MA 5755: Data Analysis & Visualization in R/Python/SQL

Week 3: Distance Geometry and K-means Clustering

Dr. Rakhi Singh
Department of Mathematics
IIT Madras

Spring 2026

Outline

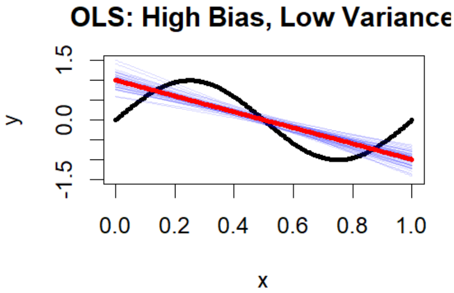
1. Review
2. Overview of Distance Geometry and k-means
3. k-medoids
4. Prototype methods (for Classification)
5. Nearest Neighbors
6. Dimension Reduction
7. Categorical Predictors
8. Computational Cost

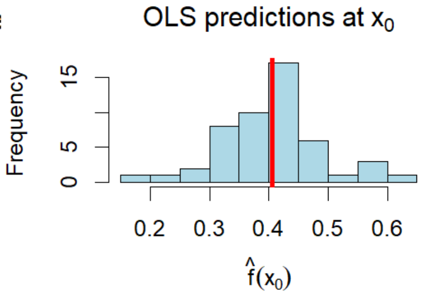
Geometry Explains Bias and Variance of OLS and kNN

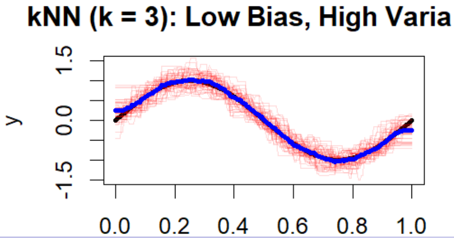
The prediction surfaces and their shapes explain bias and variance.

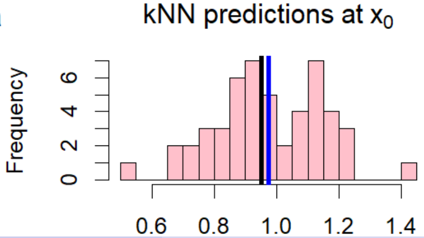
- High bias, low variance: A smooth, rigid surface (as in least squares) uses one global shape everywhere. It ignores local curvature in the data, but gives stable predictions across the space. It cannot bend to match curved patterns (high bias). It changes little if you resample data (low variance).
- Low bias, high variance: A flexible, patchy surface (as in kNN with small k) bends to follow nearby points. It captures local structure, but changes strongly when the data change. Shape is flexible and follows data closely (low bias). It changes a lot if data change (high variance).
- **The tradeoff:** Learning is choosing how much freedom the surface should have: too stiff misses structure, too flexible follows noise.

A simulation: OLS vs kNN ($k = 3$)

