Chapter 20

Fuzzy clustering and manifold learning

Fuzzy sets and clustering. Fuzzy c-means. Manifold learning. Cluster validation: internal and external indeces.

20.1 Fuzzy Clustering

In conventional set theory, elements either belong or do not belong to a set. In such case, we are dealing with *crisp* sets. In *fuzzy* set theory, each element x has a membership value $u_S(x) \in [0, 1]$ specifying by how much x belongs to set S. Thus, a *fuzzy set* S is a set of ordered pairs of elements and their respective membership function values:

$$S = \{(\mathbf{x}, u_S(\mathbf{x}))|\mathbf{x} \in X\}$$

This makes it possible to model different types of uncertainty, such as linguistic or categorical uncertainty, when we are unable to define exactly what we mean by some term or category. For example, the temperature at which something stops being cold and becomes warm or hot is not a precise (*crisp*) value. One way to account for this is to allow the membership of each temperature value to each category cold, warm or hot to vary continuously, as Figure 20.1 illustrates.

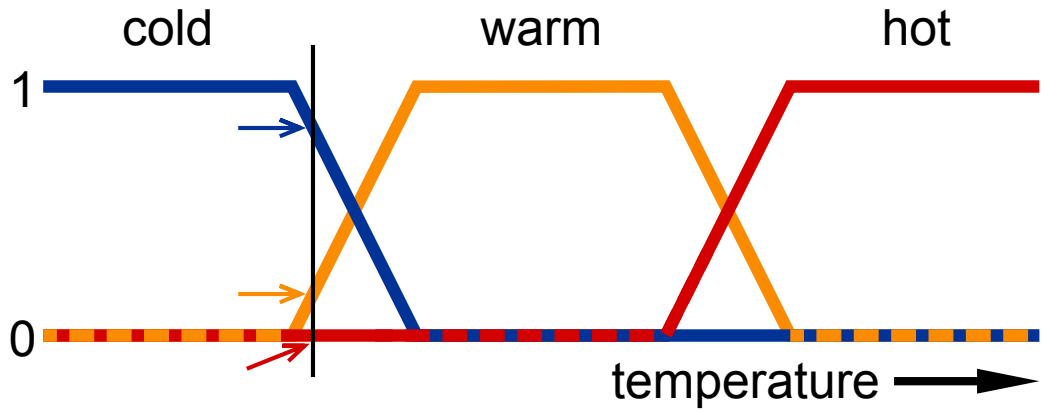

Figure 20.1: Fuzzy sets for cold, warm and hot temperatures, and respective membership values. Wikimedia, CC BY-SA 3.0 fullofstars

Fuzzy sets can also model uncertainty about information or predictions, but fuzzy membership is different from probability estimates. Conceptually, fuzzy membership is a measure of similarity to

some imprecise properties that characterize the set the element may belong to with a smaller or larger membership value, while probability is a measure either of a frequency of random events in the limit of infinite trials (frequentist interpretation) or of uncertainty but under precise definitions of concepts (Bayesian interpretation).

Fuzzy clustering rests on the notion of a *fuzzy c-partittion*. $\mathbf{U}(\mathbf{X})$ is a fuzzy c-partition of X if these three conditions hold. First, the membership values of all elements are between 0 and 1:

$$0 \leq u_k(\mathbf{x}_n) \leq 1 \qquad \forall k, n$$

Second, the total membership of each element to all c partitions is equal to 1:

$$\sum_{k=1}^{c} u_k(\mathbf{x}_n) = 1 \qquad \forall n$$

Finally, the total membership in each of the c partitions is between 0 and the total number of elements:

$$0 \leq \sum_{n=1}^{N} u_k(\mathbf{x}_n) \leq N \qquad \forall k$$

The *fuzzy c-means* algorithm is a clustering algorithm that finds a *fuzzy c-partittion* for the elements to cluster, with each partition being a cluster. From a set X of N data points, the algorithm returns the $c \times N$ membership matrix $u_k(\mathbf{x}_n)$, defining a fuzzy c-partition of X and determining the membership value of each element $\mathbf{x}_n, n \in \{1, ..., N\}$ to each cluster $k \in \{1, ..., c\}$. The *fuzzy c-means* algorithm also returns the set $\{C_1, ..., C_c\}$ of centroids of the partitions (clusters). These are found by minimizing the following squared error loss function:

$$J_m(X, C) = \sum_{k=1}^{c} \sum_{n=1}^{N} u_k(\mathbf{x}_n)^m \|\mathbf{x}_n - \mathbf{c}_k\|^2 \qquad m \geq 1$$

