December 3, 2017 Lecture 22, Nov. 27, 2017

Unsupervised Learning - *k-means*

Lecturer: Justin Domke *Scribe: Annamalai Natarajan*

1 Summary

Last time we wrapped up Bayesian Inference with MCMC sampling algorithm. In this lecture, we discuss clustering algorithms that fall under unsupervised learning which is also the last module in this course. In particular, we discuss the *k-means* clustering algorithm.

2 Unsupervised Learning - clustering

The goal of clustering algorithms in unsupervised learning given N data examples $x_1, x_2, ..., x_n$ is to assign an integer label from the set $K \in \{1, 2, , , , , k\}$.

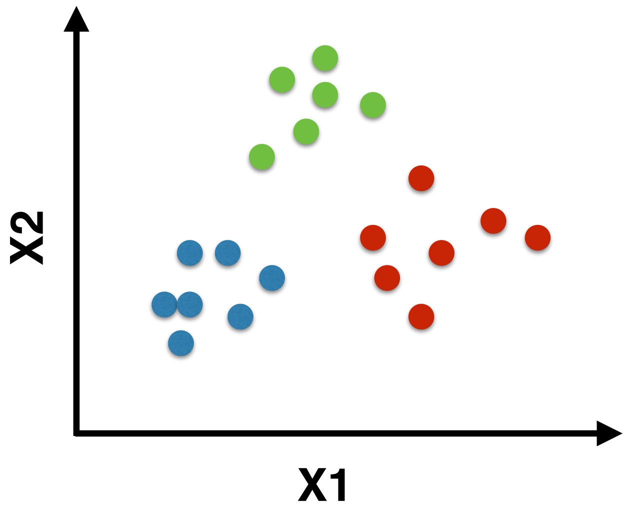
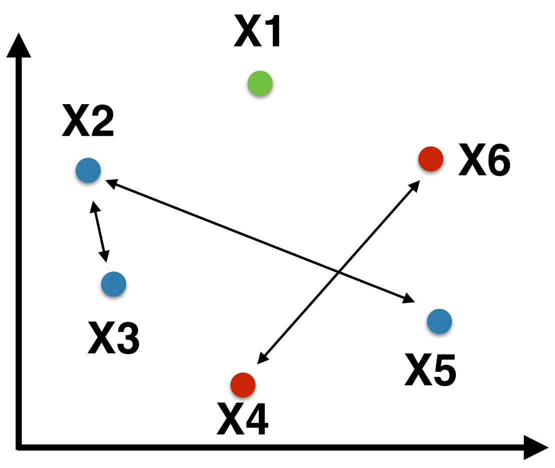


Figure 2.1: Sample clustering for 2 dimensional data. Three colors indicates three different clusters

For example in Figure 2.1 we are given data examples in 2 dimensions and one possible assignment is to label the data as belonging to clusters 1, 2 and 3 respectively. Another example would be to perform image compression, where we are given an image in which each pixel is represented as an integer triplet between 0 and 255 corresponding to R,G,B values. This image requires 24 bits per pixel which can be reduced if we represent each pixel by the mean of the cluster to which it belongs. If there are $k = 2^b$ clusters then each pixel will require b bits to denote cluster assignments and will also require $K \times \log_2(256) \times 3$ bits (assuming cluster centers are represented in integers) to represent cluster centers (centroids or means).



i	C(i)	Comparison
1	1	itself
2	2	x3 and x5
3	2	x2 and x5
4	3	x6
5	2	x2 and x3
6	3	x4

Figure 2.2: Sample within cluster scatter computation for 2 dimensional data

A critical question here, unlike supervised machine learning where we had access to labels y, is how to evaluate label assignments. One approach is to take the cluster assignments and compute the within cluster scatter as,

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i:C(i)=k} \sum_{i':C(i')=k} d(x_i, x_{i'}) \tag{2.1}$$

where, d is some distance metric that computes the distance between pairs of examples. This within scatter distance is essentially computing the distance between all pairs of examples within each cluster over all examples in the dataset. For example consider the six examples in two dimension in Figure 2.2. The colors indicate the cluster assignments and the edges between the examples indicate the pairings used to compute the within scatter distance as listed in the Table in Figure 2.2.

