Honors ML & DS Unit-1 Notes by OMG 13_159 OmGonjari

1. Support Vectors and Their Role in SVM

Support vectors are the data points that lie closest to the decision boundary, which is known as the hyperplane. These are the critical data points in a dataset that help in defining the optimal hyperplane.

The **role of these support vectors** in the Support Vector Machine (SVM) classification algorithm is fundamental. SVM aims to find an optimal hyperplane that best separates the different classes in the data. The "optimality" of the hyperplane is defined by the **margin**, which is the distance between the hyperplane and the nearest data point from either class. The support vectors are these nearest data points.

The algorithm's objective is to **maximize this margin**. The position and orientation of the optimal hyperplane are determined entirely by the support vectors. If any data point that is not a support vector is moved or removed, it will not affect the hyperplane. However, if a support vector is moved, the hyperplane will also change its position to maintain the maximum margin. Therefore, support vectors are the most influential instances for the SVM classifier.

---

2. Kernel PCA

Kernel Principal Component Analysis (Kernel PCA) is a non-linear dimensionality reduction technique. It is an extension of the standard Principal Component Analysis (PCA). While standard PCA is effective at reducing dimensions for linearly separable data, it performs poorly when the data has a complex, non-linear structure.

Kernel PCA addresses this limitation by using the **kernel trick**. The main idea is to project the data, which is not linearly separable in its original, lower-dimensional space, into a higher-dimensional space where it becomes linearly separable. This transformation is done implicitly by a **kernel function**. Instead of actually performing the complex transformation, the kernel function computes the dot products of the data points in the higher-dimensional feature space.

Once the data is in this higher-dimensional space, standard PCA is applied to identify the principal components that capture the maximum variance. This allows Kernel PCA to find non-linear patterns and relationships in the data that would be missed by linear PCA.

Common kernel functions include the Polynomial kernel, the Radial Basis Function (RBF) kernel, and the Sigmoid kernel.

---

3. K-Nearest Neighbour (KNN) Learning Algorithm

The **K-Nearest Neighbour (KNN)** algorithm is a simple, supervised machine learning algorithm that can be used for both classification and regression tasks. It is a non-parametric and instance-based learning algorithm, which means it doesn't make any assumptions about the underlying data distribution and memorizes the entire training dataset.

The working of the KNN algorithm can be summarized in these steps:

1. **Choose the number of neighbors (K)**: This is a user-defined constant.
2. **Calculate the distance**: For a new, unseen data point, calculate the distance between it and all the data points in the training set. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance.
3. **Find the K-nearest neighbors**: Identify the 'K' training data points that are closest to the new data point based on the calculated distances.
4. **Make a prediction**:
    - For **classification**, the algorithm assigns the new data point to the class that is most common among its K-nearest neighbors (majority voting).
    - For **regression**, the algorithm predicts the value for the new data point by averaging the values of its K-nearest neighbors.

Example:
Imagine we want to classify a new fruit as either an 'Apple' or an 'Orange' based on its 'Sweetness' and 'Crunchiness' ratings. Our training data has several labeled fruits. Let's say we choose K=3.
A new fruit has a 'Sweetness' of 6 and a 'Crunchiness' of 4. The KNN algorithm calculates the distance from this new fruit to all the fruits in the training data. It finds that the three closest fruits (the 3-nearest neighbors) are: an Apple, an Orange, and an Apple.

Since two out of the three nearest neighbors are Apples, the KNN algorithm will classify the new fruit as an **Apple**.

---

4. Finding the Optimal Separating Hyperplane in SVM

In the Support Vector Machine (SVM) algorithm, the **optimal separating hyperplane** is the

one that correctly separates the data points of different classes while maximizing the margin between them. The margin is defined as the distance between the hyperplane and the closest data points from either class. These closest points are the support vectors.

The process to find this optimal hyperplane involves solving a constrained optimization problem. The goal is to:

1. **Maximize the Margin**: The margin is given by 2/ // w // , where w is the weight vector normal to the hyperplane. Maximizing the margin is equivalent to minimizing // w // , or more conveniently, minimizing 21 // w // 2.
2. **Subject to the Constraint**: Every data point must be on the correct side of the margin. This is expressed mathematically as yi(w·xi−b)≥1 for all data points (xi,yi), where yi is the class label (+1 or -1).

