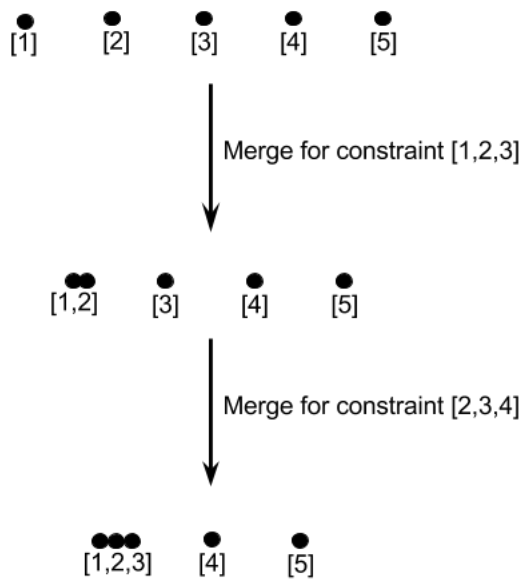


Figure 3.2. Merging the nodes when the constraints are [1,2,3] and [2,3,4]

3.1.2 Updating similarity matrix

After the data points are merged, we need to update the similarity matrix to reflect the merge process. There are two ways to update the similarity matrix.

1. Let S be the old similarity matrix, $x_1, x_2, \ldots, x_n$ be the nodes before merging, and $y_1, y_2, \ldots, y_m$ be the nodes after merging. The size of new similarity matrix is m. Let $y_i = \{x_{p_1}, x_{p_1}, \ldots, x_{p_k}\}$ and $y_j = \{x_{q_1}, x_{q_1}, \ldots, x_{q_l}\}$. An entry in the new similarity matrix S'_{ij} is updated as follows:

$$S'_{ij} = \sum_{k'=1}^{k} \sum_{l'=1}^{l} S_{p_{k'} q_{l'}}$$

2. Let $x_1, x_2, ..., x_n$ be the nodes before merging, and $y_1, y_2, .., y_m$ be the nodes after merging. In this case, we keep the size of similarity matrix the same as earlier. But we update the entries in the similarity matrix to reflect the constraints. Let

$y_i = \{x_{p_1}, x_{p_1}, .., x_{p_k}\}$ and $y_j = \{x_{q_1}, x_{q_1}, .., x_{q_l}\}$.

$$S'_{ij} = S_{ij} \times M, \quad x_{p_i} \in y_k \text{ and } x_{p_j} \in y_k, \quad \forall i, j, k$$

3.1.3 Generation of constraints

In practice, we ask users for feedback by adding constraints to the hierarchy. To simulate this behavior, we generate constraints artificially. We want the constraint chosen to be close to a constraint that would be chosen by a user. To mimic the behavior, we choose constraints such that data points from the same class are separated from data points of different class.

Algorithm to generate constraints is described in Merge Algorithm 5:

Algorithm 5. Generate Constraints

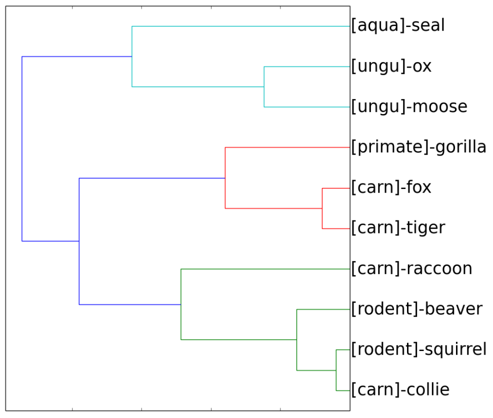
```
1: while no constraint is chosen do
2:     Randomly pick a class $c_1$ from the classes to which data belongs.
3:     Pick two data points belonging to $c_1$ : $d_1, d_2$.
4:     Pick another data $d_3$ point belonging to a different class $c_2$ != $c_1$.
5:     constraint $C = [(d_1, d_2), d_3]$.
6:     if constraint $C$ is violated in hierarchy obtained in the previous iteration then
7:         return C
```

3.2 Algorithm - triplet constraints

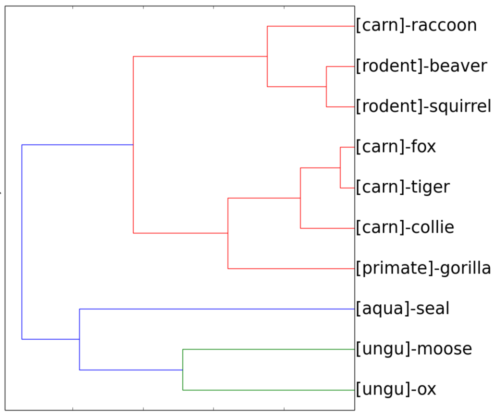
Algorithm 6 details all the steps of hierarchy interaction - merging data points, updating similarity matrix, generation of constraints, splitting nodes using spectral clustering, constructing the hierarchy etc. Input to algorithm is the feature matrix $F_{n \times f}$. Figure 3.3 shows a partial hierarchy for the animals dataset before and after applying the constraint [*(tiger, collie), gorilla*].

Algorithm 6. Triplet Constraint Algorithm

```
1: Calculate $S_{n\times n}$ = similarity matrix from $F$ (Section 1.3).
2: Initialize the constraints list $C$ = [].
3: **while** all required constraints are added **do**
4:     **while** no node has more than one data point **do**
5:         $C_1, C_2$ = clusters obtained by applying spectral clustering algorithm on $S$.
6:         Construct similarity matrices $S_1, S_2$ for each of the clusters.
7:         Merge nodes as per constraints (Algorithm 4)
8:         Update $S_1$ and $S_2$ to reflect the constraints in $C$.
9:         recursively apply spectral clustering on $S_1, S_2$.
10:    $c$ = generate a constraint using Algorithm 5
11:    $C = C \cup c$.
12: Obtain the hierarchy $H$.
13: (optional) Visualize the hierarchy.
```



(a) Before adding constraint



(b) After adding constraint

Figure 3.3. Before and after adding constraint [(tiger, collie), gorilla]

Batch Constraints

User interacts with the hierarchy, one constraint at a time. The user observes the obtained hierarchy and then decides on the constraints to add. To speed up the process, in step 9 of Algorithm 6, we generate more than one constraint. The results we discuss are from the batch constraint variation of the algorithm.

Complexity

Since we are constructing a binary hierarchy, we compute the first 2 eigen vectors. The run time of this step is $O(n^2)$where n is the size of the matrix (# of nodes at that level). Best case running time for the algorithm is $O(n^2)$ where nodes are divided in half at every level. Worst case running time is $O(n^3)$, when only one data point is separated from the rest at each level. The total number of constraints that can be added is $H^*_{triples}$. So, in the worst case, the interaction can take upto $O(n^3 H^*_{triples})$

3.3 Experiments

3.3.1 Animals Dataset

Binary features:

Figure 3.4 shows hierarchy of the animals dataset after applying 80 constraints on the hierarchy using binary feature data. In this experiment, the constraint batch size is 5. As seen in Figure 3.5, the triplet distance ratio decreases as we add constraints. As encircled in the hierarchy, a few constraints of the type [*(ungulate, ungulate), primate*] are violated. For example: [*(antelope, horse), gorilla*], [*(deer, horse), gorilla*] etc.