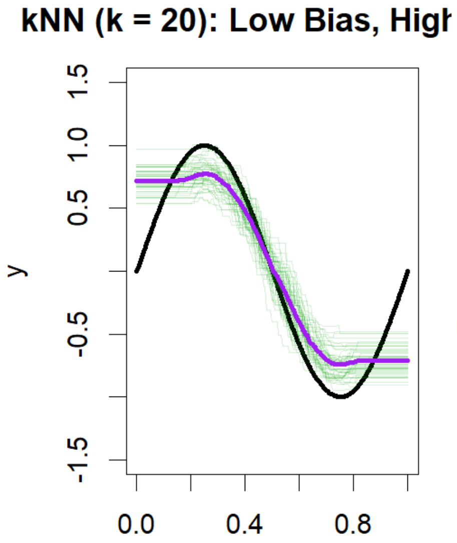


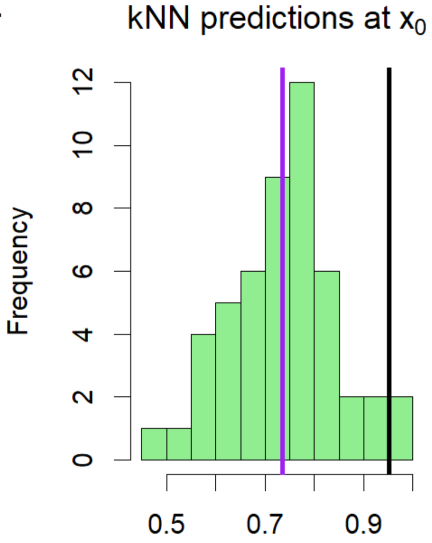






A simulation: kNN (with $k = 20$)





Bias-Variance Tradeoff of OLS

Assume the true model is linear:

$$Y = X^T\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2).$$

For least squares, the expected prediction error satisfies:

$$\mathbb{E}[\text{EPE}] \approx \sigma^2 \left(1 + \frac{p}{N}\right).$$

- There is no bias under the correct linear model.
- Variance grows only linearly with dimension.
- Strong structural assumptions prevent exponential complexity.
- But the model assumptions must hold. No guarantee if they don't.

Structure replaces locality as the defense against high dimension.

Bias–Variance Behavior of kNN

Assume the regression model

$$Y = f(X) + \varepsilon, \quad \mathbb{E}[\varepsilon] = 0,\ \mathrm{Var}(\varepsilon) = \sigma^2.$$

The expected prediction error at x_0 decomposes as

$$\mathrm{EPE}(x_0) = \sigma^2 + \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Bias}^2(\hat{f}(x_0)).$$

For kNN, asymptotically:

$$\mathrm{Var}(\hat{f}(x_0)) \gtrsim \frac{\sigma^2}{k}, \qquad \mathrm{Bias}^2(\hat{f}(x_0)) \sim \left(\frac{k}{N}\right)^{4/p}.$$

- Small k: low bias, high variance.
- Large k: higher bias, lower variance.
- As p increases, bias decays extremely slowly (*curse of dimensionality*).

Outline

1. Review
2. Overview of Distance Geometry and k-means
3. k-medoids
4. Prototype methods (for Classification)
5. Nearest Neighbors
6. Dimension Reduction
7. Categorical Predictors
8. Computational Cost

Overview

Many learning algorithms are defined entirely in terms of distances.

- Similarity, neighborhoods, and clusters are geometric notions.
- Distances determine which points interact with each other.
- Geometry becomes fragile in high dimensions.

In this lecture, we study the following three methods.

- **k-means:** An unsupervised, global clustering method that partitions the data space into k regions by minimizing within-cluster squared distances to centroids.
- **k-medoids:** An unsupervised, distance-based clustering method that represents each cluster by a real data point, improving robustness and allowing arbitrary distance metrics.
- **k-Nearest Neighbors (kNN):** A supervised, local method that predicts by averaging responses of the k closest points.

From Pairwise Distances to Clusters

Suppose we observe data points $x_1, \ldots, x_N \in \mathbb{R}^p$ with no labels.

- Clustering groups points that are close under a distance metric.
- Euclidean distance is the most common choice:

$$d(x_i, x_j) = \|x_i - x_j\|.$$

k-means seeks a partition of the data into K clusters. Let $\mathcal{C}_1, \ldots, \mathcal{C}_K$ be clusters with centroids $\mu_1, \ldots, \mu_K$. The k-means objective minimizes within-cluster distortion:

$$\sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|x_i - \mu_k\|^2.$$

- Each cluster is summarized by a single point.
- Geometry is approximated by K representatives.

Geometric Interpretation of k-means

The k-means objective has a simple geometric meaning.

- Each cluster defines a Voronoi region.
- Points are assigned to the nearest centroid.
- The centroid minimizes squared distance within its region.