Example:
Consider a 2D dataset with two classes, circles and squares, that are linearly separable. There could be many possible straight lines (hyperplanes) that can separate these two classes.

- A line that passes very close to the circles would be a poor separator because it might misclassify new circles.
- Similarly, a line that is too close to the squares would also be a bad choice.

The **optimal hyperplane** would be the line that is farthest from both the closest circle and the closest square. This distance from the hyperplane to the nearest data point on either side is the margin. The SVM algorithm finds the specific line where this margin is as wide as possible. The data points that lie on the edges of this margin are the support vectors, and they are the ones that define the optimal hyperplane.

---

5. Different Functions of a Kernel

In the context of machine learning, particularly SVMs, a **kernel** is a function that takes two data points as input and computes a similarity score between them. The primary function of kernels is to enable learning in high-dimensional feature spaces without explicitly computing the coordinates of the data in that space. This is known as the **kernel trick**.

The different functions of kernels include:

- **Linear Kernel**: This is the simplest kernel function. It computes the dot product of the input data points and is used when the data is linearly separable.
  - K(x,x′)=xTx′
- **Polynomial Kernel**: This kernel is used to model non-linear relationships. It can find curved decision boundaries in the input space.
  - K(x,x′)=(γxTx′+r)d

- **Radial Basis Function (RBF) Kernel**: This is a very popular and powerful kernel that can handle complex, non-linear relationships. It assumes that the influence of a data point is localized and diminishes with distance.
    - $K(x,x') = \exp(-\gamma \,//\, x - x' \,//\, 2)$
- **Sigmoid Kernel**: This kernel is often used in neural networks and is suitable for certain types of data.
    - $K(x,x') = \tanh(\gamma x^T x' + r)$

In essence, these kernel functions allow SVM and other algorithms to operate in a high-dimensional feature space, enabling them to find complex patterns and classify data that is not separable in its original space.

---

6. Ensemble Learning, Bagging, and Bootstrap

Ensemble Learning is a machine learning technique where multiple individual models, often called "weak learners," are strategically combined to produce a single, more powerful model, or "strong learner." The main idea is that a committee of models can often make better predictions than any single model alone. This approach helps to improve the accuracy, reduce variance, and make the model more robust.

Bootstrap is a statistical resampling method. It involves creating multiple subsets of data from the original training dataset by sampling with replacement. This means that each time a data point is selected for a subset, it is put back into the original dataset, so it can be selected again. Each of these new subsets is the same size as the original dataset.

Bagging, which stands for **Bootstrap Aggregating**, is an ensemble method that uses the bootstrap technique. The process is as follows:

1. **Bootstrap Sampling**: Multiple bootstrap samples are created from the original training data.
2. **Model Training**: A separate model (e.g., a decision tree) is trained independently on each of these bootstrap samples.
3. **Aggregation**: For a new prediction, the predictions from all the individual models are combined.
    - In a **classification** task, the final prediction is determined by a majority vote among the models.
    - In a **regression** task, the final prediction is the average of the predictions from all the models.

Bagging is particularly effective at reducing the variance of a model, helping to prevent

overfitting. The Random Forest algorithm is a well-known example of the bagging technique.

---

7. Features and Working of Random Forest Algorithm

The **Random Forest Algorithm** is a versatile and powerful ensemble learning method used for both classification and regression.

Key Features of Random Forest:

- **Ensemble of Decision Trees**: It is fundamentally a collection of multiple decision trees.
- **Bagging**: It uses the bagging technique, where each tree is trained on a different bootstrap sample of the training data.
- **Feature Randomness**: When splitting a node in a decision tree, the algorithm considers only a random subset of the features, rather than all of them. This decorrelates the trees and makes the overall model more robust.
- **High Accuracy and Robustness**: By combining many trees, it generally achieves higher accuracy than a single decision tree and is less prone to overfitting.
- **Handles Missing Values**: It has methods for handling missing data.
- **Provides Feature Importance**: It can estimate the importance of each feature in making predictions, which is useful for feature selection.

Working of Random Forest Algorithm with Example:
Let's say we want to predict whether a customer will buy a product based on their age, income, and browsing history.

