University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 16: October 18

k-Means

Lecturer: Alexander Schwing Scribe: Ashok Vardhan Makkuva

Recap:

In the lectures so far, we focused primarily on the supervised learning setting where we have access to the labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)_{i\in[n]}\}$. The main objective here is to model the relationship between the input features $\boldsymbol{x}_i$ and their corresponding labels y_i in a parametric way. For example, linear regression, logistic regression, SVM, neural networks, etc. are various instances of this paradigm. In order to learn the parameters of these models, we pose an optimization problem through the notion of a loss function and solve it using traditional algorithms such as gradient descent. Once the parameters are learned, the inference of the label y for a new input $\boldsymbol{x}$ is relatively straightforward in regression and in classification settings when the label space $\mathcal{Y}$ is relatively small. However, this can be a challenging task especially when the label domain is huge, especially in the context of structured outputs. In order to tackle this, we studied several structured prediction approaches such as message passing (belief propagation), integer linear programming, graph cut, etc. Note that all of these aforementioned methods crucially rely on lots of labeled training samples, which can be prohibitively expensive to obtain in some tasks.

In this lecture, we switch gears and focus on an equally important task: unsupervised learning. The primary goal here is to learn the structures/patterns inherent to the given data in the absence of labels and thus understand the underlying data generating distribution. Understanding the probability distribution of a given data can reveal many interesting insights and is a task of fundamental importance in statistics and machine learning. In practice, we often have lots of unlabeled samples and thus the goal is to leverage this abundance to learn about the data distribution. This is also referred to as *generative modeling* as opposed to *discriminative modeling* which we studied so far. This lecture is a first step towards understanding generative modeling in the context of mixture models.

Outline:

- Clustering
- k-Means algorithm
- Optimization view point of k-Means
- k-Means and beyond
- Applications
- Conclusion

Notation. In this lecture, we denote the Euclidean vectors by bold face lower case letters such as $\boldsymbol{x}, \boldsymbol{\mu}$ etc. $\mathbb{R}^d$ denotes the d-dimensional Euclidean space. We define $[k] \triangleq \{1, \ldots, k\}$. $\|\cdot\|$ denotes the standard Euclidean 2-norm $\|\cdot\|_2$.

Note. All the pictures not cited in this lecture are adapted from the lecture notes [Sch18].

1 Clustering

Clustering is the process of identifying groups, or clusters, of data points in a multidimensional space [Bis06]. Roughly speaking, the set of points in a cluster are similar to each other as compared to those outside the cluster. An illustrative example to understand the notion of clustering is the well-known image segmentation task. In image segmentation, the objective is to group together a set of pixels based on some similar characteristics. For example, in Figure 1, we see that all the pixels corresponding to cycles share the green patch, and those corresponding to the riders share the pink patch.

