CSC 311: Introduction to Machine Learning
Lecture 11 - K-Means and EM Algorithm

Michael Zhang       Chandra Gummaluru

University of Toronto, Winter 2023
Outline

1. K-Means for Clustering
2. Gaussian Mixture Models (optional)
3. Expectation-Maximization (E-M) (optional)
4. Why EM Works (Optional)
Final Exam

- 3 hours.
- One double-sided aid sheet, created by yourself.
- Cumulative up to k-means, with emphasis on post-midterm topics.
- Past exams posted and two review sessions next week. Upvote conceptual questions others have asked.
- Slides, recordings, office hours.

\[\text{Tuesday 4-5pm Ba 1170}\]
linear algebra
KNN
decision trees
linear regression
classification (logistic regression)
networks
probabilistic modeling
Gaussians
MLE
choose params to maximize likelihood
MAP
MLE + prior on parameters

discriminative

midterm
neural networks

generative

unsupervised

PCA
eigenvector values

data → model

optimization problem

autoencoders

NNs for compression

\[ \min (\hat{\lambda} - \lambda)^2 \]

reconstruction error

\[ \sum_i \sum_j (\hat{x}_{ij} - x_{ij})^2 \]

linear layer
many linear/nonlinear layers

If you just use a linear layer => equivalent to PCA

\[ VAE \]

minimizing reconstruction error + similarity term between \( z \) and \( N(0, I) \)
Feedback

- Please take some time to fill out the course feedback form.
- The main changes we made to the past iteration were adding the Math Diagnostic and an open-ended project option. We also made tutorials more problem-solving oriented.
- Helpful for us to hear what supports your learning.
- Feel free to write to the course email if you have more detailed thoughts.
In the previous lecture, we covered PCA, Autoencoders and Matrix Factorization—all unsupervised learning algorithms.

- Each algorithm can be used to approximate high dimensional data using some lower dimensional form.

Those methods made an interesting assumption that data depends on some latent variables that are never observed. Such models are called latent variable models.

- For PCA, these correspond to the code vectors (representation).

Today:

- K-means, a simple algorithm for clustering, i.e. grouping data points into clusters
- Reformulate clustering as a latent variable model and apply the EM algorithm
K-Means for Clustering

Gaussian Mixture Models (optional)

Expectation-Maximization (E-M) (optional)

Why EM Works (Optional)
Clustering

- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:
- Such a distribution is **multimodal**, since it has multiple **modes**, or regions of high probability mass.

**Clustering**: grouping data points into clusters, **with no observed labels**. It is an unsupervised learning technique.

- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.) But topics are never observed (unsupervised).
K-means Intuition

There are $k$ clusters, and each point is close to its cluster center, or mean (the mean of points in the cluster).

How do we compute the cluster assignments?

- Given the cluster assignments, we could easily compute the cluster centers.
- Given the cluster centers, we could easily compute the cluster assignments.
- Chicken and egg problem!
- Simple heuristic - start randomly and alternate between the two!
K-Means

- Randomly **initialize** cluster centers
- Alternate between two steps:
  - **Assignment step**: Assign each data point to the closest cluster
  - **Refitting step**: Move each cluster center to the mean of its members.
K-Means Example

Figure from Bishop

Simple demo: http://syskall.com/kmeans.js/
What is K-means Optimizing?

**K-means Objective:**
Find cluster centers \( \mathbf{m} \) and assignments \( \mathbf{r} \) to minimize the sum of squared distances of data points \( \{ \mathbf{x}^{(n)} \} \) to their assigned cluster centers

\[
\min_{\{\mathbf{m}\},\{\mathbf{r}\}} J(\{\mathbf{m}\}, \{\mathbf{r}\}) = \min_{\{\mathbf{m}\},\{\mathbf{r}\}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2
\]

subject to \( \sum_k r_k^{(n)} = 1, \forall n \), where \( r_k^{(n)} \in \{0, 1\}, \forall k, n \)

where \( r_k^{(n)} = 1 \) means that \( \mathbf{x}^{(n)} \) is assigned to cluster \( k \) (with center \( \mathbf{m}_k \))

- Finding the exact optimum can be shown to be NP-hard.
- K-means can be seen as block coordinate descent on this objective (analogous to ALS for matrix completion)
  - Assignment step = minimize w.r.t. \( \{r_k^{(n)}\} \)
  - Refitting step = minimize w.r.t. \( \{\mathbf{m}_k\} \)
Alternating Minimization

Optimization problem:

\[
\min_{\{m_k\}, \{r^{(n)}\}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2
\]

If we fix the centers \(\{m_k\}\) then we can easily find the optimal assignments \(\{r^{(n)}\}\) for each sample \(n\):

\[
\min_{r^{(n)}} \sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2
\]

