Hierarchical Clustering

Ryan P. Adams
COS 324 – Elements of Machine Learning
Princeton University

K-Means clustering is a good general-purpose way to think about discovering groups in data, but there are several aspects of it that are unsatisfying. For one, it requires the user to specify the number of clusters in advance, or to perform some kind of *post hoc* selection. For another, the notion of what forms a group is very simple: a datum belongs to cluster k if it is closer to the kth center than it is to any other center. Third, K-Means is nondeterministic; the solution it finds will depend on the initialization and even good initialization algorithms such as K-Means++ have a randomized aspect. Finally, we might reasonably think that our data are more complicated than can be described by simple partitions. For example, we partition organisms into different species, but science has also developed a rich taxonomy of living things: kingdom, phylum, class, etc. **Hierarchical clustering** is one framework for thinking about how to address these shortcomings.

Hierarchical clustering constructs a (usually binary) tree over the data. The leaves are individual data items, while the root is a single cluster that contains all of the data. Between the root and the leaves are intermediate clusters that contain subsets of the data. The main idea of hierarchical clustering is to make "clusters of clusters" going upwards to construct a tree. There are two main conceptual approaches to forming such a tree. **Hierarchical agglomerative clustering** (HAC) starts at the bottom, with every datum in its own singleton cluster, and merges groups together. **Divisive clustering** starts with all of the data in one big group and then chops it up until every datum is in its own singleton group.

1 Agglomerative Clustering

The basic algorithm for hierarchical agglomerative clustering is shown in Algorithm 1. Essentially, this algorithm maintains an "active set" of clusters and at each stage decides which two clusters to merge. When two clusters are merged, they are each removed from the active set and their union is added to the active set. This iterates until there is only one cluster in the active set. The tree is formed by keeping track of which clusters were merged.

The clustering found by HAC can be examined in several different ways. Of particular interest is the **dendrogram**, which is a visualization that highlights the kind of exploration enabled by hierarchical clustering over flat approaches such as K-Means. A dendrogram shows data items along one axis and distances along the other axis. The dendrograms in these notes will have the data on the y-axis. A dendrogram shows a collection of $\sqsupset$ shaped paths, where the legs show

Algorithm 1 Hierarchical Agglomerative Clustering *Note: written for clarity, not efficiency.*

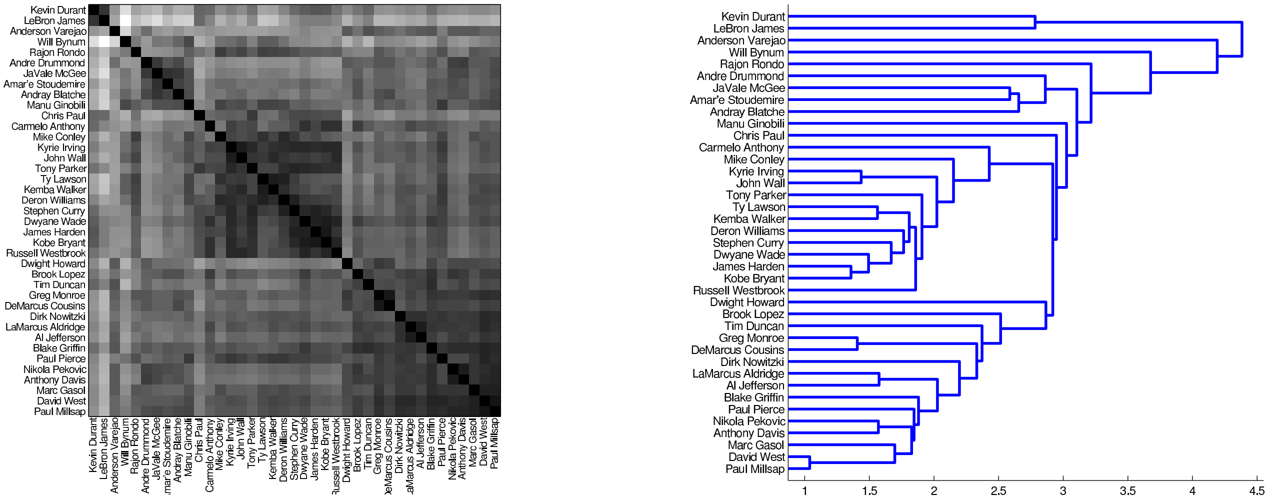
```
Input: Data vectors $\{\boldsymbol{x}_n\}_{n=1}^N$, group-wise distance $\textsc{Dist}(\mathcal{G}, \mathcal{G}')$
$\mathcal{A} \leftarrow \emptyset$                                   ▷ Active set starts out empty.
for $n \leftarrow 1 \ldots N$ do                                     ▷ Loop over the data.
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\{\boldsymbol{x}_n\}\}$          ▷ Add each datum as its own cluster.
end for
$\mathcal{T} \leftarrow \mathcal{A}$          ▷ Store the tree as a sequence of merges. In practice, pointers.
while $|\mathcal{A}| > 1$ do                       ▷ Loop until the active set only has one item.
    $\mathcal{G}_1^\star, \mathcal{G}_2^\star \leftarrow \operatorname*{arg\,min}_{\mathcal{G}_1,\mathcal{G}_2 \in \mathcal{A}\,;\,\mathcal{G}_1,\mathcal{G}_2 \in \mathcal{A}} \textsc{Dist}(\mathcal{G}_1, \mathcal{G}_2)$   ▷ Choose pair in $\mathcal{A}$ with best distance.
    $\mathcal{A} \leftarrow (\mathcal{A} \backslash \{\mathcal{G}_1^\star\}) \backslash \{\mathcal{G}_2^\star\}$      ▷ Remove each from active set.
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{G}_1^\star \cup \mathcal{G}_2^\star\}$            ▷ Add union to active set.
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{G}_1^\star \cup \mathcal{G}_2^\star\}$            ▷ Add union to tree.
end while
Return: Tree $\mathcal{T}$.
```

