Machine Learning Notes

What Is Machine Learning?

Machine Learning is a way for computers to learn from data without being explicitly programmed. Instead of following a set of instructions, the computer uses algorithms to find patterns in data and make decisions based on those patterns. It's like teaching a computer to learn from experience, similar to how humans learn.

Why Machine Learning?

Machine Learning is important because it allows computers to solve problems and make decisions on their own. It can help us make better predictions, automate tasks, and find insights in data that we might miss. For example, it can help doctors diagnose diseases, recommend movies you might like, or even drive cars!

Supervised Learning

Supervised Learning is a type of machine learning where the computer is trained with labelled data. This means the data has input-output pairs, and the computer learns to map the inputs to the correct outputs. It's called "supervised" because the learning process is guided by the known outputs, like a teacher supervising a student.

Regression

Regression is a technique in machine learning used to predict a continuous outcome variable based on one or more input variables. It helps in finding the relationship between the dependent variable and independent variables. Regression is commonly used in scenarios

where you want to predict quantities, like sales numbers or temperatures.

Linear Regression

Linear Regression is a simple method used to predict a continuous outcome variable based on the linear relationship between the input variables and the output. It fits a straight line ($y = mx + c$) to the data points, minimising the distance between the data points and the line. It's useful when the relationship between variables is approximately linear.

Backend Explanation of Linear Regression:

Imagine you have a scatter plot of points. Linear regression finds the best straight line that goes through these points. This line helps us make predictions. For example, if you know the number of hours you study and your scores, you can predict the score if you study for a new number of hours.

Polynomial Regression

Polynomial Regression is similar to linear regression but fits a polynomial equation ($y = ax^2 + bx + c$) to the data. This allows for a curve instead of a straight line, which can better fit data with a more complex relationship. It's useful when the data shows a curved trend rather than a straight-line relationship.

Backend Explanation of Polynomial Regression:

Imagine you're drawing a curve that fits through a set of points. Polynomial regression allows us to draw this curve, which can bend and twist to match the data points better than a straight line. This is helpful when the relationship between the data points isn't straight.

Decision Tree Regressor

A Decision Tree Regressor is a model that uses a tree-like structure to make predictions. The data is split into branches based on certain conditions, and each branch represents a decision made by the model. It's useful for capturing non-linear relationships and interactions between variables, making it easier to interpret.

Backend Explanation of Decision Tree Regressor:

Imagine a tree where each branch represents a question, and each leaf represents an answer. The decision tree asks a series of questions to split the data until it gets the final answer. It's like a game of 20 Questions, where each question helps narrow down the possibilities.

K Nearest Neighbour Regression

K Nearest Neighbour (KNN) Regressor predicts the value of a new data point based on the average of the k-nearest data points in the training set. It's like finding the closest neighbours to the new data point and averaging their values. It's simple and effective for many types of data but can be slow with large datasets.

Backend Explanation of K Nearest Neighbour Regression:

Imagine you want to know how much a toy costs. You ask your k nearest friends how much they paid for the same toy and then take the average of their answers. This gives you a good estimate of the toy's price.

Support Vector Regressor

Support Vector Regressor (SVR) is a type of machine learning model that uses the concept of support vectors to find the best-fit line or curve for the data. It aims to fit the data within a margin of error, balancing the model's complexity and the fit. SVR is powerful for capturing complex relationships in data.

Backend Explanation of Support Vector Regressor:

Imagine drawing lines or curves that touch the edges of the data points, creating a margin. The SVR finds the best line or curve that fits within this margin, trying to be as close to the data points as possible while staying within the allowed space. This helps it make accurate predictions.

When To Go For Regression

You should go for regression when you need to predict a continuous outcome variable based on one or more input variables. If your goal is to forecast quantities, like predicting future sales, temperatures, or prices, regression is the right choice. It's also useful for understanding relationships and trends in data.

Classification

Classification is a type of machine learning where the computer learns to categorise data into different classes or groups. Instead of predicting a continuous value, classification predicts discrete labels, like whether an email is spam or not spam. It helps in sorting data into categories based on patterns learned from past examples.