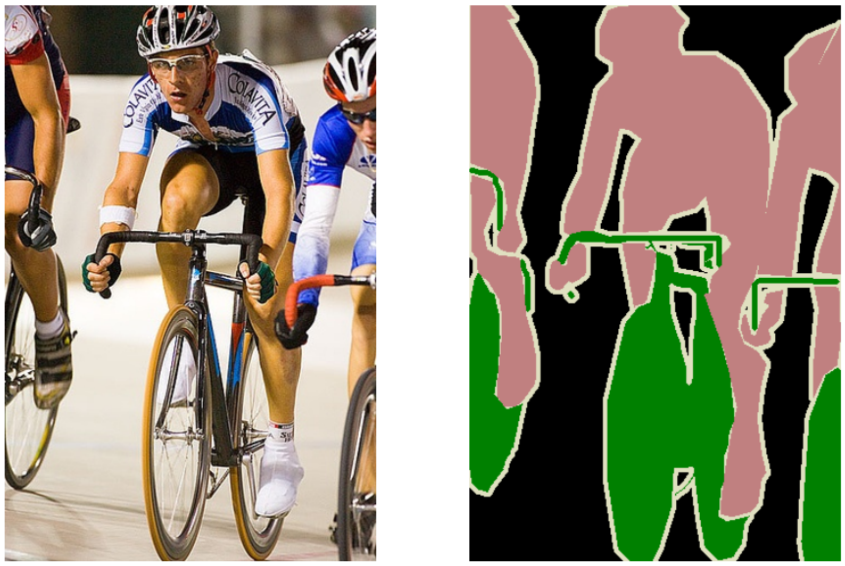


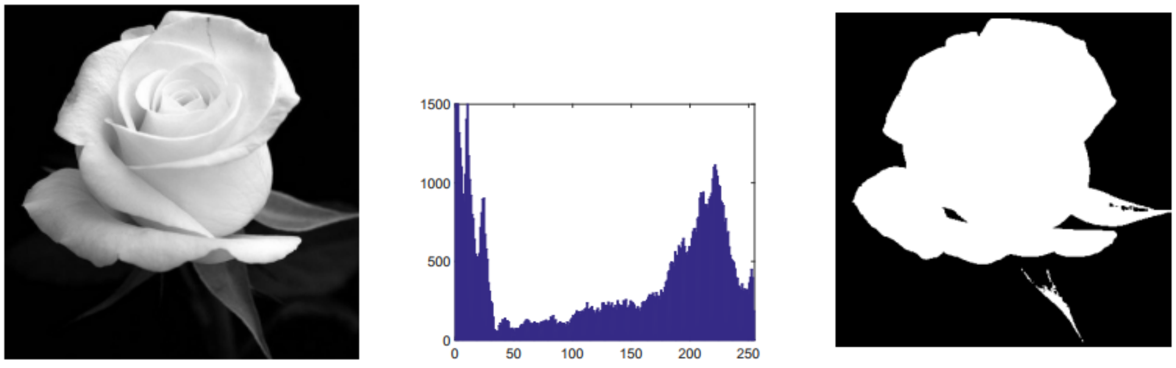
Figure 1: Image segmentation

If we represent each pixel by its feature vector in Euclidean space, for example its RGB intensities, we can see that image segmentation is equivalent to clustering these feature vectors and assigning the same color label to pixels within the same cluster. In Figure 1, we see that the image is segmented into various color patches based on the objects their corresponding pixels represent.

In the above example, we saw an important application of clustering. This begs the following question: How do we do clustering? In order to answer this, let us consider Figure 2 as a running example.

Given a grey scale image in Figure 2a, we want to segment this image into to foreground and background. Since intensity of each pixel lies between 0 to 255, a natural approach is to consider an intensity profile as in Figure 2b. From the profile, we can see that there are two distinctive peaks near 0 and 250 respectively. Thus we can chose a real number threshold to be the midpoint of location of these peaks and label all those pixels whose intensities are smaller than threshold as 0 and those with higher intensities as 1.

While this simple heuristic fulfills the task, a careful inspection highlights its shortcomings: 1) How do we chose threshold just from the data alone? 2) Is it the best possible choice? In order to find the parameter threshold, or equivalently, the two cluster centers corresponding to the peaks, adaptively, we need an algorithm that automatically determines the cluster assignment for all the points in a data dependent manner. In addition, we also want our algorithm to find the best possible cluster assignment. k-Means algorithm precisely aims to achieve this objective and will be the focus of the remaining lecture.



(a) Image (b) Intensity profile (c) Original image after clustering

Figure 2: Clustering usig gray scale intensities

2 k-Means

k-Means was introduced by [Llo57, Llo82] for pulse coded modulation in signal processing. Ever since its inception, it remains one of the most popular and influential data mining algorithms [1] and was rediscovered by several researchers among different disciplines [WKQ+08], thus highlighting its ubiquity. k-Means is a simple iterative algorithm to divide the given data points into a group of K clusters, for a fixed $K \in \mathbb{N}$. Informally, the algorithm can be summarized as follows:

Algorithm 1 k-Means algorithm

```
1: Initialization: Pick $k$ random points $\boldsymbol{\mu}_i, i \in [k]$ as cluster centers
2: Iterate:
3:     Assign data points $\boldsymbol{x}_i$ to closest cluster center according to some metric
4:     Update the cluster center to be the average of its assigned points
5:     Stopping criterion: when no points' assignments change
```

Figure 3 illustrates each step of the k-Means outlined in Algorithm 1 (adapted from [Bis06]).

[1]In fact, it is cited as one of the top 10 influential algorithms by the data mining research community [WKQ+08].

