Supervised Learning

Data: \((x_1, y_1), (x_2, y_2), \ldots \in \mathcal{X} \times \mathcal{Y}\)

Assumption: there is a (relatively simple) function \(f^* : \mathcal{X} \rightarrow \mathcal{Y}\) such that \(f^*(x_i) = y_i\) for most \(i\)

Learning task: given \(n\) examples from the data, find an approximation \(\hat{f} \approx f^*\)

Goal: \(\hat{f}\) gives mostly correct prediction on unseen examples

Supervised learning

![Diagram of supervised learning process]

- **Training Phase**
  - Labeled training data
    - (\(n\) examples from data)
  - Learning Algorithm
  - Output: \(\hat{f}\)

- **Testing Phase**
  - Unlabeled test data
    - (unseen / future data)
  - 'classifier' \(\hat{f}\)
  - Prediction
Unsupervised Learning

Data: \( \bar{x}_1, \bar{x}_2, \ldots \in \mathcal{X} \)

Assumption: there is an underlying structure in \( \mathcal{X} \)

Learning task: discover the structure given \( n \) examples from the data

Goal: come up with the summary of the data using the discovered structure

Partition the data into meaningful structures

Find a low-dimensional representation that retains important information, and suppresses irrelevant/noise information

Let’s take a closer look using an example…
Example: Handwritten digits revisited

Handwritten digit data, but with no labels

What can we do?

• Suppose know that there are 10 groupings, can we \textit{find the groups}?

• What if we don’t know there are 10 groups?

• How can we \textit{discover/explore} other structure in such data?

A 2D visualization of digits dataset
Handwritten digits visualization
Grouping The Data, aka Clustering

Data: \( \vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathcal{X} \)

Given: known target number of groups \( k \)

Output: Partition \( \vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \) into \( k \) groups.

This is called the clustering problem, also known as unsupervised classification, or quantization.
**k-means**

Given: data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Idea:
find a set of representatives $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ such that data is close to some representative

Optimization:
\[
\text{minimize}_{c_1, \ldots, c_k} \left[ \sum_{i=1}^{n} \min_{j=1, \ldots, k} \| \vec{x}_i - \vec{c}_j \|_2^2 \right]
\]

Unfortunately this is NP-hard
Even for $d=2$ and $k=2$

How do we optimize this?

How do we solve for $d=1$ or $k=1$ case?
Algorithm to approximate $k$-means

Given: data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Alternating optimization algorithm:

- Initialize cluster centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ (assuming the data partition is fixed)

Demo:
Given: data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (assuming the data partition is fixed)

Demo:
Algorithm to approximate $k$-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (assuming the data partition is fixed)

Demo:
Algorithm to approximate $k$-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Alternating optimization algorithm:

• Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (say randomly)
• Repeat till no more changes occur
  • Assign data to its closest center (this creates a partition) (assume centers are fixed)
  • Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (assuming the data partition is fixed)

Demo:
Algorithm to approximate $k$-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Alternating optimization algorithm:

• Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (say randomly)
• Repeat till no more changes occur
  • Assign data to its closest center (this creates a partition) (assume centers are fixed)
  • Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (assuming the data partition is fixed)

Demo:
Algorithm to approximate \( k \)-means

Given: data \( \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d \), and intended number of groupings \( k \)

Alternating optimization algorithm:

- Initialize cluster centers \( \vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k \) (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers \( \vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k \) (assuming the data partition is fixed)

Demo:
Algorithm to approximate $k$-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, and intended number of groupings $k$

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$ (assuming the data partition is fixed)

Demo:
Some properties of this alternating updates algorithm:

- The approximation can be arbitrarily bad, compared to the best cluster assignment!
- Performance quality heavily dependent on the initialization!

\(k\)-means:

- How to select \(k\)?

\(is\ the\ right\ k=2\ or\ k=3?\)

**Solution:** encode clustering for all values of \(k!\) (hierarchical clustering)
Example: Clustering Without Committing to $k$

$K=3$ (coarser resolution)  
$k=6$ (finer resolution)
Hierarchical Clustering

Two approaches:

Top Down (divisive):
• Partition data into two groups (say, by k-means, with k=2)
• Recurse on each part
• Stop when cannot partition data anymore (ie single points left)

Bottom Up (agglomerative):
• Start by each data sample as its own cluster (so initial number of clusters is $n$)
• Repeatedly merge “closest” pair of clusters
• Stop when only one cluster is left
Clustering via Probabilistic Mixture Modeling

Alternative way to cluster data:

Given: \( \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d \) and number of intended number of clusters \( k \).
Assume a joint probability distribution \( (X, C) \) over the joint space \( \mathbb{R}^d \times [k] \)

\[
C \sim \begin{pmatrix}
\pi_1 \\
\vdots \\
\pi_k
\end{pmatrix}
\]

Discrete distribution over the clusters \( P[C=i] = \pi_i \)

\[X|C = i \sim \text{Some multivariate distribution, e.g. } N(\mu_i, \Sigma_i)\]

Parameters: \( \theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_k, \mu_k, \Sigma_k) \)

Modeling assumption data \( (x_1,c_1), \ldots, (x_n,c_n) \) i.i.d. from \( \mathbb{R}^d \times [k] \)

BUT only get to see partial information: \( x_1, x_2, \ldots, x_n \) \( (c_1, \ldots, c_n \text{ hidden!}) \)
Gaussian Mixture Modeling (GMM)

Given: \( \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d \) and \( k \).

