K-means extensions and evaluating clusterings

CS771: Introduction to Machine Learning

Nisheeth

K-means algorithm: recap

- Notation: $z_n \in \{1,2,\ldots,K\}$ or $\boldsymbol{z}_n$ is a K-dim one-hot vector
  - ($z_{nk} = 1$ and $z_n = k$ mean the same)

K-means Algorithm

1. Initialize K cluster means $\mu_1, \ldots, \mu_K$
2. For $n = 1, \ldots, N$, assign each point $\boldsymbol{x}_n$ to the closest cluster

$$z_n = \arg\min_{k \in \{1,\ldots,K\}} ||\boldsymbol{x}_n - \mu_k||^2$$

3. Suppose $\mathcal{C}_k = \{\boldsymbol{x}_n : z_n = k\}$. Re-compute the means

$$\mu_k = \text{mean}(\mathcal{C}_k), \quad k = 1, \ldots, K$$

4. Go to step 2 if not yet converged

K-means loss function: recap

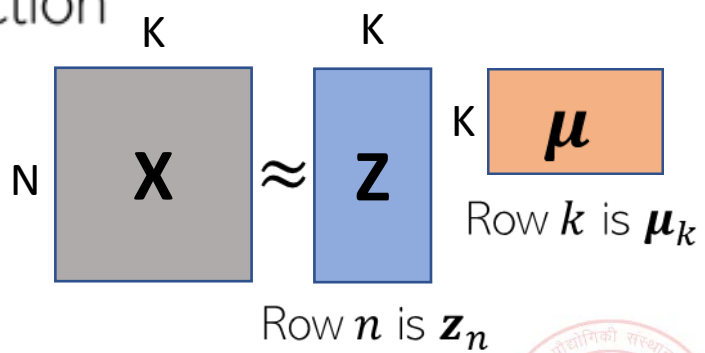
$\boldsymbol{z}_n = [z_{n1}, z_{n2}, \ldots, z_{nK}]$ denotes a length K one-hot encoding of $\boldsymbol{x}_n$

- Let $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ be the K cluster centroids/means
- Let $z_{nk} \in \{0, 1\}$ be s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $\boldsymbol{k}$, and 0 otherwise
- Define the distortion or "loss" for the cluster assignment of $\boldsymbol{x}_n$

$$\ell(\boldsymbol{\mu}, \boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2$$

- Total distortion over all points defines the K-means "loss function"

$$L(\boldsymbol{\mu}, \mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \mu_k||^2 = \underbrace{||\mathbf{X} - \mathbf{Z}\boldsymbol{\mu}||_F^2}_{\text{matrix factorization view}}$$

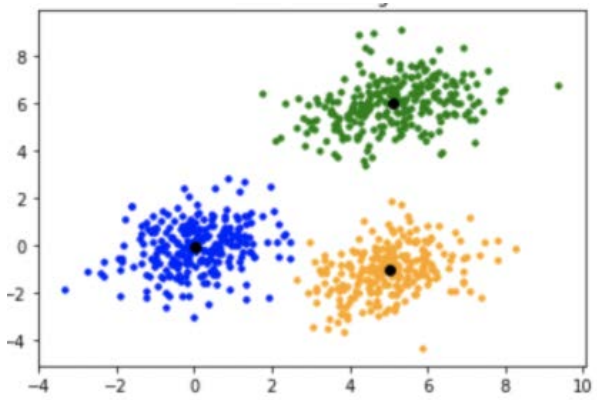


- The K-means problem is to minimize this objective w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{Z}$
  - Alternating optimization on this loss would give the K-means (Lloyd's) algorithm we saw earlier!

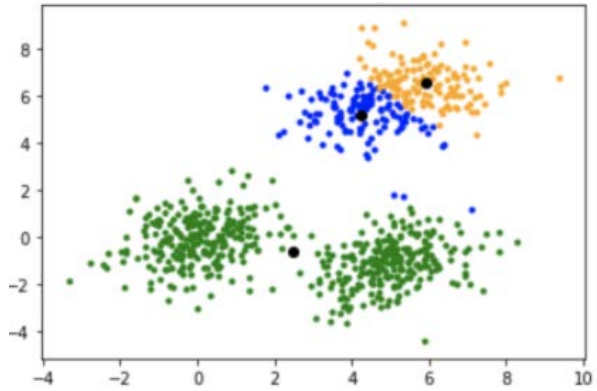
K-means++

- K-means results can be sensitive to initialization

Desired clustering



Poor initialization: bad clustering



- K-means++ (Arthur and Vassilvitskii, 2007) an improvement over K-means
  - Only difference is the way we initialize the cluster centers (rest of it is just K-means)
  - Basic idea: Initialize cluster centers such that they are reasonably far from each other
  - Note: In K-means++, the cluster centers are chosen to be K of the data points themselves

