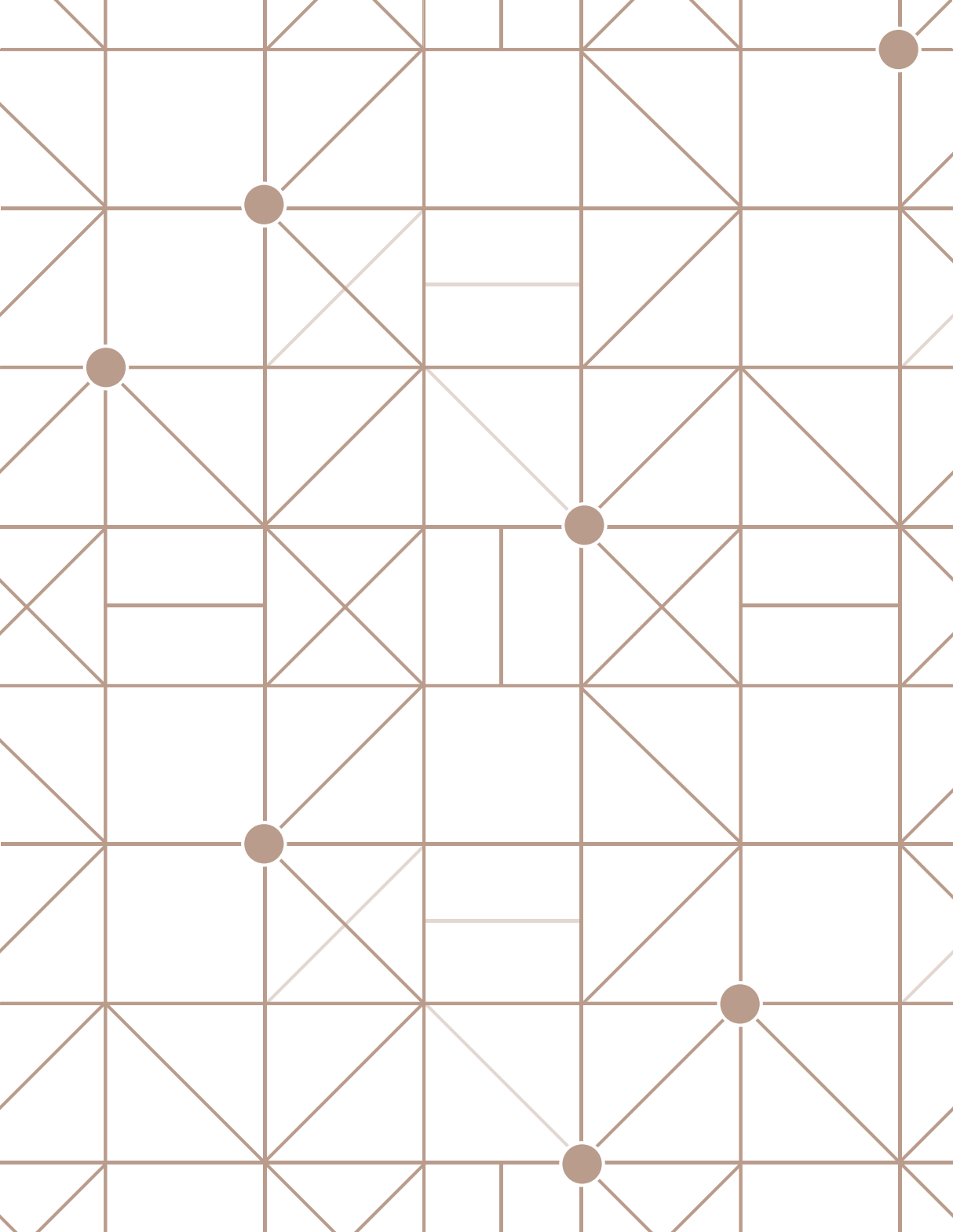


Distance & Similarity

n data pts $\begin{bmatrix} x_{11} & x_{12} & x_{1m} \\ x_{21} & \dots & \dots \\ x_{n1} & \dots & \dots \end{bmatrix}$

m features

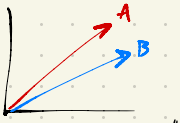
- Feature space: Context of the data

Distance function:
- A, B close
- B, C close
- A, C not close

} Triangle ineq. guarantees $d(A,C)$ is small.