the groups that have been joined together. These groups may be the base of another ⊐ or may be singleton groups represented as the data along the axis. A key property of the dendrogram is that that vertical base of the ⊐ is located along the x-axis according to the distance between the two groups that are being merged. For this to result in a sensible clustering – and a valid dendrogram – these distances must be monotonically increasing. That is, the distance between two merged groups $\mathcal{G}$ and $\mathcal{G}'$ must always be greater than or equal to the distance between any of the previously-merged subgroups that formed $\mathcal{G}$ and $\mathcal{G}'$.

Figure 1b shows a dendrogram for a set of professional basketball players, based on some per-game performance statistics in the 2012-13 season. Figure 1a on the left of it shows the pairwise distance matrix that was used to compute the dendrogram. Notice how there are some distinct groups that appear as blocks in the distance matrix and as a subtree in the dendrogram. When we explore these data, we might observe that this structure seems to correspond to position; all of the players in the bottom subtree between Dwight Howard and Paul Millsap are centers or power forwards (except for Paul Pierce who is considered more of a small forward) and play near the basket. Above these is a somewhat messier subtree that contains point guards (e.g., Stephen Curry and Tony Parker) and shooting guards (e.g., Dwayne Wade and Kobe Bryant). At the top are Kevin Durant and LeBron James, as they are outliers in several categories. Anderson Varejao also appears to be an unusual player according to these data; I attribute this to him having an exceptionally large number of rebounds for a high-scoring player.

The main decision to make when using HAC is what the distance criterion[1] should be between groups – the $\textsc{Dist}(\mathcal{G}, \mathcal{G}')$ function in the pseudocode. In K-Means, we looked at distances between data items; in HAC we look at distances between *groups* of data items. Perhaps not suprisingly, there are several different ways to think about such distances. In each of the cases below, we consider the distances between two groups $\mathcal{G} = \{\boldsymbol{x}_n\}_{n=1}^N$ and $\mathcal{G}' = \{\boldsymbol{y}_m\}_{m=1}^M$, where N and M are

[1] These are not necessarily "distances" in the formal sense that they arise from a metric space. Here we'll be thinking of distances as a measure of dissimilarity.



(a) Pairwise Distances

(b) Single-Linkage Dendrogram

Figure 1: These figures demonstrate hierarchical agglomerative clustering of high-scoring professional basketball players in the NBA, based on a set of normalized features such as assists and rebounds per game, from `http://hoopdata.com/`. (a) The matrix of pairwise distances between players. Darker is more similar. The players have been ordered to highlight the block structure: power forwards and centers appear in the bottom right, point guards in the middle block, with some unusual players in the top right. (b) The dendrogram arising from HAC with the single-linkage criterion.

not necessarily the same. Figure 2 illustrates these four types of "linkages". Figures 3 and 4 show the effects of these linkages on some simple data.