Numeric features:

In Figure 3.6, hierarchy of the animals dataset after adding 75 constraints is shown. The sub-tree circled in the hierarchy highlights the part of the hierarchy where the animals are not grouped together according to their class information. The constraints [*(gorilla,*

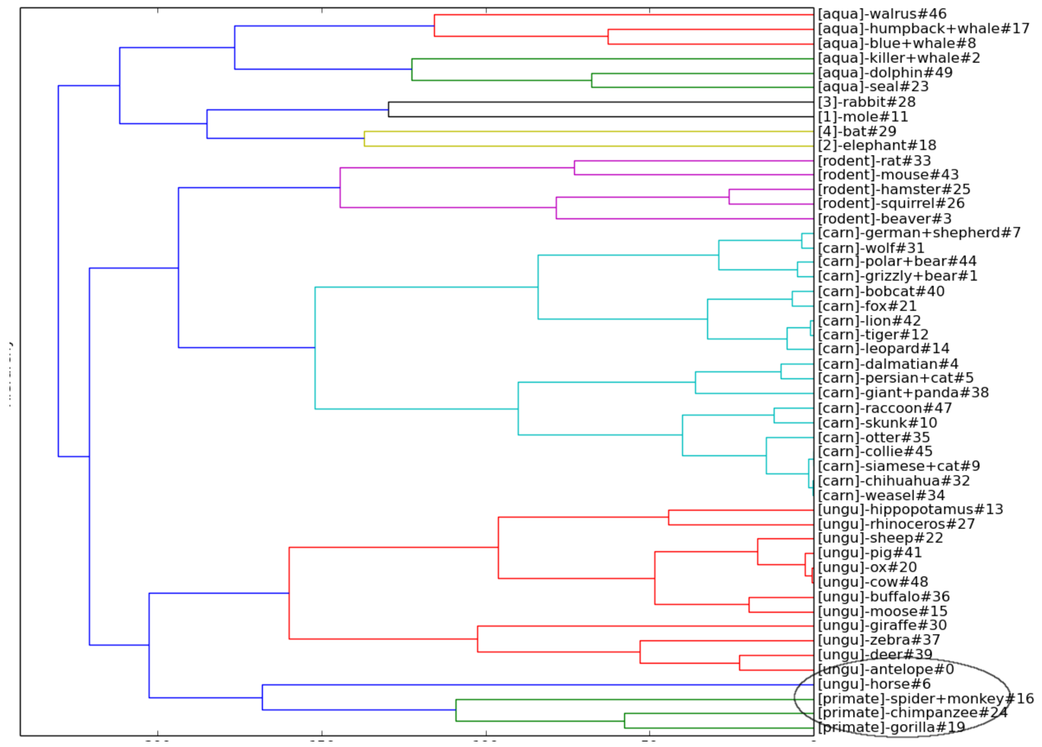


Figure 3.4. Hierarchy: Animals dataset (binary)

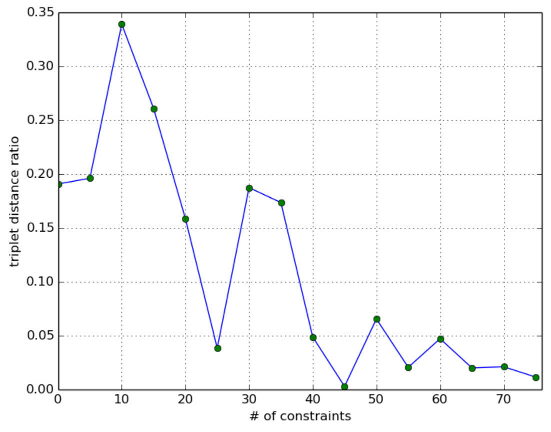


Figure 3.5. Triplet distance ratio vs # of constraints graph: Animals dataset (binary)

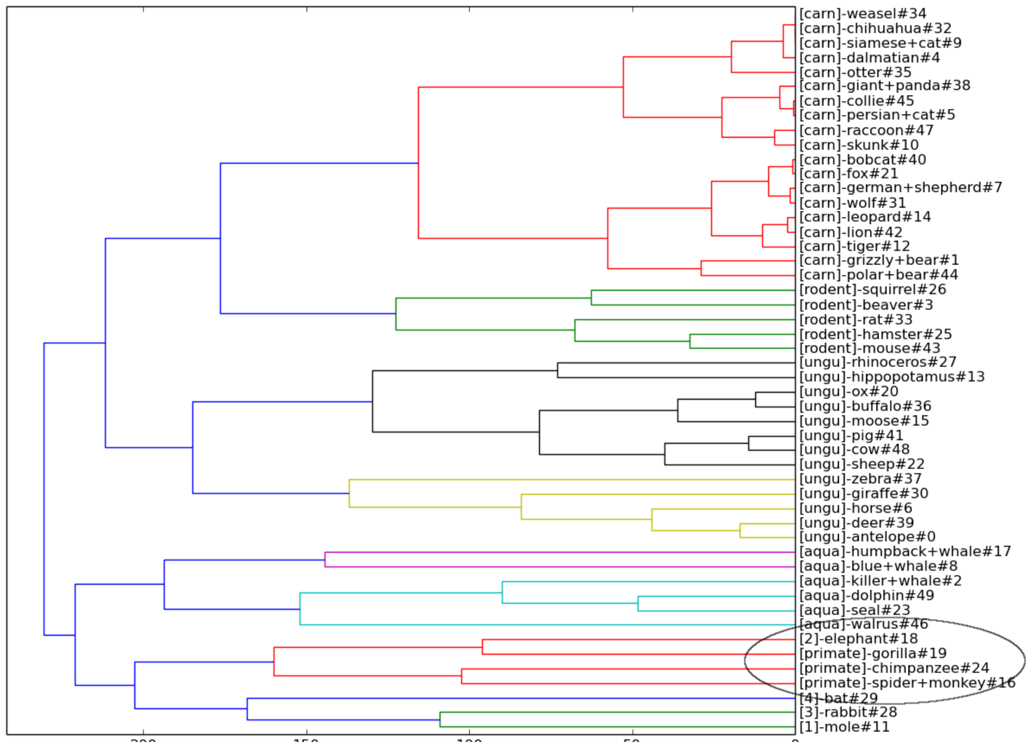


Figure 3.6. Hierarchy: Animals dataset (numeric)

chimpanzee), elephant] and [*(spider monkey, gorilla), elephant*] are violated and hence the non-zero triplet distance ratio after adding 75 constraints in Figure 3.7. It can also be noticed in Figure 3.4 and Figure 3.6 that animals belonging to the same class are grouped together in the final hierarchy.

3.3.2 MNIST dataset

Figure 3.8 shows the triplet distance ratio vs number of constraints graph on MNIST dataset for a sample of size 10000. The similarity matrix for this sample is constructed using the k-nearest neighbors method listed in Section 1.3 with $k = 1000$. As seen in the figures, the triplet distance goes down consistently as we add triplet constraints to the hierarchy. We note that the triplet distance doesn't decrease consistently during the first few iterations.

