Types of Algorithms in AI

Saksham Nandanwar
Roll No. 22111048

July 2024

1 Linear Regression

It is used for predicting a continuous target variable based on one or more variables.

1.1 Basic Concept

Linear Regression aims to fit a straight line through a set of data points, ensuring that the total of squared differences between the actual values and the values predicted by the line is minimized.

1.2 Applications

1. Biology: To predict biological responses.
2. Engineering: To understand relationship between variables in system.

1.3 Limitations

1. Results may be wrong if assumptions are ignored.

1.4

Code
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear$_m odelimportLinearRegression$

$\#Generatesomedata$
$X = np.array([[1], [2], [3], [4], [5]])$
$y = np.array([1, 3, 2, 5, 4])$

```
    # Create a linear regression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Make predictions
y_pred = model.predict(X)

#Plottheresults
plt.scatter(X, y, color =' blue')
plt.plot(X, y_pred, color =' red')
plt.xlabel('X')
plt.ylabel('y')
plt.title('LinearRegression')
plt.show()
```

2 Logistic Regression

2.1 Basic Concept

Logistic Regression is a technique used for binary classification tasks, aiming to estimate the probability that a particular input falls into one of two categories.

2.2 Application

1. Medical Diagnosis: Predicting the presence or absence of a disease.
2. Stock market.

2.3 Limitation

1. Not Suitable for Non-linear Problems
2. Limited to Binary Classification

2.4 Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_modelimportLogisticRegression

#Generatesomedata
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([0, 0, 1, 1, 1])
```

$$
\begin{aligned}
&\#Createalogisticregressionmodelandfititthedata \\
&model = LogisticRegression() \\
&model.fit(X, y)
\end{aligned}
$$

$$
\begin{aligned}
&\#Makepredictions \\
&y_prob = model.predict_proba(X)[:, 1]
\end{aligned}
$$

$$
\begin{aligned}
&\#Plottheresults \\
&plt.scatter(X, y, color =' blue') \\
&plt.plot(X, y_prob, color =' red') \\
&plt.xlabel('X') \\
&plt.ylabel('Probability') \\
&plt.title('LogisticRegression') \\
&plt.show()
\end{aligned}
$$

3 Decision Tree

A Decision Tree is a widely used machine learning algorithm for classification and regression tasks. It represents decisions and their potential outcomes through a tree-like structure, offering a clear and interpretable visualization of the decision-making process.

3.1 Advantages

1. They can capture non-linear relationships between features and the target variable without requiring feature transformations.
2. They can handle both numerical and categorical data.

3.2 Disadvantages

1.Instability: Small changes in the data can result in changes in the structure.

3.3 Applications

1. Healthcare: Diagnosing diseases.

3.4 Code

import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, $plot_tree$

$\#Generatesomedata$
$X = np.array([[6], [7], [8], [9], [10]])$
$y = np.array([0, 0, 1, 1, 1])$

$\#Createadecisiontreemodelandfititthedata$
$model = DecisionTreeClassifier()$
$model.fit(X, y)$

$\#Plotthetree$
$plt.figure(figsize = (10, 6))$
$plot_tree(model, filled = True, feature_names = ['X'], class_names = ['0','1'])$
$plt.title('DecisionTree')$
$plt.show()$

4 Support Vector Machine (SVM)

Support Vector Machine (SVM) is an effective supervised learning technique employed for classification and regression problems. The main objective of SVM is to identify the optimal hyperplane that maximally differentiates between various classes in the feature space.

4.1 Types of SVM

1. Linear SVM:
2. Non-Linear SVM

4.2 Advantages

1. Effective in High-Dimensional Spaces
2. Versatile

4.3 Disadvantages

1. Computational Complexity

4.4 Applications

1. Image Classification, Object recognition, facial recognition.

4.5 Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC

# Generate some data
X = np.array([[1, 1], [2, 2], [3, 3], [4, 4], [1, 2], [2, 3], [3, 4], [4, 5]])
y = np.array([0, 0, 0, 0, 1, 1, 1, 1])

# Create a SVM model and fit it to the data
model = SVC(kernel='linear')
model.fit(X, y)

# Plot the decision boundary
w = model.coef_[0]
b = model.intercept_[0]
x_plot = np.linspace(0, 5)
y_plot = -(w[0]/w[1]) * x_plot - b/w[1]

plt.scatter(X[:, 0], X[:, 1], c = y)
plt.plot(x_plot, y_plot, color =' red')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.title('SupportVectorMachine')
plt.show()
```

5 K-Means Clustering

K-Means Clustering is a widely used unsupervised learning technique designed to divide a dataset into distinct groups or clusters by evaluating the similarities between features.

5.1 Advantages

1. Easy to understand and implement.
2. Efficient for large datasets.
3. Can be applied for variety of applications like image compression.

5.2 Disadvantages

1. The outcome can be affected by the initial placement of centroid.
2. Sensitive to Outliers.

5.3 Applications

1. Image compression by reducing the no of colors.
2. Grouping genes or proteins with similar functions.

5.4 Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Generate some data
X = np.array([[1, 2], [2, 1], [3, 4], [4, 3], [5, 5], [6, 5]])

# Create a K-means model and fit it to the data
kmeans = KMeans(n_clusters = 2)
kmeans.fit(X)

#Plottheclusters
plt.scatter(X[:, 0], X[:, 1], c = kmeans.labels_, cmap =' viridis')
plt.scatter(kmeans.cluster_centers_[ :, 0], kmeans.cluster_centers_[ :, 1], s = 300, c ='
red')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.title('K - MeansClustering')
plt.show()
```