Soft Clustering

So far, clustering was done using hard assignments (1 point → 1 cluster)
Sometimes this doesn't accurately represent the data: it seems reasonable to have overlapping clusters.
Using a soft assingment, where we're assigning a probability to belonging to a given cluyster might more accurately reflect the data than hard assignments.

Mixture Model

You can imagine that these two curves represent the distribution of weights of these two species. We are collecting weights from these different species. We just know that we are collecting weights. Our goal is to create a clustering that accurately represents the given data. We would naturally expect there to be two distinct clusters, but there is some overlap in the two curves. We want to capture the uncertainty of belonging to one group or another. We just know within a certain degree of confidence a point belongs to based on these distributions.

There's a lot of probabilities at play here, such as the probability of coming from one of these populations. We could have a population of species where 80% come from the red species and 20% come from the green, so there would be a greater chance of coming from the red species. Once we know which curve it comes from, we know that it's weight is distributed by the red distribution.

Suppose we have k different species. We have an inherent probability of coming from a certain species, and then we have the probability of your weight given that you come form that species. Given that you come from the green cluster, what is your weight distribution.

We call this the mixture proportion. Using this mixture model, what is the probability distribution of this certain point? It would be the probability of being in the red cluster times the probability of seeing this weight when we're in the red curve plus the probability of being in the green cluster times the probability of seeing this weight when we're in the green curve.

The Gaussian Mixture Model is where within C_i, the probability distribution of the mixture model is the normal distribution.

GMM Clustering

Address a much simpler problem first: estimating the probability distribution of flipping a coin. We want to find the parameters that describe the distribution, to find the GMM that maximizes

the distribution of the data we saw. The best we can do is to assume that we've seen a likely data set and to say, what are the parameters of the distribution that maximize the probability of the distribution given the data that we saw?

We know that each of H,T,T,H,T flips is a Bernoulli random variable with probability p, Bern(p). We don't know the parameter p that generated this distribution. We are estimating the parameter p given the data that we saw from the coin flip. For example, p hat = ⅗. We hope that given a big enough sample, we can estimate this parameter p hat that tries to maximize the probability of having seen the data that we saw. What's the probability of having seen the data that we saw? P(x=H) P(x=T) = p(1-p). Find the value of p that maximizes this probability distribution.

Now, for GMM, it will be more complicated because we don't just have one parameter. We have to find the parameters of the GMM that maximize the probability of having seen the data that we saw. We want the data we saw to be a good representation. On the next slide, we will see a product and a sum. Every point follows a Gaussian distribution and the probability of seein the data we saw is the product of the probabilities of observing each data point, assuming they are independent and identically distributed.

What are the parameters that uniquely characterize a distribution? We have k mixture components so we have k means and k variances and the probability of it coming from a certain cluster. P(Ci) and Mu i and sigma i for all k components. How do we find the parameters that maximize something? To get the maximum of a function, we might try setting the derivative to 0, get the critical points, maybe set the second derivative to 0 to understand the critical points.

Call theta the set of all these parameters. Our goal is to find theta star, the set of parameters that maximize the following function. The argument that maximizes the following, hence arg max. The function is the joint distribution of data points of n Gaussian mixture distributions, assuming independence. It would just be the product of Gaussian distributions for each data point. Just like when we have the probability of H times the probability of T.

We would automatically go through the process of finding derivatives of this function. Taking the log of a given function does not change its critical points, so we will do that, which will give us a sum of logs, which is much easier to work with than the product of terms in the original function. We can solve the partial derivatives with respect to the different parameters we are interested in learning. If we solve for this, we get a formula for Mu j hat, Sigma j hat, and P(Cj) hat.

We are still missing the term P(Xi | Ci) in these formulas. We can use Baye's rule to compute this, but for that we will need P(Xi | Cj), which is uniquely characterized by our Mu and Sigma parameters, and we will need P(Cj). Thus, we are stuck in a loop where we are trying to find the parameters, but we need the parameters in our process of doing so. What do we do?

Expectation Maximization Algorithm. We start with a random theta. Then, we compute P(Cj | Xi) for all Xi by using theta. We can use that value to update our theta value, our parameters mu, sigma, and probability of being in Cj. We can repeat that until convergence.

