Analysis of K-Nearest Neighbors (KNN)

Your Name
sayan.chaki@inria.fr

March 19, 2025

K-Nearest Neighbors (KNN) Formulation

- KNN is a non-parametric, instance-based learning algorithm.
- Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$, the goal is to classify a new point x_{new}.
- Compute the distance between x_{new} and all training points using a metric such as:
  - **Euclidean Distance:** $d(x, x') = \sqrt{\sum_{j=1}^{d}(x_j - x'_j)^2}$
  - **Manhattan Distance:** $d(x, x') = \sum_{j=1}^{d} |x_j - x'_j|$
  - **Minkowski Distance:** $d(x, x') = \left(\sum_{j=1}^{d} |x_j - x'_j|^p\right)^{\frac{1}{p}}$
- Select the k closest neighbors and assign the class by majority vote:

$$\hat{y} = \arg \max_{y \in \{0,1\}} \sum_{i \in \mathcal{N}_k} \mathbb{1}(y_i = y)$$

Algorithm for KNN Classification

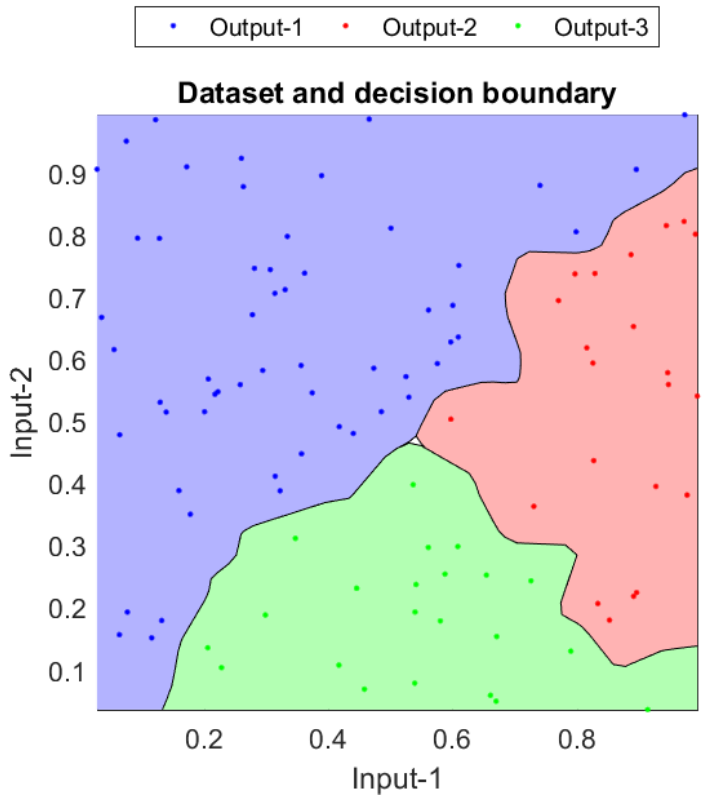
1: **Input:** Training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, query point x_{new}, number of neighbors k
2: **for** each $x_i \in \mathcal{D}$ **do**
3: Compute $d(x_i, x_{\text{new}})$
4: **end for**
5: Sort distances and select k nearest neighbors.
6: Assign the majority class among k neighbors to x_{new}.

Choosing the Right k for KNN

- The choice of k significantly affects KNN's performance.
- **Small k (e.g., $k = 1$)**:
  - Very sensitive to noise.
  - Decision boundary is highly irregular.
- **Large k (e.g., $k = 10$)**:
  - Provides smoother decision boundaries.
  - Reduces sensitivity to noise but may misclassify points near class boundaries.
- Common practice: Use cross-validation to find the optimal k.

Visualizing KNN Decision Boundaries

- Decision boundaries are determined by the class distribution in the neighborhood.
- Boundaries can be highly non-linear depending on k and data distribution.
- Illustration of decision regions for a toy dataset:



Factors Affecting KNN Performance

- **Choice of k:** Small k leads to high variance, large k can oversmooth boundaries.
- **Distance metric:** Euclidean is common but alternatives like cosine similarity may work better.
- **Feature scaling:** Important for distance-based models. Use normalization (e.g., min-max or z-score scaling).