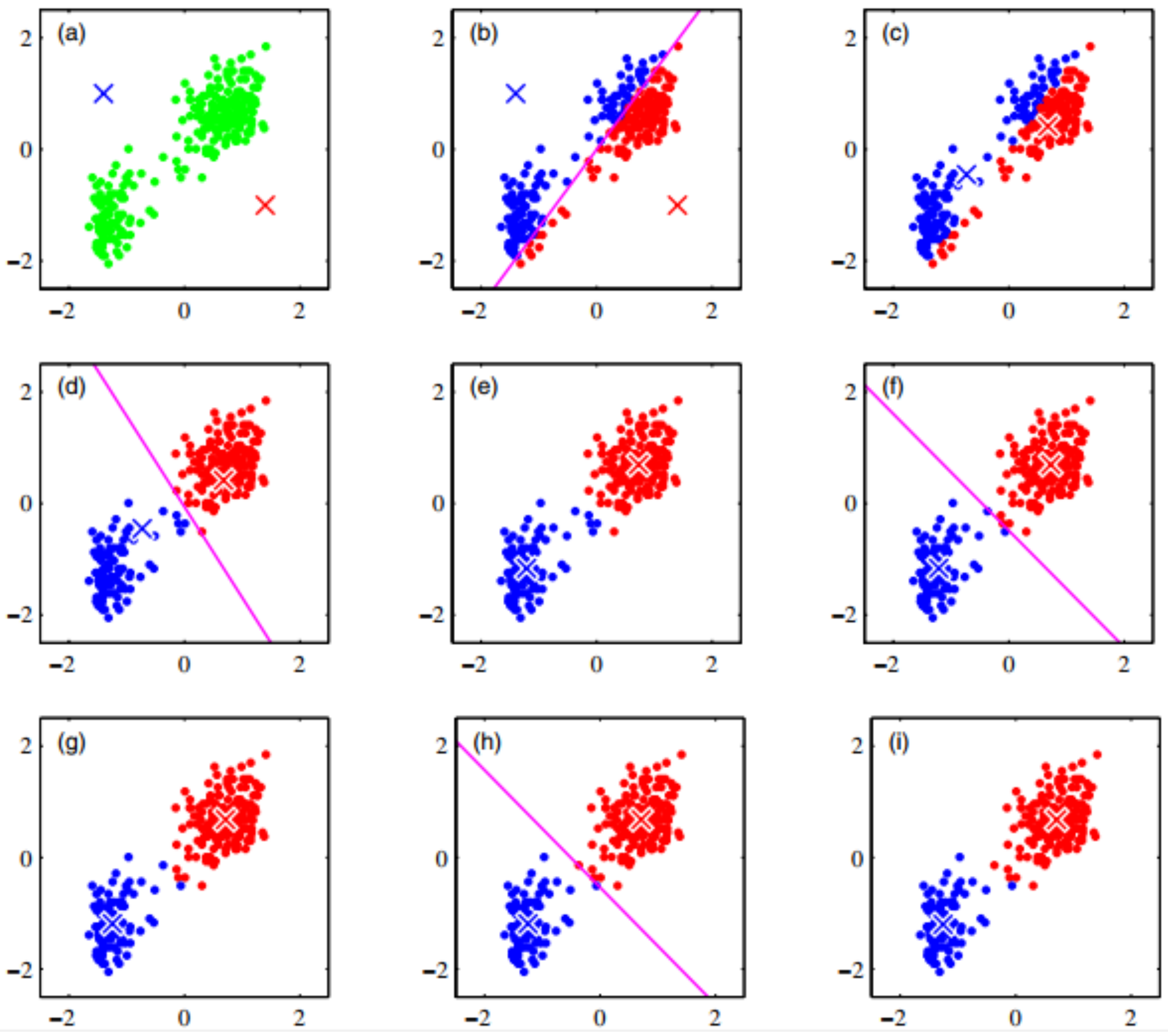


Figure 3: Illustration of the k-Means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centers $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm.