The Single-Linkage Criterion: The single-linkage criterion for hierarchical clustering merges groups based on the **shortest** distance over all possible pairs. That is

$$\textsc{Dist-SingleLink}(\{\boldsymbol{x}_n\}_{n=1}^{N}, \{\boldsymbol{y}_m\}_{m=1}^{M}) = \min_{n,m} ||\boldsymbol{x}_n - \boldsymbol{y}_m||, \tag{1}$$

where $||\boldsymbol{x} - \boldsymbol{y}||$ is an appropriately chosen distance metric between data examples. See Figure 2a. This criterion merges a group with its nearest neighbor and has an interesting interpretation. Think of the data as the vertices in a graph. When we merge a group using the single-linkage criterion, add an edge between the two vertices that minimized Equation 1. As we never add an edge between two members of an existing group, we never introduce loops as we build up the graph. Ultimately, when the algorithm terminates, we have a tree. As we were adding edges at each stage that minimize the distance between groups (subject to not adding a loop), we actually end up with the tree that connects all the data but for which the sum of the edge lengths is smallest. That is, single-linkage HAC produces the *minimum spanning tree* for the data.

Eponymously, two merge two clusters with the single-linkage criterion, you just need one of the items to be nearby. This can result in "chaining" and long stringy clusters. This may be good or bad, depending on your data and your desires. Figure 4a shows an example where it seems like a good thing because it is able to capture the elongated shape of the pinwheel lobes. On the other hand, this effect can result in premature merging of clusters in the tree.

The Complete-Linkage Criterion: Rather than choosing the shortest distance, in complete-linkage clustering the distance between two groups is determined by the **largest** distance over all possible pairs, i.e.,

$$\textsc{Dist-CompleteLink}(\{\boldsymbol{x}_n\}_{n=1}^N, \{\boldsymbol{y}_m\}_{m=1}^M) = \max_{n,m} ||\boldsymbol{x}_n - \boldsymbol{y}_m||, \tag{2}$$

where again $||\boldsymbol{x} - \boldsymbol{y}||$ is an appropriate distance measure. See Figure 2b. This has the opposite of the chaining effect and prefers to make highly compact clusters, as it requires *all* of the distances to be small. Figures 3b and 4b show how this results in tighter clusters.

The Average-Linkage Criterion: Rather than the worst or best distances, when using the average-linkage criterion we average over all possible pairs between the groups:

$$\textsc{Dist-Average}(\{\boldsymbol{x}_n\}_{n=1}^N, \{\boldsymbol{y}_m\}_{m=1}^M) = \frac{1}{NM} \sum_{n=1}^N \sum_{m=1}^M ||\boldsymbol{x}_n - \boldsymbol{y}_m||. \tag{3}$$

This linkage can be thought of as a compromise between the single and complete linkage criteria. It produces compact clusters that can still have some elongated shape. See Figures 2c, 3b, and 4c.

The Centroid Criterion: Another alternative approach to computing the distance between clusters is to look at the difference between their centroids:

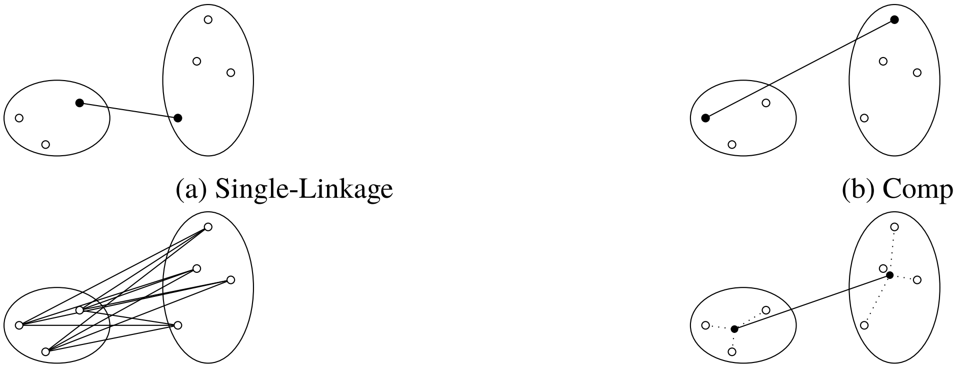
$$\textsc{Dist-Centroid}\{\boldsymbol{x}_n\}_{n=1}^N, \{\boldsymbol{y}_m\}_{m=1}^M) = || \left(\frac{1}{N} \sum_{n=1}^N \boldsymbol{x}_n \right) - \left(\frac{1}{M} \sum_{m=1}^M \boldsymbol{y}_m \right) ||. \tag{4}$$

Note that this is something that only makes sense if an average of data items is sensible; recall the motivation for K-Medoids versus K-Means. See Figure 2d, 3d and 4d.

Although this criterion is appealing when thinking of HAC as a next step beyond K-Means, it does present some difficulties. Specifically, the centroid linkage criterion breaks the assumption of monotonicity of merges and can result in an *inversion* in the dendrogram.