Failure Cases and Considerations

- Sensitive to choice of k: Too small may lead to noise, too large can blur decision boundaries.
- Computationally expensive: Requires storing all training data and computing distances for every query.
- Feature scaling matters: PCA or normalization can improve performance.

Elbow Method

The Elbow Method is a heuristic used to determine the optimal number of neighbors k in K-Nearest Neighbors (KNN)

Why is Choosing k Important?

- Small k (e.g., 1 or 3): Leads to high variance and overfitting, making the model sensitive to noise.
- Large k (e.g., 20 or 50): Leads to high bias and underfitting, as the decision boundary becomes too smooth.

Steps of the Elbow Method

- Train KNN models for different values of k (e.g., from 1 to 50).
- Compute the classification error or distance-based metric (such as mean squared error or misclassification rate).
- Plot the error rate against k to visualize the trend.
- Identify the "elbow point": This is the value of k where the error rate stops decreasing significantly. It represents the best trade-off between bias and variance.

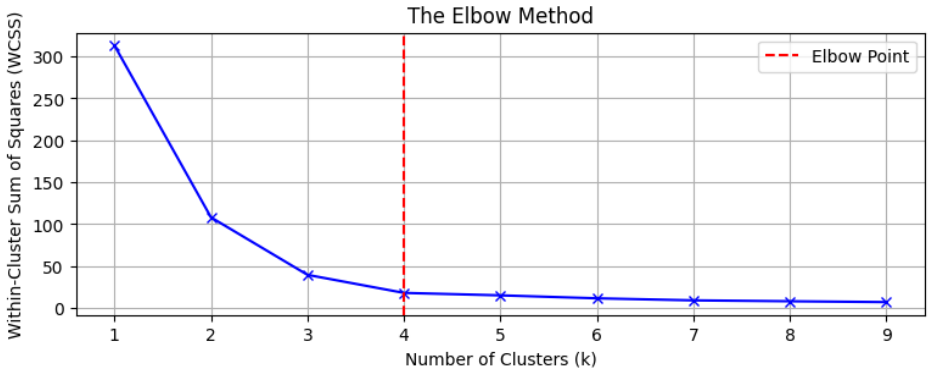


Figure: Elbow Method

Task Example: Classifying Digits 0 and 1 on MNIST

- **Dataset:** MNIST subset with only digits 0 and 1.
- **Feature Representation:** Each image is 28×28, vectorized to $\mathbb{R}^{784}$.
- **Classification Procedure:**
  1. Compute the Euclidean distance between a new digit and all training samples.
  2. Select k nearest samples.
  3. Perform a majority vote among selected neighbors.

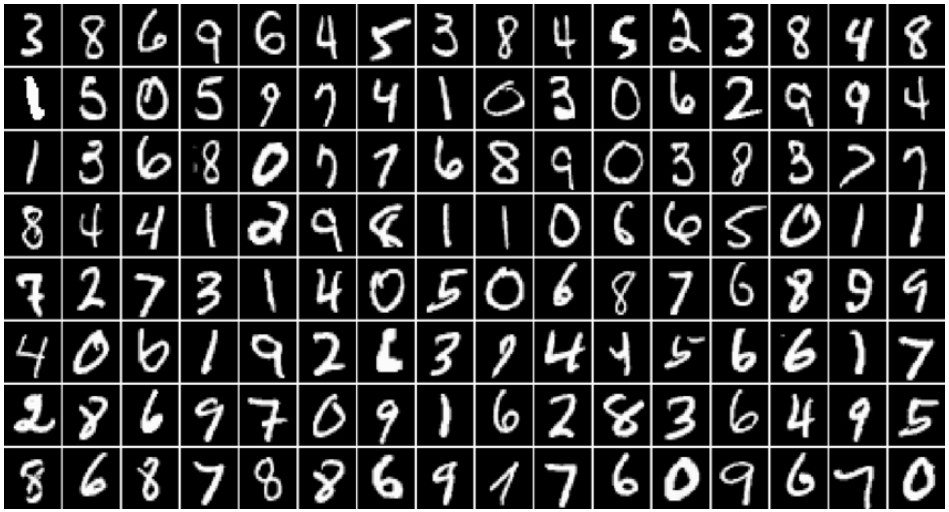


Figure: MNIST Dataset