3 Optimization view point of k-Means

While Algorithm 1 informally details the k-Means procedure, it is natural to ask if there is any underlying objective behind this. More formally, we would like to understand if this algorithm is a natural outcome of some optimization problem. To establish this, we start with some preliminary notation (we closely follow the setting in [Bis06]). Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ be the set of data points we would like to cluster into $K \in \mathbb{N}$ groups. Let $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ denote the cluster centers. Each $\boldsymbol{\mu}_k$ can be thought of as a representative candidate for each cluster $k \in [K]$. Let $r_{ik} \in \{0, 1\}$ be the binary indicator variable which captures the assignment of each data point $\boldsymbol{x}_i$ to a cluster k, i.e.

$$r_{ik} = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \text{ is assigned to cluster } k \\ 0 & \text{otherwise} \end{cases}$$

Let $r \triangleq (r_{ik})_{i \in [n], k \in [K]}$ contain all the indicator variables and similarly for the cluster centers $\boldsymbol{\mu} \triangleq (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$. We now define the objective function $J(r, \boldsymbol{\mu})$ as

$$J(r, \boldsymbol{\mu}) \triangleq \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k \right\|^2 \tag{1}$$

which measures the sum of the squared Euclidean distance between each data point and its assigned cluster center. Our objective is to find the pair of best cluster assignments and cluster centers $(r^*, \boldsymbol{\mu}^*)$ minimizing the above objective function, i.e.

$$J(r^*, \boldsymbol{\mu}^*) = \min_{r, \boldsymbol{\mu}} J(r, \boldsymbol{\mu}) \text{ s.t } \sum_{k=1}^{K} r_{ik} = 1, r_{ik} \in \{0, 1\}, \forall i \in [n], k \in [K]. \tag{2}$$

Since r_{ik} are binary variables with linear constraints and $\boldsymbol{\mu}_k$ are Euclidean vectors, joint optimization of $J(r, \boldsymbol{\mu})$ in (2) can be hard in general. In fact, it is well-known that in the worst case scenario it is NP-hard [Das]. In order to tackle this, we solve the problem using *alternate minimization.* Here the idea is to fix one set of variables and optimize the objective with respect to the remaining variables and then fix these variables and find the optimal first set of variables. This process is iterated until convergence or using some pre-defined stopping criterion. In our context, this boils down to

1. For a given cluster centers $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$, find the best cluster assignment $r = (r_{ik})$.
2. For a given cluster assignment $r = (r_{ik})$, find the best cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$.
3. Keep iterating Step 1 and Step 2.

In order to see what each individual step comprises of, first let us fix the cluster centers $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$. Now our goal is to solve

$$\min_{r_{ik}} \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k \right\|^2 \text{ s.t } \sum_{k=1}^{K} r_{ik} = 1, r_{ik} \in \{0, 1\}, \forall i \in [n], k \in [K]. \tag{3}$$

Since the cost function J decomposes over $i \in [n]$, for a fixed $i \in [n]$, the optimization problem reduces to

$$\min_{r_{ik}} \sum_{k=1}^{K} r_{ik} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k \right\|^2 \text{ s.t } \sum_{k=1}^{K} r_{ik} = 1, r_{ik} \in \{0, 1\}, \forall k \in [K]. \tag{4}$$

Since r_{ik}'s are binary and sum to 1, we can see that the optimal solution to (4) is to assign the data point $\boldsymbol{x}_i$ to its nearest cluster center. Formally,

$$r_{ik} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j \right\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

Now suppose that we fix the cluster assignments $r = (r_{ik})$. To obtain the best cluster centers, we solve

$$\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K} \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{5}$$

Since the above objective is quadratic in $\boldsymbol{\mu}_k$ for any $k \in [K]$, we obtain the optimal solution by equating the gradients to zero, i.e.

$$\frac{\partial J}{\partial \boldsymbol{\mu}_k} = 0 \Rightarrow \sum_{i=1}^{n} r_{ik}(\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0 \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_{ik}\boldsymbol{x}_i}{\sum_{i=1}^{n} r_{ik}}.$$

Thus $\boldsymbol{\mu}_k$ equals the mean of all the data points assigned to cluster k, hence the name k-Means . Using the solutions for these alternate minimization problems, k-Means can be expressed formally as follows:

Algorithm 2 k-Means algorithm

```
1: Input: Dataset $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$, number of clusters $K$
2: Initialization: Pick $k$ random points $\boldsymbol{\mu}_i^{(0)}, i \in [k]$ as cluster centers
3: for $t = 0, 1, \ldots$ do
4:     E step: For each $i \in [n]$ and $k \in [K]$, compute
           $r_{ik}^{(t+1)} = \begin{cases} 1 & \text{if } k = \text{argmin}_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(t)}\|^2 \\ 0 & \text{otherwise.} \end{cases}$
5:     M step: For each $k \in [K]$, compute
           $\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} r_{ik}^{(t+1)} \boldsymbol{x}_i}{\sum_{i=1}^{n} r_{ik}^{(t+1)}}$
```

Here the computation of new cluster assignment using the cluster centers at the previous iteration is called the *E step* where as calculating the new cluster centers is termed the *M step*. The reason is that k-Means is very closely related to a popular algorithm to learn parameters in latent variable models, called EM algorithm, which will be the focus of next lecture. In particular, if we allow 'soft' cluster assignments in $[0, 1]$ instead of the hard binary assignments $\{0, 1\}$, we obtain the EM algorithm. Thus k-Means can be viewed as non-probabilistic version of EM for Gaussian mixture models.

