Chapter 19

Hierarchical clustering

Hierarchical Clustering. Agglomerative and Divisive Clustering. Clustering Features.

19.1 Hierarchical clustering

Deciding the best number of clusters is often difficult, as the structure of the data may not provide an obvious solution for this problem. For example, if we want to cluster all living organisms, it is not clear how many clusters we should have. In this case, the reason is that living organisms are related in a family tree, in a range of degrees of distance. The best option is to represent this structure in a series of nested clusters, and clusters of clusters, and so on. This is done with *hierarchical clustering*. Figure 19.1 shows two examples of hierarchical clustering. The tree of life, a hierarchical clustering of living species that also represents their evolutionary relations, and hierarchical clustering for analysing similarities in gene expression patterns in different organisms.

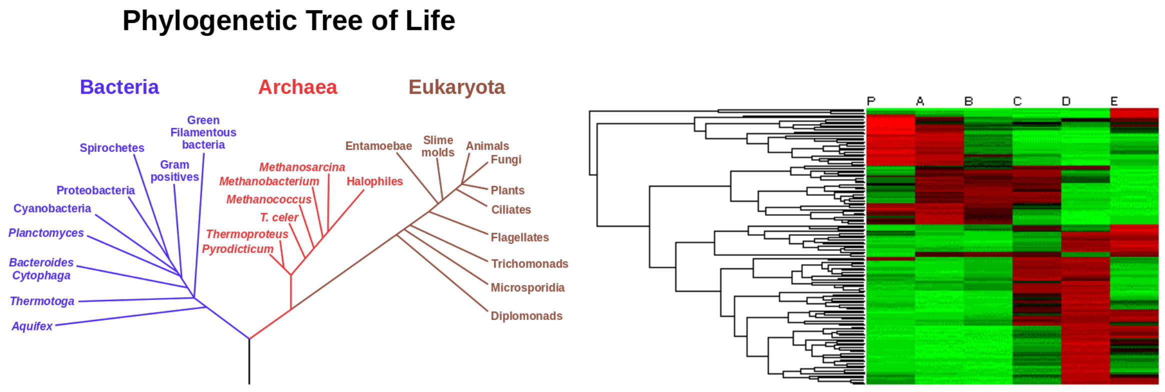

Figure 19.1: Examples of hierarchical clustering. Left panel, hierarchical clustering of living organisms, indicating evolutionary relations. Image source: Wikipedia. On the right panel, hierarchical clustering of gene expression data (Mulvey and Gingold, Online Computational Biology Textbook).

A hierarchical clustering can be represented as a dendrogram (a tree) by joining together first the examples that are more similar and then gradually joining the most similar clusters until all links are found, as shown in Figure 19.2. This means that we need to define how to measure the similarity, or dissimilarity, between examples in our dataset but also how to measure similarity between clusters of examples, because we need to decide how to cluster clusters into sets of larger clusters.

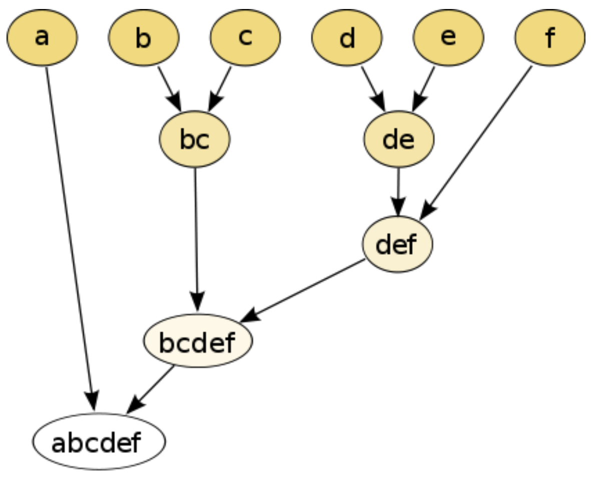

Figure 19.2: Hierarchical clustering represented as a dendrogram. Image source: Wikipedia.

There are several ways of thinking about this problem. We can think about *proximity* between examples as a generic term of “likeness”, without any precise definition. *Similarity*is more well defined, generally a number between 0 and 1 that indicates how alike examples are. *Dissimilarity* is also a quantitative measure, in this case of difference between examples, and *distance* is a special case of a dissimilarity measure that respects the algebraic properties of a distance. Namely, not negative, symmetrical and respecting the triangle inequality:

$$d(x, y) \geq 0 \;,\; d(x, y) = d(y, x) \;,\; d(x, z) \leq d(x, y) + d(y, x)$$

There are many possible distance measures. Some of the most used are Euclidean, Manhattan and squared Euclidean distance.