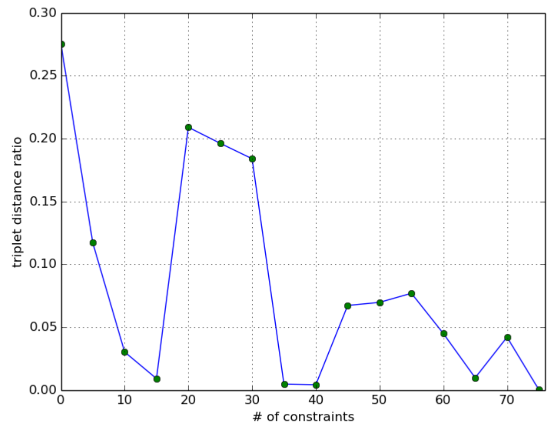


Figure 3.7. Triplet distance ratio vs # of constraints graph: Animals dataset (numeric)

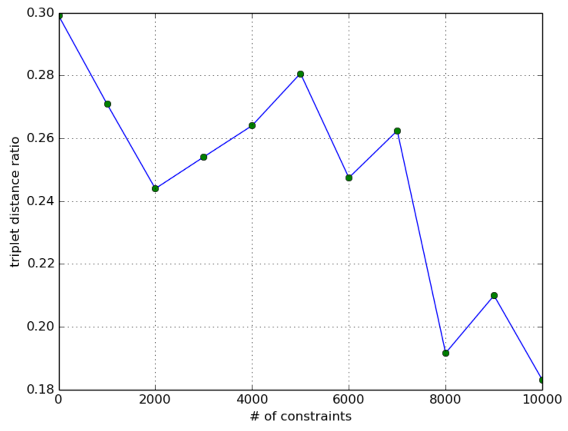


Figure 3.8. Triplet distance ratio vs # of constraints graph: MNIST dataset

Chapter 4

Other Hierarchy Interaction

In the previous chapter, we looked at triplet constraint interaction with hierarchy. In this chapter, we discuss other types of interactions using class information and triplet constraints. In Section 4.1, providing feedback as class information is discussed. In every interaction, we provide feedback of the form [data point, class]. All data points belonging to the same category (have same label) are merged into one node to enforce the constraints. We also look at a variant of triplet constraints, using the category information in addition to triplet constraints in Section 4.2

4.1 Interaction by providing class information

In this type of interaction, user provides class information for few data points. To simulate this behavior, we randomly pick a data point. The data point and its class information is added to the hierarchy. This type of interaction introduces a different form of constraints. The data points for which the class information is known and belong to the same class, they cannot be separated from each other till other data points are separated from these data points. These constraints are enforced strictly. Apart from the merging data points algorithm, the algorithm is similar to triplet interaction.

4.1.1 Merge Algorithm

Let $M = \{m_1, m_2, \ldots, m_n\}$ represent a node in the tree. Each m_i is a set containing one single data point. M should now be split into child clusters. Before splitting M we merge data points in M to reflect the constraints added to the hierarchy. We start the merge algorithm with M and the constraint set C.

Algorithm 7. Merge Algorithm

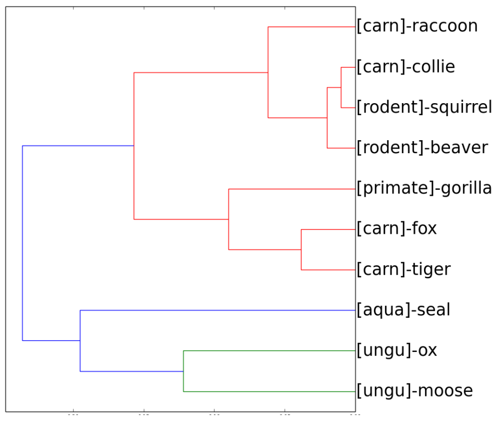
```
1: for constraint $[d, cl]$ in C do
2:     if $d \notin M$ then
3:         continue
4:     $A = \{m_i \,|\, \forall m \in m_i, m \in cl$ and $m_i \in M\}$
5:     $B = \{m_i \,|\, d \in m_i, m_i \in M\}$
6:     $A = A \cup B$
7:     $M = M - B$
8: return $M$
```

For this type of interaction, we merge all the data points which belong to the same class (according to the feedback). Figure 4.1a shows a partial hierarchy before applying any constraints.

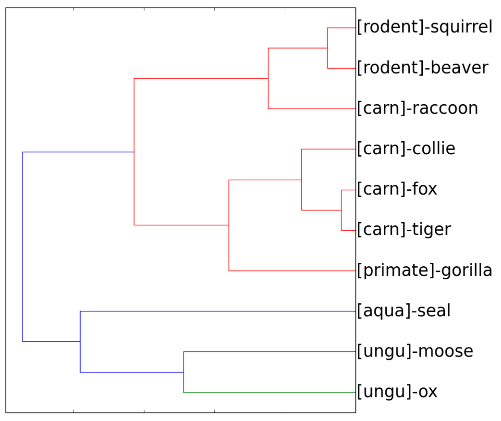
Figure 4.1b shows partial hierarchy after applying constraints on 5 animals: *beaver*(rodent), *squirrel*(rodent), *collie*(carn), *fox*(carn), *tiger*(carn). Rodent class animals : *beaver* and *squirrel* are grouped together and carnivorous animals : *collie*, *fox* and *tiger* are grouped together in the figure. The carnivorous animals (for which the class information is added) are not separated from each other till they are separated from other animals. This applies to the rodents *beaver* and *squirrel* as well. This can be clearly seen in Figure 4.1b.

4.1.2 Animals Dataset

Figure 4.2 and Figure 4.3 show the triplet distance ratio graph as we add constraints to the Animals dataset. Figure 4.2 is for binary valued attributes and Figure 4.3 is



(a) Before adding constraint



(b) After adding constraint

Figure 4.1. Before and after adding constraints for beaver, squirrel, collie, fox, tiger

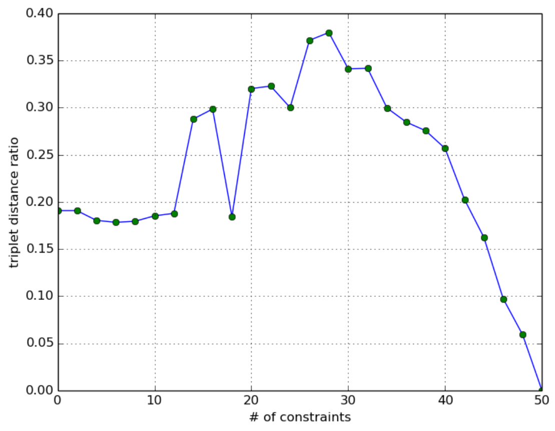


Figure 4.2. Triplet distance ratio vs # of constraints: Animals dataset (binary)

for numeric attributes.

4.1.3 MNIST dataset

Figure 4.4 plots the triplet distance ratio as we add the class information for each of the data points. In each iteration, the class information for 1000 new data points is given to the system.