1.1 Discussion

Hierarchical agglomerative clustering is our first example of a *nonparametric*, or *instance-based*, machine learning method. When thinking about machine learning methods, it is useful to think about the space of possible things that can be learned from data, i.e., our *hypothesis space*. Parametric methods such as K-Means decide in advance how large this hypothesis space will be; in clustering that means how many clusters there can be and what possible shapes they can have. Nonparametric methods such as HAC allow the effective number of parameters to grow with the



(a) Single-Linkage

(b) Complete-Linkage

(c) Average-Linkage

(d) Centroid

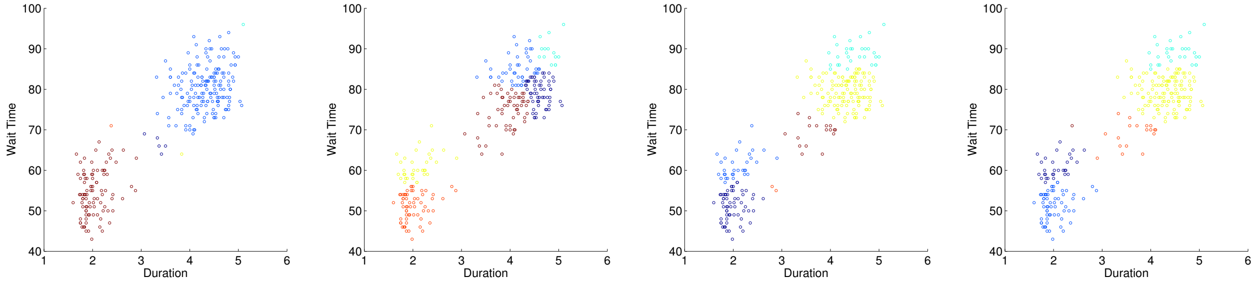
Figure 2: Four different types of linkage criteria for hierarchical agglomerative clustering (HAC). (a) Single linkage looks at minimum distance between all inter-group pairs. (b) Complete linkage looks at the maximum distance between all inter-group pairs. (c) Average linkage uses the average distance between all inter-group pairs. (d) Centroid linkage first computes the centroid of each group and then looks at the distance between them. Inspired by Figure 17.3 of Manning et al. (2008).

size of the data. This can be appealing because we have to make fewer choices when applying our algorithm. The downside of nonparametric methods is that their flexibility can increase computational complexity. Also, nonparametric methods often depend on some notion of distance in the data space and distances become less meaningful in higher dimensions. This phenomenon is known as the **curse of dimensionality** and it unfortunately comes up quite often in machine learning. There are various ways to get an intuition for this behavior. One useful way to see the curse of dimensionality is observe that squared Euclidean distances are sums over dimensions. If we have a collection of random variables, the differences in each dimension will also be random and the central limit theorem results in this distribution converging to a Gaussian. Figure 5 shows this effect in the unit hypercube for 2, 10, 100, and 1000 dimensions.

1.2 Example: Binary Features of Animals

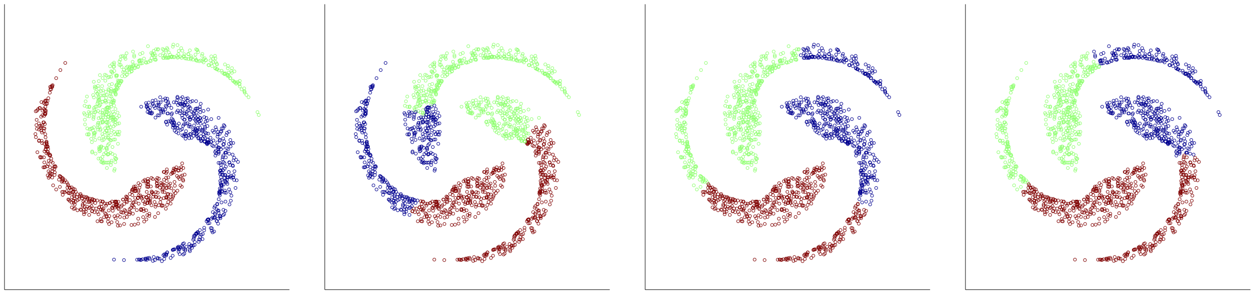
We look again at the binary feature data for animals we examined in the K-Means notes.[2] These data are 50 binary vectors, where each entry corresponds to properties of an animal. The raw data are shown as a matrix in Figure 6a, where the rows are animals such as “lion” and “german shepherd” while the features are columns such as “tall” and “jungle”. Figure 6b shows a matrix of Hamming distances in this feature space, and Figure 6c shows the resulting dendrogram using the average-link criterion. The ordering shown in Figures 6b and 6c is chosen so that there are no overlaps.