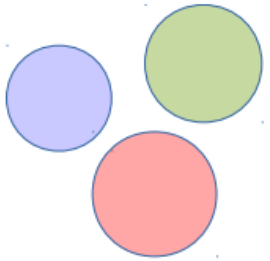
K-means++

- K-means++ works as follows
  - Choose the first cluster mean uniformly randomly to be one of the data points
  - The subsequent $K-1$ cluster means are chosen as follows
    1. For each unselected point $\boldsymbol{x}$, compute its smallest distance $D(\boldsymbol{x})$ from already initialized means
    2. Select the next cluster mean unif. rand. to be one of the unselected points based on probability prop. to $D(\boldsymbol{x})^2$
    3. Repeat 1 and 2 until the $K-1$ cluster means are initialized
  - Now run standard K-means with these initial cluster means
- K-means++ initialization scheme sort of ensures that the initial cluster means are located in different clusters

Thus farthest points are most likely to be selected as cluster means

K-means: Soft Clustering

- *K*-means makes hard assignments of points to clusters
  - Hard assignment: A point either completely belongs to a cluster or doesn't belong at all



Hard-assignment okay

Hard-assignment tricky

A more principled extension of K-means for doing soft-clustering is via probabilistic mixture models such as the Gaussian Mixture Model

  - When clusters overlap, soft assignment is preferable (i.e., probability of being assigned to each cluster: say $K = 3$ and for some point $\boldsymbol{x}_n$, $p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$)

- A heuristic to get soft assignments: Transform distances from clusters into prob.

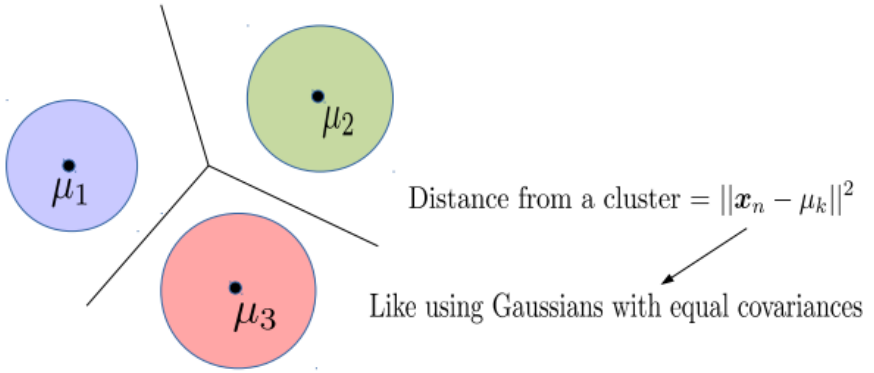
$$\sum_{k=1}^{K} \gamma_{nk} = 1$$

$$\gamma_{nk} = \frac{\exp(-||\boldsymbol{x}_n - \mu_k||^2)}{\sum_{\ell=1}^{K} \exp(-||\boldsymbol{x}_n - \mu_\ell||^2)} \quad \text{(prob. that } \boldsymbol{x}_n \text{ belongs to cluster } k\text{)}$$

- Cluster mean updates also change: $\mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_{nk}}$ (all points contribute, fractionally)

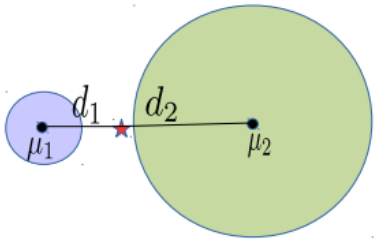
K-means: Decision Boundaries and Cluster Sizes/Shapes

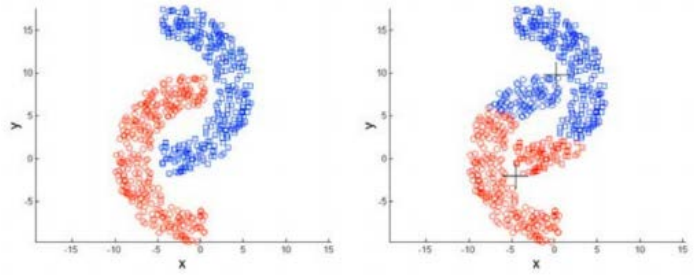
- K-mean assumes that the decision boundary between any two clusters is linear
- Reason: The K-means loss function implies assumes equal-sized, spherical clusters



Reason: Use of Euclidean distances

- May do badly if clusters are not roughly equi-sized and convex-shaped





Kernel *K*-means

Helps learn non-spherical clusters and nonlinear cluster boundaries

- Basic idea: Replace the Eucl. distances in K-means by the kernelized versions

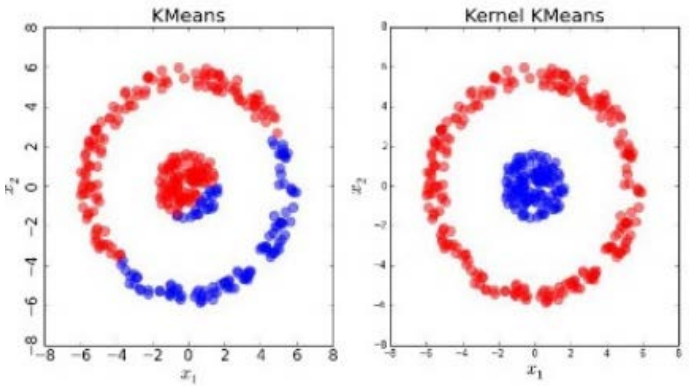
$$||\phi(\boldsymbol{x}_n) - \phi(\boldsymbol{\mu}_k)||^2 = ||\phi(\boldsymbol{x}_n)||^2 + ||\phi(\boldsymbol{\mu}_k)||^2 - 2\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{\mu}_k)$$
$$= k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{\mu}_k, \boldsymbol{\mu}_k) - 2k(\boldsymbol{x}_n, \boldsymbol{\mu}_k)$$