As seen in the graphs listed, the triplet distance increases for the first few con-

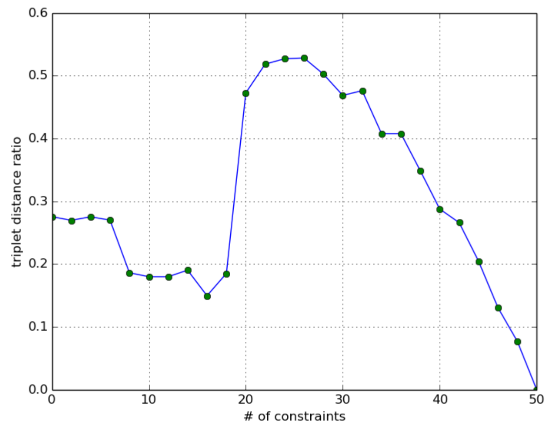


Figure 4.3. Triplet distance ratio vs # of constraints: Animals dataset (numeric)

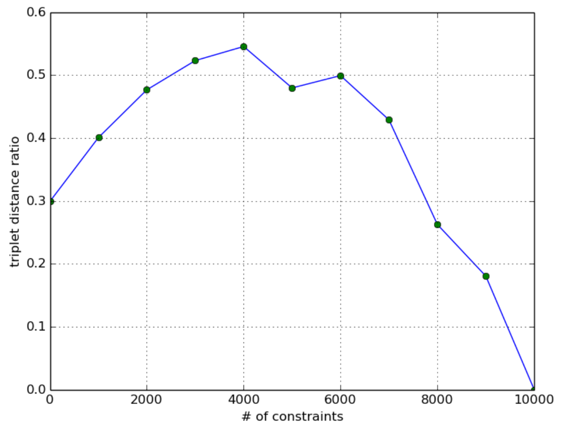


Figure 4.4. Triplet distance ratio vs # of constraints : MNIST Dataset

straints and then decreases as more class information is provided to the hierarchy. After providing class information for all the data points, the triplet distance goes down to zero. As observed in the case of triplet constraints, we don't see a steady decrease in the triplet distance. The triplet distance increases for the first few iterations and then it decreases ot zero.

4.2 Interaction with triplets and class information

These constraints are a combination of triplet constraints (Chapter 3) and the class information (Section 4.1). The feedback provided to the hierarchy is of the form $[(d_1,d_2),d_3,c_1,c_2,c_3]$, where $d_i \in c_i$, $i \leq 3$. Here d_i represents a data points and c_j represents the class to which data points belongs. Data points are first merged according the triplet constraint $[(d_1,d_2),d_3]$. They are then merged according the the provided class information.

4.2.1 Experiments

Figure 4.5 and Figure 4.6 show the triplet distance ratio graph as we add constraints to the Animals dataset. Figure 4.5 is for binary valued attributes and Figure 4.6 is for numeric valued attributes. Figure‘4.7 shows the graph for MNIST sample of 10000 images.

The quality of tree produced by using triplet+class information is not better than the one produced by using just the triplet constraints. This can be seen from the triplet distance ratio graphs for Animals dataset and MNIST dataset. In the case of MNIST dataset, the triplet distance is higher than what we started with, after adding about 10000 constraints.

We expected a stable decrease in the triplet distance as constraints are added, but these type of constraints are not very helpful as seen from the results.

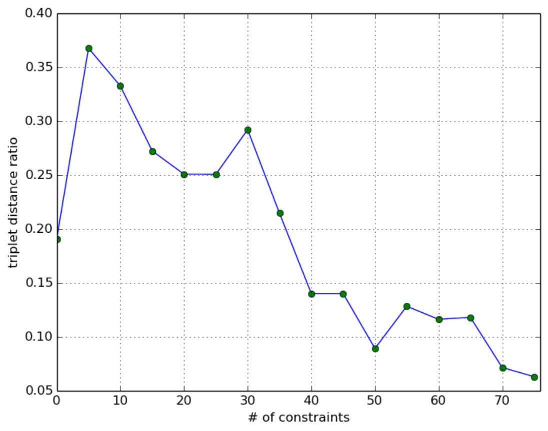


Figure 4.5. Triplet distance ratio vs # of constraints: Animals dataset (binary)

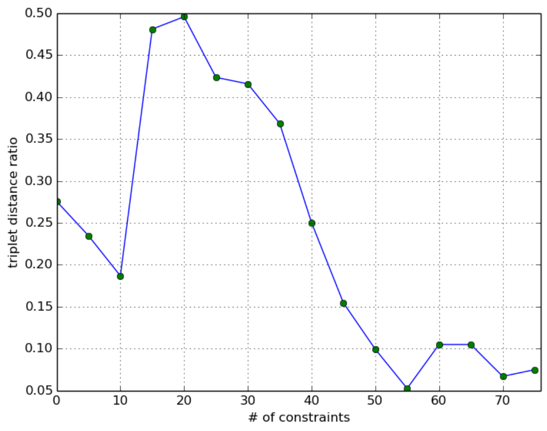


Figure 4.6. Triplet distance ratio vs # of constraints: Animals dataset (numeric)

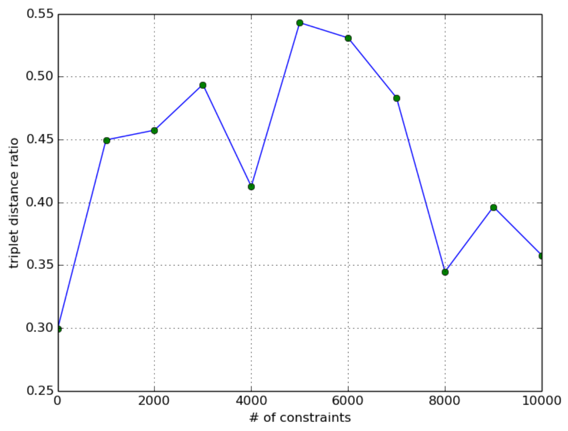


Figure 4.7. Triplet distance ratio vs # of constraints: MNIST dataset

Chapter 5

Interpretable Hierarchical Clustering

In previous chapters, we looked at spectral partitional methods for hierarchical clustering and various ways of interacting with the hierarchy. In this chapter, we will discuss methods to construct the hierarchical clustering in a top-down approach which is also interpretable. At each level of the hierarchy, the clusters are split into two child clusters. Each binary split in the interpretable hierarchy is guided by a single feature and a threshold on feature value of data points in the parent cluster. The clusters thus formed are interpretable, can be described in terms of the features. For example, cluster with features $\{A, B, C\}$ and without $\{D, E\}$; cluster with feature A $> V_A$ and feature B $< V_B$ etc.

We will look at Animals with Attributes dataset and Mushrooms dataset [22] which have meaningful attributes. We discuss the construction on binary attribute values (Animals with Attributes data set), numeric attribute values (Animals with Attributes dataset), categorical attributes (Mushrooms dataset). We also discuss interaction with the hierarchy using triplet constraints and results of these interactions on the hierarchy.

5.1 Binary Valued Data

We are given a dataset with binary valued attributes. We start with a cluster of all data points and recursively partition the data to obtain the hierarchy. We iterate through possible partitions, and we choose that split which results in the lowest split cost.