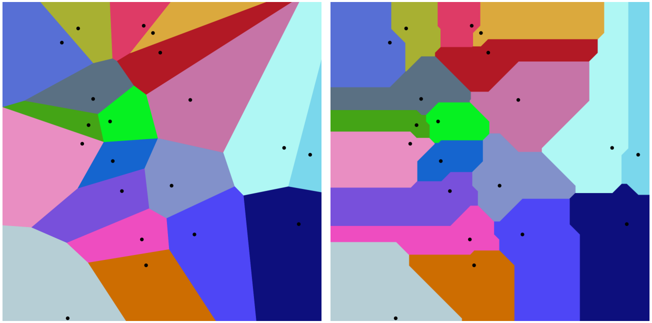


Figure: Left: Voronoi tessellation induced by (left) Euclidean distance and (right) Manhattan distance.

The k-means Algorithm

Fix a desired number of cluster centers, say R, and the K-means procedure iteratively moves the centers to minimize the total within cluster variance. Given an initial set of centers, the Kmeans algorithm alternates between the two steps:

Assignment step: For each center, identify the subset of training points that is closer to it than any other center;

$$\mathcal{C}_k = \{x_i : \|x_i - \mu_k\| \leq \|x_i - \mu_\ell\| \ \forall \ell\}.$$

Update step: The means of each feature in each cluster are computed, and this mean vector becomes the new center for that cluster.

$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} x_i.$$

Each step reduces the objective.

Why Does the Representative Depend on the Distance?

For a fixed cluster $\mathcal{C}$, the representative is defined as

$$\hat{\mu} = \arg\min_{\mu} \sum_{i \in \mathcal{C}} d(x_i, \mu).$$

For the Euclidean distance, $d(x, \mu) = \|x - \mu\|_2^2$. The objective is differentiable and strictly convex. Setting the gradient to zero gives:

$$\hat{\mu} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} x_i.$$

For the Manhattan distance, $d(x, \mu) = \|x - \mu\|_1 = \sum_{j=1}^{p} |x_{ij} - \mu_j|$. The problem separates coordinatewise. Each coordinate μ_j is any median of $\{x_{ij}\}$.

$$\hat{\mu}_j = \text{median}\{x_{ij}\}.$$

Convergence Properties and Initialization

The objective of k-means decreases at each step and is bounded below by zero. Only finitely many labelings of N points into K clusters exist. Therefore, the algorithm terminates in a finite number of iterations at a fixed point.

Initialization sensitivity.

- The objective is nonconvex in the pair (labels, centroids).
- Different initial centroids lead to very different results.
- Convergence is guaranteed only to a local minimum or saddle point.

Practical consequences.

- Poor initialization can yield suboptimal distortion.
- Multiple random restarts are used to improve solutions.
- No polynomial-time algorithm is known for the global optimum.

An example

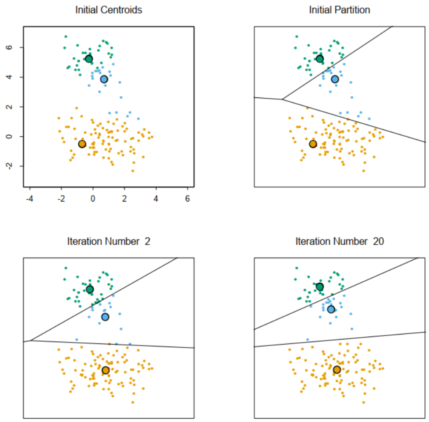


FIGURE 14.6. *Successive iterations of the K-means clustering algorithm for the simulated data of Figure 14.4.*

Geometry, Dimensionality, and k-means

Distance-based methods degrade in high dimensions:

- Pairwise distances concentrate.
- Nearest and farthest points become indistinguishable.

As a consequence:

- kNN neighborhoods cease to be local.
- k-means Voronoi cells lose geometric meaning.

k-means performs well only under strong structural assumptions:

- clusters are approximately spherical,
- within-cluster variances are comparable,
- most dimensions are informative.

