clustering-kmeans-em

October 28, 2024

1 Clustering, K-means, Expectation Maximization

1.1 *Vocabulary*

na

2 Lecture Notes

2.1 *K-Means*

2.1.1 Clustering

Clustering is an important **unsupervised** learning task. The **goal** is to partition the dataset into similar groups. **Use cases** include categorizing news stories for easier sorting and locating by readers, or discovering distinct groups in a customer base for marketing campaigns.

2.1.2 K-means Algorithm

Context for K-means K-means takes n objects, where each x_i could be a vector itself where each coordinate represents different information. For example, x_i^1 might represent the dollars this particular customer spends on electronics, whereas x_i^2 might represent dollars spent by this customer in another category.

We also have K, which is the number of clusters we want to partition the data points into. K must be given in K-means algorithm, so if you don't know the best K, you will have to experiment and choose the best one. There are also data driving approaches that allows us to estimate the best K automatically, however these appraoches will not be covered in class.

With the data and K given, we want to decide how to partition the data. We also want to determine a centroid for each of the clusters. The **centroid** is the representative data point for each cluster. Empirically, the centroid will be the average or geometric center within the cluster.

Goal of K-means So, the K-means algorithm must jointly determine the assignment and the centroids that provides an optimal partition of the data.

High Level Algorithm The K-means algorithms works by using the idea that we don't know the centroid and the assignments, so we start from **random initialization**. After randomly initializing either the centroids or the assignments, we iteratively update them so that they can be improved sequentially. These steps will be repeated until the the solution converges.

Illustrative Example Here we have a simple dataset of 3 points in red. We wan to classify these three data points using clustering into 2 groups, $K = 2$. The process is as follows:

1. Start with random initialization of the centroids. This location will of course not be a reasonable estimation of the centroids.
2. Perform the assignment step of the data points to centroids, then update the centroids. Perform this iteratively until convergence.

Formal Algorithm Recap **Input:** Data points $x_1, \dots, x_n$, number of clusters K.

Algorithm: - Randomly place K points as the centroids - Iterate until convergence:

1. Assign each data point to the closest centroid, where $z_i \in \{1, \dots, K\}$ is the centroid that data point is assigned to, and μ_k represents the center of k's cluster.

$$z_i = \underset{k=1,\dots,K}{argmin} ||x_i - \mu_k||$$

2. Recalculate the position of the K centroids, where S_k is the set of data points belonging to centroid k, and μ_k is the center of the centroid. Here we are assigning mu_k to thea average mean of the datapoints assigned to centroid k.

2.1.3 K-means as Optimization

Defining K-means as an Optimization Problem K-means is really just an optimization algorithm (something like coordinate descent), a very nice one, that guarantees convergence.

Given ${x_i}_{i=1}^n$, we want ${\mu_k}_{k=1}^K$, ${z_i}_{i=1}^n$. We will refer to the set of centers of the centroids as μ, and he set of data points belonging to each centroid as z.

We need to define an objective function (loss function) that defines how well the data is partitioned:

$$L(\mu, z) = \sum_{i=1}^{n} ||x_i - \mu_{z_i}||^2$$

Remember that z_i is the index of the centroid assigned to x_i.

So, viewing K-means as an optimization problem, our goal is to solve:

$$\min_{\mu,z} L(\mu, z)$$

Which is actually a mized optimization problem, since μ is continuous ($\mu \in \mathbf{R}^{d \times K}$) and z is discrete ($z \in \{1, \dots, K\}^n$), so it's a tricky optimization problem. It is not a convex optimization problem, and in fact there will be many local minima. However, the K-means algorithm we just introduce actually tries to exactly minimize the optimization problem by using the idea of coordinate descent, which allows us to update μ and z alternatively.

Coordinate Descent The steps of coordinate descent are: - Initialize μ - Repeat: - Update z, with fixed μ - Update μ, with fixed z

Solving mathematically:

2.1.4 Things to Pay Attention to for This Problem

Local vs. Global Minima The loss function for this problem is not convex, meaning there will be a lot of local minumums. Since we are starting with random initialization, we may just end up finding a random local optimal solution when running K-means. (Finding the global minimum is a very difficult problem). This means that K-means can be **very sensitive to the initialization**. In practice, you can try several different initializations, and choose the best solution that minimizes the loss.

Coordinate Descent Minimization Guarantee The coordinate descent algorithm is actually guaranteed to find the local optimal solution. This is due to the nature of alternatively solving for μ and z. Since we are finding the minimum solution while the other coordinate is fixed, the proceude can only decrease the loss function. Thus, we can say that coordinate descent monotonically decreases.

2.2 Expectation Maximization

2.2.1 Context

EM vs. K-means Expectation maximization is another clustering algorithm which **generalizes the K-means algorithm**. EM solves the problem in K-means where, for data points which are tied between multiple centroid distances during assignment, you must randomly choose which centroid to assign the point to. **EM allows us to make probabilistic assignments of the data points**.

In EM, we make probabilistic assignments by calculating the probability of $z_i = k$ for any particular value instead of making a deterministic assignment for each data point.

Probabilistic Assignment of Data Points The probability calulated for each z_i is taken to be proportional to distance, an example being taking the assignment using some type of sigmoid function (here lambda is some constant that is decided in the complete EM algorithm):

$$Prob(z_i = k) = \frac{exp(-||x_i - \mu_k||^2/\lambda)}{\sum_{i=1}^{K} exp(-||x_I -]\mu_k||^2/\lambda)}$$

Probabilistic Update of Centroids Since we are defining randomizd assignments and we only maintain our probability, we must modify the mean to be a **weighted** average. The calculation of μ_k for K-means can be written as an indicator function:

$$\mu_k = \frac{\sum_{i=1}^{n} \mathbb{I}(z_i = k)x_i}{\sum_{i=1}^{n} \mathbb{I}(z_i = k)}$$

For EM, we replace the deterministic indicator function with a soft probabalistic indicator:

$$\mu_k = \frac{\sum_{i=1}^{n} Prob(z_i = k) x_i}{\sum_{i=1}^{n} Prob(z_i = k)}$$

Example First we will calculate the probabilities that the data points belong to each centroid:

After we have these probabilites, we can update the centroids. We do a weighted average using the probaility that each data point belongs to z_i:

This process will be iterated until it converges.

Result using EM vs. K-means We can see that the outcome will be different if we use EM or K-means, since K-means is uses deterministic (sometimes random) assignments.

Visualization

2.2.2 Subsection

3 Personal Notes