Collaborative Learning

Academic Year 2021-2022 Even Semester

Degree, Semester & Branch: VI Semester B.TECH.IT

Course Code & Title: JIT1007 Machine Learning Techniques

Name of the Faculty member (s): MS.Aishwarya S

Innovative Practice Description

- **Unit / Topic: Unit II /** Maximum margin classification
- **Course Outcome:** CO2
- **Topic Learning Outcome:** TLO2
- **Activity Chosen:** Design-Thinking Process
- **Justification:**

Maximum margin classification is a principle used in machine learning, particularly in support vector machines (SVMs), to achieve the best possible separation between different classes in a dataset. The goal is to find a decision boundary (hyperplane) that maximizes the margin, which is the distance between the closest points of the classes to the hyperplane. Here's a more detailed explanation:

Concept

1. **Hyperplane**: In an n-dimensional space, a hyperplane is a flat affine subspace of one dimension less than the space itself. For example, in a 2D space, a hyperplane is a line, and in a 3D space, it's a plane.
2. **Margin**: The margin is defined as the distance between the hyperplane and the nearest data points from any class. These nearest data points are called support vectors.
3. **Maximum Margin**: The maximum margin classifier aims to maximize this margin. By doing so, it increases the classifier's robustness to errors and improves its generalization to unseen data.

Mathematical Formulation

Given a training dataset (x1,y1),(x2,y2),…,(xn,yn)(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)(x1,y1),(x2,y2),…,(xn,yn) where xix_ixi represents the feature vectors and yiy_iyi represents the class labels (yi∈{1,−1}y_i \in \{1, -1\}yi∈{1,−1}), the goal is to find a hyperplane defined by w·x+b=0w \cdot x + b = 0w·x+b=0 that maximizes the margin.

Objective

Minimize ‖w‖2/2\|w\|^2 / 2‖w‖2/2 subject to yi(w·xi+b)≥1y_i (w \cdot x_i + b) \geq 1yi(w·xi +b)≥1 for all iii.

This is a convex optimization problem that can be solved using quadratic programming.

Time Allotted for the Activity:

To implement maximum margin classification using a Support Vector Machine (SVM), we can use popular machine learning libraries such as scikit-learn in Python. Here's a step-by-step guide to implementing SVM for maximum margin classification:

Step 1: Import Libraries

First, you need to install scikit-learn if you haven't already. You can install it using pip:

```
sh
Copy code
pip install scikit-learn
```

Then, import the necessary libraries:

```
python
Copy code
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

Step 2: Load and Prepare the Data

For this example, we'll use the Iris dataset, which is a classic dataset for classification problems:

```
python
Copy code
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data[:, :2]  # We'll use only the first two features for
simplicity
y = iris.target

# We'll only use two classes to make it a binary classification problem
X = X[y != 2]
y = y[y != 2]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Step 3: Train the SVM Model

We'll use the `SVC` class from scikit-learn with a linear kernel to achieve maximum margin classification:

```
python
Copy code
# Create and train the SVM model
svm_model = SVC(kernel='linear', C=1.0)
svm_model.fit(X_train, y_train)
```

Step 4: Make Predictions and Evaluate the Model

```
python
Copy code
# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

Step 5: Visualize the Decision Boundary

To visualize the decision boundary and the support vectors, you can plot the data points and the hyperplane:

```
python
Copy code
# Function to plot the decision boundary
def plot_decision_boundary(model, X, y):
    h = .02  # Step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
    plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
facecolors='none', edgecolors='r', marker='o', s=100)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary with Support Vectors')
    plt.show()

# Plot the decision boundary
plot_decision_boundary(svm_model, X_train, y_train)
```

Full Code

Here is the full code together:

```
python
Copy code
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the Iris dataset
```

```
iris = datasets.load_iris()
X = iris.data[:, :2]  # We'll use only the first two features for
simplicity
y = iris.target