1. **Bootstrap Sampling**: The algorithm creates multiple random samples (with replacement) from the customer dataset. Let's say it creates 100 different datasets.
2. **Building Trees**: A decision tree is built for each of these 100 datasets. For each split in each tree, only a random subset of the features (e.g., just 'age' and 'income', or 'income' and 'browsing history') is considered to find the best split.
3. **Making a Prediction**: Now, a new customer comes in. To predict if they will buy the product, their data is run through all 100 trees.
4. **Voting**: Each tree makes a prediction ('buy' or 'not buy'). Let's say 70 trees predict 'buy' and 30 trees predict 'not buy'.
5. **Final Decision**: The final prediction is taken by a majority vote. In this case, since the majority of trees voted 'buy', the Random Forest algorithm predicts that the new customer will buy the product.

---

8. Working of Support Vector Machine (SVM) Classification Algorithm

The **Support Vector Machine (SVM)** is a supervised machine learning algorithm used for classification and regression tasks. In classification, its primary goal is to find the best possible decision boundary that separates data points of different classes.

The working of the SVM classification algorithm can be explained as follows:

1. **Plotting Data in n-Dimensional Space**: The algorithm begins by representing the data points as points in an n-dimensional space, where 'n' is the number of features.
2. **Finding the Hyperplane**: The core of SVM is to find a **hyperplane** that best separates the data points into their respective classes. A hyperplane is a decision boundary. In a 2D space, it's a line; in a 3D space, it's a plane, and so on.
3. **Maximizing the Margin**: For a given dataset, there might be many hyperplanes that can separate the classes. SVM seeks the **optimal hyperplane**, which is the one with the maximum **margin**. The margin is the distance between the hyperplane and the nearest data point from each class. A larger margin leads to a more confident and robust classification.
4. **Identifying Support Vectors**: The data points that are closest to the optimal hyperplane and lie on the margin are called **support vectors**. These are the critical points that "support" or define the hyperplane.
5. **Handling Non-Linear Data with the Kernel Trick**: If the data is not linearly separable, SVM can use the **kernel trick**. It projects the data into a higher-dimensional space where it can be separated by a hyperplane. This is done using kernel functions (like RBF, Polynomial, etc.) without the need to explicitly compute the new dimensions, which is computationally efficient.

In essence, SVM is about finding the "best street" to separate neighborhoods of data, where the "street" (margin) is as wide as possible.

---

9. Feature Selection in Random Forest and Its Difference in Classification vs. Regression

Random Forest selects features at each node split and also provides a measure of **feature importance**, which can be used for feature selection.

How Random Forest Selects Features during Tree Building:
At each node of each decision tree, instead of considering all the available features to find the best split, Random Forest selects a random subset of features. The best split is then determined from this smaller, random set. This process introduces randomness and diversity among the trees, which is key to the algorithm's performance and helps to prevent overfitting.

How Random Forest Calculates Feature Importance:
After the forest is trained, it can calculate the importance of each feature. A common method is Mean Decrease in Impurity. For each feature, the algorithm measures how much the Gini impurity (for classification) or the Mean Squared Error (for regression) decreases on average across all the trees in the forest whenever that feature is used to split a node. A higher mean decrease indicates a more important feature.
Another method is **Permutation Importance**. The importance of a feature is determined by randomly shuffling the values of that feature in the validation dataset and measuring the resulting decrease in the model's accuracy or increase in its error. A larger drop in performance means the feature is more important.

Difference for Classification and Regression:
The fundamental mechanism of feature selection and importance calculation in Random Forest is similar for both classification and regression, but the metric used to evaluate the splits and importance differs.

- **For Classification**: The algorithm typically uses metrics like **Gini Impurity** or **Information Gain** to determine the quality of a split. When calculating feature importance, it measures the mean decrease in Gini impurity.
- **For Regression**: The algorithm uses metrics like **Mean Squared Error (MSE)** or **Variance Reduction** to find the best split. For feature importance, it calculates the mean decrease in MSE.

So, while the process of randomly selecting features for splits and then aggregating importance scores is the same, the underlying criteria for what makes a "good" split or an "important" feature are tailored to the specific task—minimizing classification error or minimizing prediction error.