Chapter 18

Clustering: beyond prototypes

Affinity Propagation clustering and problems with prototype-based clustering. Density Clustering. Clustering validation.

18.1 Affinity Propagation clustering

Clustering using prototypes can be useful in some situations but inadequate in others. Because prototype-based clustering assigns data points to clusters according the their similarity to the prototypes, this type of clustering does not work well with clusters that are not globular or have different variances. Another difficulty arises if, like with k-means, we need to specify the number of clusters, which can lead to a poor clustering in some cases, although, in others, this can be a useful feature as we saw in the case of vector quantization, where the ability to specify the number of prototypes allows us to control the quantization. Figure 18.1 shows these aspects of prototype-based clustering.

Affinity Propagation solves the problem of having to pre-determine the number of clusters to generate. This algorithm is based, conceptually, on the idea of data points passing messages of "responsibility" sent by each data point to the candidates for cluster prototypes, indicating how suitable each candidate is according to that data point, and messages of "availability", sent by each prototype candidate to the data points indicating how adequate the candidate seems to be based on the support it has for being a prototype. Thus, we define:

- The similarity matrix $s_{i,k}$, indicating how alike two data points are, with $s_{k,k}$ indicating the propensity for being a prototype.
- Responsibility matrix $\mathbf{R}$, with $r_{i,k}$, indicating the suitability of k as prototype for i as estimated by i
- Availability matrix $\mathbf{A}$, with $a_{i,k}$, indicating the suitability of k to be prototype for i as estimated by k

The algorithm begins by initializing $\mathbf{R}$ and $\mathbf{A}$ to zero. Then assign to each value of $r_{i,k}$ the similarity between point i and prototype candidate k minus the largest sum of the affinity and similarity between i and any other prototype:

$$r_{i,k} \leftarrow s_{i,k} - \max_{k' \neq k} \left(a_{i,k'} + s_{i,k'} \right)$$

