4/2/2024: Learning GMMs

Given data, how to <u>learn</u> $\pi_{1:K}$, $\mu_{1:K}$, and $\Sigma_{1:K}$?

Algorithm: Expectation-Maximization (EM)

Used wherever you have:

1. Latent variables (that are unknown) $z_1, \dots, z_n$
2. Unknown parameters (need to learn) $\pi_{1:K}, \mu_{1:K}, \Sigma_{1:K}$

Strategy: Alternate between guessing each one

alternate

1. <u>E-step</u> ("Expectation"):
   Infer latent variable distribution using current guess of parameters

   <u>In K-Means</u>
   $\approx$ make assignments based on current centroids

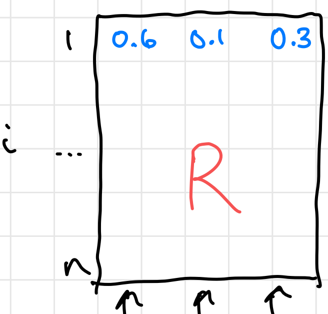
2. <u>M-step</u> ("Maximization"):
   Choose parameters that fit the data best based on inferred distribution of latent variables

   $\approx$ choose new centroids based on new assignment

$\boxed{\text{E-step}}$ For each $i = 1, \dots, n$, we infer:

$$R_{ij} = p\left(z_i = j \mid x_i = x^{(i)} ; \text{ current guess of } \pi_{1:K}, \mu_{1:K}, \Sigma_{1:K}\right)$$

Run inference using Bayes Rule



prob of being from cluster j

$\boxed{\text{M-step}}$ Inputs are:

- Actual values of X_i's
- Inferred distribution of all Z_i's

Goal: Find the best π, μ, Σ

Imagine: we actually had true value of Z_i for each i

Then we could apply Maximum Likelihood Estimation:

$$\text{maximize} \sum_{i=1}^{n} \log P(Z_i, X_i)$$

$$= \sum_{i=1}^{n} \log P(Z_i) + \log P(X_i \mid Z_i)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{K} \mathbb{1}[Z_i = j] \left[\log P(Z_i = j) + \log P(X_i \mid Z_i = j)\right]$$

Actual M-step: Approximate the MLE objective using inferred R_{ij}'s

Expected Complete Log-Likelihood (ECLL)

$$ECLL(\pi_{1:K}, \mu_{1:K}, \Sigma_{1:K}) =$$

$$\underbrace{\sum_{i=1}^{n} \sum_{j=1}^{K} \underbrace{R_{ij}}_{\substack{\text{How much to weight the} \\ \text{possibility that } Z_i = j}} \cdot \underbrace{\left[\log P(Z_i = j) + \log P(X_i = x^{(i)} \mid Z_i = j)\right]}_{\substack{\text{"Complete log likelihood"} \\ \text{includes } x\text{'s and } Z\text{'s}}}}_{\text{Expected value with } R_{ij}\text{'s as weights}}$$

Now: Derive formula for μ_j to maximize ECLL, discuss formulas for π_j & Σ_j

Plan: Take ∇_{μ_j} ECLL, set to 0, solve for μ_j

$$\nabla_{\mu_j} \sum_{i=1}^{n}\sum_{j=1}^{k} R_{ij} \cdot \Big[\underbrace{\cancel{\log P(z_i=j)}}_{\text{Depends only on } \pi_j \text{; } \nabla \text{ will be } 0} + \log \underbrace{P(X_i = x^{(i)} \mid z_i = j)}_{\text{For each } j\text{, this depends on that cluster's } \mu \text{ \& } \Sigma} \Big]$$