In practice:

- initialization critically affects the solution,
- dimensionality reduction (e.g. PCA) and scaling often help,
- failures are geometric, not algorithmic.

Outline

1 Review

2 Overview of Distance Geometry and k-means

3 k-medoids

4 Prototype methods (for Classification)

5 Nearest Neighbors

6 Dimension Reduction

7 Categorical Predictors

8 Computational Cost

k-medoids: Definition and Objective

k-medoids is an unsupervised clustering method related to k-means, but with a different notion of cluster representative.

Representative. Each cluster $\mathcal{C}_k$ is represented by a *medoid* m_k, which must be one of the observed data points:

$$m_k \in \mathcal{C}_k.$$

Objective. k-medoids minimizes total within-cluster distance:

$$\sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} d(x_i, m_k),$$

where $d(\cdot, \cdot)$ is a chosen distance metric.

- No coordinate-wise averaging is performed.
- The objective is well-defined for arbitrary distance functions.
- The cluster representative depends directly on the loss induced by d.

k-means vs k-medoids: What Actually Changes

- **Loss being minimized**
  - k-means minimizes squared Euclidean distance.
  - k-medoids minimizes unsquared distance under a chosen metric.
- **Cluster representative**
  - k-means: arithmetic mean (may not correspond to any data point).
  - k-medoids: an observed point that minimizes total distance.
- **Assumptions and robustness**
  - k-means implicitly assumes Euclidean structure and is sensitive to outliers.
  - k-medoids allows general distances and is more robust to extreme points.
- **Computation**
  - k-means updates are closed-form and fast.
  - k-medoids requires combinatorial updates and is more expensive.

Outline

1 Review

2 Overview of Distance Geometry and k-means

3 k-medoids

4 Prototype methods (for Classification)

5 Nearest Neighbors

6 Dimension Reduction

7 Categorical Predictors

8 Computational Cost

Prototype Methods for Classification

Training data consist of N labeled observations: $(x_1, g_1), \ldots, (x_N, g_N)$, $g_i \in \{1, 2, \ldots, K\}$. The response has K classes.

Each class is represented by one or more points (prototypes) in feature space. Prototypes typically are *not* training samples, except in 1-nearest-neighbor classification.

Given a query point x, assign it to the class of the closest prototype:

$$\hat{g}(x) = g_{k^*}, \quad k^* = \arg\min_{k} \; d(x, m_k),$$

where m_k denotes a prototype and $d(\cdot, \cdot)$ is a distance metric.

- Typically, Euclidean distance is used.
- Standardizing each feature to mean 0 and variance 1.
- Sufficiently many prototypes can find highly irregular class boundaries.
- How to choose the number of prototypes and their locations?

Method 1: k-means clustering

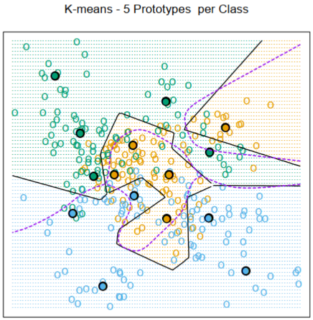
To use k-means clustering for classification of labeled data, the steps are:

- apply K-means clustering to the training data in each class separately, using R prototypes per class;
- assign a class label to each of the $K \times R$ prototypes;
- classify a new feature x to the class of the closest prototype.

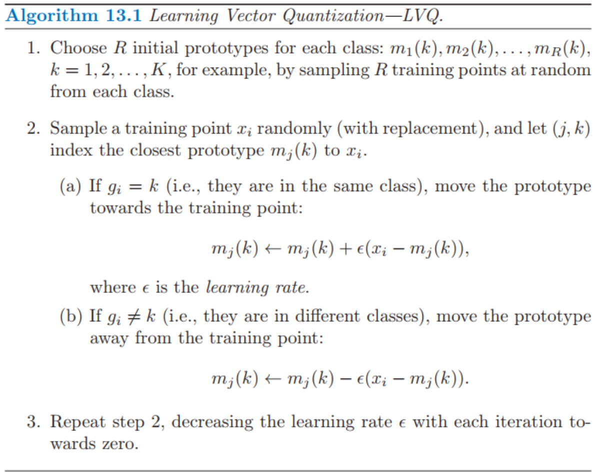
A simulated example with $K = 3$, $p = 2$, and $R = 5$.