4 k-Means and beyond

4.1 Properties of k-Means

The following lemma shows that k-Means always decreases the cost function J.

Lemma 1. *For the* k-Means *in Algorithm 2, if $(r^{(t)}, \boldsymbol{\mu}^{(t)})$ denote the variable pair at iteration t, we have that*

$$J(r^{(t+1)}, \boldsymbol{\mu}^{(t+1)}) \leq J(r^{(t)}, \boldsymbol{\mu}^{(t)}).$$

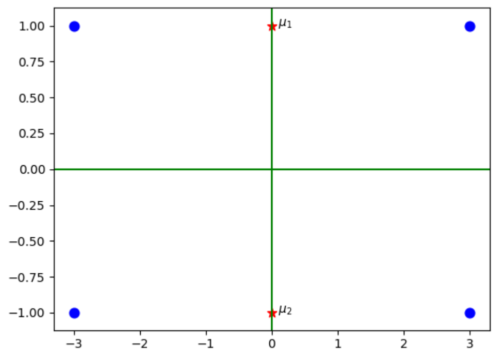
Proof. By definition, we have that

$$J(r^{(t+1)}, \boldsymbol{\mu}^{(t+1)}) \overset{(a)}{\leq} J(r^{(t+1)}, \boldsymbol{\mu}^{(t)}) \overset{(b)}{\leq} J(r^{(t)}, \boldsymbol{\mu}^{(t)}),$$

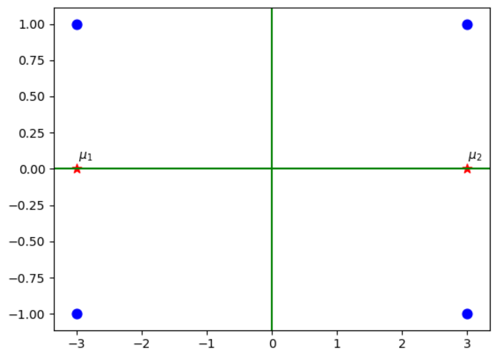
where (a) follows from the fact that $\boldsymbol{\mu}^{(t+1)}$ minimizes $J(r^{(t)}, \cdot)$ and (b) follows from the fact that $r^{(t+1)}$ minimizes $J(\cdot, \boldsymbol{\mu}^{(t)})$. $\square$

The following are some basic properties of the k-Means algorithm:

- **Convergence**: Since k-Means decreases the objective value J (Lemma 1) at each iteration, it always converges. However, only convergence to a local minimum is assured rather than a global minimum. In Figure 4, we see that when the inital clusters are initalized on the Y-axis, the algorithm gets stuck there. In contrast, the global minimum is on the X-axis. See [M+67] for details about convergence of k-Means algorithm.
- **Running time**: In Algorithm 1, the running complexity of the E-step is $O(Knd)$, where as for the M-step, the complexity is $O(n)$. Several variants have been proposed in the literature to speed up the E-step using some clever tree data structures [Bis06].



(a) Local minimum



(b) Global minimum

Figure 4: Illustration that k-Means can get stuck in local minimum. (a) Blue dots indicate the data points and the red dots indicate the initial cluster centers. The two data points in the upper half-plane are assigned to $\boldsymbol{\mu}_1$ and those in the lower half are assigned to $\boldsymbol{\mu}_2$. Since the cluster means again equal their original centers, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ remain unchanged. (b) If instead the cluster centers are located on the X-axis at $(1, 0)$ and $(-1, 0)$ respectively, we obtain a better cluster assignment. In fact, one can show that it is a globally optimal configuration (Exercise).