# We'll only use two classes to make it a binary classification problem
X = X[y != 2]
y = y[y != 2]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train the SVM model
svm_model = SVC(kernel='linear', C=1.0)
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Function to plot the decision boundary
def plot_decision_boundary(model, X, y):
    h = .02  # Step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
    plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
facecolors='none', edgecolors='r', marker='o', s=100)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary with Support Vectors')
    plt.show()

# Plot the decision boundary
plot_decision_boundary(svm_model, X_train, y_train)
```

This code trains an SVM model on the Iris dataset, evaluates its accuracy, and visualizes the decision boundary along with the support vectors.

- **Procedure of the Implementation:**
  - The design thinking process is a set of structured strategies that identify challenges, gather information, generate potential solutions, refine ideas, and test solutions.
  - Collaborative, active learning strategy in which students work on a different application problem posed by the instructor, first thinking independently, then forming groups as Customer, Project Manager, Development, Testing and

Maintenance team then finally working together to interact with one another on their given problem.

- Once interaction is finished between the groups, finally the development team is asked to interact with the customer team and then the developer team interacts with their own team about the process.

Details of the Implementation

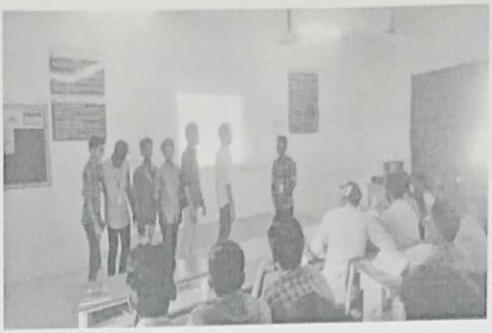
- Teacher gave a different application problem to the different teams about the various process models. Students think about what they know or have learned in software process model, and come up with their own idea to the problem. [Takes 1-3 Minutes].
- The teacher asks them to select the process model based on their problem and form a group Customer, Project Manager, Development, Testing and Maintenance team to discuss the problem. Then different teams interact with each other about their problems to deepen understanding of the process model. The Customer team is interacting with the developer's teams, and the developer teams collect the detailed requirements from the customers during their interaction. Then, the developer's team discussed the problem with their own team. They share their thoughts with each other and proceed with the task. [Takes 5-7 Minutes].
- Students share their solution with the entire class. Teacher moderates the discussion and highlights important points. [Takes 05-10 minutes].

- **CO – PO / PSO mapping:**

CO	PO1	PO2	PO3	PO4	PO8	PO10	PO11	PO12
CO1	3	3	3	1	1	1	1	2

(1 – Low 2 – Moderate 3 – High)

- **Images / Screenshot of the practice:**



- **Reflective Critique:**
  - ***Feedback of practice from students and other stakeholders:***
  - Most of the students actively participated and enjoyed. Students were given a solution with various formats.
  - Bright students enjoyed with peer learning.
  - ***Benefit of the practice:*** (E.g.: Outcome attainment would have increased due to innovative practice over conventional practice)
    - Students are actively engaged.
    - Students learn from each other.
    - Makes class interactive.
    - Builds a friendly, yet academic atmosphere.
    - Includes all the students in the teaching-learning process.
  - ***Challenges faced in implementation:***
    - Students were not aware of Design-Thinking Process even though made an announcement and posted the content of Design-Thinking Process in the course canvas.
    - Slow learners were not interested to participate in practice.
    - Noise level of the class room goes up and a transition was needed to get back on the topic.
    - It was hard to keep every student on task.

References:

1. https://www.forbes.com/sites/robynshulman/2018/11/19/10-ways-educators-can-make-classrooms-more-innovative/?sh=2fc797577f87

2. https://uxdesign.cc/3-design-thinking-exercises-to-make-problem-solving-more-exciting-98bc3bb67350

Signature of Faculty Member

HOD-IT

Dr. K. SUNDARAMOORTHY
Professor & HOD
Department of Information Technology
Jerusalem College of Engineering (Autonomous)
Pallikkaranai, Chennai-600 100.