and subject to the constraint

$$\sum_{k=1}^{c} u_k(\mathbf{x}_n) = 1 \qquad \forall n$$

</p> The parameter m, typically $m = 2$, is the *degree of fuzzification*. The derivative of the loss function with respect to the membership values is zero at the points:

$$u_k(\mathbf{x}_n) = \frac{\left(\frac{1}{\|\mathbf{x}_n - \mathbf{c}_k\|^2}\right)^{\frac{2}{m-1}}}{\sum_{j=1}^{c} \left(\frac{1}{\|\mathbf{x}_n - \mathbf{c}_j\|^2}\right)^{\frac{2}{m-1}}}$$

and with respect to the centroids C_k:

$$C_k = \frac{\sum_{n=1}^{N} u_k(\mathbf{x}_n)^m \mathbf{x}_n}{\sum_{n=1}^{N} u_k(\mathbf{x}_n)^m}$$

That is, each centroid C_k is the weighted mean of the example vectors using the membership values. This algorithm is similar to the k-means algorithm, but using a continuous membership function instead of the $0, 1$ membership of crisp sets.

Like k-means, the fuzzy c-means algorithm also uses an approach that is similar to Expectation-Maximization (EM), even though, formally, it is not quite EM. First, the membership values are computed from a random initial set of centroids $\{C_1, ..., C_c\}$:

$$u_k(\mathbf{x}_n) = \frac{\left(\frac{1}{\|\mathbf{x}_n - \mathbf{c}_k\|^2}\right)^{\frac{2}{m-1}}}{\sum_{j=1}^{c}\left(\frac{1}{\|\mathbf{x}_n - \mathbf{c}_j\|^2}\right)^{\frac{2}{m-1}}}$$

This, in turn, allows the update of the centroid coordinates by maximizing the adequacy of the centroid to the cluster assuming the computed membership values:

$$C_k = \frac{\sum_{n=1}^{N} u_k(\mathbf{x}_n)^m \mathbf{x}_n}{\sum_{n=1}^{N} u_k(\mathbf{x}_n)^m}$$

These steps are then repeated until convergence, as usual in algorithms based on the EM method. The stopping criteria for the fuzzy c-means algorithm are generally either reaching a predetermined number of iterations or the change in the centroid positions falling below some initially specified value.

The result is similar to a k-means clustering, but with continuous membership values. Figure 20.2 illustrates the clustering along with the plot of the membership function for each cluster.

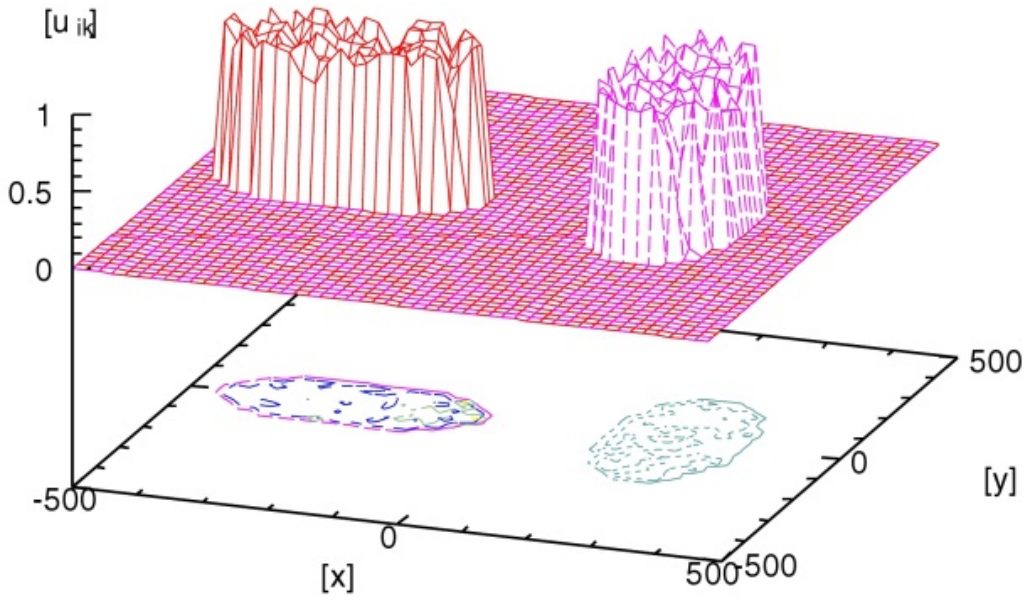

Figure 20.2: Fuzzy c-means example, from "Simulated Annealing - Advances, Applications and Hybridizations", Ed. Marcos de Sales Guerra Tsuzuki, CC BY 3.0

To convert a fuzzy clustering into a crisp clustering — *defuzzification* — it is simply necessary to convert the continuous membership function into $\{0, 1\}$ crisp membership values. This can be done by, for each data point, setting to 1 the largest membership value, and to 0 the remainder, thus assigning the data point to the cluster to which it has the largest membership, or assigning the data point to the cluster with the nearest centroid, as in the k-means algorithm.