4.2 Extensions

There are several variants and extensions to the standard k-Means algorithm we studied so far. The following are some modifications:

1. k-Means++: As highlighted in Figure 4, the standard k-Means is very sensitive to initializations and can get stuck in local optimum. In addition, the approximate solution can be arbitrarily bad compared to the optimal solution [Wik17]. In order to address these shortcomings, two independent works [AV07, ORSS06] introduced a variant called k-Means++, which gives a new

initialization procedure of cluster centers as opposed to random initialization in the standard version. The key idea here is to find a initial set of cluster centers that are well separated since it will lead to better clustering. Note that k-Means++ is only a new initialization scheme for the cluster centers before the standard k-Means algorithm is run. The following is the actual algorithm [Wik17]:

- Randomly choose first center from among the data points
- For each data point $\boldsymbol{x}_i$, compute the distance to its nearest neighbor
- Pick a new center from among the data points where $\boldsymbol{x}_i$ is chosen with probability proportional to $\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$, where $\boldsymbol{\mu}_k$ is its nearest neighbor
- Repeat until K centers are chosen

This new variant is known to perform empirically better than the standard version in a variety of settings. Moreover, the approximation ratio for the k-Means++ algorithm is $O(\log K)$ in expectation [AV07]. In other words, on an average over randomness in the initialization process, k-Means++ is guaranteed to find a solution that is at most $O(\log K)$ worse than the optimal solution.

2. **Generalization of distances**: Note that the standard version of k-Means relies on the Euclidean norm $\|\cdot\|_2$ to compute the distances. The distance between a data point and its cluster center, $\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|$ can be thought of as a measure of dissimilarity between $\boldsymbol{x}_i$ and $\boldsymbol{\mu}_k$. Thus we can generalize this to incorporate other distance measures such as cosine similarity and also various norms such as $\|\cdot\|_1, \|\cdot\|_\infty$, etc.

3. **Kernel k-Means** : k-Means can also be viewed as a linear separation of given data points into clusters. This is because the assignment of a data point to any cluster center is determined by the linear hyperplane separating them. In order to account for non-linear separation, a standard approach is to embed this data in high dimensional feature space using some non-linear transformations and then do linear separation in this high-dimensional space, which is equivalent to using kernels as a measure of dissimilarity instead of the standard Euclidean distance. See [DGK04] for an interesting connection between kernel k-Means clustering and spectral clustering.

5 Applications

5.1 Image segmentation

As we saw in Section 1, one of the important practical applications of k-Means is image segmentation. Figure 5 below highlights the trade-off between the number of clusters K and the quality of the clustering [Bis06].