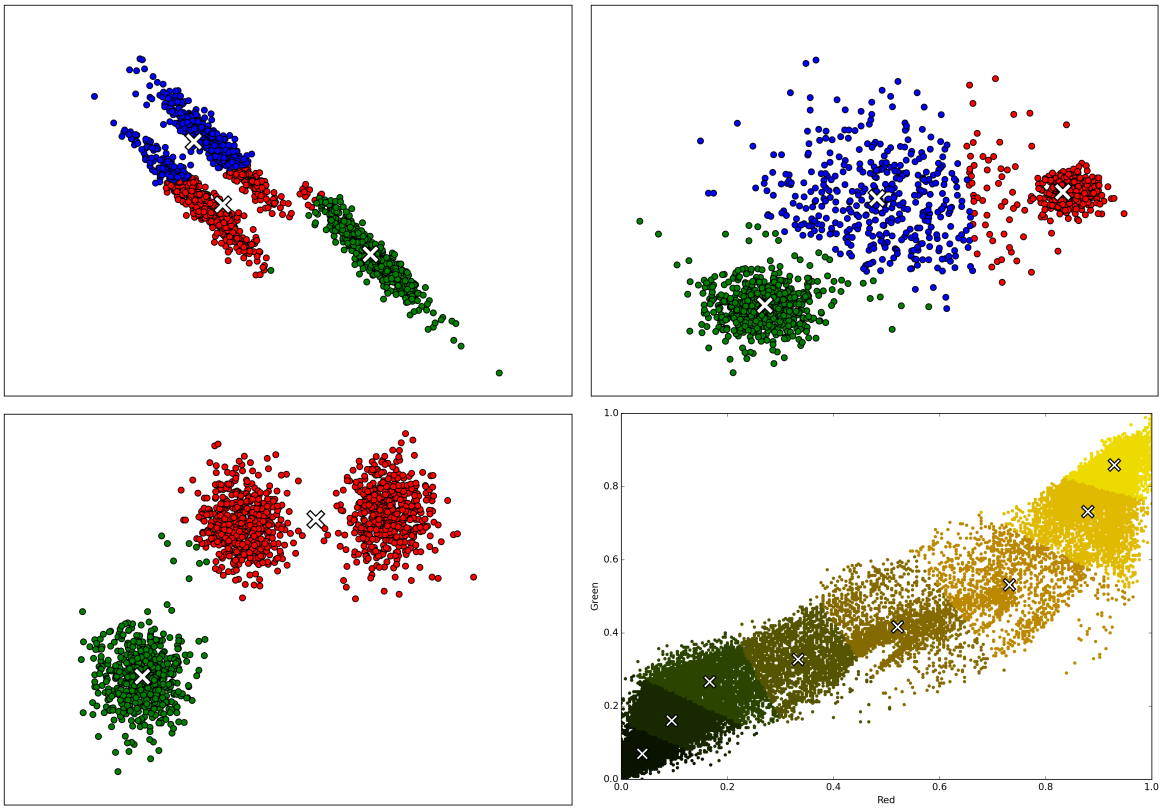

Figure 18.1: Prototype clustering fares poorly when the clusters are not globular (top-left panel), have different variances (top-right) or the number of prototypes chosen is incorrect (bottom-left). However, these features may be useful in some cases, such as in vector quantization (bottom-right).

In the first iteration, with $\mathbf{A}$ set to zero, this is simply the similarity between i and k. After the first iteration, this also discounts the availability of the other candidates as prototypes for point i, the values of $a_{i,k'}$, which are negative numbers. So, basically, the responsibility sent from i to k will depend on how better than other candidates, in similarity and availability, k seems to be for i. In other words, candidate prototypes are competing for the votes of the data points.

Then the algorithm updates the $\mathbf{A}$ matrix by considering the self responsibility of k and the votes k gets from other data points.

$$a_{i,k(i\neq k)} \leftarrow \min\left(0, r_{k,k} + \sum_{i' \notin \{i,k\}} \max(0, r_{i',k})\right)$$

The self availability is updated as follows, serving as evidence that k is a prototype:

$$a_{k,k} \leftarrow \sum_{i' \neq k} \max\left(0, r_{i',k}\right)$$

To identify prototypes, at each iteration, the algorithm computes for each i the k' point with the largest sum of availability and responsibility:

$$k' = \arg\max_{k} a_{i,k} + r_{i,k}$$

If $k' = i$, then i is a prototype of a cluster. Otherwise, i belongs to the cluster with prototype k'. Figure 18.2 shows some stages of the affinity propagation algorithm. Initially, nearly all points consider

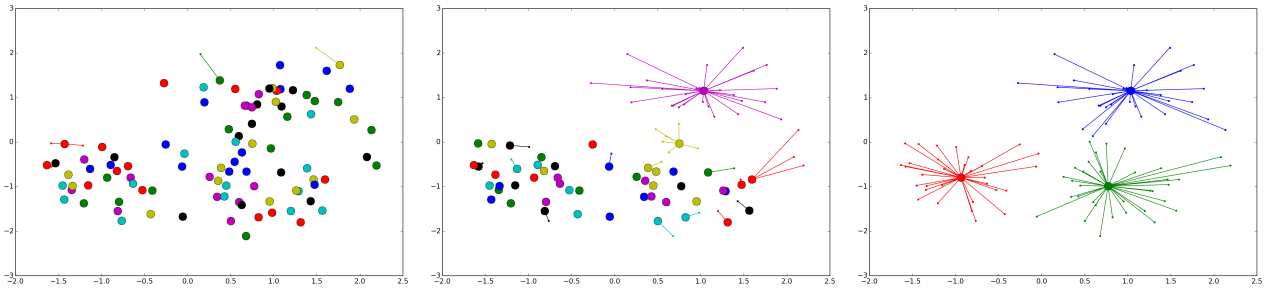

Figure 18.2: Stages of training in affinity propagation clustering.

themselves the best prototype for their own clusters. But as availability and responsibility is propagated, more votes will accumulate with some candidates, becoming prototypes for larger groups of points.

This solves the problem of determining the number of clusters, since these result automatically from the algorithm and depend on the structure of the data. However, it does not solve other problems with prototype-based clustering that stem from attributing points to clusters based on the similarity to the cluster prototypes. Figure 18.3 illustrates this problem. Since all it matters is the similarity to the prototype, and all points must be long to the cluster with the closest prototype, prototype-based clustering is unable to account for some relations between the points.