20.2 Manifold Learning

Mathematically, an n-dimensional manifold, or n-manifold, is a set of points such that each point and its neighbours form an approximately Euclidean space. For example, seismic events at the surface of the Earth form a two-dimensional surface in three-dimensional space. Figure 20.3 shows two examples of manifolds.

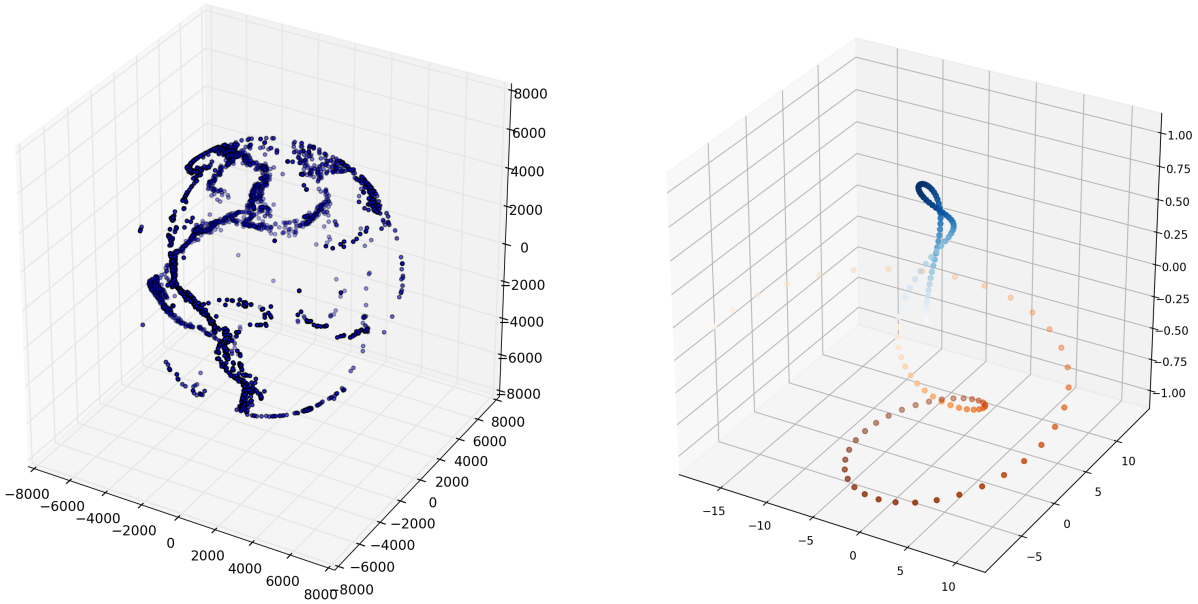

Figure 20.3: Examples of two manifolds in three-dimensional space: the 2-dimensional manifold of seismic events on the surface of the Earth and a 1-dimensional manifold of hypothetical data that follows a line in 3D.

In machine learning, this is a useful concept because it is often the case that data do not span all possible combinations of feature values. Thus, data sets are usually sets of points that can be approximated by manifolds fewer dimensions than the number of attributes. Finding these manifolds is a useful way of reducing the dimensionality of our data.

There are many different algorithms for finding manifolds, using different approaches and criteria [1]. Here we will see two as examples: Isomap and t-distributed stochastic neighbor embedding (t-SNE).

t-SNE

The t-distributed stochastic neighbor embedding algorithm [16] (t-SNE) projects the data into lower dimensions – typically two dimensions, for visualization – while trying to keep the distribution of distances between points approximately analogous. First, it considers the probability of point x_i choosing point x_j as a neighbour to be a Gaussian distribution dependent on the distance between the points, $||x_i - x_j||$. This can be imagined as the conditional probability of point x_j being picked as a neighbour given that x_i is the centre of the neighbourhood:

[1] See, for example, Scikit-Learn documentation on Manifold Learning at https://scikit-learn.org/stable/modules/manifold.html

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{i \neq j} exp(-||x_i - x_j||^2/2\sigma_i^2)} \qquad (p_{i|i} = 0)$$

To avoid the problem of computing the joint probability of finding x_i and x_j in the same neighbourhood from the conditional probabilities, and also to make the distribution vanish less quickly for longer distances, in t-SNE the joint probability is taken as the average of the conditional probabilities:

$$p_{j,i} = (p_{j|i} + p_{i|j})/2$$

Note that $p_{j|i}$ and $p_{i|j}$ may be different because each Gaussian centred on each point has its own σ value. In addition, though these steps are computationally convenient, this means that the probabilistic framework of t-SNE is not formally correct but rather a practical approximation.