Given this objective function the goal is to minimize the within cluster distance. One approach is to repeatedly randomly flip the cluster assignments (*i.e. colors in Figure 2.2*) and pick the assignment that resulted in lowest within cluster scatter. But this is *NP*-hard to compute and is likely to get struck in local optima (a solution that is optimal in the neighborhood of possible cluster assignments but not globally the best solution that one could achieve). It is possible to achieve reasonably good results if we make some assumptions on the distance function d.

3 *k-means*

k-means is a type of unsupervised clustering algorithm which sets the distance function, d, to be $\|x_i - x_i'\|_2^2$ – which is the square of the ℓ_2 norm between pairs of examples in the same cluster. We rewrite the within cluster scatter as,

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i:C(i)=k}\sum_{i':C(i')=k}\|x_i - x_i'\|_2^2 \tag{3.1}$$

Suppose we assume that the distance function $d = \|x_i - x_i'\|_2^2$ and $\bar{x} = 0$ then,

Lemma 3.1. *We assume without loss of generality that $\bar{x} = 0$*

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\|x_i - x_j\|_2^2 = 2N\sum_{i=1}^{N}\|x_i - \bar{x}\|_2^2 \tag{3.2}$$

$$\tag{3.3}$$

The proof of Lemma 3.1 is as follows,

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=1}^{N}\|x_i - x_j\|_2^2 &= \sum_{i=1}^{N}\sum_{j=1}^{N}\|x_i\|_2^2 - 2x_i^T x_j + \|x_j\|_2^2 &(3.4)\\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\|x_i\|_2^2 + \|x_j\|_2^2 &(3.5)\\ &= 2N\sum_{i=1}^{N}\|x_i\|_2^2 &(3.6)\end{aligned}$$

where, $2x_i^T x_j$ term reduces to zero since $\sum_{j=1}^{N} x_j = \bar{x} \times N = 0$ since we assumed that $\bar{x} = 0$.

Applying this lemma to within cluster scatter we get,

$$\begin{aligned}W(C) &= \frac{1}{2}\sum_{k=1}^{K}\sum_{i:C(i)=k}\sum_{i':C(i')=k} d(x_i, x_{i'}) &(3.7)\\ &= \frac{1}{2} \times 2\sum_{i=1}^{K} N_k \sum_{i:C(i)=k} d(x_i, x_{i'}) &(3.8)\\ &= \sum_{i=1}^{K} N_k \sum_{i:C(i)=k} \|x_i - \bar{x}_k\|_2^2 &(3.9)\end{aligned}$$

where $N_k = \sum_{i=1}^{N} I[C(i) = k]$ and $\bar{x}_k = \frac{1}{N_k}\sum_{i:C(i)=k} x_i$. The interpretation of the above equation is that it is computing the average distance between each element from the its respective cluster centroid. This holds for this specific distance metric d only. Given this representation the goal of *k-means* is to find the best c that minimizes the $W(C)$. This is mathematically represented as,

$$\arg\min_{C} W(C) \tag{3.10}$$

$$\arg\min_{C} \sum_{i=1}^{K} N_k \sum_{i:C(i)=k} \|x_i - \bar{x}_k\|_2^2 \tag{3.11}$$

There are $|K|^N$ possible labellings. But we make the observation that for examples belonging to each cluster we have,

$$\arg\min_{m} \sum_{C(i)=k} \|x_i - m\|_2^2 = \bar{x}_k \tag{3.12}$$

where m is the cluster center. The term on the LHS can be minimized by setting m to be the respective cluster centers, $\bar{x}_k$. This reduces the problem of minimizing both c and $\bar{x}_{1,...,K}$ into minimizing C and $\bar{x}_1$,...,$\bar{x}_K$ separately – this leverages the fact that computing the cluster center is only dependent on examples that belong to that specific cluster. Hence this simplifies the problem as,

$$\min_{C} \min_{m_1}, ... \min_{m_K} \sum_{C(i)=k} \|x_i - m_k\|_2^2 \tag{3.13}$$