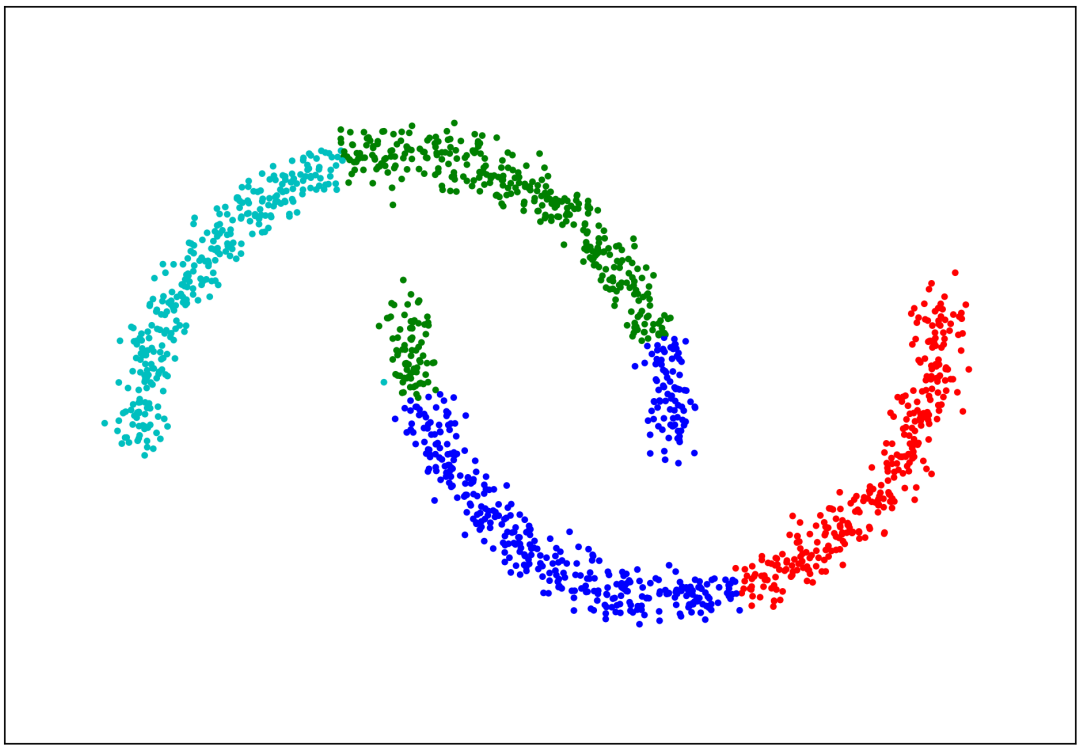

Figure 18.3: Affinity propagation does not solve all problems with prototype-based clustering.

Affinity propagation clustering in Scikit-learn can be done with the `AffinityPropagation` class in the `cluster` module.

18.2 Density-based clustering

The *Density-based spatial clustering of applications with noise* (DBSCAN) algorithm[10] takes a different approach, which solves these problems with prototype-based clustering. Defining the ϵ

neighbourhood N_ϵ of each point as the set of points within distance ϵ, a point p is a *core point* if the number of points in the N_ϵ of p is at least equal to a parameter `minPts`. A point q is *reachable* from p if p is a core point and q is in the neighbourhood N_ϵ of p or in the neighbourhood N_ϵ of any core point that is reachable from p. This recursive definition means that q is reachable from p if there is a path of reachable core points from p to q. The DBSCAN algorithm proceeds as follows. For each point p, if the number of points in the neighbourhood N_ϵ of p is less than `minPts`, p is presumed to be noise. Otherwise, a cluster is created for p, which is a core point, and all neighbours of p are added to the cluster. If any neighbour of p is a core point belonging to another cluster, the clusters are merged.

This algorithm solves the problem of the shape of the clusters that arises with prototype clustering, since the cluster membership propagates along the paths of nearby core points. Figure 18.4 shows the result of clustering with the DBSCAN algorithm. The blue and green points are assigned to the two different clusters, while the black points are considered noise and not assigned to any cluster.