On the side of the embedded manifold, in lower dimensions, we can consider a similar distribution or probabilities for points being neighbours, also dependent on their distance. For this, t-SNE uses Student's t-distribution. Thus, with y_i and y_j being the images of x_i and x_j in the lower dimensional space, the probability of y_j being in the neighbourhood of y_j is:

$$q_{j|i} = \frac{(1 + ||y_i - y_j||)^{-1}}{\sum_{i \neq j}(1 + ||y_i - y_j||)^{-1}} \qquad (q_{i|i} = 0)$$

Since Student's t-distribution has no parameters, $q_{j|i} = q_{i|j}$ and we can just take this as if it was the joint distribution. Another reason for choosing Student's t-distribution is that it makes it easier to adjust the y values in order to bring this distribution close to the Gaussian distributions for the original points. This is done by minimizing the Kullback–Leibler divergence of q_{ij} with respect to p_{ij}:

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

which measures how the distributions differ. By minimizing this measure the distribution of the embedded points will be locally similar to what it is in the original space. Figure 20.4 illustrates the result of applying t-SNE to a three-dimensional data set that is distributed along a one-dimensional line.

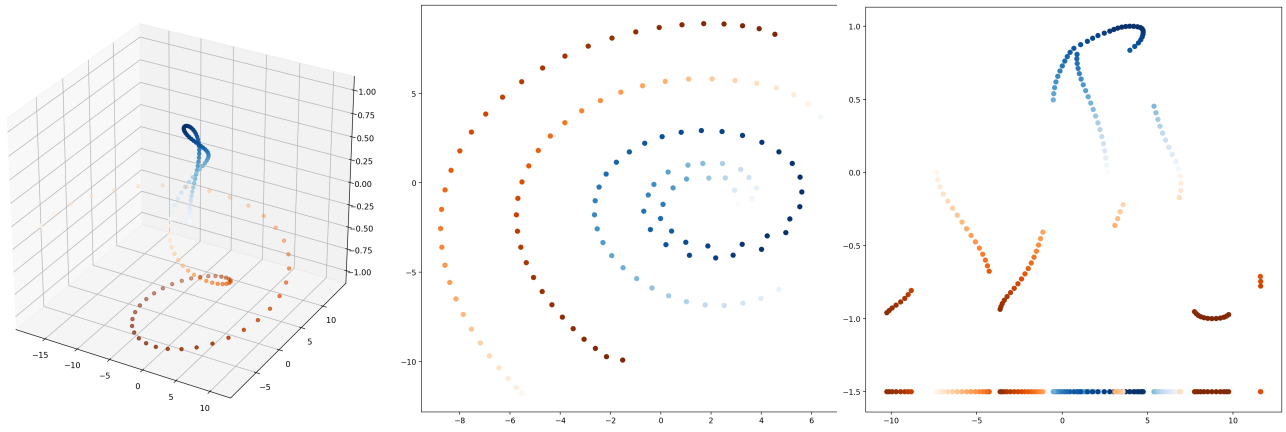

Figure 20.4: Original data set (left panel), t-SNE projection to two dimensions (center) and to one dimension (right panel, bottom line).

Isomap

The Isomap algorithm [21] tries to create a low-dimensional manifold from high-dimensional data while preserving as much as possible the distances between the nearest neighbours. The outline of the algorithm is as follows:

1. Create a k-nearest neighbours graph connecting each point to its k-nearest neighbours.
2. Compute pairwise distances for all pairs of points by adding the distances of all steps in the shortest path between two points on the k-nearest neighbours graph. This is an estimate of the distance between points in the manifold.
3. Compute the distribution of the points in the lower-dimensional manifold with multidimensional scaling, which finds an embedding that preserves as much as possible the distances between pairs of points.

Figure 20.5 illustrates the result of applying Isomap to the data set shown previously. We can reduce from three to two dimensions, or even one dimension, and still preserve most of the local structure in the data because this embedding in lower dimensions approximates the distances measured along the k-neighbours graph.

