Machine Learning

Video 101:

AdaBoost Hyperparameters | GridSearchCV in Adaboost

Example:

Code link:

https://github.com/campusx-official/100-days-of-machine-learning/blob/main/day66-adaboost/adaboost_demo.ipynb

Video 102:

Bagging Vs Boosting | What is the difference between Bagging and Boosting

Bagging (Bootstrap Aggregating) and Boosting are both ensemble learning techniques in Machine Learning that aim to improve the performance of models by combining multiple weak learners. However, they work in fundamentally different ways.

1. Bagging (Bootstrap Aggregating)

- **Concept:** Multiple models (often the same type, e.g., decision trees) are trained independently on different random subsets of the dataset.
- **How it Works:**
    - The training data is sampled **with replacement** (bootstrap sampling).
    - Each model is trained independently on a different subset of the data.
    - The predictions of all models are aggregated, usually by **majority voting** (for classification) or **averaging** (for regression).
- **Goal:** Reduce variance and prevent overfitting by averaging out errors.
- **Example Algorithm: Random Forest** (combination of decision trees using bagging).

2. Boosting

- **Concept:** Models are trained sequentially, where each new model corrects the mistakes of the previous ones.
- **How it Works:**
    - The first model is trained on the original dataset.
    - The second model is trained by giving **more weight** to the misclassified points from the first model.
    - This process continues, with each new model focusing on the hardest-to-classify points.
    - The final prediction is a weighted combination of all models.

- **Goal:** Reduce bias by creating a strong learner from multiple weak learners.
- **Example Algorithms: AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost.**

Key Differences:

Feature	Bagging	Boosting
Training Process	Parallel (independent models)	Sequential (each model depends on the previous one)
Focus	Reducing variance (prevents overfitting)	Reducing bias (improves weak models)
Data Sampling	Random sampling with replacement	Weighted sampling (focus on hard examples)
Combination Method	Majority voting (classification) / Averaging (regression)	Weighted sum of weak learners
Common Algorithms	Random Forest	AdaBoost, Gradient Boosting, XGBoost

When to Use What?

- **Bagging (e.g., Random Forest):** When you want to reduce overfitting and variance in high-variance models like decision trees.
- **Boosting (e.g., XGBoost, AdaBoost):** When you need to improve model accuracy and reduce bias, especially in weak models.

Would you like a more practical example with code? 🚀

Video 103:

K-Means Clustering Algorithm | Geometric Intuition | Clustering | Unsupervised Learning

What is Clustering?

Clustering is an **unsupervised learning** technique that groups similar data points into clusters based on patterns or similarities. It helps in discovering hidden structures in data without predefined labels. Common algorithms include **K-Means, Hierarchical Clustering, and DBSCAN**, used in applications like customer segmentation, anomaly detection, and image recognition.

What is K-Means clustering?

K-Means is an **unsupervised learning** algorithm used for **clustering** data into **K** groups based on feature similarity.

How It Works:

1. **Choose K** (the number of clusters).
2. **Initialize K centroids** randomly.
3. **Assign each data point** to the nearest centroid.
4. **Recalculate centroids** as the average of assigned points.
5. **Repeat steps 3-4** until centroids no longer change.

