Chapter 20

Probabilistic clustering

Fuzzy sets and clustering. Fuzzy c-means. Probabilistic Clustering: mixture models. Expectation-Maximization revisited.

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.

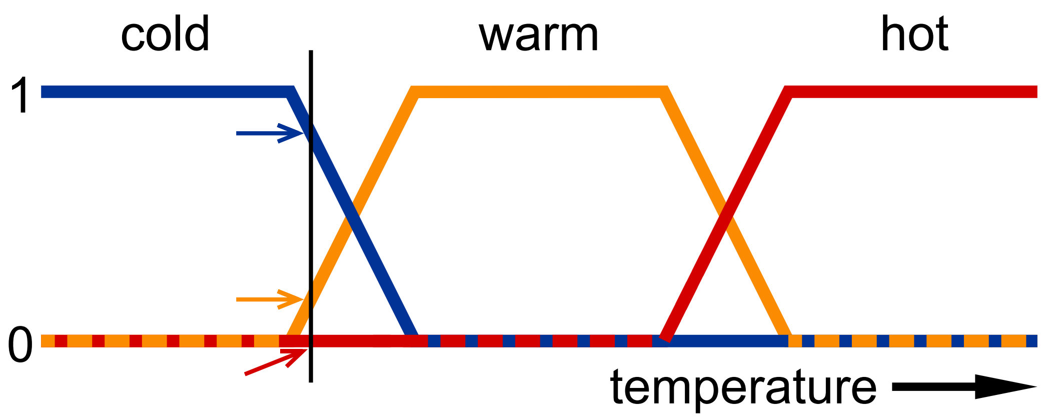

Figure 20.1: Fuzzy sets for cold, warm and hot temperatures, and respective membership values. Wikimedia, CC BY-SA 3.0 fullofstars

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:

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

and with respect to the centroids C_k:

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

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 the Expectation-Maximization method. First, the expected value of the latent variables, 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 likelihood 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.

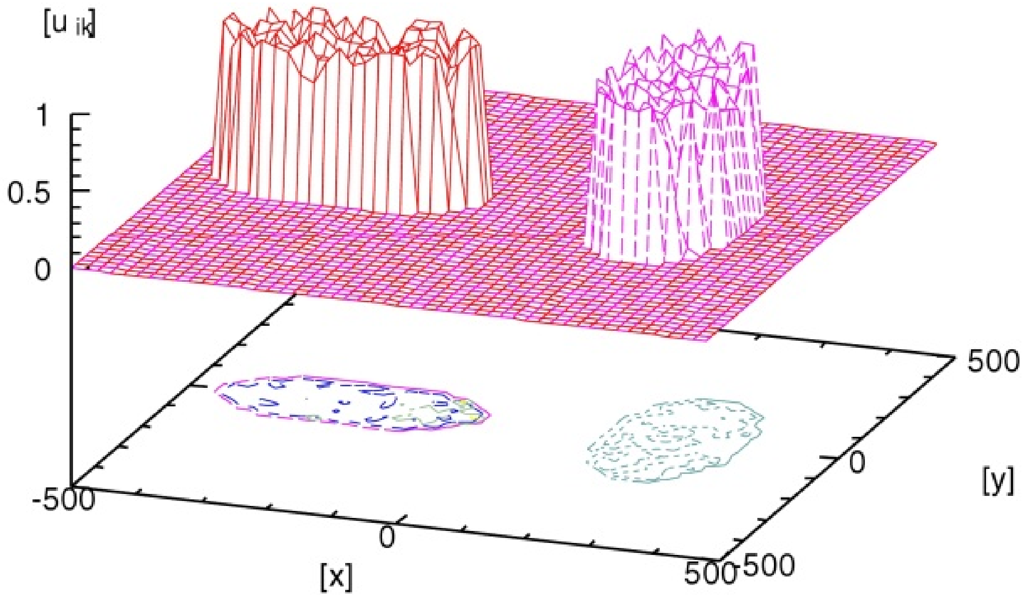

Figure 20.2: Fuzzy c-means example, from “Simulated Annealing - Advances, Applications and Hybridizations”, Ed. Marcos de Sales Guerra Tsuzuki, CC BY 3.0

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 Probabilistic Clustering

Suppose we have a probabilistic model for the distribution of our data, X, given some random distributions from which our data points are drawn, Y:

$$P(X,Y) = P(X|Y)P(Y)$$

This model would allow us to cluster examples from X from the probability of each point given each of the Y distributions and, from this joint probability, we would also be able to generate new points. The problem is to find Y.

Let us suppose that this abstract Y can be decomposed into a set of parameters θ and a set of latent (hidden) variables Z. The likelihood function for our parameters θ given complete information on X and Z would be:

$$L(\theta; X, Z) = p(X, Z|\theta)$$

We do not know Z, but we can use the Expectation-Maximization method to solve this problem iteratively, as we saw before.