- Euclidean: $\|x-y\|_2 = \sqrt{\sum\limits_d (x_d - y_d)^2}$
- Squared Euclidean: $\|x-y\|_2^2 = \sum\limits_d (x_d - y_d)^2$
- Manhattan: $\|x-y\|_1 = \sum\limits_d (x_d - y_d)^2$
- Mahalanobis (normalized by variance): $\sqrt{(x-y)^T Cov^{-1}(x-y)}$

For strings and sequences in general, some useful measures are the Hamming distance, which is the count of differences between the strings, or the Levenshtein distance, or edit distance, counting the number of single-character edits (insertions, deletions or substitutions) needed to transform one string into the other.

Apart from a way to measure similarity or distance between examples, we must also measure distance between clusters. The method for evaluating cluster distance is the *linkage*, and there are also several ways of doing this.

- *Single linkage*: distance between clusters is the distance between the closest points.

$$dist(C_j, C_k) = min\left(dist(x \in C_j, y \in C_k)\right)$$

- *Complete linkage*: distance between the most distant points.

$$dist(C_j, C_k) = max\left(dist(x \in C_j, y \in C_k)\right)$$

- *Centroid linkage*: distance between the centroids of the two clusters.

$$dist(C_j, C_k) = dist\left(\frac{\sum x \in C_j}{|C_j|}, \frac{\sum y \in C_k}{|C_k|}\right)$$

- *Average linkage*: average distance between all pairs of points from the different clusters.

$$dist(C_j, C_k) = mean\left(dist(x \in C_j, y \in C_k)\right)$$

- *Median linkage*: median distance between all pairs of points from the different clusters.

$$dist(C_j, C_k) = median\left(dist(x \in C_j, y \in C_k)\right)$$

- *Ward linkage*: join clusters that minimize Sum of Squares Error:

$$\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|x_n - \mu_k\|^2$$

Figure 19-linkage illustrates some examples of linkage methods.

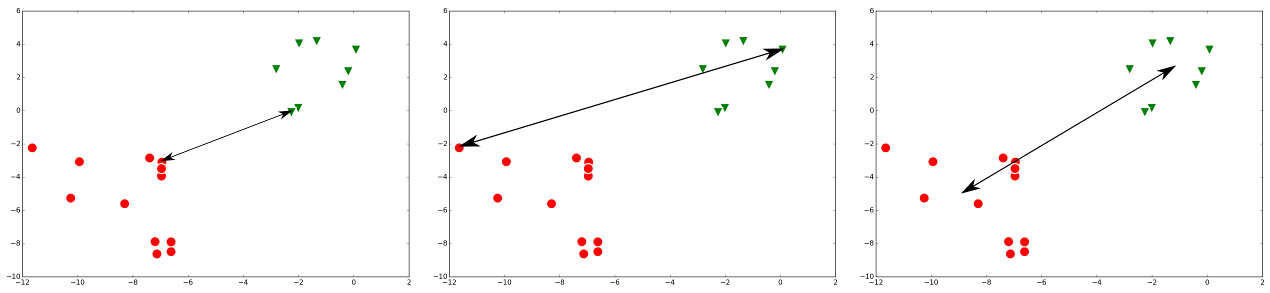

Figure 19.3: Single, complete and centroid linkage methods.

The obvious advantages of hierarchical clustering is avoiding the need to specify a number of clusters, both before or after clustering, and the possibility of revealing some hierarchical structure in the data. The disadvantages are that hierarchical clustering must generally be done in a single pass, with a greedy algorithm, which may introduce errors, and if the hierarchical structure assumed by this type of clustering does not exist in the data the result may be confusing or misleading.

Agglomerative clustering is a bottom-up approach that begins with singleton clusters and repeatedly joins the best two clusters, according to the linkage method used, into a higher level cluster until all elements are joined. The time complexity of agglomerative clustering is generally $O(n^3)$, but can be improved with linkage constraints.

Divisive clustering is a top-down approach that begins with a single cluster containing all examples and iteratively picks a cluster to split and separates it into smaller clusters until some number of clusters is reached. The theoretical time complexity for divisive clustering is $O(2^n)$ for an exhaustive search and this approach needs an additional clustering algorithm for splitting each cluster. However, the time complexity in practice can be lower, depending on the clustering algorithm used, and it may be better than agglomerative clustering if we only want a few levels of hierarchical clustering.