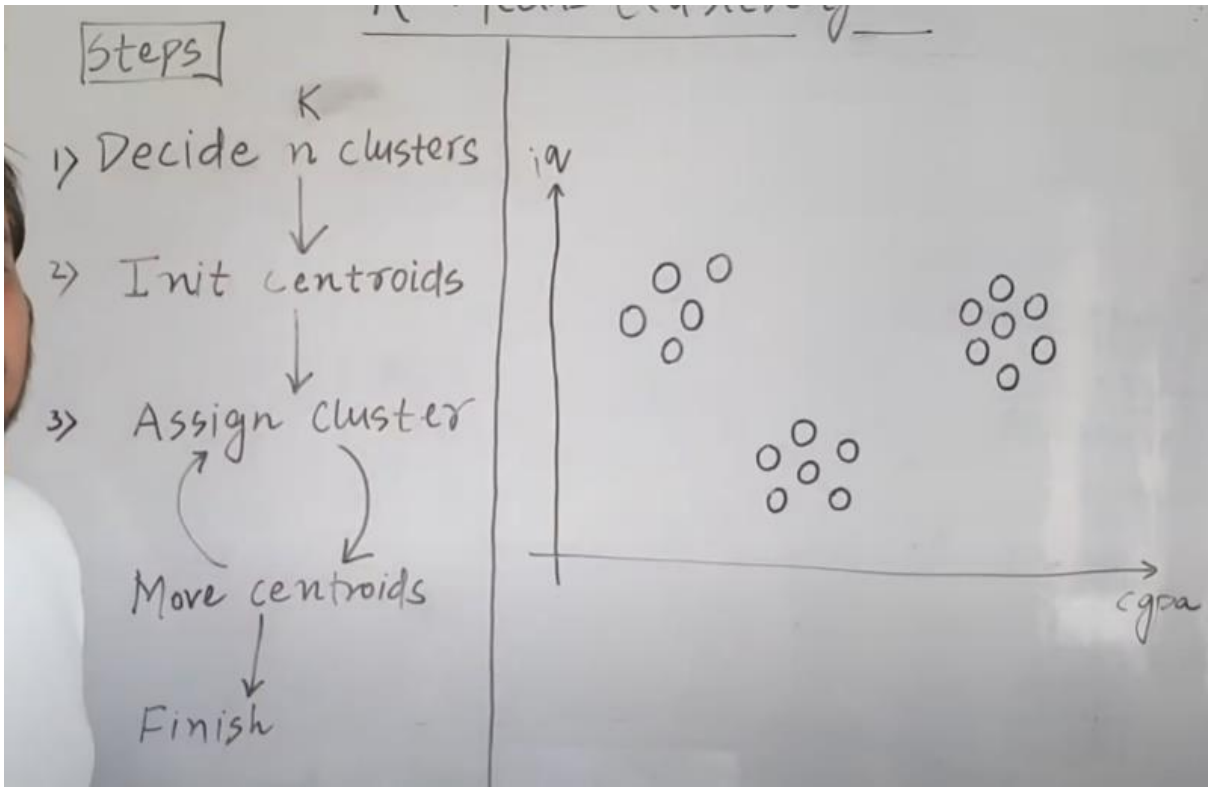
Key Points:

- Uses **Euclidean distance** to measure similarity.
- Sensitive to the choice of **K** and initial centroids.
- Works well for **spherical** clusters but struggles with non-linear shapes.

Applications:

- Customer segmentation
- Image compression
- Anomaly detection

Use **K-Means clustering** when you need to **group similar data points** without predefined labels. It's best for **large datasets** with **well-separated, spherical clusters**. Ideal applications include **customer segmentation, anomaly detection, image compression, and pattern recognition**. Avoid it for **non-linear, imbalanced, or overlapping clusters**, as it assumes equal cluster sizes.



How to decide the number of clusters?

How to Decide the Number of Clusters in K-Means?

One of the most common techniques is the **Elbow Method**.

Elbow Method:

1. **Run K-Means** with different values of KK (e.g., 1 to 10).
2. **Calculate the Within-Cluster Sum of Squares (WCSS)** for each KK.
    - WCSS measures how close data points are to their assigned cluster centroids.
3. **Plot KK vs. WCSS** (Elbow Curve).
4. **Look for the "elbow point"** where WCSS stops decreasing significantly.
    - This point suggests the optimal number of clusters.

Video 104:

K-Means Clustering Algorithm in Python | Practical Example | Student Clustering Example | sklearn

Example:

Code link:

https://colab.research.google.com/drive/1GQakj73nEfubpfTEK6kPYvlxItIpiZNf?usp=sharing

Video 105:

K-Means Clustering Algorithm From Scratch In Python | ML Algorithms From Scratch

Example:

Code link:

https://github.com/campusx-official/100-days-of-machine-learning/blob/main/kmeans/kmeans.py