CSC 411 Lecture 15: K-Means

Mengye Ren and Matthew MacKay

University of Toronto

Motivating Examples

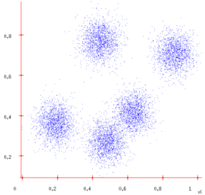
- Some examples of situations where you'd use unsupervised learning
  - You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
  - You're a biologist studying animal behavior, so you want to infer a high-level description of their behavior from video. You don't know the set of behaviors ahead of time.
  - You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (refrigerator, washing machine, etc.).
- Common theme: you have some data, and you want to infer the structure underlying the data.
- This structure is **latent**, which means it's never observed.

Overview

- In last lecture, we looked at density modeling where all the random variables were fully observed.
- The more interesting case is when some of the variables are latent, or never observed. These are called **latent variable models**.
  - Today's lecture: K-means, a simple algorithm for **clustering**, i.e. grouping data points into clusters
  - Next 2 lectures: reformulate clustering as a latent variable model, apply the EM algorithm

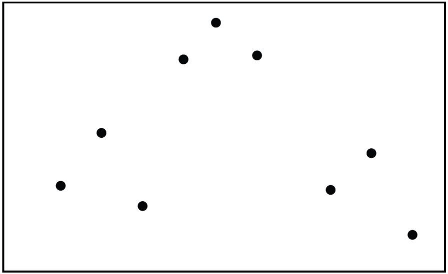
Clustering

- Sometimes the data form clusters, where examples within a cluster are similar to each other, and examples in different clusters are dissimilar:



- Such a distribution is **multimodal**, since it has multiple **modes**, or regions of high probability mass.
- Grouping data points into clusters, with no labels, is called **clustering**
- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.)

Clustering



- Assume the data $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ lives in a Euclidean space, $\mathbf{x}^{(n)} \in \mathbb{R}^d$.
- Assume each data point belongs to one of K clusters
- Assume the data points from same cluster are similar, i.e. close in Euclidean distance.
- How can we identify those clusters (data points that belong to each cluster)?

K-means Objective

Let's formulate this as an optimization problem

- **K-means Objective**:
  Find cluster centers $\{\mathbf{m}_k\}_{k=1}^{K}$ and assignments $\{\mathbf{r}^{(n)}\}_{n=1}^{N}$ to minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned cluster centers
- Mathematically:

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

  where $r_k^{(n)} = \mathbb{I}[\mathbf{x}^{(n)}$ is assigned to cluster $k]$

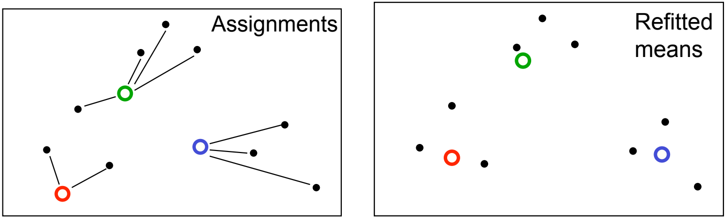
- Finding an optimal solution is an NP-hard problem!

Coordinate Descent

- But note:
  - If we fix the centers $\{\mathbf{m}_k\}$ then we can easily find the optimal assignments $\{\mathbf{r}^{(n)}\}$
    - Assign each point to the cluster with the nearest center (check!)
  - Likewise, if we fix the assignments $\{\mathbf{r}^{(n)}\}$ then can easily find optimal centers $\{\mathbf{m}_k\}$
    - Set each cluster's center to the average of its assigned data points (check!)
- Let's alternate between minimizing $J(\{\mathbf{m}\}, \{\mathbf{r}\})$ with respect to $\{\mathbf{m}\}$ and $\{\mathbf{r}\}$
- This is called **coordinate descent**

High level overview of algorithm:

- **Initialization**: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - **Assignment step**: Assign each data point to the closest cluster
  - **Refitting step**: Move each cluster center to the mean of the data assigned to it