19.2 Hierarchical to partitional

Although a hierarchical clustering is a tree of clusters inside other clusters, we can convert it into a partitional clustering, with a set of clusters at the same level, by cutting some arcs of the tree. The farther we go from the root of the tree, the greater the number of clusters generated. Figure ?? illustrates this process.

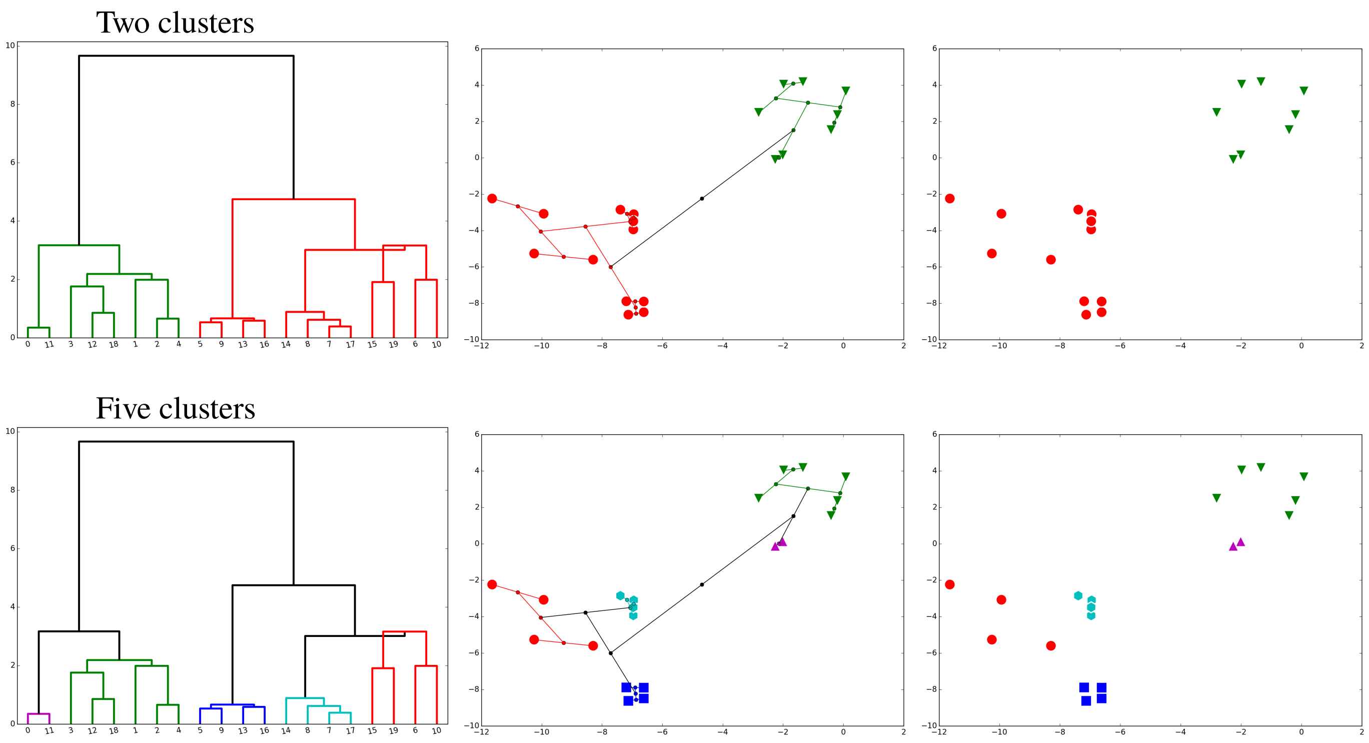

Figure 19.4: Partitioning a hierarchical clustering by cutting the tree at the desired level.

19.3 Connectivity constraints

In agglomerative clustering, we can restrict which clusters to join by adding connectivity constraints. These constraints specify which examples are considered connected and only clusters with connected examples, from one cluster to the other, can be joined into larger clusters. This helps solve some problems like Figure 19.5 illustrates. The left panel shows the result of agglomerative clustering without connectivity constraints. Since the linkage method used (Ward) takes into account only distances between the points, in order to minimize the SSE, the clusters include examples across the gap separating different stretches of the “ribbon” in which the data is structured. A connectivity constraint that restricts the connection of each example only to the 10 nearest neighbours creates a graph of connections that respects the structure of the data and prevents these inadequate clusters from forming.