Kernelized distance between input $\boldsymbol{x}_n$ and mean of cluster k

- Here $k(.,.)$ denotes the kernel function and ϕ is its (implicit) feature map
- Note: $\phi(\mu_k)$ is the mean of ϕ mappings of the data points assigned to cluster k

Not the same as the ϕ mapping of the mean of the data points assigned to cluster k

$$\phi(\mu_k) = \frac{1}{|\mathcal{C}_k|} \sum_{n:z_n=k} \phi(\boldsymbol{x}_n)$$



Can also used landmarks or kernel random features idea to get new features and run standard k-means on those

Note: Apart from kernels, it is also possible to use other distance functions in K-means. Bregman Divergence* is such a family of distances (Euclidean and Mahalanobis are special cases)

*Clustering with Bregman Divergences (Banerjee et al, 2005)

Overlapping Clustering

Kind of unsupervised version of multi-label classification (just like standard clustering is like unsupervised multi-class classification)

Example: Clustering people based on the interests they may have (a person may have multiple interests; thus may belong to more than one cluster simultaneously)

- Have seen hard clustering and soft clustering
- In hard clustering, z_n is a one-hot vector
- In soft clustering, z_n is a vector of probabilities
- Overlapping Clustering: A point can simultaneously belong to multiple clusters
  - This is different from soft-clustering
  - z_n would be a binary vector, rather than a one hot or probability vector, e.g.,

$$z_n = [1\ 0\ 0\ 1\ 0]$$

K=5 clusters with point x_n belonging (in whole, not in terms of probabilities) to clusters 1 and 4

- In general, more difficult than hard/soft clustering (for N data points and K clusters, the size of the space of possible solutions is not K^N but 2^{NK} - exp in both N and K)
- K-means has extensions* for doing overlapping clustering. There also exist latent variable models for doing overlapping clustering

*An extended version of the k-means method for overlapping clustering (Cleuziou, 2008); Non-exhaustive, Overlapping k-means (Whang et al, 2015)

Evaluating Clustering Algorithms

- Clustering algos are in general harder to evaluate since we rarely know the ground truth clustering (since clustering is unsupervised)

- If ground truth labels not available, use output of clustering for some other task
  - For example, use cluster assignment z_n (hard or soft) as a new feature representation
  - Performance on some task using this new rep. is a measure of goodness of clustering

- If ground truth labels are available, can compare them with clustering based labels
  - Not straightforward to compute accuracy since the label identities may not be the same, e.g.,

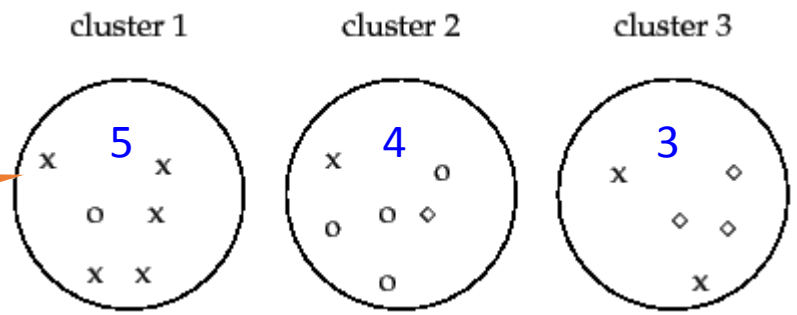
Ground truth = [1 1 1 0 0 0] Clustering = [0 0 0 1 1 1]

(Perfect clustering but zero "accuracy" if we just do a direct match)

  - There are various metrics that take into account the above fact
    - Purity, Rand Index, F-score, Normalized Mutual Information, etc

Evaluating Clustering Algorithms

- Purity: Looks at how many points in each cluster belong to the majority class in that cluster



3 classes (x, o , △, assuming known ground truth labels)

Sum and divide by total number of points

$$\text{Purity} = (5+4+3)/17 \approx 0.71$$

Close to 0 for bad clustering, 1 for perfect clustering

Also a bad metric if number of clusters is very large – each cluster will be kind of pure anyway

- Rand Index (RI): Can also look at what fractions of pairs of points with same (resp. different) label are assigned to same (resp. different) cluster

F_β score is also popular

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \qquad F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

Precision

Recall

True Positive: No. of pairs with same true label and same cluster

True Negative: No. of pairs with diff true label and diff clusters

$$\text{RI} = \frac{TP+TN}{TP+FP+FN+TN}$$

False Positive: No. of pairs with diff true label and same cluster

False Negative: No. of pairs with same true label and diff cluster