[2] `http://www.psy.cmu.edu/~ckemp/code/irm.html`



(a) Single-Linkage (b) Complete-Linkage (c) Average-Linkage (d) Centroid

Figure 3: These figures show clusterings from the four different group distance criteria, applied to the joint durations and waiting times between Old Faithful eruptions. The data were normalized before clustering. In each case, HAC was run, the tree was truncated at six groups, and these groups are shown as different colors.

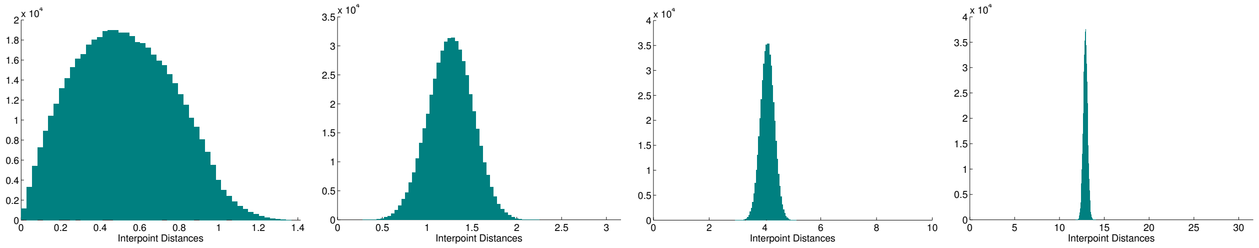


(a) Single-Linkage (b) Complete-Linkage (c) Average-Linkage (d) Centroid

Figure 4: These figures show clusterings from the four different group distance criteria, applied to 1500 synthetic "pinwheel" data. In each case, HAC was run, the tree was truncated at three groups, and these groups are shown as different colors. (a) The single-linkage criterion can give stringy clusters, so it can capture the pinwheel shapes (b-d) Complete, average, and centroid linkages try to create more compact clusters and so tend to not identify the lobes.

1.3 Example: National Governments and Demographics

These data are binary properties of 14 nations (collected in 1965), available at the same URL as the animal data of the previous section. The nations are Brazil, Burma, China, Cuba, Egypt, India, Indonesia, Israel, Jordan, the Netherlands, Poland, the USSR, the United Kingdom, and the USA. The features are various properties of the governments, social structures, and demographics. The data are shown as a binary matrix in Figure 7a, with missing data shown in gray. Figure 7b shows the matrix of pairwise distances, with darker indicating greater similarity. Figure 7c shows a dendrogram arising from the complete-linkage criterion.



(a) 2 Dimensions (b) 10 Dimensions (c) 100 Dimensions (d) 1000 Dimensions

Figure 5: Histograms of inter-point Euclidean distances for 1000 points in a unit hypercube of increasing dimensionality. Notice how the distribution concentrates relative to the minimum (zero) and maximum ($\sqrt{D}$) values. The curse of dimensionality is the idea that this concentration means differences in data will be come less meaningful as dimension increases.

1.4 Example: Voting in the Senate

These data are voting patterns of senators in the 113th United States Congress. In total there are 104 senators and 172 roll call votes. The resulting binary matrix is shown in Figure 8a, with abstentions/absences shown as grey entries. Pairwise Euclidean distances are shown in Figure 8b, with darker being more similar. Note the very clear block structure; the names have been ordered to make it clear. Figure 8c shows a dendrogram using the average-linkage criterion. There are two very large clusters apparent, which have been colored using the conventional red/blue distinction.

2 Divisive Clustering

Agglomerative clustering is a widely-used and intuitive procedure for data exploration and the construction of hierarchies. While HAC is a bottom-up procedure, *divisive clustering* is a top-down hierarchical clustering approach. It starts with all of the data in a single group and then applies a flat clustering method recursively. That is, it first divides the data into K clusters using, e.g., K-Means or K-Medoids, and then it further subdivides each of these clusters into smaller groups. This can be performed until the desired granularity is acheived or each datum belongs to a singleton cluster. One advantage of divisive clustering is that it does not require binary trees. However, it suffers from all of the difficulties and non-determinism of flat clustering, so it is less commonly used than HAC. A sketch of the divisive clustering algorithm is shown in Algorithm 2.

3 Additional Reading

- Chapter 17 of Manning et al. (2008) is freely available online and is an excellent resource.
- Duda et al. (2001) is a classic and Chapter 10 discusses these methods.