This allows us to decouple m and C thus allowing us to optimize C (or m) by fixing m (or C) respectively. We minimize the within cluster scatter,

$$W(C) = \sum_{i=1}^{K} N_k \sum_{i:C(i)=k} \|x_i - \bar{x}_k\|_2^2 \tag{3.14}$$

as,

$$W(C, m_1, , , , , m_K) = \sum_{i=1}^{K} N_k \sum_{i:C(i)=k} \|x_i - m_k\|_2^2 \tag{3.15}$$

The *k-means* algorithm is shown in Algorithm 1

Algorithm 1 *k-means* Algorithm

```
1: while not converged do
2:     Fix $m_1, m_2, ..., m_K$ and set $C(i) = \arg\min_{1\leq k\leq K} \|x_i - m_k\|_2^2$ for all $i$
3:     Fix $C$ and set $m_k = \frac{1}{N_k}\sum_{i:C(i)=k} x_i$ for all $k$
4: end while
5: return $C(i)$
```

The algorithm terminates when there is no change in the label assignments between two consecutive iterations. This algorithm proceeds by first fixing the cluster means and minimizing C and then optimizes for cluster centers, m, while fixing C. Alternately, one can reverse the order of these two steps and get identical results.

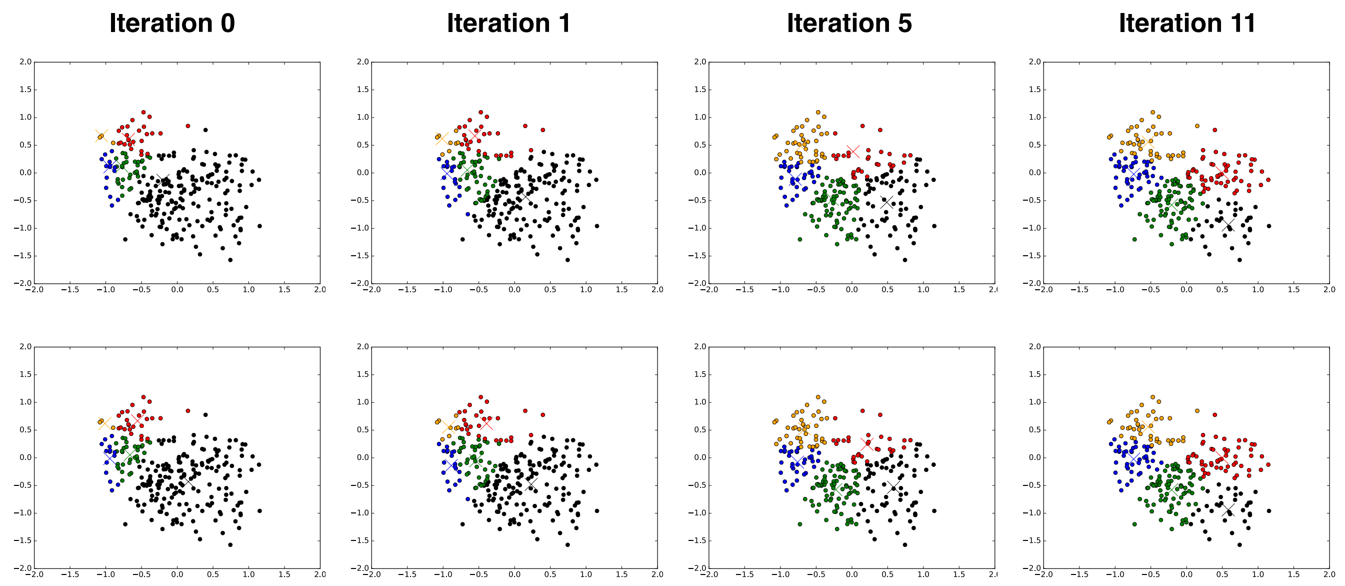


Figure 3.1: Snapshot of four iterations of *k-means* algorithm. The algorithm was initialized with randomly chosen data examples to be cluster centers. The top row plots the results after updating cluster identities C and the bottom rows plots the results after updating cluster centers $m_1, ..., m_K$. The five different colors represent cluster labels and the colored 'X' represent the cluster centers.