Let us assume that our data set X was drawn from a set of Gaussian distributions. In one dimension, the Gaussian distribution is characterized by the means, μ, and standard deviation σ:

$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

We can create a mixture of Gaussian distributions by adding different Gaussian distributions with different weights, so that the weights all add up to 1. In more than one dimension, the Gaussian distributions are:

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

Where Σ is now the covariance matrix. A mixture model of k gaussians is:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

</p> where the π_k are the mixing coefficients, or the weight of each Gaussian distribution, and they have to be such that their sum is equal to one, so that the probabilities are normalized:

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leq \pi_k \leq 1 \; \forall k$$

For each data point $\mathbf{x}$, let us create a variable $\mathbf{z}$ so that:

$$z_k \in \{0,1\} \qquad \sum_k z_k = 1$$

In other words, $\mathbf{z}$ has a value of 1 in one of its dimensions and 0 in all other, with the dimensions corresponding to the Gaussian distributions. This means that $\mathbf{z}$ specifies from which Gaussian distribution the point $\mathbf{x}$ was drawn. We can also define the joint distribution of $\mathbf{x}$ and $\mathbf{z}$ to be $p(\mathbf{x},\mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$. The marginal probability of any z_k being equal to one, marginalized for all possible values of $\mathbf{x}$, is equal to the weight of the corresponding Gaussian:

$$p(z_k = 1) = \pi_k$$

Since $z_k = 1$ for a particular point means that point was drawn from distribution k, then the conditional probability of the points given $z_k = 1$ is the respective Gaussian:

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

This checks with our initial assumption regarding the probability of $\mathbf{x}$ as drawn from a mixture of gaussians:

$$p(\mathbf{x}) = \sum_z p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

but now we can explicitly include the latent variables $\mathbf{z}$ and thus solve this problem with the EM method. Using Bayes' rule, we know that the probability of $z_k = 1$ given the point $\mathbf{x}$ is:

$$p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

where the denominator is the marginal probability of the point $\mathbf{x}$, marginalized over all possible source distributions for that point. Given what we know about the probability distribution of $\mathbf{x}$, we can rewrite this as:

$$p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}$$

This is the posterior probability that Gaussian k is responsible for point $\mathbf{x}$:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} \tag{20.1}$$

The log-likelihood of our parameters π, μ and Σ, describing, respectively, the weights, means and covariance matrices of our Gaussian distributions in the mixture, is the log-probability of the observed data given the parameters:

$$\ln p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right)$$

The derivative of the log-likelihood function is zero at:

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)} \sum_k (\mathbf{x}_n - \mu_k)$$

Which we can rewrite as a function of μ_k using Equation 20.1:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n$$

Where N_k is the effective number of points in component k:

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Doing the same for Σ_k, we get this expression:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T$$

which is the sum of each covariance weighted by the corresponding $\gamma(z_{nk})$. Finally, for the parameters π_k:

$$\pi_k = \frac{N_k}{N}$$

This means that we can find the maximum likelihood solution of our parameters π, μ and Σ if we know the values of the $\gamma(z_{nk})$, which in turn depend both on X and our parameters. Once again, the solution is to use the Expectation-Maximization method. Starting from random values for π, μ and Σ, we iteratively recompute the $\gamma(z_{nk})$ and recompute the parameters until convergence. Figure 20.3 shows what happens to the Gaussian distributions after one, two and three passes of this method.

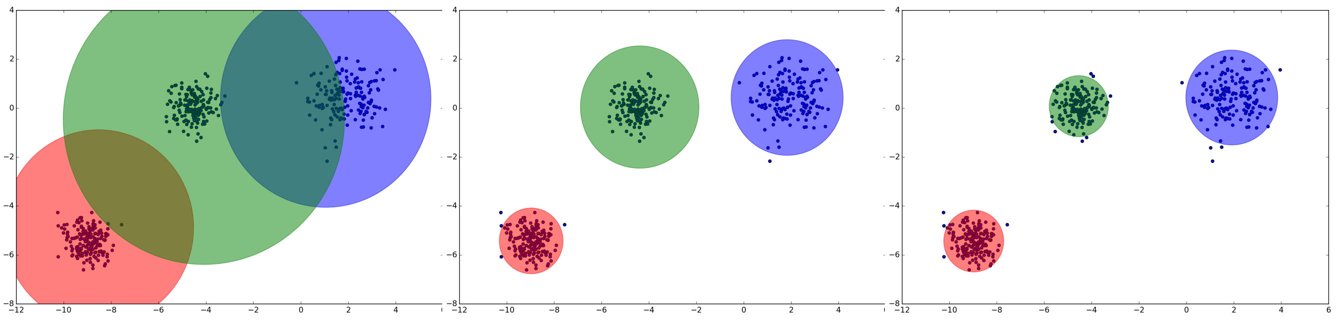

Figure 20.3: Three iterations of the EM method for the computation of a mixture model of 3 Gaussian distributions.

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:

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

20.4 Further Reading

1. Bishop [4], Section 9.2
2. Scikit-Learn demo on Mixture Models: http://scikit-learn.org/stable/modules/mixture.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.