Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.S064 Introduction to Machine Learning

Phase 1: clustering (lecture 6)

In this lecture, we will continue our discussion of the K-means algorithm. In particular, we will discuss two issues crucial for finding good clusters: an initialization procedure and a method for selecting the number of clusters. We begin by introducing a slight variant of the K-means algorithm.

1 The K-medoids algorithm

We had previously defined the cost function for the K-means algorithm in terms of squared Euclidean distance of each point $x^{(i)}$ to the closest cluster representative. We showed that, for any given cluster, the best representative to choose is the mean of the points in the cluster. The resulting cluster mean typically does not correspond to any point in the original dataset. The K-medoids algorithm operates exactly like K-means but, instead of choosing the cluster mean as a representative, it chooses one of the original points as a representative, now called an *exemplar*. Selecting exemplars rather than cluster means as representatives can be important in applications. Take, for example, Google News, where a single article is used to represent a news cluster. Blending articles together to evaluate the "mean" would not make sense in this context. Another advantage of K-medoids is that we can easily use other distance measures, other than the squared Euclidian distance.

The K-medoids objective is very similar to the K-means objective:

$$Cost(C^1, \ldots, C^k, z^{(1)}, \ldots, z^{(k)}) = \sum_{j=1}^{k} \sum_{i \in C^j} d(x^{(i)}, z^{(j)}) \tag{1}$$

The algorithm:

1. Initialize exemplars: $\{z^{(1)}, \ldots, z^{(k)}\} \subseteq \{x^{(1)}, \ldots, x^{(n)}\}$ (exemplars are k points from the original dataset)
2. Repeat until there is no further change in cost:
   (a) for each j: $C^j = \{i : x^{(i)}$'s closest exemplar is $z^{(j)}\}$
   (b) for each j: set $z^{(j)}$ to be the point in C^j that minimizes $\sum_{i \in C^j} d(x^{(i)}, z^{(j)})$

In order to update $z^{(j)}$ in step (b), we can consider each point in turn as a candidate exemplar and compute the associated cost. Among the candidate exemplars, the point that produces the minimum cost is chosen as the exemplar.

2 Initialization

In the previous lecture, we demonstrated that the K-means algorithm monotonically decreases the cost (the same holds for the K-medoids algorithm). However, K-means (or K-medoids) only guarantees that we find a local minimum of the cost, not necessarily the optimum. The quality of the clustering solution can depend greatly on the initialization, as shown in the example below[1]. The example is tailored for K-means.

Given: N points in 4 clusters with small radius δ, and a large distance B between the clusters. The cost of the optimal clustering will be $\approx O(\delta^2 N)$. Now consider the following initialization:

After one iteration of K-means, the center assignment will be as follows:

This cluster assignment will not change during subsequent iterations. The cost of the resulting clustering is $O(B^2 N)$. Given that B can be arbitrary large, K-means produces a solution that is far from the optimal.

One failure of the above initialization was that two centers were placed in close proximity to each other (see cluster 3) and therefore many points were left far away from any center/representative. One possible approach to fixing this problem is to pick k points that are far away from each other. In the example above, this initialization procedure would indeed yield one center per cluster. But this solution is sensitive to outliers. To correct this flaw, we will add some randomness to the selection process: the initializer will pick the k-centers one at a time, choosing each center at random from the set of remaining points. The probability that a given point is chosen as a center is proportional to that point's squared distance from the centers chosen already thereby steering them towards distinct clusters. This initialize procedure is known as the K-means++ initializer.

K-means++ initializer
pick x uniformly at random from the dataset, and set T=x
while $|T| \leq k$
pick x at random, with probability proportional to the cost $\min_{z \in T} ||x - z||^2$
$T = T \cup \{x\}$

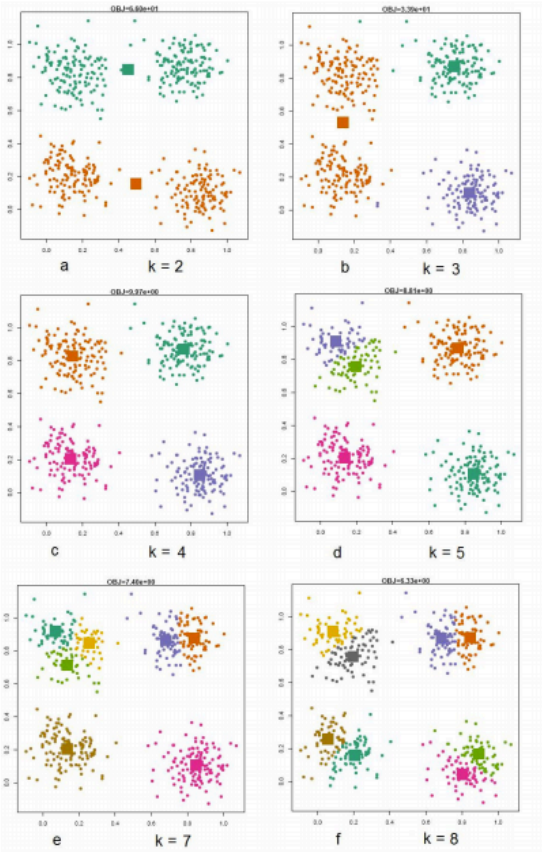
We can offer some guarantees for this initialization procedure:

[1] This example is taken from Sanjoy Dasgupta.