Clustering Aggregation

Clarification: When we say clustering, we are talking about the output from a clustering algorithm. A cluster is a set of points.

We want to be able to meaningfully compare the output of clustering algorithms. Also, maybe we can combine the strengths of these clustering algorithms to create a super clustering that can cluster everything well.

Comparing Clusterings

We have too many cost functions, and one might not be appropriate for a certain other clustering. The output for the error function of one clustering method might not be a good metric for another evaluating clustering method. We need to compare clusterings by looking at their assignment of points to clusters.

This is difficult because there's no convention for how we name our clusters. If suppose we have an output from two different clustering algorithms, C1 and C2, and we have data x1, x2, x3, x4, Clustering 1 could say they belong in red, blue, blue red cluster, and clustering 2 could say they belong in clusters 1, 2, 3, 4. Even if we enforce a convention, we could still have 2, 1, 1, 2 clusters and 1, 2, 3, 4 clusters. Thus, we can't just iterate over the clusters we create because there's no convention for which points we have in what cluster and what cluster is that. It's nondeterministic. Our trick is to look at which points are in what cluster, rather than look at which clusters. For each pair of points, we're going to look at whether they should be clustered together. x1 and x3 are clustered together in C1 and C2. We can start to build an understanding of which points are clustered together, which is all we need.

We're going to define a distance function between these two clusterings that tells us whether or not they agree whether pairs of points should be clustered together or not. This is equivalent to our original question of are you clustering points the same way? For each pair of points, they either agree or disagree on whether or not they should be clustered together. This distance function is called the disagreement distance. It is the sum of the indicator distance I. I is 1 if P and C disagree on which clusters x and y belong to. It is 0 otherwise. If there are many disagreements, the value of the distance will be large, and the inverse is also true. The trick is to show that this works for the triangle inequality.

In the example, we count the total number of disagreements between P and C in the given data samples. This is the disagreement distance between P and C. In the next slide, we formally verify that the disagreement distance is a distance function. 1. D(C, P) = 0 iff C = P. 2. D(C,P) = D(P, C). 3. Triangle Inequality. This is the trickier part. We wanna show that $I_{C1, C3}$ (x, y) ≤ $I_{C1, C2}$(x, y) + $I_{C2, C3}$ (x, y). The only way for this to be false is if C1 and C3 disaagree, but both of C1, C2 and C2, C3 agree. This can never be the case because if C1 and C2 agree and C2 and C3 agree, then C1 and C3 agree, which contradicts the fact that C1 and C3 disagree. Thus, we know that it upholds triangle inequality.

The goal of aggregate clustering is, from a set of clusterings C1, …, Cm, to generate a clustering C* that minimizes the sum of the disagreement distances of these clusterings C1, ... , Cm. This means the result clustering will be closer to all of these clusterings on average. The question is, how do we find this minimum? One thing to note is that this progblem is equivalent to clustering categorical data. If you have categorical data, that is itself, a clustering of your data in a way. The features City, Profession, Nationality provide clusterings of our data. For example, all entries with city Boston.

The benefits of aggregate clustering are that it can identify the best number of clusters, it can handle/ detect outliers, it can provide a more robust clustering by utilizing the strengtrhs of all the different clustering methods that we've seen so far, and interestingly, it can provide a privacy preserving clustering. You can send data to a company without identifiers, but just some clustering, and we can all share data where information is stripped, but we can still see which points belong to which cluster. We can share this data without accessing the data itself, but just the labels or clusters that we assigned each data point.

The problem is NP-Hard. Not only are we doing all possible pairs of points, but we are finding the optimal. We have to go about it with approximations and brute force. Don't shy away from implementing an NP-Hard algorithm. One interesting note is, can't we just ask all our clusterings, what clustering should they be in, and if a majority say they belong in the same cluster, then maybe we assign them to that cluster? Why would that work or not work? It would not work because it would not always produce a clustering. You can ask them should they be clustered together, and they can all bring this inconsistent result where it's not always a clustering. For example, you can have a majority saying x1 and x2 together, x2 and x3 together, x1 and x3 separate.