Algorithm 8

```
1: while Clusters have more than one data point do
2:     prev-cost = $\infty$
3:     for $f$ in features do
4:         $S_1$ = data points with $f$
5:         $S_2$ = data points without $f$
6:         Calculate the cost for th split $(S_1, S_2)$
               $$cost = \frac{w(S_1,S_2)}{|S_1||S_2|}.$$
7:         if $cost \leq$ prev-cost then
8:             save $S_1$ and $S_2$
9:             prev-cost = $cost$
10:    choose recently saved $(S_1, S_2)$ as the split for this level
```

Complexity

The inner for loop at step 3 is $O(nf)$, where n is the number of data points in parent cluster and f is the number of features in the data. If the data is split into half at each level, the running time of this algorithm is $O(fn\log n)$. If only one data point is split away at each level, the running time is $O(n^2 f)$.

Mushrooms dataset

We also look at Mushrooms dataset for the Interpretable hierarchical clustering case. The dataset [22] includes descriptions of hypothetical samples to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. The data points are classified into two classes - *edible* and *poisonous*. Each data point has 22 features each with a few discrete values. For eg.,

1. cap-shape: bell=b,conical=c,convex=x,flat=f,knobbed=k,sunken=s

2. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s

3. gill-spacing: close=c,crowded=w,distant=d

The dataset contains 3916 instances belonging to poisonous class and 4208 instances belonging to the edible class.

Similar to the case of binary data, we iterate through the attributes and all combinations of the attribute values. For example, for the attribute *gill-spacing*, a possible split is all the nodes with feature value *close* and *crowded* into one set and with feature value *distant* into another. All possible splits to be examined for *giil-spacing* attribute are listed below.

1. [close, crowded], [distant]

2. [close, distant], [crowded]

3. [distant, crowded], [close]

5.2 Numerical Data

We are given a dataset with continuous valued attributes. We start with a cluster of all data points and recursively partition the data to obtain the hierarchy. We iterate through the possible partitions, and we choose that split which results in the lowest cost function value.

Algorithm 9

```
1:  while Clusters have more than one data point do
2:      prev-cost = $\infty$
3:      for $d$ in data points in parent cluster do
4:          for $f$ in features do
5:              $t$ = value of feature $f$ for data point $d$
6:              $S_1$ = data points with value of feature $f \geq t$
7:              $S_2$ = data points with value of feature $f < t$
8:              Calculate the $cost$ for the split $(S_1, S_2)$

                    $$cost = \frac{w(S_1, S_2)}{|S_1||S_2|}.$$

9:              if $cost \leq$ prev-cost then
10:                 save $S_1$ and $S_2$
11:                 prev-cost = $cost$
12:     choose recently saved $(S_1, S_2)$ as the split for this level
```

Complexity

The inner for loop at step 3 is $O(n^2 f)$, where n is the number of data points in parent cluster and f is the number of features in the data. If the data is split into half at each level, the running time of this algorithm is $O(fn^2)$. If only one data point is split away at each level, the running time is $O(n^3 f)$.

5.3 Experiments

Hierarchy of Animals dataset using binary data:

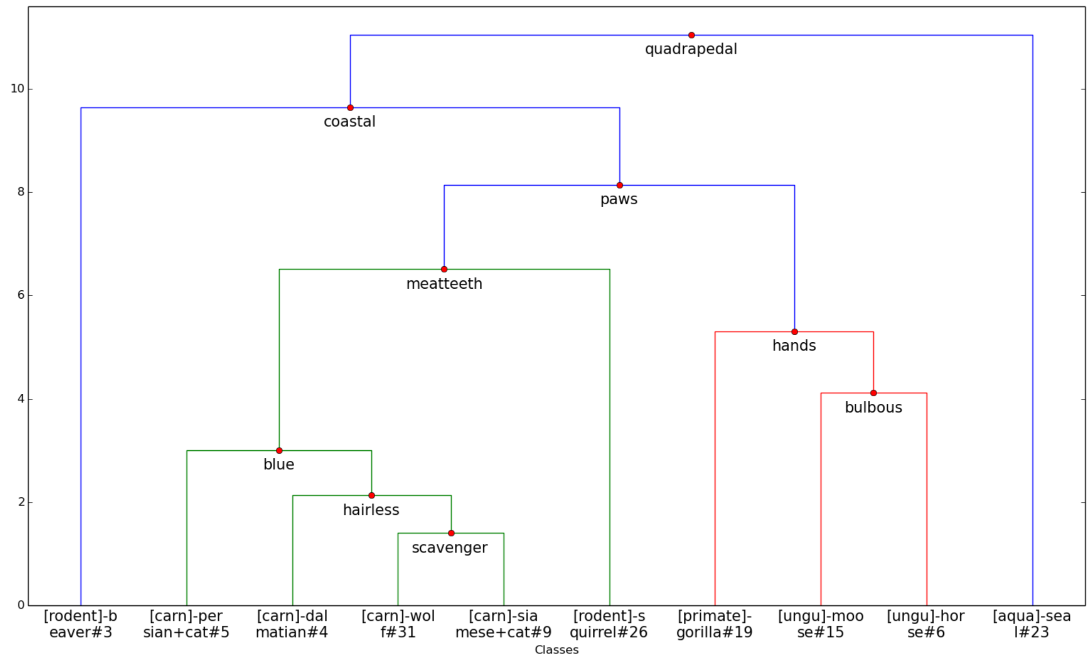


Figure 5.1. Animals dataset - hierarchy - triplet distance ratio = 0.1607

Figure 5.1 shows a partial hierarchy of animals obtained by applying the algorithm 8 on the binary valued data of animals dataset.The first feature upon which the data is split is whether or not an animal is a *quadrapedal*. The animals on the left side of the split are all 4-legged animals and the one on the right is not. Similarly, beaver is a *coastal* animal and the rest 8 are not. The resultant hierarchy is at a triplet distance of 0.1607 from the optimal hierarchy shown in Figure 1.3.

Hierarchy of Animals dataset using numeric data:

Figure 5.2 shows the resultant partial hierarchy when the algorithm 9 is applied on the continuous feature valued animals dataset. The first feature *flippers* differentiates the animals with and without flippers. The animals on the right have *flipper* feature value as 0 and the animals on the right have a very high feature value (around 75). Feature *fierce* separates leopard from the other animals based on dividing criterion used in the algorithm.

We note that the triplet distance is higher for the hierarchy obtained in the numeric case compared to when we use only the binary data.

