Revision Notes: K-Nearest Neighbors (KNN) Class

Agenda Overview

In this session, we focused on the K-Nearest Neighbors (KNN) algorithm, a foundational concept in supervised machine learning. The class covered several core aspects of KNN, including its intuition, implementation, and its role in classification tasks. Additionally, we touched on related algorithms and practical applications【4:1†source】【4:9†source】 .

Introduction to KNN

What is KNN?

- K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for classification and regression.
- It classifies a data point based on how its neighbors are classified 【4:10†source】 .

Key Characteristics

- **Non-parametric:** KNN does not assume any underlying data distribution.
- **Instance-based learning:** It makes decisions based on data points in the closest proximity to the query point .
- **Lazy-learning algorithm:** The algorithm makes decisions whenever a query point is to be classified, rather than after an intensive training phase 【4:19†source】 .

How Does KNN Work?

1. **Calculate Distance:**
   - Compute the distance between the query point and all points in the dataset, commonly using Euclidean distance: $Distance =$

【4:16†source】 .

2. **Find K Nearest Neighbors:**
   - Sort all distances and select the top K smallest distances. These are the nearest neighbors【4:19†source】.
3. **Voting Mechanism:**
   - In classification, each of the K neighbors votes for their respective classes, and the class with the maximum votes is assigned to the query point .
   - For regression, the average of the values of its K nearest neighbors is used【4:8†source】.
4. **Assign Class:**
   - The class that gets the maximum votes is assigned to the query point【4:2†source】【4:8†source】.

Key Considerations in KNN

- **Choice of K:**
  - Choosing K is crucial as it impacts model performance. A smaller K is sensitive to noise, while a larger K produces smoother decision boundaries .
  - Typically, K is chosen as an odd number to avoid ties in voting【4:2†source】.
- **Distance Metric:**
  - The choice of distance metric, such as Euclidean or Manhattan, can affect the classification results【4:8†source】.
- **Dimensionality Reduction:**
  - Techniques such as PCA can be used to reduce dimensionality before applying KNN, as high dimensionality can overly complicate calculations【4:18†source】.

Advantages and Limitations

Advantages

- **Simplicity:** Easy to understand and implement.

Limitations

- **Computationally Intensive:**
    - KNN is computationally expensive in terms of both time and memory as it requires storing all training data and recalculating distances for each classification【4:19†source】.
- **Performance on Large Datasets:**
    - Not recommended on large datasets due to higher computational overhead【4:11†source】.
- **Feature Scaling:**
    - KNN is sensitive to the scale of the input data. Therefore, normalization or standardization is necessary .

Practical Tips for KNN

- Always normalize or standardize your data before using KNN to ensure distance calculations are meaningful .
- Consider using KNN on smaller datasets where computational resources are limited【4:11†source】.
- Use odd K values to avoid ties during classification【4:2†source】.

Conclusion

KNN remains a fundamental algorithm in machine learning due to its simplicity and effectiveness in basic applications. Despite its limitations regarding large datasets and high computational cost, it serves well in educational contexts and smaller practical scenarios 【4:18†source】 .