Algorithm 2 *K*-Wise Divisive Clustering *Note: written for clarity, not efficiency.*

```
Input: Data vectors $\{\boldsymbol{x}_n\}_{n=1}^N$, Flat clustering procedure FlatCluster($\mathcal{G}$, $K$)

function SubDivide($\mathcal{G}$, $K$)                        ▷ Function to call recursively.
    $\{\mathcal{H}_k\}_{k=1}^K \leftarrow$ FlatCluster($\mathcal{G}$, $K$)    ▷ Perform flat clustering of this group.
    $\mathcal{S} \leftarrow \emptyset$
    for $k \leftarrow 1 \ldots K$ do                          ▷ Loop over the resulting partitions.
        if $|\mathcal{H}_k| = 1$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{H}_k\}$                        ▷ Add singleton.
        else
            $\mathcal{S} \leftarrow \mathcal{S} \cup$ SubDivide($\mathcal{H}_k$, $K$)        ▷ Recurse on non-singletons and add.
        end if
    end for
    Return: $\mathcal{S}$                                    ▷ Return a set of sets.
end function

Return: SubDivide($\{\boldsymbol{x}_n\}_{n=1}^N$, $K$)         ▷ Call and recurse on the whole data set.
```

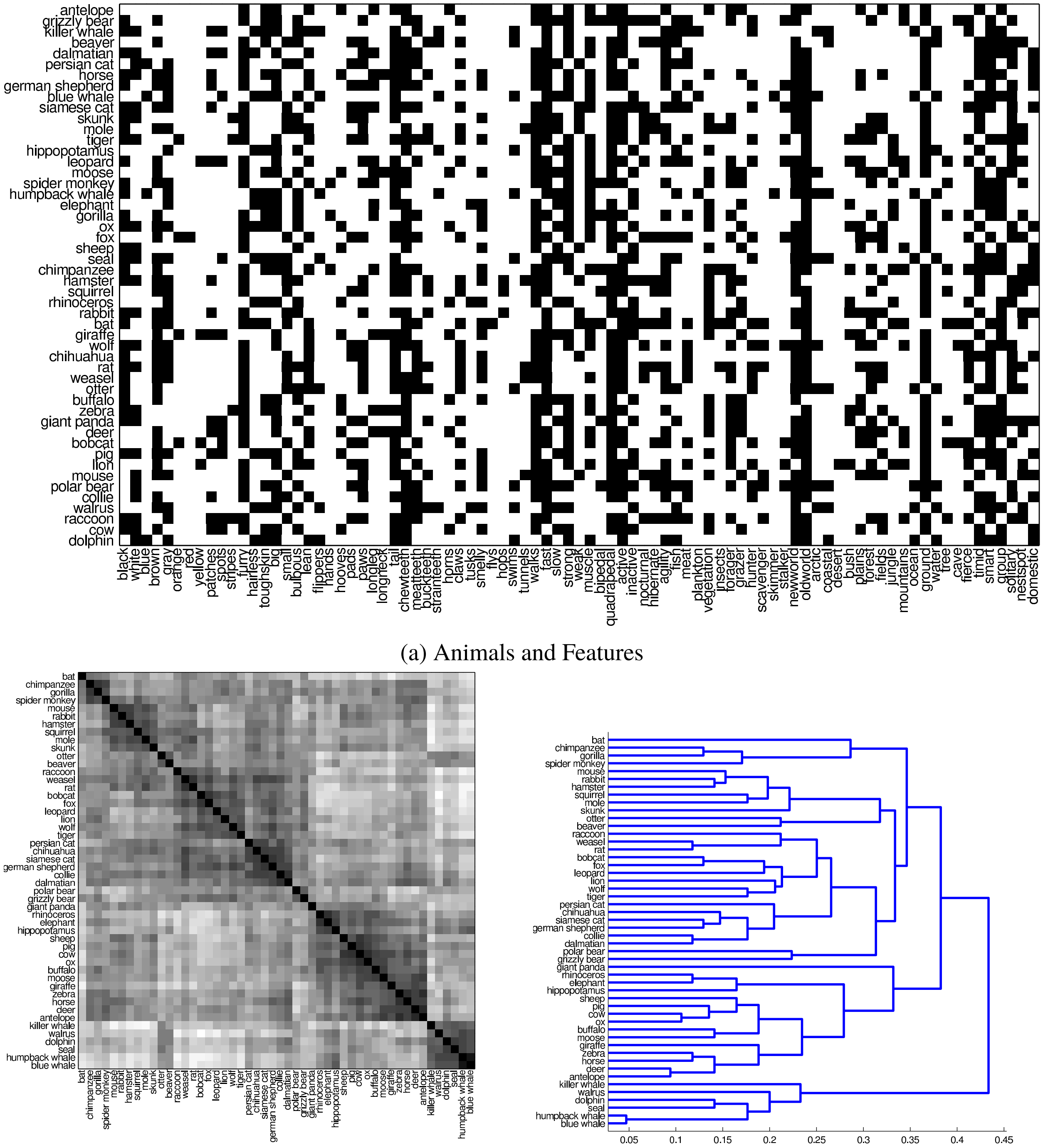
References

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL `http://nlp.stanford.edu/IR-book/`.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

Changelog

- 2 November 2018 – Initial revision of CS181 notes.

