Support Vector Machines (SVM)

🎯 Learning Outcomes

By the end of this lecture, you should be able to:

1. Understand the concept of hyperplanes and margins in SVM.
2. Distinguish between linear and non-linear SVMs.
3. Apply kernels in SVM to handle non-linearly separable data.
4. Implement SVM using `scikit-learn` on real datasets.
5. Tune hyperparameters using `GridSearchCV`.
6. Interpret results using classification metrics and plots.

---

1. Theoretical Background

1.1 What is an SVM?

Support Vector Machine (SVM) is a supervised learning algorithm used primarily for **classification** tasks (also applicable to regression). It finds the **optimal hyperplane** that separates classes with the **maximum margin**.

1.2 Key Concepts

- **Hyperplane**: A decision boundary that separates classes.
- **Margin**: Distance between the hyperplane and the nearest data point (support vector).
- **Support Vectors**: Points closest to the decision boundary.
- **Kernel Trick**: A technique to project data into a higher-dimensional space for non-linear separation.

1.3 Types of Kernels

- **Linear**: Straight-line separation.
- **Polynomial**: Curved boundary with polynomial transformation.

- **RBF (Gaussian)**: Maps data into infinite-dimensional space.
- **Sigmoid**: Similar to neural networks.

---

2. Practical Implementation with Detailed Explanation

🔧 Step 1: Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, Confusi
```

📊 Step 2: Load and Explore Dataset

```
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

print("Feature shape:", X.shape)
print("Class distribution:")
print(y.value_counts())
```

⚙️ Step 3: Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

📏 Step 4: Feature Scaling

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

✅ Step 5: Train a Linear SVM Model

```
svm_linear = SVC(kernel='linear', C=1.0)
svm_linear.fit(X_train_scaled, y_train)
y_pred_linear = svm_linear.predict(X_test_scaled)
```

📈 Step 6: Evaluate Linear SVM

```
print("Classification Report:\n", classification_report(y_test, y_pred_linear))
print("Accuracy:", accuracy_score(y_test, y_pred_linear))
ConfusionMatrixDisplay.from_estimator(svm_linear, X_test_scaled, y_test)
plt.show()
```

🔄 Step 7: Train an RBF Kernel SVM

```
svm_rbf = SVC(kernel='rbf', C=10, gamma=0.01)
svm_rbf.fit(X_train_scaled, y_train)
y_pred_rbf = svm_rbf.predict(X_test_scaled)
```

📊 Step 8: Evaluate RBF Model

```
print("Classification Report:\n", classification_report(y_test, y_pred_rbf))
print("Accuracy:", accuracy_score(y_test, y_pred_rbf))
ConfusionMatrixDisplay.from_estimator(svm_rbf, X_test_scaled, y_test)
plt.show()
```

🔍 Step 9: Hyperparameter Tuning with GridSearchCV

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf']
}

grid_search = GridSearchCV(SVC(), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)
print("Best Parameters:", grid_search.best_params_)
```

🧪 Step 10: Evaluate Best Estimator

```
best_model = grid_search.best_estimator_
y_best_pred = best_model.predict(X_test_scaled)
print("Accuracy:", accuracy_score(y_test, y_best_pred))
ConfusionMatrixDisplay.from_estimator(best_model, X_test_scaled, y_test)
plt.show()
```

📚 Assignment

Use this dataset: Heart Disease UCI

Tasks:

- EDA and preprocessing
- Train SVM (linear & RBF)
- Use `GridSearchCV`
- Report: Accuracy, confusion matrix, ROC-AUC, F1-score

📘 Further Reading

- Scikit-learn SVM Docs
- "Pattern Recognition and Machine Learning" by Christopher Bishop
- CS229 Notes by Andrew Ng

🔜 **Next Lecture**: Kernel SVM Decision Boundaries and Introduction to SVR