10/26/2023: K-Means Clustering

Machine Learning

Supervised Learning

Training Dataset:

$D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$

input to model

desired output

Goal: Learn function mapping x to y

Unsupervised Learning

Training Dataset only contains x's

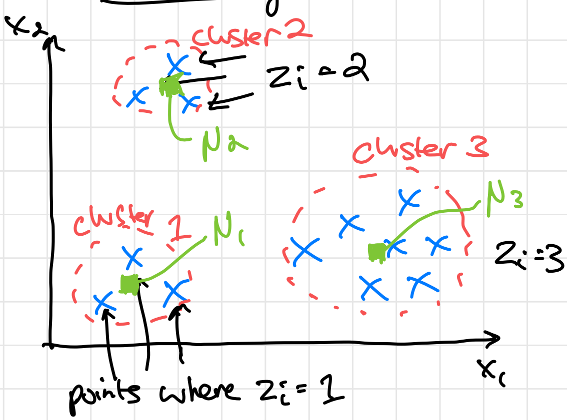
$D = \{x^{(1)}, \ldots, x^{(n)}\}$

No "correct output" per example

Goal: Learn what structure is present in the data

1. Sub-population / group / clusters
2. low-dimensional structure (subspace structure)
3. Similarity / relationship structure

Clustering



Input:

Dataset $= \{x^{(1)}, \ldots, x^{(n)}\}$

Given: # of clusters in data K

Output: An assignment $z_1, z_2, \ldots, z_n$ where $z_i \in \{1, \ldots, K\}$ denotes cluster we assign to $x^{(i)}$

Today: K-Means Clustering Algorithm

Idea 1: Write down a loss function to define what is a good assignment $z_1, \ldots, z_n$

Idea 2: Each cluster has a centroid N_j for $j = 1, \ldots, K$

Loss = how far is $x^{(i)}$ from centroid it was assigned to

Loss function (formally):

$$L(\underbrace{z_{1:n}}_{=[z_1,\ldots,z_n]}, \mu_{1:k}) = \sum_{i=1}^{n} \| x^{(i)} - \mu_{z_i} \|^2$$

z_i: cluster ID assigned to $x^{(i)}$

μ_{z_i}: centroid for the assigned cluster of $x^{(i)}$

"Reconstruction Error" Loss

If we only knew the cluster assignments $z_{1:n}$ and cluster means $\mu_{1:k}$

how well could we reconstruct $\{x^{(1)}, \ldots, x^{(n)}\}$?

Goal: minimize $L(z_{1:n}, \mu_{1:k})$ with respect to $z_{1:n}, \mu_{1:k}$

Note: Can't do gradient descent because z_i's are discrete

Can't take a derivative w.r.t. z_i

Strategy: Alternating Minimization

1. Start with a random choice of $\mu_1, \ldots, \mu_k$

Alternate until $\boxed{\text{convergence}}$ ← when $z_{1:n}$'s and $\mu_{1:k}$'s stop changing

2. Choose $z_{1:n}$ to minimize L given current $\mu_{1:k}$
3. Choose $\mu_{1:k}$ to minimize L given current $z_{1:n}$

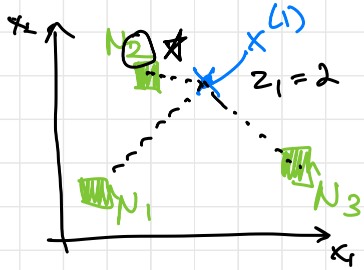
$\boxed{\text{Step 1}}$: Choose each μ_j to be a different random $x^{(i)}$ from the dataset

$\boxed{\text{Step 2}}$: Minimizing L w.r.t. $z_{1:n}$

For each $i = 1, \ldots, n$:

set $z_i = \underset{j=1,\ldots,k}{\operatorname{argmin}} \| x^{(i)} - \mu_j \|^2$

English: choose cluster whose μ_j is closest

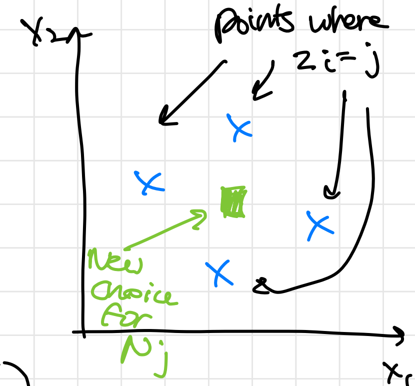


Step 3: Minimizing L w.r.t. $\mu_{1:K}$

Intuitively: μ_j should be mean of all points where $z_i = j$

minimize $\displaystyle\sum_{i=1}^{n} \|x^{(i)} - \mu_{z_i}\|^2$

$$= \sum_{j=1}^{K} \sum_{i: z_i = j} \|x^{(i)} - \mu_j\|^2$$



For particular value of j:

$$\underset{\mu_j}{\text{minimize}} \sum_{i: z_i = j} \|x^{(i)} - \mu_j\|^2$$

Take gradient w.r.t. μ_j, set $= 0$

$$\sum_{i: z_i = j} \cancel{2} \cdot (x^{(i)} - \mu_j) \cdot \cancel{(-1)} = 0$$

$$\sum_{i: z_i = j} x^{(i)} = |\{i : z_i = j\}| \cdot \mu_j$$

$$\Rightarrow \quad \mu_j = \frac{1}{|\{i : z_i = j\}|} \sum_{i: z_i = j} x^{(i)}$$

Average of points in cluster j

Note: Eventually we will converge
why? At every step L gets better

<u>No guarantee</u> of converging to optimal solution
Random initialization influences final clustering

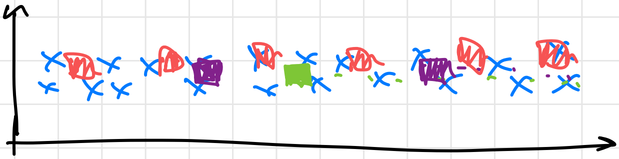
ie global minimum of L

How do you choose k?

this is a hyperparameter

Wrong answer: Fit on a dev set

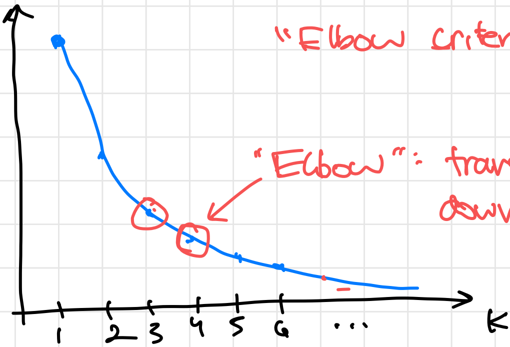
Why? larger k always makes loss go down



$k=1$: Loss large

$k=2$: loss goes down

$k=6$: loss even lower



"Elbow criterion"

"Elbow": transition between loss going down rapidly vs. loss going down slowly

Choose k at elbow

K-means looks for spherical clusters (because it uses Euclidean distance)

Goal: New algorithm that can learn location and shape of clusters