In Figure 3.1, we show the cluster label assignments for four iterations in the *k-means* algorithm. The five different colors represent cluster labels and the colored 'X' represent the cluster centers. The algorithm was initialized with randomly chosen data examples to be cluster centers. Following this the algorithm alternates between assigning cluster labels and recomputing cluster centers. We make similar plots in Figure 3.2 when the two steps are reversed and also by initializing the algorithm with randomly chosen cluster labels $k \in 1, ..., K$ for each $C(i)$. Like we observe for this example the approach to first initialize cluster centers converges to cluster label assignments quickly (11 vs 17 iterations).

One concern with *k-means* algorithm comes from the computation of the cluster means in line #3 where N_k could be zero. The fix to this problem is to reinitialize *k-means* with different cluster centers.

4 *k-means++*

k-means++ is an extension of the *k-means* algorithm where the cluster centers are carefully chosen to be initialized. The claim is that this in principle leads to constant factor improvement. The typical approach is to choose a data example as the center for cluster 1, choose another data example which is atleast ϵ distance away from the first example as center for cluster 2, etc. In practice *k-means* gives better results than *k-means++*. For more on *k-means++* algorithm please refer to [1].

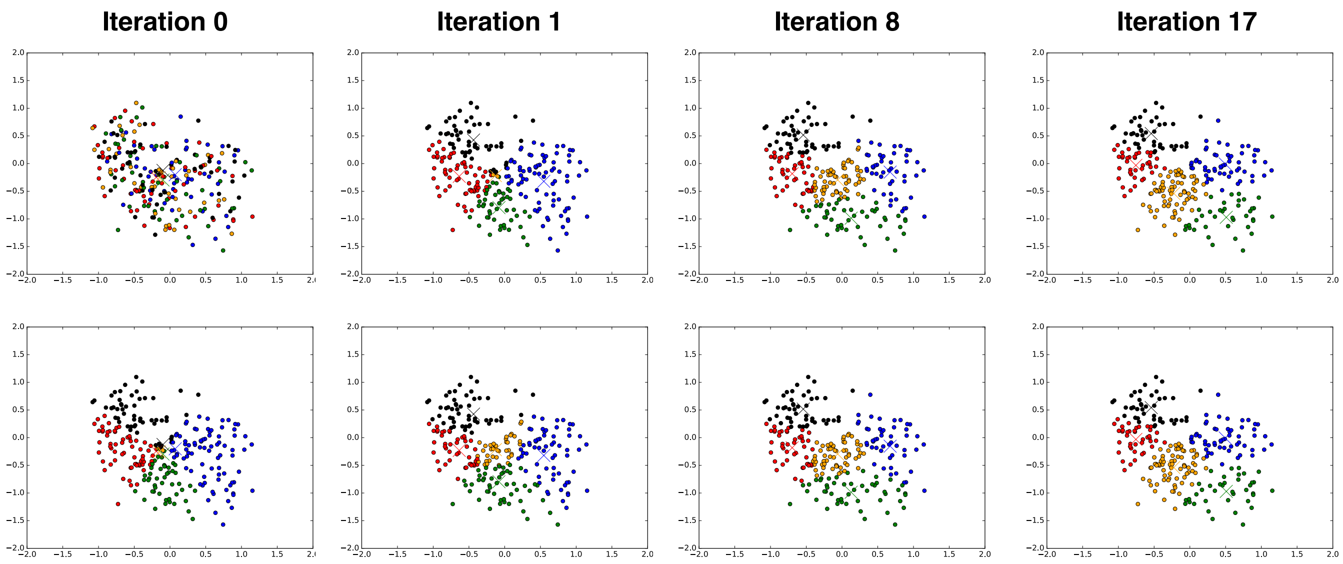


Figure 3.2: Snapshot of four iterations of *k-means* algorithm. The algorithm was initialized by randomly choosing a cluster label $k \in 1, ..., K$ for each $C(i)$. The top row plots the results after updating cluster centers $m_1, ..., m_K$ and the bottom rows plots the results after updating cluster identities C. The five different colors represent cluster labels and the colored 'X' represent the cluster centers.

References

[1] Arthur, D., and Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics (2007), 1027–1035.