To create this matrix with the connectivity constraints, we can use the `kneighbors_graph` function from the `neighbors` module of the Scikit-learn, and then use the connectivity constraints matrix in the `AgglomerativeClustering` class, as shown below. The result is shown in the right panel of Figure 19.5.

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.neighbors import kneighbors_graph
```

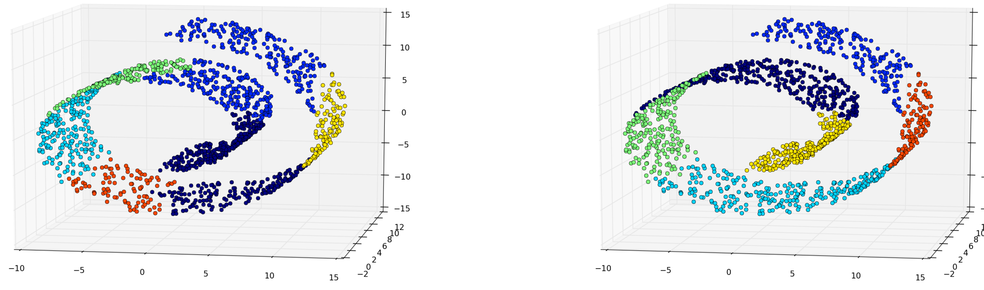

Figure 19.5: Agglomerative clustering with Ward linkage, without connectivity constraints (left panel) and with connectivity constraints connecting only the 10 nearest neighbours of each example.