Note that R cluster centers are used as prototypes

A shortcoming: for each class, the other classes do not have a say in the positioning of the prototypes for that class.



Method 2: Learning Vector Quantization



Algorithm 13.1 *Learning Vector Quantization—LVQ.*

1. Choose R initial prototypes for each class: $m_1(k), m_2(k), \ldots, m_R(k)$, $k = 1, 2, \ldots, K$, for example, by sampling R training points at random from each class.
2. Sample a training point x_i randomly (with replacement), and let (j, k) index the closest prototype $m_j(k)$ to x_i.
   (a) If $g_i = k$ (i.e., they are in the same class), move the prototype towards the training point:
   $$m_j(k) \leftarrow m_j(k) + \epsilon(x_i - m_j(k)),$$
   where ϵ is the *learning rate*.
   (b) If $g_i \neq k$ (i.e., they are in different classes), move the prototype away from the training point:
   $$m_j(k) \leftarrow m_j(k) - \epsilon(x_i - m_j(k)).$$
3. Repeat step 2, decreasing the learning rate ϵ with each iteration towards zero.

Since no optimization is being done, it is difficult to study these algorithms.

Outline

1 Review

2 Overview of Distance Geometry and k-means

3 k-medoids

4 Prototype methods (for Classification)

5 Nearest Neighbors

6 Dimension Reduction

7 Categorical Predictors

8 Computational Cost

k-NN classifiers

These classifiers require no model to be fit.

Given a query point x_0, we find the k training points $x(r)$, $r = 1, \ldots, k$ closest in distance to x_0, and then classify using majority vote among the k neighbors.

Ties are broken at random.

Euclidean distances are used for continuous predictors.

Despite its simplicity, k-nearest-neighbors has been successful in a large number of classification problems, including handwritten digits, and satellite image scenes.

It is often successful when each class has many possible cluster centers, and the decision boundary is very irregular.

Why kNN and k-means Are Not the Same

kNN:

- Neighborhood depends on x
- Different x sees different subsets of training data
- Effective classifier changes across space

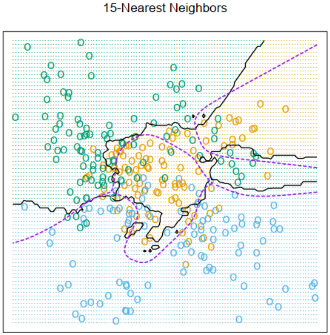
k-means classifier:

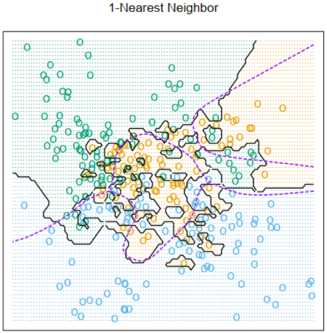
- Partition is fixed after training
- Same centroid used for all x in its Voronoi cell
- Classification ignores local density

Key distinction:

$$\text{kNN adapts at query time} \neq \text{k-means adapts at training time}$$

Example 1





The decision boundary of 15-nn is fairly smooth compared to the right panel, where a 1-nearest neighbor classifier was used.

Example 2

Textbook compares nearest-neighbors, k-means, and LVQ classifiers on two simulated two-class problems. Let $X_j \overset{iid}{\sim} \mathrm{Unif}[0,1]$, where $j = 1, \ldots, 10$.

Problem 1 (easy):

$$Y = \mathbf{1}\{X_1 > 1/2\}$$

Single relevant feature; linear decision boundary.