Assign each point to the cluster with the nearest center

\[
r_k^{(n)} = \begin{cases} 
1 & \text{if } k = \text{arg min}_j \|m_j - x^{(n)}\|^2 \\
0 & \text{otherwise} 
\end{cases}
\]

E.g. if \(x^{(n)}\) is assigned to cluster \(\hat{k}\),

\[
r^{(n)} = [0, 0, \ldots, 1, \ldots, 0]^T
\]

Only \(\hat{k}\)-th entry is 1.
Alternating Minimization

Likewise, if we fix the assignments \( \{ r^{(n)} \} \) then can easily find optimal centers \( \{ m_k \} \)

\[
0 = \frac{\partial}{\partial m_l} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \left\| m_k - x^{(n)} \right\|^2
\]

\[
= 2 \sum_{n=1}^{N} r_l^{(n)} (m_l - x^{(n)}) \quad \Rightarrow \quad m_l = \frac{\sum_n r_l^{(n)} x^{(n)}}{\sum_n r_l^{(n)}}
\]

K-Means simply alternates between minimizing w.r.t. assignments and centers. This is an instance of alternating minimization, or block coordinate descent.

\[\text{Fix the points (and their associated centers)}\]

\[\text{Update } m_k \rightarrow \text{pick the average of all points assigned to center}\]

\[\text{look at the points assigned to } m_l\]

\[\text{monotonically decreasing sequence that is bounded below } \Rightarrow \text{converges}\]

\[\text{sequence: k-means loss}\]
The K-means Algorithm

- **Initialization**: Set K cluster means \( m_1, \ldots, m_K \) to random values.

- Repeat until convergence (until assignments do not change):
  - **Assignment** (Optimize w.r.t. \( \{r\} \))
    Each data point \( x^{(n)} \) assigned to nearest center.
    \[
    r_k^{(n)} = \begin{cases} 
    1 & \text{if } k = \arg \min_j \|m_j - x^{(n)}\|^2 \\
    0 & \text{otherwise}
    \end{cases}
    \]
  - **Refitting** (Optimize w.r.t. \( \{m\} \))
    Each center is set to mean of data assigned to it.
    \[
    m_k = \frac{\sum n r_k^{(n)} x^{(n)}}{\sum n r_k^{(n)}}.
    \]
Why K-means Converges

- K-means algorithm reduces the cost at each iteration.

- If the assignments do not change in the assignment step, we have converged (to at least a local minimum).

- Convergence will happen after a finite number of iterations, since the number of possible cluster assignments is finite.
Local Minima

- The objective $J$ is non-convex.
- Coordinate descent on $J$ is not guaranteed to converge to the global minimum.
- Nothing prevents k-means getting stuck at local minima.
- We could try many random starting points

A bad local optimum
K-means for Vector Quantization

Given image, construct “dataset” of pixels represented by their RGB pixel intensities

Run k-means, replace each pixel by its cluster center
K-means for Image Segmentation

- Given image, construct “dataset” of pixels, represented by their RGB pixel intensities and grid locations
- Run k-means (with some modifications) to get superpixels
Soft K-means

- Instead of making hard assignments of data points to clusters, we can make soft assignments.

- For example, one cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.

- This allows a cluster to use more information about the data in the refitting step.

- How do we decide on the soft assignments?

- We already saw this in multi-class classification: 1-of-$K$ encoding vs softmax assignments.
Soft K-means Algorithm

- **Initialization**: Set K means \( \{m_k\} \) to random values

- Repeat until convergence (measured by how much \( J \) changes):
  - **Assignment**: Each data point \( n \) given soft “degree of assignment” to each cluster mean \( k \), based on responsibilities
    
    \[
    r_k(n) = \frac{\exp[-\beta \|m_k - x(n)\|^2]}{\sum_j \exp[-\beta \|m_j - x(n)\|^2]}
    \]

    \[\beta \to \infty\] k-means algorithm

    \[\beta \to 0\] uniform dist. \( \implies r(n) = \text{softmax}(-\beta \{\|m_k - x(n)\|^2\}_k) \]

  - **Refitting**: Cluster centers are adjusted to match sample means of datapoints they are responsible for:

    \[
    m_k = \frac{\sum_n r_k(n) x(n)}{\sum_n r_k(n)}
    \] weighted average

  *rather than \( r \) be i-hot, Softer version of \( k \)-means algorithm*
Questions about Soft K-means

Some remaining issues

- How to set $\beta$?
- Clusters with unequal weight and width?

These aren’t straightforward to address with K-means.
Instead, we’ll reformulate clustering using a generative model.
As $\beta \to \infty$, soft k-Means becomes k-Means! (Exercise)