- Dist. function is symmetric

Dissimilarity –
- Large if pts are far
- Small if pts are close



dissim(A,B) is small

Minkowski Distance

$$L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

P=2 → Euclidean distance (circle)
P=1 → Manhattan distance (diamond)

For $p \geq 1$

- Not a dist. function when $0 < p < 1$
  - Violates triangle inequality

Cosine Similarity

- $s(x,y) = \cos\theta$ – Returns large value if objects are similar
  - θ: angle btwn vectors x and y
    - Proportional vectors: 1
    - Orthogonal vectors: 0
    - Opposite vectors: −1
- Use $s(x,y)$ when direction matters more than magnitude

Jaccard Similarity –

- Use Manhattan Similarity to return size of set difference
- $Jsim(x,y) = \frac{|x \cap y|}{|x \cup y|}$ – intersection of pts / Union of pts

$Jdist(x,y) = 1 - Jsim(x,y)$

Norms

$d(A,B) = \|A - B\|$
$d(0,x) = \|x\|$

- Not all distances create a norm
- Minkowski dist = Lp norm

Clustering

- Grouping similar data points into clusters (or dissimilar to other groups)
- Clusters can be ambiguous – depend on circumstances
- Types:
  - Partitional - Each object belongs to 1 cluster – Partition into k-partitions
  - Hierarchical - Set of nested clusters in tree
  - Density-based - Defined based on local density of points
  - Soft clustering - Each point assigned to every cluster based on probability

- Goal of clustering – Minimize variance
- "Cost function" – small output if good, big output if bad
  - $\sum_{i}^{k} \sum_{x \in C_i} d(x_j, \mu_i)^2$
  - Way to evaluate/compare solutions

$k = 1$: One large cluster

$k = n$: Every pt is a cluster

- k-means - Given $X = \{x_1, \ldots, x_n\}$ dataset & euclidean dist., and k
  Find k centers that minimize cost function

Lloyd's Algorithm

1. Randomly pick k centers $\{\mu_1, \ldots, \mu_k\}$
2. Assign each pt to closest center
3. Compute new centers as means of each cluster
4. Repeat 2 and 3 until convergence

kmeans ++

Goal - Put together similar points

Does Lloyd's always converge?

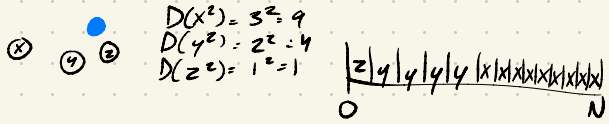
Proof (by contradiction): Suppose it does not converge. Then either:

1. Finite points, finite partitions
   - Sequence of costs cannot decrease towards infinity — impossible bc finite partitions, thus cannot go to infinity
2. Algorithm stuck in cycle/loop
   - Impossible

- Thus, algorithm always converges, but not to optimal solution

Farthest-first traversal: Pick a point, then pick farthest pt
Problem: Datasets are weird, so it usually chooses outliers :(

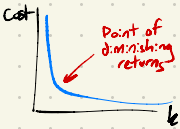
k-means ++: Pick points at random, but assign prob of being picked based on distance from center
- Further you are, higher prob of being picked



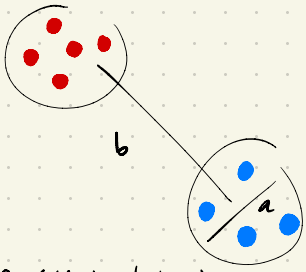
- If box picks 12 - x
- If box picks 4 - y

- Box generates only 0 or 1?
  - Scale everything by 1/4.

How to choose right k?
1. Iterate through different k-values (elbow method)



- Minimizes cost
  - Cons: Computationally expensive; cost isn't always significant + doesn't always represent best outcome
- Evaluation: Goal is that similar pts in same cluster (kmeans only cares about this)
  - + dissimilar in diff. clusters



- If (b-a) large: b > a, good

Normalize - $\frac{(b-a)}{\max(a,b)}$
- If close to 1: ☺
  If close to 0: :(

- a - avg. in-cluster dist
- b - avg. intra-cluster dist
  - (b-a): 0 — clusters are next to each other
  - large b is good

Silhouette Score - for each data pt i:

a_i = Mean dist. from pt i to every other pt in cluster

b_i = Smallest mean dist from pt i to every pt in another cluster

$S_i = \frac{(b_i - a_i)}{\max(a_i, b_i)}$