Problem 2 (difficult):

$$Y = \mathbf{1}\left\{\mathrm{sign}\left(\prod_{j=1}^{3}(X_j - 1/2)\right) > 0\right\}$$

Checkerboard structure in first three coordinates.

Training size $N = 100$, test size $= 1000$.

Example 2 - Results

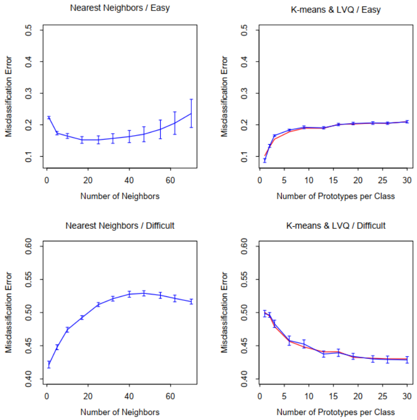


FIGURE 13.5. *Mean ± one standard error of misclassification error for nearest-neighbors, K-means (blue) and LVQ (red) over ten realizations for two simulated problems: "easy" and "difficult," described in the text.*

k-means ≈ LVQ across k; k strongly problem-dependent.

Invariant Metrics for Nearest Neighbors

Inputs: $x \in \mathbb{R}^{256}$ for (16×16) grayscale image

Natural invariances: rotation, translation (2D), scaling (2D), shear, stroke thickness. Total: 7 transformation degrees of freedom.

Problem with Euclidean distance:

$$\|x - x'\|_2 \text{ is large even if } x' \text{ is a small rotation of } x$$

Define invariance manifold for image x:

$$\mathcal{M}(x) = \{ T_\theta(x) : \theta \in \Theta \}$$

Invariant distance:

$$d_{\mathrm{inv}}(x, x') = \min_{\theta, \theta'} \| T_\theta(x) - T_{\theta'}(x') \|_2$$

Nearest-neighbor classification using d_{inv} exploits invariance. But computation of d_{inv} is infeasible and large transformations are undesirable (e.g. $6 \leftrightarrow 9$).

Tangent Distance Approximation

Approximate $\mathcal{M}(x)$ locally by its tangent space.

$$T_\theta(x) \approx x + \sum_{l=1}^{q} \theta_l v_l, \quad v_l = \left. \frac{\partial T_\theta(x)}{\partial \theta_l} \right|_{\theta=0} \text{ are tangent vectors.}$$

Tangent distance between images x and x':

$$d_T(x, x') = \min_{\alpha,\beta} \left\| x + \sum_l \alpha_l v_l - \left(x' + \sum_l \beta_l v'_l \right) \right\|_2$$

Equivalent to distance between two affine subspaces.

Classification rule works by computing tangent spaces for each training image and query image and then performing 1-NN using d_T.

This method but avoids explicit data augmentation. Tangent spaces are almost surely non-intersecting in $\mathbb{R}^{256}$.

Example 3 - Hand-written digits

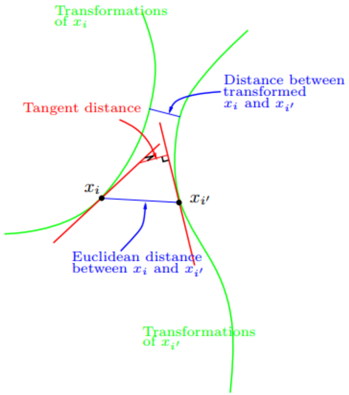

Transformations of x_i
Distance between transformed x_i and $x_{i'}$
Tangent distance
x_i
$x_{i'}$
Euclidean distance between x_i and $x_{i'}$
Transformations of $x_{i'}$

FIGURE 13.11. *Tangent distance computation for two images x_i and $x_{i'}$. Rather than using the Euclidean distance between x_i and $x_{i'}$, or the shortest distance between the two curves, we use the shortest distance between the two tangent lines.*

Outline

1 Review

2 Overview of Distance Geometry and k-means

3 k-medoids

4 Prototype methods (for Classification)

5 Nearest Neighbors

6 Dimension Reduction

7 Categorical Predictors

8 Computational Cost

Global Dimension Reduction for Nearest Neighbors

Nearest-neighbor methods can benefit from applying the rule in a lower-dimensional subspace of the original feature space.