```

connectivity = kneighbors_graph(X, n_neighbors=10, include_self=False)
ward = AgglomerativeClustering(n_clusters=6, connectivity=connectivity,
                               linkage='ward').fit(X)
```

19.4 Choosing the linkage method

Scikit-Learn currently offers three linkage methods for agglomerative clustering: complete, average and Ward linkage. Figure 19.6 shows an example data set clustered to three clusters using agglomerative clustering and the three linkage methods. Complete linkage tends to favour larger clusters, so leads to a poor relation between the clusters and the data structure in some cases, as the figure shows (left panel). Average linkage is better, in these cases (middle panel), and Ward linkage (right panel), minimizing the SSE measured in the clusters, seems to work best. However, Ward linkage can only be used when the dissimilarity measure is the Euclidean distance, so if another measure must be used average linkage tends to be the best option.

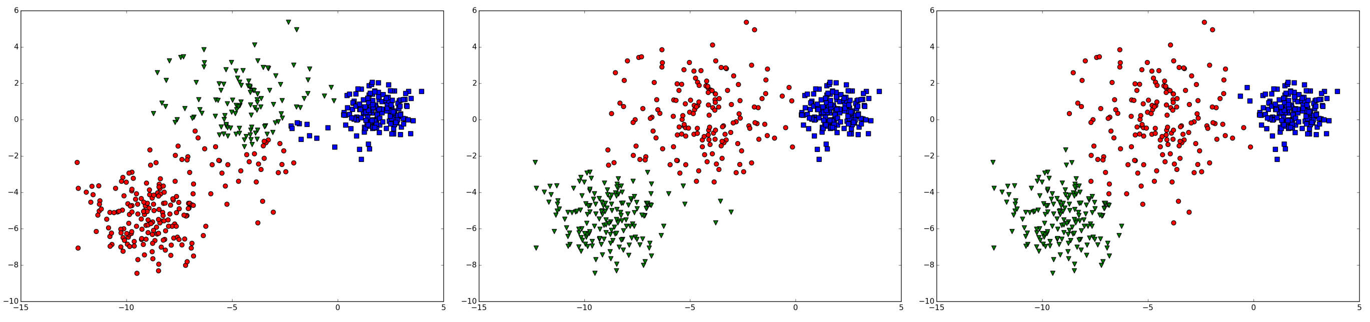

Figure 19.6: Agglomerative clustering of the same data set with (left to right) complete, average and Ward linkage.

19.5 Bisecting k-means

An example of a divisive hierarchical clustering algorithm is the *bisecting k-means*. The algorithm is:

1. Start with all the examples in a single cluster.

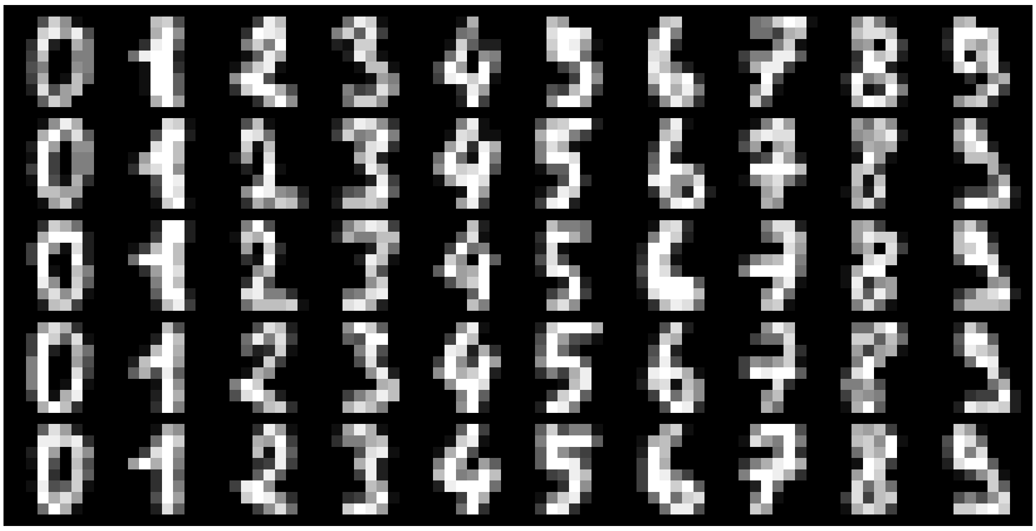

Figure 19.7: Some examples from the handwritten digits dataset.

2. Choose the best cluster for splitting (*e.g.* the largest or the one with the lowest score).
3. Split the best candidate with k-means, using $k = 2$.
4. Repeat steps 2 and 3 until the desired number of clusters is reached.

Although the time complexity for an exhaustive search in divisive clustering is $O(2^n)$, using the k-means algorithm reduces the complexity (although at the cost of having a more greedy divisive clustering) and the possibility of stopping at the desired level may make this algorithm preferable to agglomerative clustering in some cases, since agglomerative clustering must run until the complete tree is generated.

19.6 Clustering features

Conceptually, clustering features is the same as clustering examples. We need but imagine that we transpose the matrix with all examples in rows and features in columns and obtain a new matrix were the examples are in the columns, and are now considered features, and the original features, now in rows, are examples. Clustering features allows us to identify similar features and reduce the dimensionality of the data by grouping these together in a single feature. With hierarchical clustering we have a simple way of controlling how many groups of features we use and thus the dimensionally of the resulting data set.

To illustrate this, we will simplify the handwritten digits data set, which consists of digitized handwritten digits into grayscale bitmaps of 64 pixels. Figure 19.7 shows these data.

The data set has a total of 1797 examples with 64 features each so, for feature clustering, we will convert it into a set of 64 examples each with 1797 features. Then we cluster it into 16 clusters, corresponding to 16 features in the original data set, which we can extract by averaging all features in each cluster. We also add a connectivity constraint to restrict clustering to neighbouring pixels in the original image. Feature clustering is done automatically in the `FeatureAgglomeration` class, so the complete code, including loading the data set, is simply:

```
import numpy as np
from sklearn import datasets, cluster
```

```
 3 from sklearn.feature_extraction.image import grid_to_graph
 4
 5 digits = datasets.load_digits()
 6 images = digits.images
 7 X = np.reshape(images, (len(images), -1))
 8 connectivity = grid_to_graph(images[0].shape[0],images[0].shape[1])
 9 agglo = cluster.FeatureAgglomeration(connectivity=connectivity, n_clusters=16)
10 agglo.fit(X)
11 X_reduced = agglo.transform(X)
12 X_restored = agglo.inverse_transform(X_reduced)
```

Lines 5-7 are for loading the data and converting the image matrices into a matrix of examples (rows) and features (columns). Line 8 is for creating the connectivity matrix with the neighbours of each pixel in the 64×64 image matrix. Lines 9 and 10 create the agglomerative clusterer and fit the data, and the last two lines complete the reduced dataset, with only 16 features, and a 64 features dataset with the feature values aggregated, averaging the features in the same cluster. Figure 19.8 shows the result. Although the digits in the reduced dataset are no longer recognizable as digits, it is easy to see that the patterns are different from digit to digit, so this process reduced the number of features without losing much information.

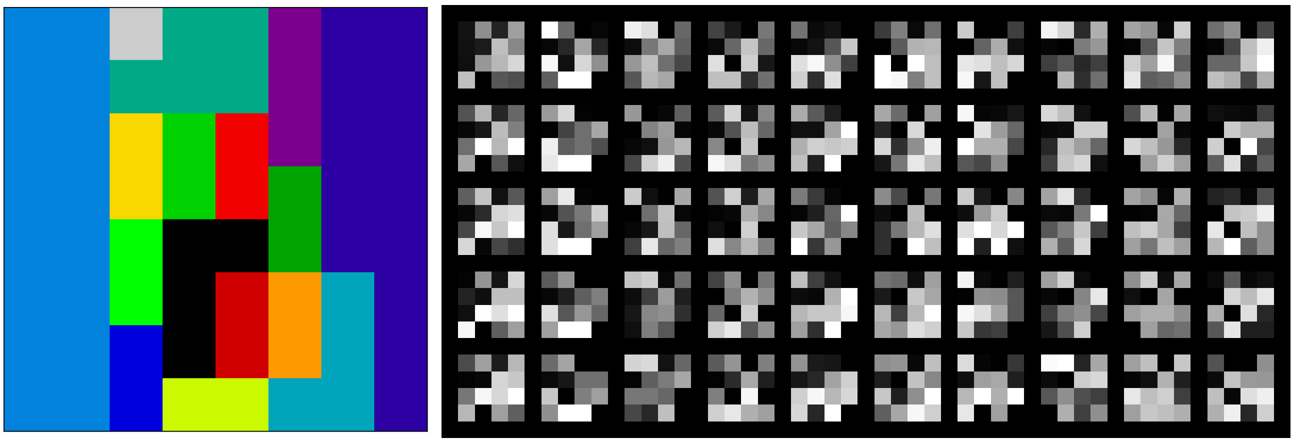

Figure 19.8: Feature clustering. The original handwritten digits features were clustered as shown in the left panel. Using only these 16 clusters as 16 features, the reduced data set is illustrated on the right panel.

19.7 Further Reading

1. Scikit-learn documentation on clustering:`http://scikit-learn.org/stable/modules/clustering.html`
2. Alpaydin [2], Section 7.7

Bibliography

[1] Uri Alon, Naama Barkai, Daniel A Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[3] David F Andrews. Plots of high-dimensional data. *Biometrics*, pages 125–136, 1972.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, New York, 1st ed. edition, oct 2006.

[5] Deng Cai, Xiaofei He, Zhiwei Li, Wei-Ying Ma, and Ji-Rong Wen. Hierarchical clustering of www image search results using visual. Association for Computing Machinery, Inc., October 2004.

[6] Guanghua Chi, Yu Liu, and Haishandbscan Wu. Ghost cities analysis based on positioning data in china. *arXiv preprint arXiv:1510.08505*, 2015.

[7] Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404. Morgan Kaufmann, 1990.

[8] Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning. Stanford CA Morgan Kaufmann*, pages 231–238, 2000.

[9] Hakan Erdogan, Ruhi Sarikaya, Stanley F Chen, Yuqing Gao, and Michael Picheny. Using semantic analysis to improve speech recognition performance. *Computer Speech & Language*, 19(3):321–343, 2005.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[11] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[12] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[13] Patrick Hoffman, Georges Grinstein, Kenneth Marx, Ivo Grosse, and Eugene Stanley. Dna visual and analytic data mining. In *Visualization'97., Proceedings*, pages 437–441. IEEE, 1997.

[14] Chang-Hwan Lee, Fernando Gutierrez, and Dejing Dou. Calculating feature weights in naive bayes with kullback-leibler measure. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1146–1151. IEEE, 2011.

[15] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[16] 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.

[17] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[18] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[19] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[20] Roberto Valenti, Nicu Sebe, Theo Gevers, and Ira Cohen. Machine learning techniques for face analysis. In Matthieu Cord and Pádraig Cunningham, editors, *Machine Learning Techniques for Multimedia*, Cognitive Technologies, pages 159–187. Springer Berlin Heidelberg, 2008.

[21] Giorgio Valentini and Thomas G Dietterich. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *The Journal of Machine Learning Research*, 5:725–775, 2004.

[22] Jake VanderPlas. Frequentism and bayesianism: a python-driven primer. *arXiv preprint arXiv:1411.5018*, 2014.