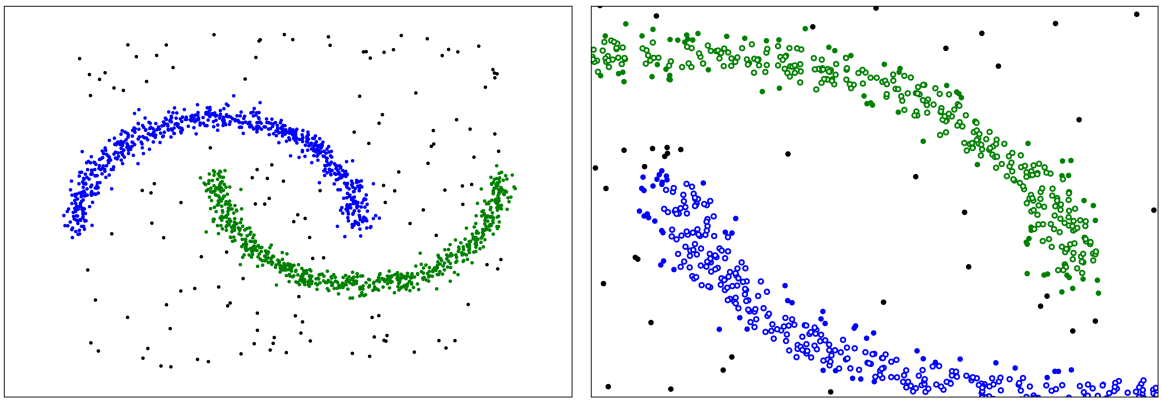

Figure 18.4: DBSCAN clustering. The right panel shows the core points (white centers), the non core points assigned to clusters (filled, green and blue) and points not assigned to clusters, which are considered noise, in low density regions (black).

The `cluster` module of the Scikit-Learn library offers a `DBSCAN` class for clustering with this algorithm.

18.3 Clustering Validation

In supervised learning we can always evaluate the results by measuring the error between the predictions and the data labels. But with clustering we may not have that possibility, if we use unlabelled data. So we may need to assess clusters by some measure of the "goodness" of the clustering. This may be necessary in different contexts, such as to determine if the data has structure, if the right number of clusters were predicted, if the clusters fit the data structure and if they fit some external information, such as class labels, if such information is available. It may also be necessary to evaluate clustering to compare different sets of clusters. Note that, in the following subsections, the similarity measure can be substituted by a dissimilarity or distance measure. The only difference is that the meaning of lower or higher scores will be reversed, as a high score using a similarity measure will, for the same clusters, be a low score using a distance measure, and the quality of the clusters may be proportional or inversely proportional to the score depending on the functions used.

Cluster cohesion

Cluster cohesion is a measure of the similarity of points within each cluster. For each cluster, the cluster cohesion score is:

$$cohesion(C_i) = \sum_{x \in C_i} \sum_{y \in C_i} S(x, y)$$

For prototype-based clusters, the distance or similarity can be measured with respect to the prototype of each cluster:

$$cohesion(C_i) = \sum_{x \in C_i} S(x, \mathbf{c}_i)$$

Cluster separation

Cluster separation measures the similarity, or dissimilarity, of examples between different clusters, summing the similarity measure between all pairs of points in different clusters.

$$separation(C_i, C_j) = \sum_{x \in C_i} \sum_{y \in C_j} S(x, y)$$

For prototype-based clusters, the distance or similarity can be measured with respect to the prototype of each cluster:

$$separation(C_i, C_j) = S(\mathbf{c}_i, \mathbf{c}_j)$$

Sum of Squared Errors

For prototype-based clustering, we can define a sum of squared errors as the sum of the distance to the prototype or, more often, the sum of the squared distance.

$$SSE = \sum_{k} \sum_{x \in C_k} dist(x, \mathbf{c}_k)$$

Silhouette Score

Given $a(i)$ as the average distance between point i and all other points in the same cluster and $b(i)$ as the average distance of point i to all points in the nearest cluster, with the nearest cluster being the one with the smallest average distance to i, the silhouette score for point i is the fraction:

$$s(i) = \frac{b(i) - a(i)}{\max\left(a\left(i\right), b\left(i\right)\right)}$$

Averaging over all points, we can obtain the silhouette score for the clustering, a value ranging from -1 to 1 with a higher value the more the distance between clusters, $b(i)$, is greater than $a(i)$. The silhouette score is available on the Scikit-Learn library as the `silhouette_score` function in the `metrics` module. Figure **??** shows the silhouette score for different clusterings using the k-means algorithm with different numbers of clusters and data sets.