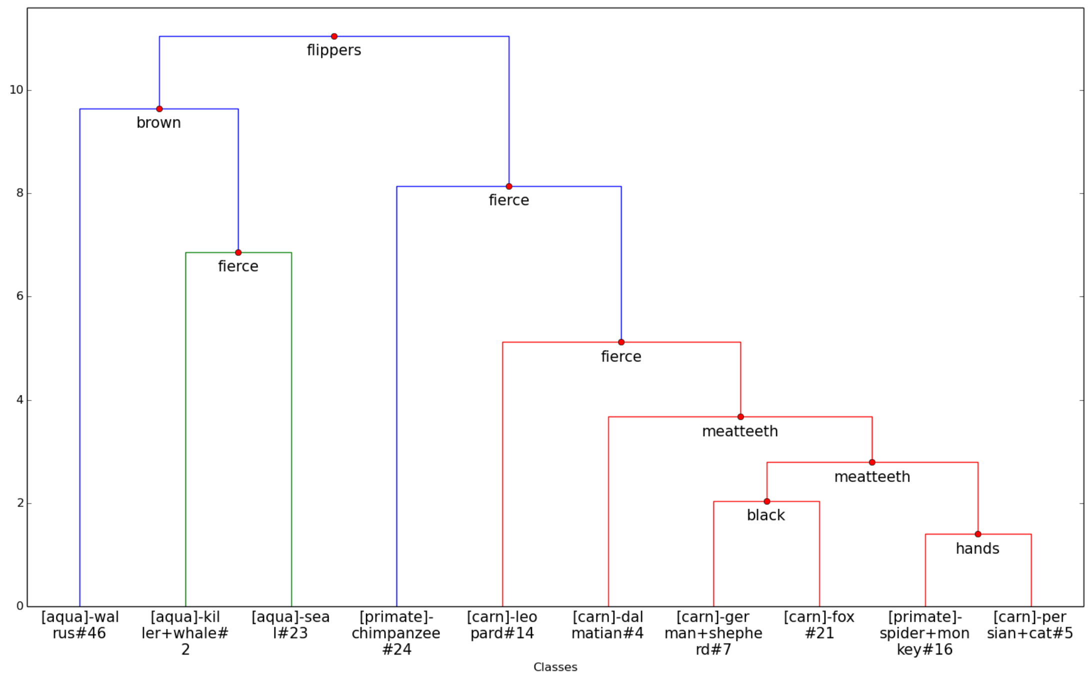


Figure 5.2. Animals dataset - hierarchy - triplet distance ratio = 0.4391

Hierarchy of Mushrooms dataset:

Figure 5.3 shows the hierarchy and the features that are used to split for a sample of 15 data points.

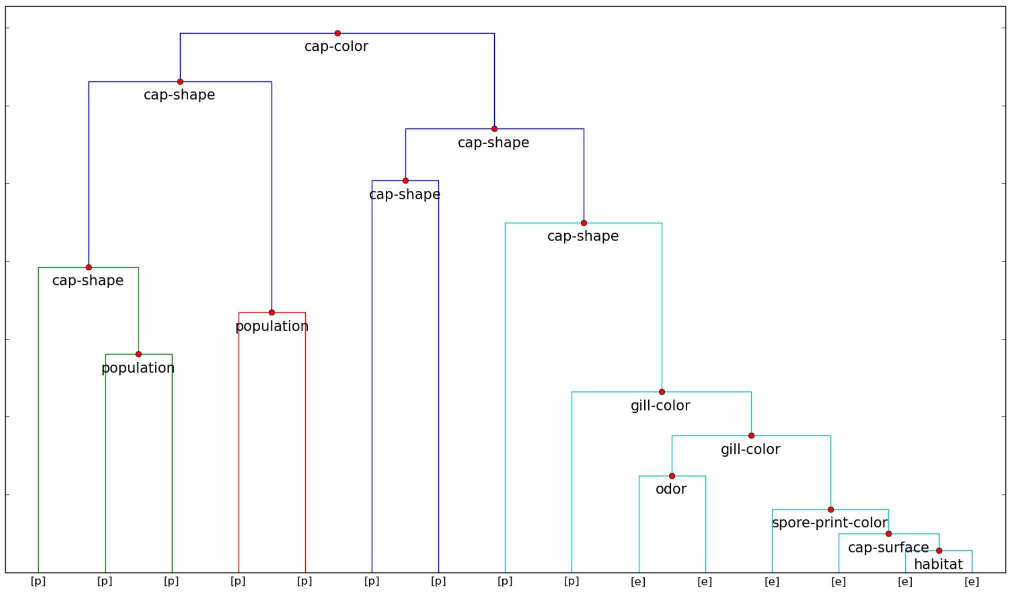


Figure 5.3. Mushrooms dataset - hierarchy - triplet distance ratio = 0.6139

5.4 Interaction

We interact with the Interpretable hierarchy through triplet constraints discussed in Section 2.1. As stated in the algorithms 8 and 9, we iterate through all the possible splits and choose that split which results in the lowest cost function value at that level. Through the process of interaction, we add constraints to the hierarchy. These constraints are hard constraints. To enforce the constraints, at each level, we choose the split of nodes with lowest cost function value among the splits which respect the constraints.

Let $s_1, s_2, s_3, ..., s_n$ be all possible splits sorted in the order of the split cost at a given level of hierarchy. We choose the first s_p in the list which respects all the constraints in the constraint list. If there is no split which respects all the constraints, we choose s_1 (This case has not occurred in our experiments).

Figure 5.4 shows a partial hierarchy of the animals dataset before any constraint is applied. After applying the constraint $[ox, cow, fox]$, the hierarchy changes to reflect that constraint, which can be seen in Figure 5.5. This can be seen from the highlighted boxes in both the figures. In Figure 5.4, the animals *ox* and *cow* are split at the first level, but in Figure 5.5 the animals *ox* and *cow* are split from *fox* in the first level, in the desired way.