Logistic Regression

Logistic Regression is a method used in classification to predict the probability of a data point belonging to a certain class. It's used when the outcome is categorical, like yes/no or true/false. It works by fitting a logistic function (an S-shaped curve) to the data, which helps in making predictions about the probability of the different classes.

Backend Explanation of Logistic Regressor:

Imagine you're deciding whether a fruit is an apple or an orange based on its colour and size. Logistic regression finds a line or curve that separates apples from oranges in a way that makes it easy to predict which category a new fruit belongs to based on its features.

Decision Tree Classifier

A Decision Tree Classifier is a model that uses a tree-like structure to make decisions. The data is split into branches based on certain conditions, and each branch represents a decision made by the model. It’s useful for classifying data into categories and is easy to understand because it shows a clear path of decisions.

Backend Explanation of Decision Tree Classifier:

Imagine a tree where each branch represents a question (like "Is the fruit red?"), and each leaf represents a final category (like "Apple" or "Cherry"). The decision tree asks questions about the data, following the branches until it reaches a decision.

K Nearest Neighbour Classifier

K Nearest Neighbour (KNN) Classifier determines the class of a new data point based on the majority class of its k-nearest neighbors in the training set. It’s like looking at the closest examples in the data and seeing which class they belong to. The class that appears most frequently among the neighbours is assigned to the new data point.

Backend Explanation of K Nearest Neighbour Classifier:

Imagine you want to classify a new fruit based on the types of fruits that are closest to it. You look at the k closest fruits and see what types they are. If most of them are apples, you classify the new fruit as an apple too.

Support Vector Classifier

Support Vector Classifier (SVC) is a model that finds the best boundary (line or hyperplane) that separates different classes of data. It aims to create a margin between the classes, making sure that the boundary is as far away as possible from the nearest data points of each class. This helps in making accurate and robust classifications.

Backend Explanation of Support Vector Classifier:

Imagine drawing a line between two groups of points on a graph. The SVC tries to draw this line in such a way that it is as far away as possible from the closest points of each group. This helps the model classify new points accurately by ensuring a clear separation between the classes.

Naive Bayes Classifier

Naive Bayes Classifier is a simple probabilistic model based on Bayes' theorem. It assumes that the features used to classify a data point are independent of each other, which makes it "naive." Despite this simple assumption, it works well for many types of classification problems, especially with text data.

Backend Explanation of Naive Bayes Classifier:

Imagine you want to guess whether an email is spam based on words it contains. The Naive Bayes Classifier looks at each word independently and calculates the probability of the email being spam or not spam based on the words it has. It then makes a classification based on these probabilities.

When To Go For Classification

You should go for classification when you need to assign categories or labels to data points. If your goal is to sort items into distinct groups, like identifying whether an email is spam or predicting the type of animal in a picture, classification is the right choice. It helps in making decisions where outcomes are discrete categories.

Boosters

Boosters are a type of machine learning technique that improves the performance of models by combining the predictions of several weaker models to make a stronger prediction. The idea is to build a series of models where each new model tries to correct the mistakes made by the previous ones. This approach helps in making more accurate predictions.

Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. Each decision tree is trained on a different subset of the data, and their predictions are averaged or voted on to get the final result. This helps in improving accuracy and reducing overfitting compared to using a single decision tree.

Random Forest Classifier

A Random Forest Classifier is used for classification tasks where it predicts the category or class of a data point. By combining the outputs of multiple decision trees, it makes a final decision based on the majority vote from all trees. It's effective in handling large datasets and complex relationships between features.

Backend Explanation of Random Forest Classifier:

Imagine having several decision trees, each making a prediction about whether an email is spam or not. Each tree might give a different answer, but by taking a vote among all trees, the Random Forest Classifier makes a final decision that's more reliable and accurate.

Random Forest Regressor

A Random Forest Regressor is used for regression tasks where it predicts a continuous value. Similar to the classifier, it uses multiple decision trees, but instead of voting, it averages the predictions from all trees to get the final result. This helps in making more precise predictions and handling various types of data.

Backend Explanation of Random Forest Regressor:

Imagine having several decision trees, each predicting the price of a house. Each tree might give a slightly different price, but by averaging all the predictions, the Random Forest Regressor provides a more accurate estimate of the house price.

AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble technique that combines multiple weak models to create a strong model. It works by giving more weight to the data points that were misclassified by previous models, making sure the next model focuses more on these difficult cases. This helps in improving the overall performance of the model.

AdaBoost Classifier

An AdaBoost Classifier is used for classification tasks. It combines several weak classifiers to form a strong classifier by focusing on the errors made by previous classifiers. This approach helps in achieving high accuracy even with simpler base models.

Backend Explanation of AdaBoost Classifier:

Imagine you have several simple classifiers, each trying to categorise emails as spam or not spam. AdaBoost trains these classifiers one after the other, giving more attention to the emails that were misclassified earlier. By combining their results, it makes a stronger and more accurate classifier.

AdaBoostRegressor

An AdaBoost Regressor is used for regression tasks. It combines multiple weak regression models to form a strong regressor by focusing on the errors made by previous regressors. This helps in improving the accuracy of predictions for continuous outcomes.

Backend Explanation of AdaBoost Regressor:

Imagine you have several simple regression models predicting house prices. AdaBoost trains these models in sequence, giving more importance to the houses where the previous models made mistakes. By averaging their predictions, it provides a more accurate estimate of house prices.

When To Go For Boosting

You should consider using boosting when you need to enhance the performance of your machine learning model. Here's when boosting is a good choice:

1. **Improving Accuracy**: When you need to achieve high accuracy and your current model isn't performing well enough, boosting can help. By combining the strengths of multiple weak models, boosting can make a more accurate and robust model.
2. **Fixing Model Issues**: If your initial model is underfitting (not capturing enough detail) or if it's giving poor results, boosting can help correct these issues. It does this by focusing on the mistakes made by the previous models and improving performance.
3. **Complex Data**: For datasets where relationships between features and outcomes are complex and not easily captured by a single model, boosting can help. It builds multiple models in sequence to handle these complexities better.
4. **Reducing Overfitting**: Boosting can help reduce overfitting, where a model performs well on training data but poorly on new data. By focusing on correcting errors, boosting helps the model generalize better to new, unseen data.
5. **Combining Weak Models**: When you have several simple models (weak learners) that are not very powerful on their own, boosting can combine them into a strong model. This approach takes advantage of each model's strengths to create a more effective overall model.

In summary, boosting is useful when you need to improve accuracy, fix issues with existing models, handle complex patterns in data, reduce overfitting, or combine multiple weaker models into a stronger one.

Unsupervised Learning

Unsupervised Learning is a type of machine learning where the computer tries to find patterns or groupings in data without any pre-labeled outcomes. Unlike supervised learning, where we provide the correct answers, unsupervised learning lets the computer discover hidden structures or groupings on its own. It's like exploring data without knowing what to expect.

What Is Cluster

A cluster is a group of data points that are similar to each other based on certain features. In clustering, the goal is to group similar data points together so that data within the same cluster is more alike than data in different clusters. It's like sorting different fruits into baskets based on their types, so each basket contains similar fruits.

K-Means Cluster

K-Means Clustering is a popular method used to group data points into a specified number of clusters. It works by first guessing where the centre of each cluster should be (called centroids), then assigning each data point to the closest centroid. After all points are assigned, it updates the centroids and repeats the process until the clusters don't change much.

Backend Explanation of K-Means Clustering:

Imagine you have a bunch of different coloured balls and you want to group them into 3 baskets based on colour. K-Means starts by randomly placing the baskets, then moves them around to better match the colour of the balls. It keeps adjusting the basket positions until all balls are in the basket that best matches their colour.

What Is Dimensionality Reduction ?

Dimensionality Reduction is a technique used to reduce the number of features or dimensions in data while preserving important information. It helps in simplifying data and making it easier to visualise or analyse. It's like summarising a long book into a shorter version while keeping the main story.

PCA (Principal Component Analysis)

Principal Component Analysis (PCA) is a method used for dimensionality reduction. It works by finding the most important directions (principal components) in the data that capture the most variation. By projecting the data onto these directions, PCA reduces the number of features while retaining the essential information.

Backend Explanation of PCA:

Imagine you have a very detailed map with lots of roads and landmarks. PCA helps you create a simpler map that still shows the most important features, like major roads and landmarks, while removing less important details. This makes it easier to understand and work with the data.