When working with labelled data, in supervised learning, we can also evaluate the clusters by comparing cluster to the known structure of the data. While the previous scores we saw depend on *internal indices*, computed solely from the clusters and the structure of the data, with labelled data it is possible to use this information to obtain *external indices* for evaluating clusters.

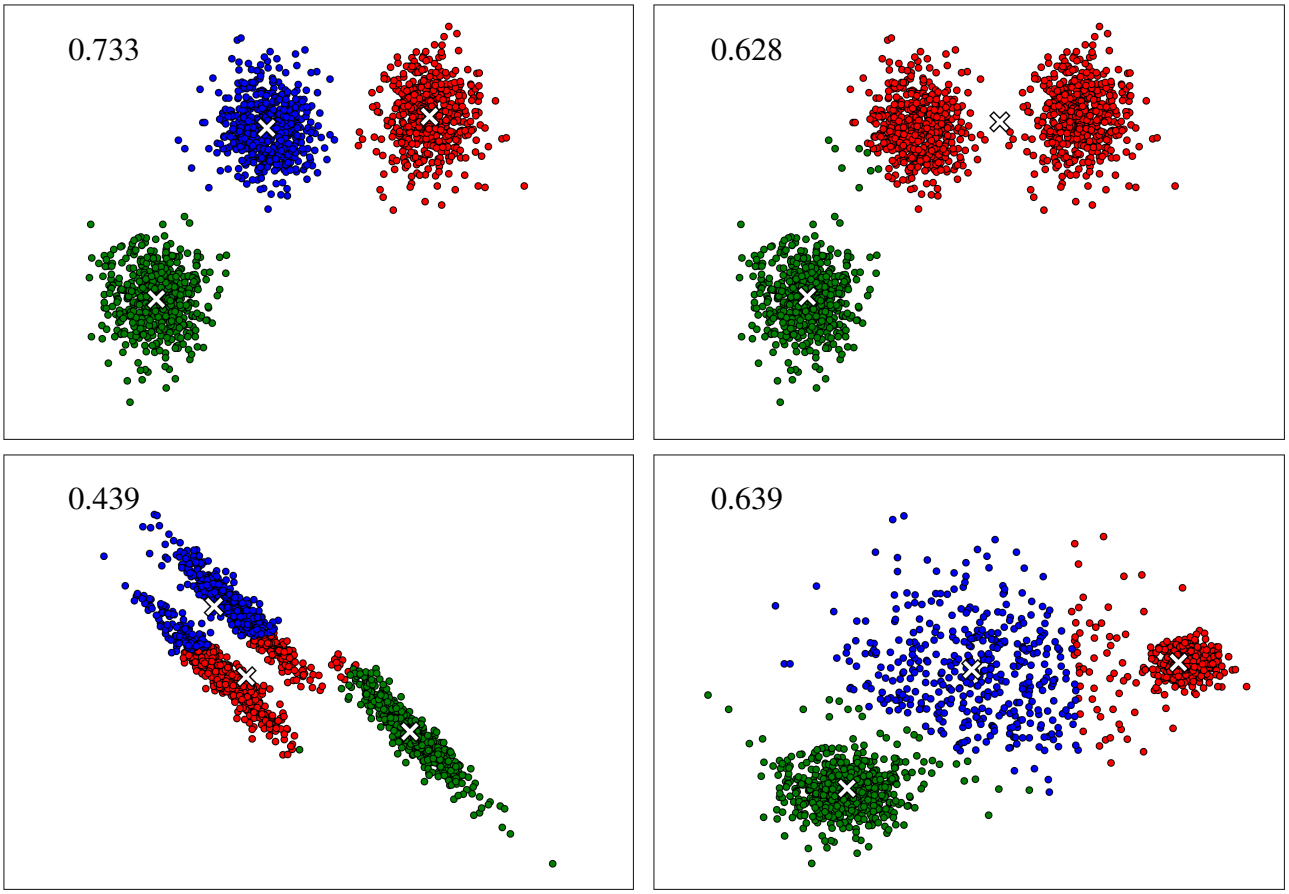

Figure 18.5: Silhouette score for different clusterings.

18.4 Further Reading

1. Frey *et. al.*, Clustering by passing messages between data points [11]
2. Ester *et. al.*, A density-based algorithm for discovering clusters in large spatial databases with noise. [10]
3. Scikit-learn documentation on clustering:`http://scikit-learn.org/stable/modules/clustering.html`

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.