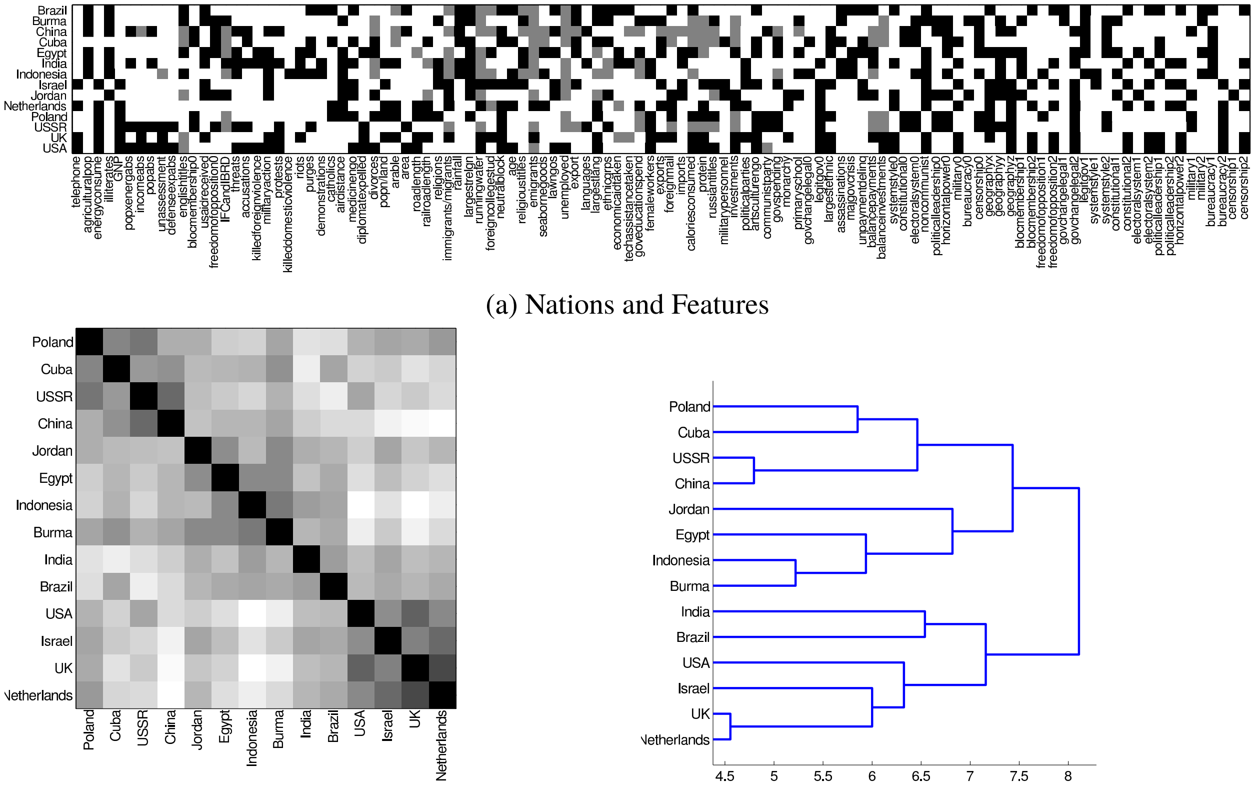


(a) Animals and Features

(b) Hamming Distances

(c) Average-Linkage Dendrogram

Figure 6: These figures show the result of running HAC on a data set of 50 animals, each with 85 binary features. (a) The feature matrix, where the rows are animals and the columns are binary features. (b) The distance matrix computed using pairwise Hamming distances. The ordering shown here is chosen to highlight the block structure. Darker colors are smaller distances. (c) A dendrogram arising from HAC with average-linkage.