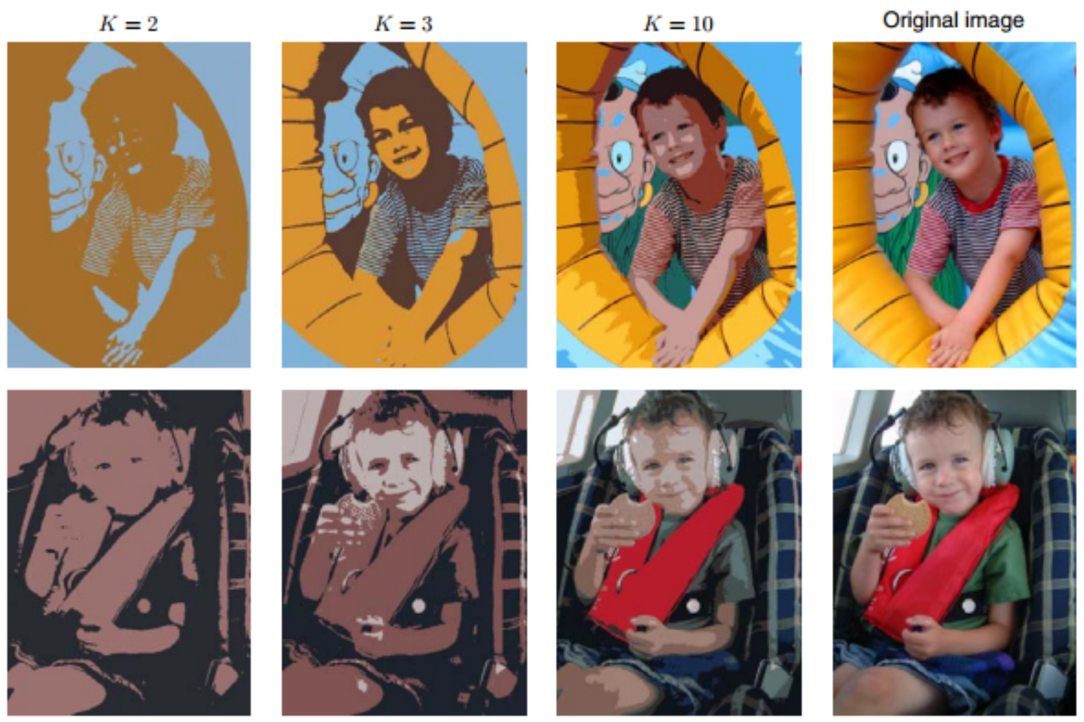


Figure 5: Two examples of the application of the k-Means clustering algorithm to image segmentation showing the initial images together with their segmentations obtained using various values of K. This also illustrates of the use of vector quantization for data compression, in which smaller values of K give higher compression at the expense of poorer image quality

5.2 Superpixel segmentation and Image compression

Another example is illustrated in Figure 6, where we do superpixel segmentation [Sch18]. Here in addition to the usual color intensities as feature vectors, we also augment their spatial co-ordinates to ensure spatial smoothness. The corresponding distance metric is modified accordingly to treat the color and spatial co-ordinates differently. These compressed features obtained after segmentation can be used in further downstream tasks as opposed to the standard high-dimensional features, which can be viewed as an instance of *image compression*.

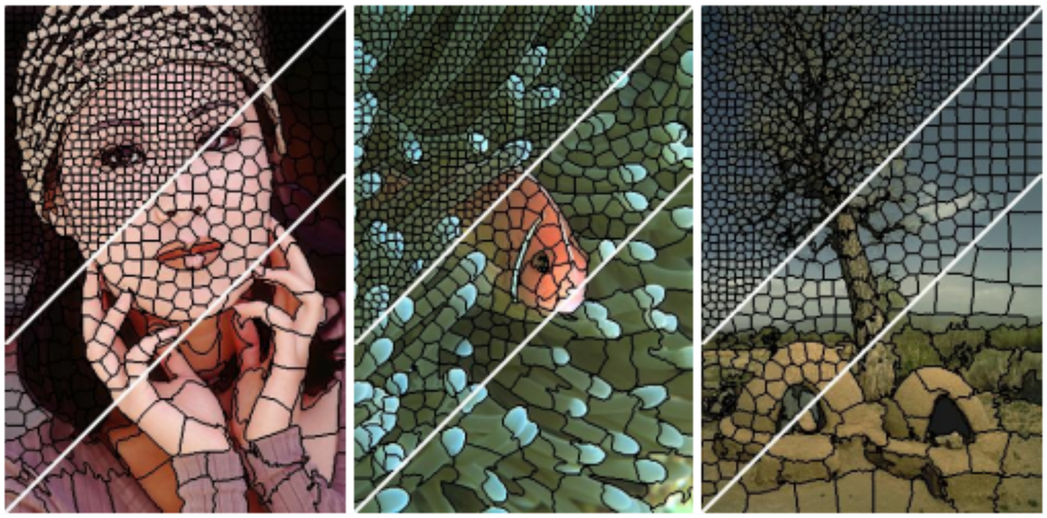


Figure 6: Superpixel segmentation

6 Conclusion

In this lecture, we studied the standard k-Means algorithm in detail. In particular, we derived it as a natural solution to the optimization problem involving the squared Euclidean distances to its nearest cluster centers. We further discussed several possible generalizations and extensions of this procedure. Finally, we also some interesting applications of this in several popular computer vision tasks. In the next lecture, we study a probabilistic generalization of this algorithm in the context of Gaussian mixture models, EM algorithm.

References

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[Das] Sanjoy Dasgupta. The hardness of k-means clustering. Technical report.

[DGK04] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. ACM, 2004.

[Llo57] Stuart Lloyd. Least squares quantization in pcm. *Unpublished Bell Lab. Tech. Note, portions presented at the Institute of Mathematical Statistics Meeting Atlantic City, NJ*, September 1957.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[M$^+$67] 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.

[ORSS06] Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 165–176. IEEE, 2006.

[Sch18] A. Schwing. Ece 544 pattern recognition, lecture 16: kmeans. Technical report, 2018.

[Wik17] Wikipedia contributors. K-means++ — Wikipedia, the free encyclopedia. 2017. [Online; accessed 25-October-2018].

[WKQ$^+$08] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.