$\Rightarrow$ Ignore all terms except for the μ_j we care about

$$= \sum_{i=1}^{n} R_{ij} \nabla_{\mu_j} \log P(X_i = x^{(i)} \mid z_i = j)$$

Plug in multivariate gaussian pdf formula

$$= \underbrace{\cancel{\frac{1}{(2\pi)^{d/2}}}}_{\text{constant}} \underbrace{\cancel{\frac{1}{\sqrt{\det(\Sigma_j)}}}}_{\text{doesn't depend on } \mu_j} \exp\left(-\frac{1}{2}(x^{(i)}-\mu_j)^T \Sigma_j^{-1} (x^{(i)}-\mu_j)\right)$$

$$= \sum_{i=1}^{n} R_{ij} \nabla_{\mu_j} \left(-\frac{1}{2}(x^{(i)}-\mu_j)^T \Sigma_j^{-1} (x^{(i)}-\mu_j)\right)$$

$\boxed{\text{Fact: } \nabla_x x^T A x = 2Ax}$

$$= \sum_{i=1}^{n} R_{ij} \left(\cancel{-\frac{1}{2}} \cdot \cancel{2}\, \cancel{\Sigma_j^{-1}} (x^{(i)}-\mu_j) \cdot \cancel{-1}\right) = 0$$

multiply by Σ_j on left

$$\sum_{i=1}^{n} R_{ij} (x^{(i)}-\mu_j) = 0$$

$$\sum_{i=1}^{n} R_{ij}\, x^{(i)} = \sum_{i=1}^{n} R_{ij}\, N_j$$

$$\boxed{N_j = \frac{\sum_{i=1}^{n} R_{ij}\, x^{(i)}}{\sum_{i=1}^{n} R_{ij}}} \quad \star$$

$P(x_1$ is in cluster $j)$
$P(x_2$ is in cluster $j)$
$\vdots$
$+\, P(x_n$ is in cluster $j)$

Expected # of points in cluster j

Weighted average of $x^{(i)}$ weighted by the probability that each $x^{(i)}$ is in cluster j

M-step formulas for π_j & Σ_j

$$\pi_j = \frac{\sum_{i=1}^{n} R_{ij}}{n}$$

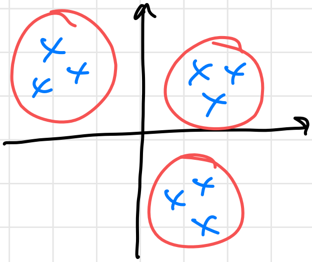
"soft" version of $\dfrac{\#\text{ of points in cluster } j}{\text{total } \#\text{ points in data}}$

$$\Sigma_j = \boxed{\frac{\sum_{i=1}^{n} R_{ij}\,(x^{(i)} - N_j)(x^{(i)} - N_j)^T}{\sum_{i=1}^{n} R_{ij}}}$$

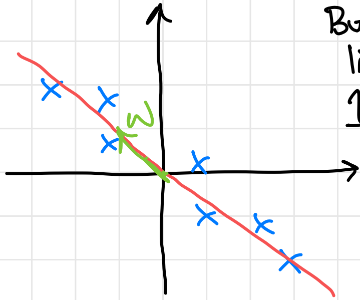
Formula for covariance matrix is $\frac{1}{n}\sum_{i=1}^{n} (x^{(i)} - N)(x^{(i)} - N)^T$

"weighted average" version of covariance matrix formula

Dimensionality Reduction



Clustering



Data in $\mathbb{R}^2$
But "mostly" lies in 1-dimensional subspace

Dimensionality Reduction:
Find a low-dimensional subspace that preserves most information in the dataset

Method: Principal Components Analysis (PCA)

commonly: map high-dim data $\rightarrow$ 2 dimensions

Starting Point: Try to find best 1-D subspace

Key Assumption: Data has mean 0

i.e. $\frac{1}{n}\sum_{i=1}^{n} x^{(i)} = 0$

Enforce by computing mean of data, then subtract it from each example

What is our parameter to learn?

Learn a single parameter vector $w \in \mathbb{R}^{\boxed{d}}$ that defines the 1-D subspace to project onto

original dimension of data

We'll focus only on w where $\|w\| = 1$ since we only care about direction

"Reconstruction Error": How well can we reconstruct $\{x^{(1)}, \dots, x^{(n)}\}$ based on only the projections of $\{x^{(1)}, \dots, x^{(n)}\}$ onto subspace defined by w