Assume a joint probability distribution \( (X, C) \) over the joint space \( \mathbb{R}^d \times [k] \):

\[
C \sim \begin{pmatrix}
\pi_1 \\
\vdots \\
\pi_k
\end{pmatrix}
\]

\[
X | C = i \sim N(\mu_i, \Sigma_i)
\]

\[
\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_k, \mu_k, \Sigma_k)
\]

The Gaussian Mixture Model:

\[
P[\vec{x} | \theta] = \sum_{i=1}^{k} \pi_i \frac{1}{\sqrt{(2\pi)^d \det(\Sigma_i)}} \exp \left\{ -\frac{1}{2} (\vec{x} - \mu_i)^T \Sigma_i^{-1} (\vec{x} - \mu_i) \right\}
\]

*Example in \( \mathbb{R}^2 \times [3] \):*
GMM: Parameter Learning

\[ P[\mathbf{x} | \theta] = \sum_{i=1}^{k} \pi_i \frac{1}{\sqrt{(2\pi)^d \det(\Sigma_i)}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \bar{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x} - \bar{\mu}_i) \right\} \]

\[ \theta = (\pi_1, \bar{\mu}_1, \Sigma_1, \ldots, \pi_k, \bar{\mu}_k, \Sigma_k) \]

So... how to learn the parameters \( \theta \)?

MLE approach:

Given data \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d \) i.i.d.

\[ \theta_{\text{MLE}} := \arg \max_{\theta} \sum_{i=1}^{n} \ln P[\mathbf{x} | \theta] \]

\[ = \arg \max_{\theta} \sum_{i=1}^{n} \ln \left[ \sum_{j=1}^{n} \pi_j \frac{1}{\sqrt{(2\pi)^d \det(\Sigma_j)}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \bar{\mu}_j)^T \Sigma_j^{-1} (\mathbf{x} - \bar{\mu}_j) \right\} \right] \]

umm... now what? Cannot really simplify further!
MLE for Mixture modeling (like GMMs) is NOT a convex optimization problem.

In fact **Maximum Likelihood Estimate** for GMMs is degenerate!

\[ X = \mathbb{R}, \ k = 2 \] (fit two Gaussian in 1d):

Which pair of Gaussians gives higher likelihood?

As \( \sigma \to 0 \), MLE \( \to \infty \)!

Aside: why doesn’t this occur when fitting one Gaussian?
GMM: (local) Maximum Likelihood

So, can we make any progress?

Observation: even though a global MLE maximizer is not appropriate, several local maximizers are desirable!

An example non-maximized likelihood

(Do a few steps of gradient ascent)

Reaches a desirable local maximum!

A better algorithm for finding good parameters: Expectation Maximization (EM)
Expectation Maximization (EM) Algorithm

Similar in spirit to the alternating update for $k$-means algorithm

Idea:

- Initialize the parameters arbitrarily
- Given the current setting of parameters find the best (soft) assignment of data samples to the clusters (Expectation-step)
- Update all the parameters with respect to the current (soft) assignment that maximizes the likelihood (Maximization-step)
- Repeat until no more progress is made.
EM for GMM

Initialize $\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_k, \mu_k, \Sigma_k)$ arbitrarily

**Expectation-step:** For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$ compute the assignment $w_{j}^{(i)}$ of data $x_i$ to cluster $j$

$$w_{j}^{(i)} := \frac{\pi_j \sqrt{\det(\Sigma_j^{-1})} \exp \left( -\frac{1}{2} (\bar{x} - \mu_j)^T \Sigma_j^{-1} (\bar{x} - \mu_j) \right)}{\sum_{j'=1}^{k} \pi_{j'} \sqrt{\det(\Sigma_{j'}^{-1})} \exp \left( -\frac{1}{2} (\bar{x} - \mu_{j'})^T \Sigma_{j'}^{-1} (\bar{x} - \mu_{j'}) \right)}$$

**Maximization-step:** Maximize the log-likelihood of the parameters (with respect to complete data)

$$\pi_j := \frac{1}{n} \sum_{i=1}^{n} w_{j}^{(i)} \quad \mu_j := \frac{1}{n} \sum_{i=1}^{n} w_{j}^{(i)} \bar{x}_i$$

$$\Sigma_j := \frac{1}{n} \sum_{i=1}^{n} w_{j}^{(i)} (\bar{x}_i - \mu_j)(\bar{x}_i - \mu_j)^T$$

*Why?*
EM for GMM in Action

Arbitrary $\theta$
assignment
EM for GMM in Action

$E$ step: soft assignment of data
EM for GMM in Action

$L = 1$

*M step: Maximize parameter estimate*
EM for GMM in Action

After two rounds

$L = 2$
EM for GMM in Action

After five rounds

$L = 5$
EM for GMM in Action

After twenty rounds
What We Learned...

- Unsupervised Learning problems:
  - Clustering and Dimensionality Reduction
- K-means
- Hierarchical Clustering
- Gaussian Mixture Models
- EM algorithm
Questions?
Next time...

Dimension reduction!