Suppose the data $\mathbb{X}$ live in $\mathbb{R}^p$, but only an unknown $L \ll p$ dimensional subspace carries information relevant for discrimination; remaining directions are noise.

Let (θ_ℓ, e_ℓ) be the eigenpairs of $\mathbb{X}$, ordered by decreasing θ_ℓ. The leading L eigenvectors span the optimal global subspace. The rank-L approximation

$$\mathbb{X}_{[L]} = \sum_{\ell=1}^{L} \theta_\ell e_\ell e_\ell^\top .$$

Nearest-neighbor classification is then performed in the reduced L-dimensional space obtained by methods such as PCA.

Outline

1 Review

2 Overview of Distance Geometry and k-means

3 k-medoids

4 Prototype methods (for Classification)

5 Nearest Neighbors

6 Dimension Reduction

7 Categorical Predictors

8 Computational Cost

Categorical Predictors and k-means

k-means requires a numeric representation and a notion of averaging.

Ordinal predictors (ordered categories):

- Encode levels as integers (e.g. low$<$medium$<$high $\rightarrow$ 1,2,3).
- Treat as numeric and apply k-means with Euclidean distance.
- Implicit assumption: equal spacing between adjacent levels.

Centroids may be non-integer; interpretation is approximate but common in practice.

Nominal predictors (unordered categories):

- No meaningful subtraction or averaging.
- Euclidean distance and means are undefined.
- Vanilla k-means is inappropriate.

Standard alternatives:

- one-hot encoding $+$ k-means (heuristic),
- k-modes (principled replacement).

Mixed Attributes: Modified Distances and Prototypes

For data with continuous and categorical predictors, replace squared Euclidean loss with something like:

$$d(x, \mu) = \sum_{j \in \text{cont}} (x_j - \mu_j)^2 \; + \; \lambda \sum_{j \in \text{cat}} \mathbf{1}\{x_j \neq \mu_j\}$$

- Continuous variables $\rightarrow$ means.
- Categorical variables $\rightarrow$ modes.
- λ controls relative weighting.

Summary:

- k-means applies only when averaging is meaningful.
- Ordinal data can be coerced into k-means with care.
- Nominal data require mode-based prototypes.
- Mixed data require mixed distances.

Outline

1 Review

2 Overview of Distance Geometry and k-means

3 k-medoids

4 Prototype methods (for Classification)

5 Nearest Neighbors

6 Dimension Reduction

7 Categorical Predictors

8 Computational Cost

Computational Cost of Nearest-Neighbor Methods

Nearest-neighbor methods place most of their burden at prediction time.

For N training points in $\mathbb{R}^p$, a single kNN query requires:

$$O(Np)$$

operations to compute distances and identify neighbors.

- The full training set must be stored.
- Prediction cost grows linearly with both N and p.
- This contrasts with parametric models, where prediction is $O(p)$.

Various data structures (e.g., k-d trees, ball trees) can reduce average search cost, but their effectiveness deteriorates as dimension increases.

Editing and Condensing the Training Set

Reducing storage and computation motivates pruning the training data.

Only a subset of training points is needed for accurate nearest-neighbor prediction, especially those near decision boundaries.

- **Editing methods:** remove points that are consistently misclassified.
- **Condensing methods:** keep only points essential for defining class boundaries.

Examples.

- **Multi-edit algorithm (Devijver & Kittler, 1982):** iteratively deletes misclassified points using cyclic train–test splits.
- **Condensed NN (Hart, 1968):** incrementally builds a minimal subset by adding points only when needed to correct misclassification.

These methods can reduce storage substantially, but lack strong guarantees and have seen limited systematic evaluation.