K-means++ guarantee Let T be the initial set of centers chosen by K-means++. Let T^* be the set of optimal cluster centers. Then, $E|cost(T)| \leq cost(T^*)O(logK)$, where the expectation is over the randomness in the initialization procedure, and $cost(T) = \sum_{i=1}^{n} \min_{z \in T} \|x^{(i)} - z\|^2$

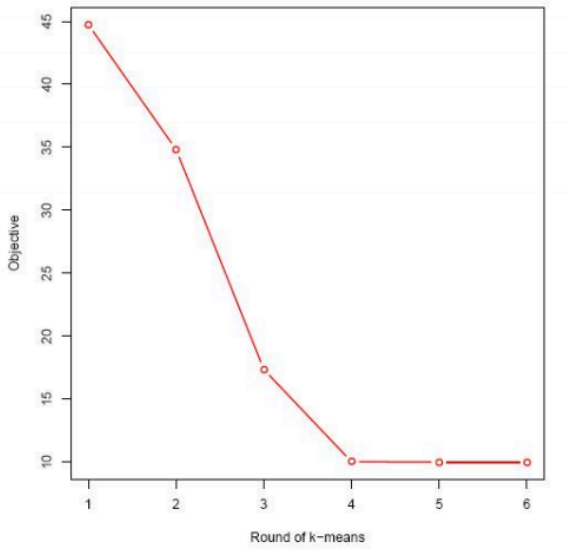
3 Choosing K

The selection of k greatly impacts the quality of your clustering solution. Figure 3 shows how the output of K-means changes as a function of k. In some applications, the desired k is intuitively clear based on the problem definition. For instance, if we wish to divide students into recitations based on their interests, we will choose k to be 4 (the number of recitation times for the class). In most problems, however, the optimal k is not given and we have to select it automatically.



Let's start by understanding the connection between the number of clusters k and the cost function. If every point belongs to its own cluster, then the cost is equal to zero. At the other extreme, if all the points belong to a single cluster with a center z,

then the cost is the maximum: $\sum_{i=1}^{n} ||x^{(i)} - z||^2$. Figure 3 shows how the cost decreases as a function of k. Notice that for $l = 4$, we observe a sharp decrease in cost. While the decrease continues for $k \geq 4$, it levels off. This observation motivates one of the commonly used heuristics for selecting k, called the "elbow method". This method suggests that we should choose the value of k that results in the highest relative drop in the cost (which corresponds to an "elbow" in the graph capturing as a function of k). However, it may be hard to identify (or justify) such a sharp point. Indeed, in many clustering applications the cost decreases gradually.



There are a number of well-founded statistical criteria for choosing the number of clusters. These include, for example, the *minimum description length principle* (casting clustering as a communication problem) or the *Gap statistics* (characterizing how much we would expect the cost to decrease when no additional cluster structure exists). We will develop here instead a simpler approach based on assessing how useful the clusters are as inputs to other methods. In class, we used clustering to help semi-supervised learning.

In a semi-supervised learning problem we assume that we have access to a small set of labeled examples as well as a large amount of unannotated data. When the input vectors are high dimensional, and there are only a few labeled points, even a linear classifier would likely overfit. But we can use the unlabeled data to reduce the dimensionality. Consider, for instance, a document classification task where the goal is to label documents based on whether they involve a specific topic such as ecology. As you have seen in project 1, a typical mapping from documents to feature vectors is bag-of-words. In this case, the feature vectors would be of dimension close to the size of the English vocabulary. However, we can take advantage of the unannotated documents by clustering them into semantically coherent groups. The clustering would

not tell us which topics each cluster involve, but it would put similar documents in the same group, hopefully placing documents involving pertinent topics in distinct clusters. If so, knowing the group to which the document belongs should help us classify it. We could therefore replace the bag-of-words feature vector by one that indicates only to which cluster the document belongs to. More precisely, given k clusters, a document that belongs to cluster j can be represented by a k dimensional vector with the j-th coordinate equal to 1 and the rest set to zero. This representation is admittedly a bit coarse – all the documents in the same cluster will be assigned the same feature vector, and therefore end up with the same label. A bit better representation would be to compile the feature vectors by appending relative distances of the document to the k clusters. In either case, we obtain low dimensional feature vectors that can be more useful for topic classification. At the very least, the lower dimensionality will guard against over-fitting. But how to choose k in this case? Note that none of the training labels were used to influence what the clusters were. As a result, we can use cross-validation, with the k dimensional feature vectors, to get a sense of how well the process generalizes. We would select k (the number of clusters) that minimizes the cross-validation error.