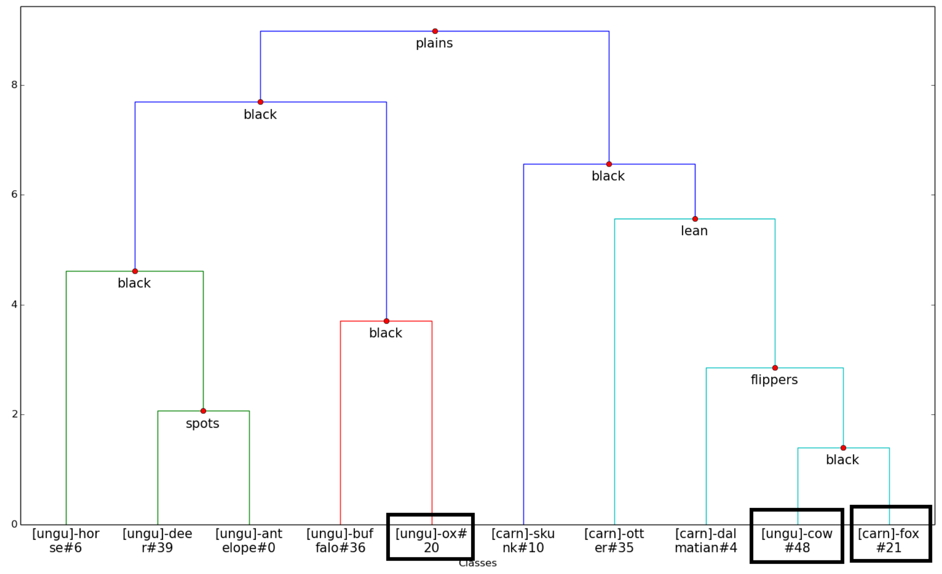


Figure 5.4. Before applying [ox, cow, fox] constraint

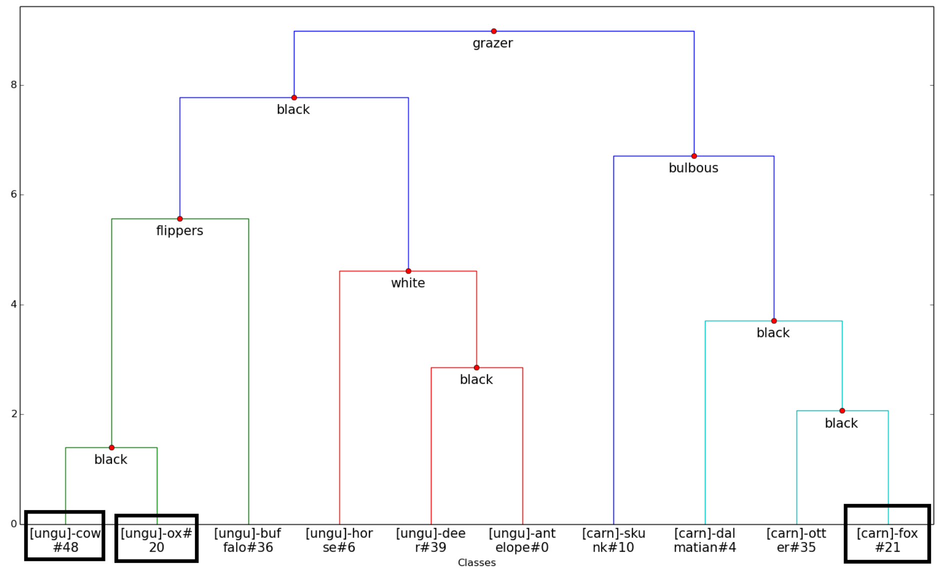


Figure 5.5. After applying [ox, cow, fox] constraint

Animals Dataset using Binary data:

Figure 5.6 shows the final interpretable hierarchy of animals dataset using binary data. The highlighted boxes show that the animals belonging to classes *aqua*, *ungulate* and *carnivore* are grouped together in the hierarchy. Figure 5.7 shows a consistent decrease in triplet distance as we add constraints to the hierarchy. The ratio goes down to 0.05 on adding 7 constraints and remains in the range even after adding about 20 constraints.

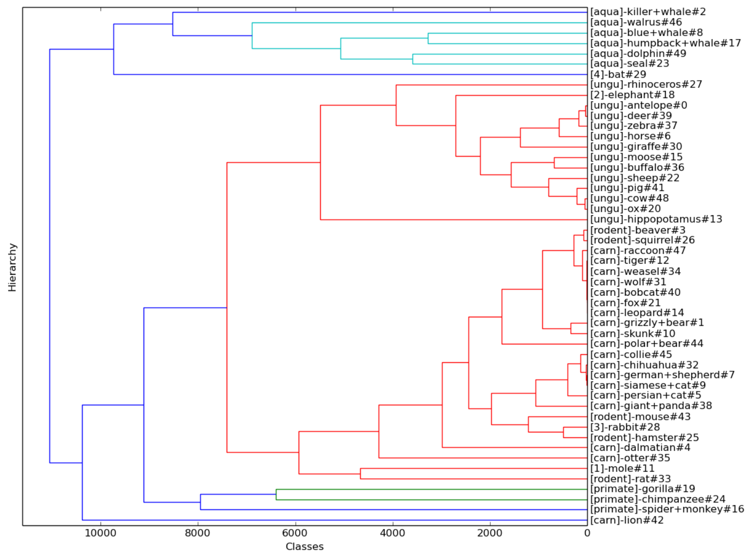


Figure 5.6. Hierarchy: Animals dataset (binary)

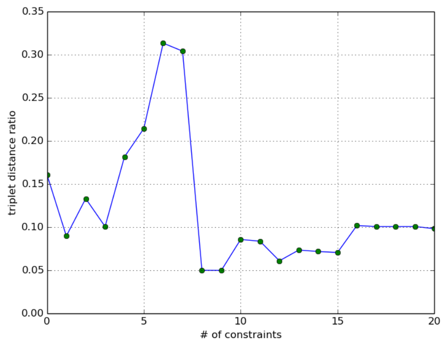


Figure 5.7. Triplet distance ratio vs # of constraints: Animals dataset (binary)

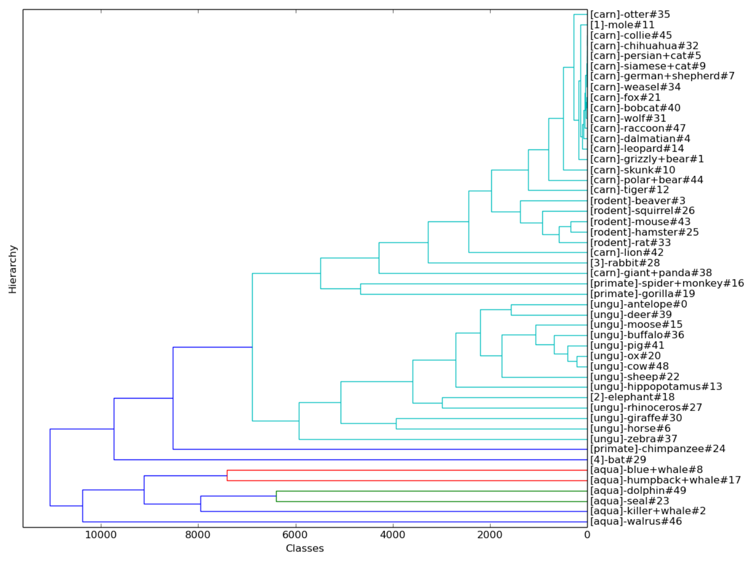


Figure 5.8. Hierarchy: Animals dataset (numeric)

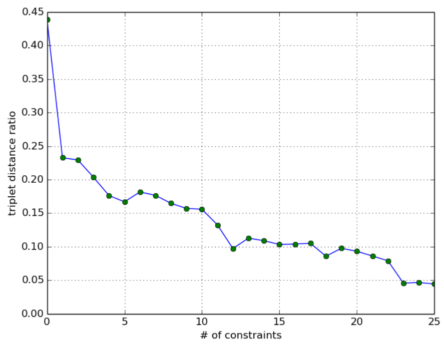


Figure 5.9. Triplet distance ratio vs # of constraints: Animals dataset (numeric)

Animals dataset using Numeric data:

Figure 5.8 shows the final interpretable hierarchy of animals dataset using numeric data. The highlighted boxes show that the animals belonging to classes *carnivore*, *ungulate* and *aqua* are grouped together in the hierarchy. Figure 5.9 shows the triplet distance ratio decreasing as we add upto 24 constraints.

Mushrooms dataset

A small sample of 100 data points of the Mushrooms dataset is used for this experiment. Figure 5.10 shows the decrease in triplet distance as we add constraints to the hierarchy. The triplet distance goes to 0 after adding 10 constraints. This shows that after adding 10 constraints the splits in the hierarchy are now perfectly split and align with the optimal tree.