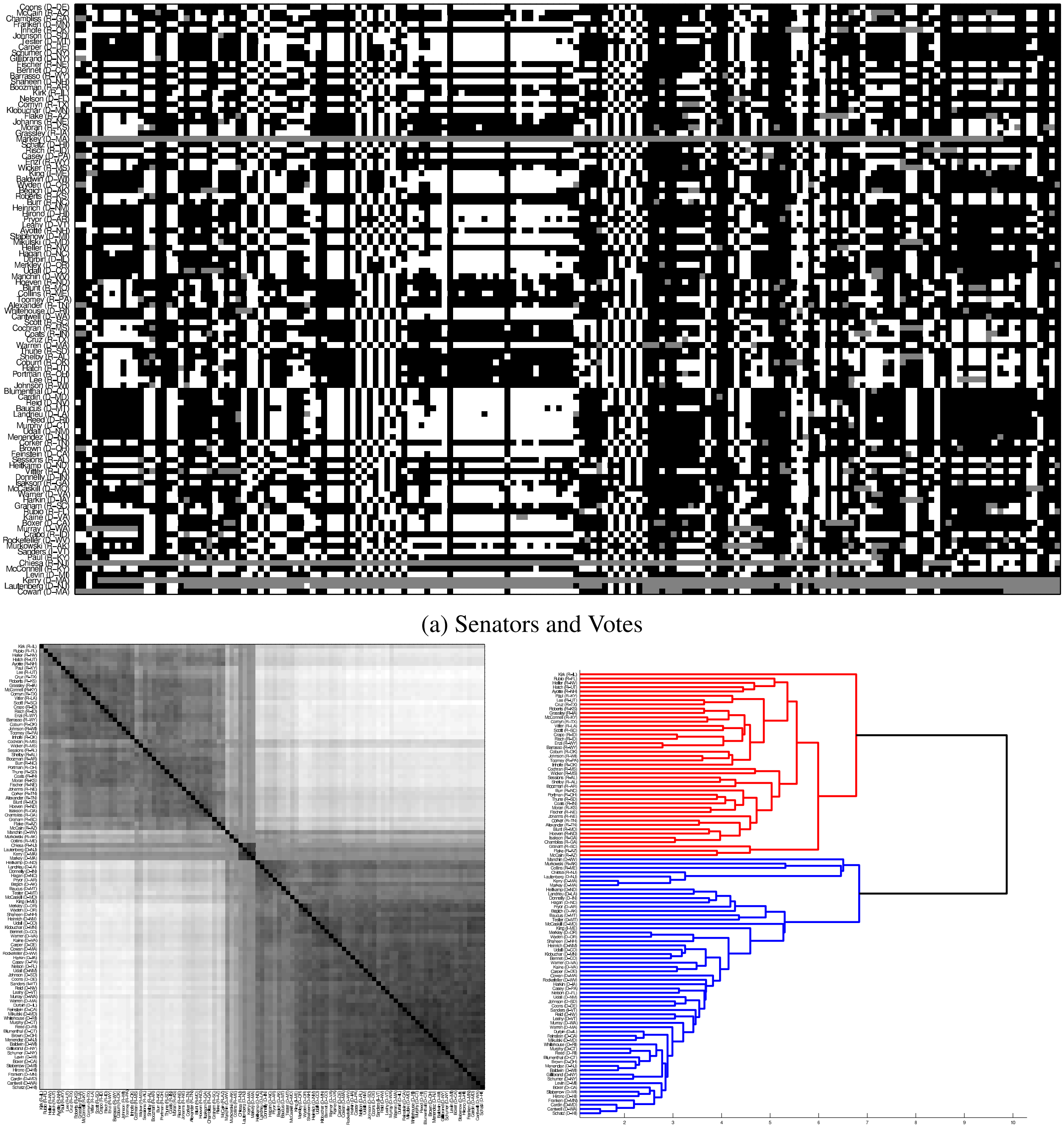


(a) Nations and Features

(b) Euclidean Distances

(c) Complete-Linkage Dendrogram

Figure 7: These figures show the result of running HAC on a data set of 14 nations, with binary features. (a) The feature matrix, where the rows are nations and the columns are binary features. When a feature was missing it was replaced with 1/2. (b) The distance matrix computed using pairwise Euclidean distances. The ordering shown here is chosen to highlight the block structure. Darker colors are smaller distances. (c) A dendrogram arising from HAC with complete linkage.



(a) Senators and Votes

(b) Euclidean Distances

(c) Average-Linking Dendogram

Figure 8: These figures show the result of running HAC on a data set of 104 senators in the 113th US congress, with binary features corresponding to votes on 172 bills. (a) The feature matrix, where the rows are senators and the columns are votes. When a vote was missing or there was an abstention it was replaced with 1/2. (b) The distance matrix computed using pairwise Euclidean distances. The ordering shown here is chosen to highlight the block structure. Darker coloring corresponds to smaller distance. (c) A dendrogram arising from HAC with complete linkage, along with two colored clusters.