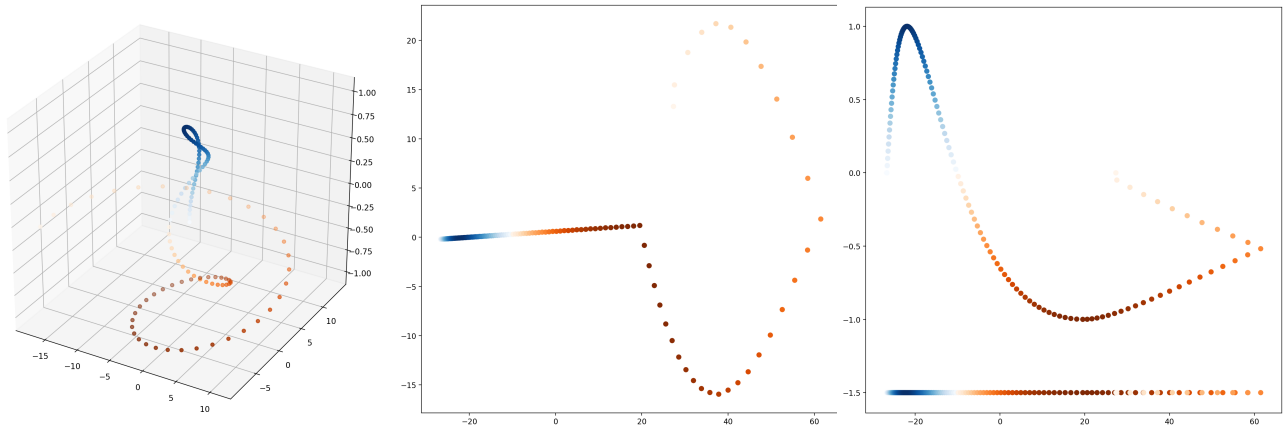

Figure 20.5: Original data set (left panel), Isomap projection to two dimensions (center) and to one dimension (right panel, bottom line).

20.3 The Rand index

Previously, we saw the silhouette score as an internal index to evaluate clusterings, and we also talked about the possibility of using external indexes. The Rand index is an example of an external index we can use when we want to compare a clustering with some other partition of our data, such as another clustering, classification labels or any other way of organizing our data into different groups.

Let us suppose our N examples are grouped into some partition X composed of groups $\{X_1, X_2, X_3, ...\}$ and we have a clustering Y with clusters $\{Y_1, Y_2, Y_3, ...\}$. Note that this is not a supervised learning problem, so we are not trying to predict the exact groups each example will fall into. However, we would like our clustering to place in the same cluster of Y examples that belong in the same group of X and in different clusters points that belong to different groups.

To measure this, we can consider all $N \times (N-1)/2$ pairs of examples and label any pair "positive" if the two examples belong in the same group of X and "negative" if they belong to different groups. This way, we can make an analogy to the true and false positives, and true and false negatives, of supervised learning:

- True Positive: a pair of examples from the same group placed in the same cluster
- True Negative: a pair of examples from different groups placed in different clusters
- False Positive: a pair of examples from different groups placed in the same cluster
- False Negative: a pair of examples from the same group placed in different clusters

This makes it easy to understand the Rand index as analogous to the accuracy of a classifier:

$$Rand = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{N(N-1)/2}$$

One shortcoming of the Rand index is that it does not account for the possibility of the clustering of pairs of examples matching the groups with which we compare them. The Adjusted Rand Index solves this problem by subtracting the expected index values if the clustering was uncorrelated to the groups it is being compared to. The Adjusted Rand Index varies from -1 to 1, with 0 indicating no correlation, and can be computed in Scikit Learn using the `adjusted_rand_score` function from the `sklearn.metrics` module.

Following this analogy with classification, we can also compute scores analogous to precision, recall and the F1 measure:

$$Precision = \frac{TP}{FP + TP} \qquad Recall = \frac{TP}{FN + TP} \qquad F1 = 2\frac{Precision \times Recall}{Precision + Recall}$$

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*, page 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, textual and link information. In *MULTIMEDIA '04 Proceedings of the 12th annual ACM international conference on Multimedia*, page 952–959. 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*, page 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*, page 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*, page 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*, page 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*, page 1146–1151. IEEE, 2011.

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

[16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[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, page 281–297. Oakland, CA, USA., 1967.

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

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

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

[21] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[22] 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, page 159–187. Springer Berlin Heidelberg, 2008.

[23] 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.

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