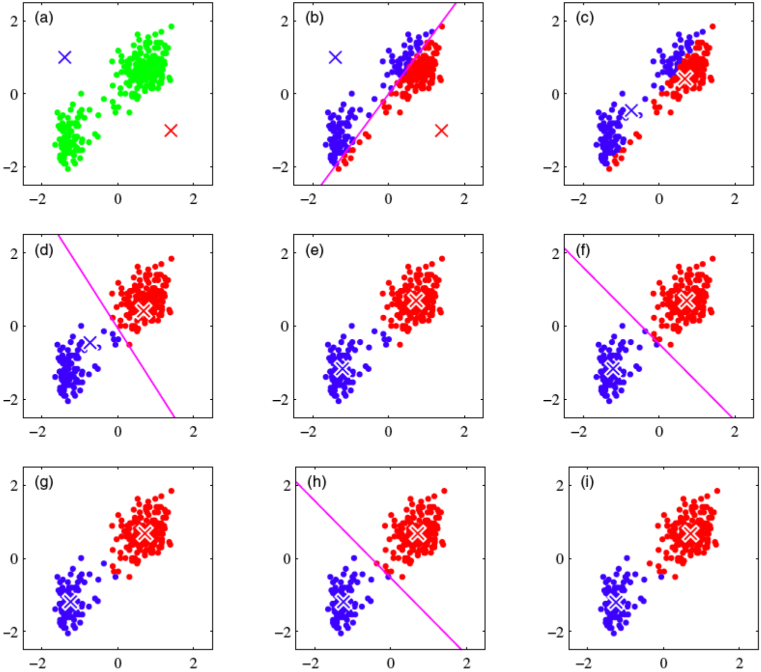


Figure from Bishop

Simple demo: http://syskall.com/kmeans.js/

The K-means Algorithm

- **Initialization**: Set K cluster means $\mathbf{m}_1, \ldots, \mathbf{m}_K$ to random values
- Repeat until convergence (until assignments do not change):
  - **Assignment**: Optimize J w.r.t. $\{\mathbf{r}\}$: Each data point $\mathbf{x}^{(n)}$ assigned to nearest center

    $$\hat{k}^{(n)} = arg \min_k ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

    and **Responsibilities** (1-hot encoding)

    $$r_k^{(n)} = \mathbb{I}[\hat{k}^{(n)} = k]$$

  - **Refitting:** Optimize J w.r.t. $\{\mathbf{m}\}$: Each center is set to mean of data assigned to it

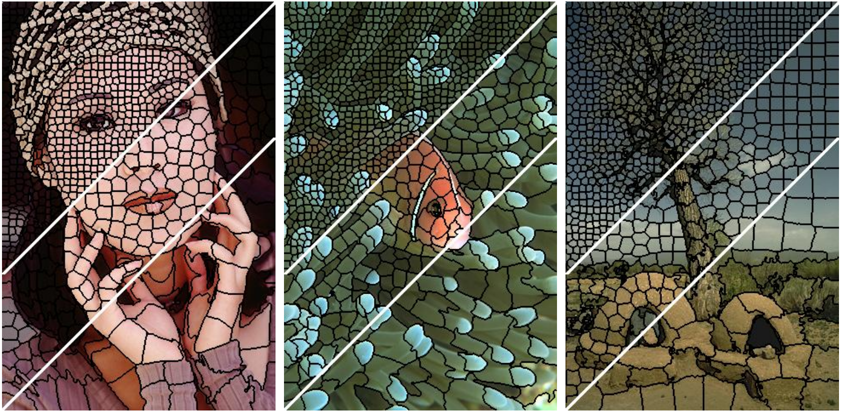
    $$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

K-means for Vector Quantization

Figure from Bishop

- Given image, construct “dataset“ of pixels represented by their RGB pixel intensities
- Run k-means, replace each pixel by its cluster center

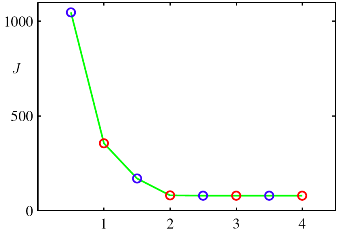
K-means for Image Segmentation



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

Why K-means Converges

- Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, J is reduced.
- **Test for convergence**: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).
- This will always happen after a finite number of iterations, since the number of possible cluster assignments is finite

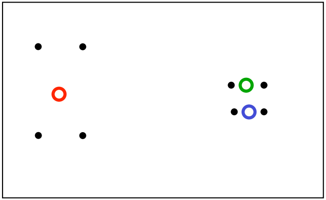


- K-means cost function after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step

Local Minima

- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves:
  - Simultaneously **merge** two nearby clusters
  - and **split** a big cluster into two

A bad local optimum



Soft K-means

- Instead of making hard assignments of data points to clusters, we can make **soft assignments**. One cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.
  - Allows a cluster to use more information about the data in the refitting step.
  - How do we decide on the soft assignments?

Soft K-means Algorithm

- **Initialization**: Set K means $\{\mathbf{m}_k\}$ 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

$$r_k^{(n)} = \frac{\exp[-\beta d(\mathbf{m}_k, \mathbf{x}^{(n)})]}{\sum_j \exp[-\beta d(\mathbf{m}_j, \mathbf{x}^{(n)})]}$$

$$\implies \mathbf{r}^{(n)} = \text{softmax}(-\beta[d(\mathbf{m}_k, \mathbf{x}^{(n)})]_{k=1}^{K})$$

  - **Refitting:** Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

Questions about Soft K-means

Some remaining issues

- How to set β?
- Clusters with unequal weight and width?

These aren't straightforward to address with K-means. Instead, next lecture, we'll reformulate clustering using a generative model.