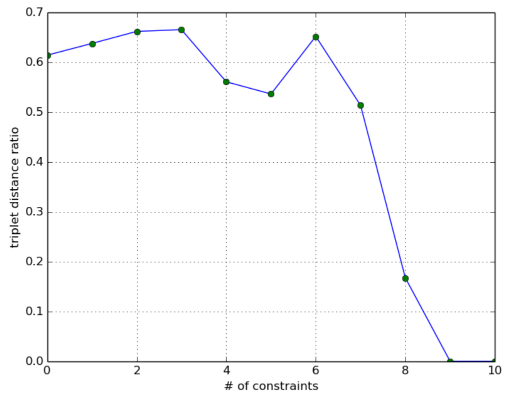


Figure 5.10. Triplet distance ratio vs # of constraints: Mushrooms dataset

From the triplet distance ratio graphs, we see that interpretable hierarchy achieves a better hierarchy in fewer iterations compared to interaction using spectral methods.

Chapter 6

Conclusion and Future work

In this thesis, we discussed interaction methods to improve the hierarchical clustering. We explored providing feedback using triplet constraints, by adding class information and triplet constraints along with class information. We also looked into interpretable hierarchical clustering and interacting with the hierarchy using triplet constraints. We experimented with datasets with various feature values - binary valued data (*Animals*), numeric data (*Animals*,*MNIST*), discrete valued data (*Mushrooms*). Triplet distance metric is introduced to measure the quality of hierarchy. We observed this metric as we interact with hierarchies of different datasets with different types of constraints. The results are summarized in Table 6.1 and 6.2.

In Table 6.1, the highlighted entries are results of applying spectral clustering without any interaction on Animals and MNIST datasets. These are followed by results after interacting with different types of constraints on the dataset. In Table 6.2, the triplet distance ratio before and after adding the noted number of constraints is listed for Animals and Mushrooms dataset (using Interpretable hierarchical clustering). The triplet distance ratio decreased as we added constraints to the hierarchy for the datasets we tested on, and for various types of constraints that are added. From the results, we can see that a triplet distance ratio ≤ 0.1 is achieved by adding just $\sim$25 constraints using Interpretable Hierarchy where as $\sim$75 constraints were needed for those using spectral methods for Animals dataset. Interpretable Hierararchical Clustering seems to achieve a good quality hierarchy after adding fewer constraints comparatively.

Table 6.1. Summary of Hierarchical Clustering (using spectral methods)

Dataset	Interaction method	# of constraints	triplet ratio
Animals(binary)	-	-	**0.19**
Animals(binary)	triplet	75	0.01
Animals(binary)	label	50	0
Animals(binary)	triplet+label	75	0.06
Animals(numeric)	-	-	**0.27**
Animals(numeric)	triplet	75	0.01
Animals(numeric)	label	50	0
Animals(numeric)	triplet+label	75	0.07
MNIST	-	-	**0.29**
MNIST	triplet	10000	0.18
MNIST	label	10000	0
MNIST	triplet+label	10000	0.35

Table 6.2. Summary of Interpretable Hierarchical Clustering

Dataset	initial hierarchy	# of constraints	final hierarchy
Animals(binary)	0.16	20	0.09
Animals(numeric)	0.44	25	0.04
Mushrooms	0.61	9	0

In all the discussed scenarios, we expect a steady decrease in triplet distance ratio with interactions. As per our experiments, it takes a few iterations before the triplet distance decreases consistently. This might be a result of random ordering of constraints, unstable grouping of data points during the first few iterations etc.

Hierarchical clustering methods, introduced in 1960s, is still a challenging problem. It has wide applications in fields like information retrieval ([10], [11]), protein sequence analysis ([5], [14]), chemoinformatics [20] etc. These methods are greedy in nature and proceed through each level of clustering without any backtracking. Hierarchies produced by such algorithms can be improved through interaction. This is evident from our experiments on various datasets using different types of constraints as summarized in Tables 6.1 and 6.2. This work is a step towards the goal of producing better clustering through feedback/interaction. Here are a few interesting questions to pursue as a part of future work:

- We generate and add random constraints to the hierarchy. Can we improve the convergence rate of triplet distance ratio vs # of constraints graph by adding better constraints? What is the most useful constraint? Is there an ordering of constraints which will produce a good hierarchy in fewer iterations?
- We explored triplet constraints with two points belonging to the same class. What is the behavior when points belong to different classes but are meaningful?
- Currently, we restart the process of hierarchy construction on adding the constraint. Can we interact with the hierarchy such that only a sub-tree of hierarchy is to be manipulated? This will improve the running time of the algorithm.
- Bounds on running time to produce a good hierarchy - this includes the loop of hierarchy construction and addition the constraints.

Bibliography

[1] Alfred V. Aho, Yehoshua Sagiv, Thomas G. Szymanski, and Jeffrey D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

[2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

[3] James C Bezdek, Chris Coray, Robert Gunderson, and James Watson. Detection and characterization of cluster substructure i. linear structure: Fuzzy c-lines. *SIAM Journal on Applied Mathematics*, 40(2):339–357, 1981.

[4] Gerth Stølting Brodal, Rolf Fagerberg, Thomas Mailund, Christian NS Pedersen, and Andreas Sand. Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1814–1832. SIAM, 2013.

[5] Florence Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, 16(22):10881–10890, 1988.

[6] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. *arXiv preprint arXiv:1510.05043*, 2015.

[7] Renato Cordeiro de Amorim. Constrained clustering with minkowski weighted k-means. In *Computational Intelligence and Informatics (CINTI), 2012 IEEE 13th International Symposium on*, pages 13–17. IEEE, 2012.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[9] Kristen Grauman, Fei Sha, and Sung Ju Hwang. Learning a tree of metrics with disjoint visual features. In *Advances in neural information processing systems*, pages 621–629, 2011.

[10] Alan Griffiths, Lesley A Robinson, and Peter Willett. Hierarchic agglomerative clustering methods for automatic document classification. *Journal of Documentation*, 40(3):175–205, 1984.

[11] Nick Jardine and Cornelis Joost van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information storage and retrieval*, 7(5):217–240, 1971.

[12] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[13] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[14] Antje Krause, Jens Stoye, and Martin Vingron. Large scale hierarchical clustering of protein sequences. *BMC bioinformatics*, 6(1):1, 2005.

[15] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

[16] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[17] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[18] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[19] Fionn Murtagh and Pedro Contreras. Methods of hierarchical clustering. *arXiv preprint arXiv:1105.0121*, 2011.

[20] Fionn Murtagh, Geoff Downs, and Pedro Contreras. Hierarchical clustering of massive, high dimensional data sets by exploiting ultrametric embedding. *SIAM Journal on Scientific Computing*, 30(2):707–730, 2008.

[21] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[22] Jeffrey Curtis Schlimmer. Concept acquisition through representational adjustment. 1987.

[23] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[24] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[25] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097, 2000.

[26] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[27] Stella X Yu and Jianbo Shi. Multiclass spectral clustering. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 313–319. IEEE, 2003.

[28] Stella X Yu and Jianbo Shi. Segmentation given partial grouping constraints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):173–183, 2004.