Support Vector Machine

GitHub Repository:
https://github.com/st24aaq/MLNN

Support Vector Machine (SVM)

1. What is a Support Vector Machine (SVM)?

Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification and regression tasks (López, López and Crossa, 2022). It is a powerful model that works by finding an optimal hyperplane that best separates data points of different classes (Sun, Yuan and Yang, 2024).

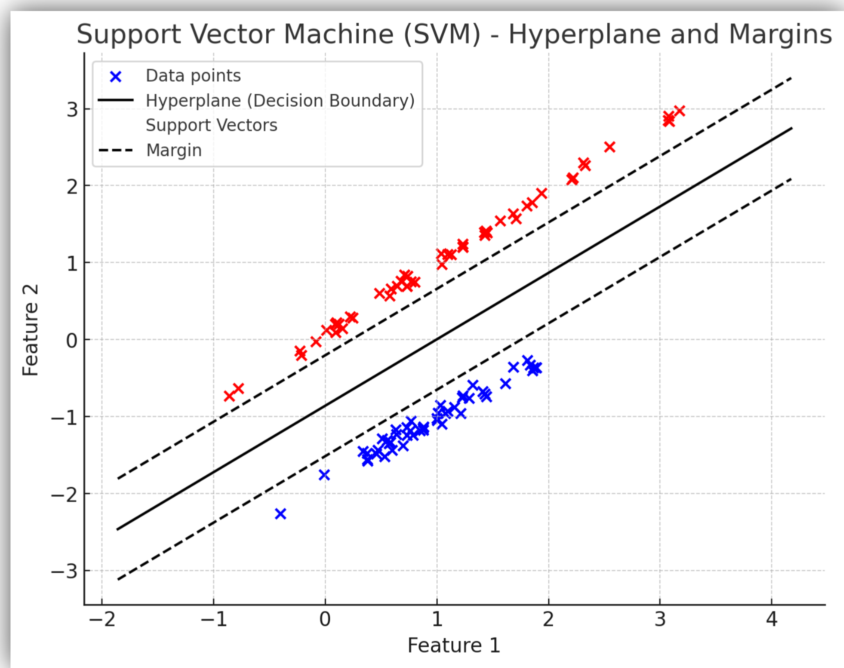

Figure 1 SVM

Here is a visualization of a Support Vector Machine (SVM) classifier showing:

- The decision boundary (black solid line - hyperplane)
- The support vectors (black circled points)
- The margin (black dashed lines)

This plot illustrates how SVM finds the optimal hyperplane that maximizes the margin between two classes. The data points closest to the hyperplane (support vectors) determine its position.

1.1 How Does SVM Work?

SVM operates based on the following key concepts:

- **Hyperplane:** A decision boundary that separates different classes in the dataset.
- **Support Vectors:** The closest data points to the hyperplane that influence its position.
- **Margin:** The distance between the hyperplane and the nearest support vectors. SVM aims to maximize this margin for better generalization.

1.2 Types of SVM

1. **Linear SVM:** Used when data is linearly separable (i.e., can be separated by a straight line) (Chen, Yin and Tian, 2024).
2. **Non-Linear SVM:** Used when data is not linearly separable. In such cases, SVM applies the **kernel trick** to map data into a higher-dimensional space where it becomes linearly separable (Sepahvand, 2021).

1.3 Kernel Functions in SVM

- **Linear Kernel:** Used when data is linearly separable (Reza et al., 2023).
- **Polynomial Kernel:** Maps input space into a higher degree polynomial space (Kamran et al., 2021).
- **Radial Basis Function (RBF) Kernel:** Useful when data is not linearly separable, as it transforms data into infinite-dimensional space (Virmani and Pandey, 2022).
- **Sigmoid Kernel:** Similar to a neural network activation function, useful for binary classification (Xu et al., 2021).

2. Applications of SVM

SVM is widely used in various domains due to its ability to handle both linear and non-linear problems efficiently.

2.1 Medical Diagnosis

- **Disease Prediction:** SVM is used to diagnose diseases like diabetes, cancer, and PCOS by classifying patient data based on medical attributes.
- **Medical Image Analysis:** Applied in MRI and CT scan image classification to detect anomalies.

2.2 Text Classification & Natural Language Processing (NLP)

- **Spam Email Filtering:** Classifies emails as spam or non-spam.
- **Sentiment Analysis:** Used to classify text-based reviews into positive, negative, or neutral sentiment.

- **Fake News Detection:** Distinguishes between real and fake news based on linguistic patterns.

2.3 Financial and Risk Analysis

- **Credit Risk Assessment:** SVM helps banks and financial institutions predict the probability of loan defaults.
- **Stock Market Prediction:** Used to classify stock trends based on historical market data.

2.4 Image Processing & Computer Vision

- **Facial Recognition:** SVM classifies facial features to identify individuals.
- **Handwritten Digit Recognition:** Used in Optical Character Recognition (OCR) systems (e.g., recognizing digits in postal code automation).

2.5 Intrusion Detection in Cybersecurity

- **Network Anomaly Detection:** Identifies unauthorized access attempts and security breaches in network traffic.
- **Malware Classification:** Differentiates between benign and malicious software.

3. Usage of SVM in Real-World Scenarios

3.1 When to Use SVM?

- When the dataset is small to medium-sized (SVM struggles with very large datasets due to computational complexity).
- When the data is high-dimensional (e.g., text classification with many features).
- When classes are well-separated or can be separated using a kernel trick.

3.2 Advantages of SVM

✓ Works well on high-dimensional data.

✓ Effective in cases where the number of dimensions exceeds the number of samples.

✓ Robust to overfitting when using appropriate kernel functions.

3.3 Limitations of SVM

✗ Computationally expensive for very large datasets.

✗ Choosing the right kernel function requires trial and error.

✗ Difficult to interpret compared to decision trees or logistic regression.

A. How Different Kernels Change the Behavior of SVM

1. Types of SVM Kernels and Their Effects on Model Behavior

1.1 Linear Kernel

Equation:

$$K(xi, xj) = xi \cdot xj$$

Behavior:

- The Linear Kernel is used when data is linearly separable.
- It performs well in high-dimensional datasets.
- Computationally efficient, making it ideal for large-scale problems.

Use Cases:

- Spam classification
- Document classification

1.2 Polynomial Kernel

Equation:

$$K(xi, xj) = (xi \cdot xj + c)^d$$

Behavior:

- Allow SVM to create curved decision boundaries.
- Higher-degree polynomials capture more complexity but may lead to overfitting.
- Computational cost increases with higher polynomial degrees.

Use Cases:

- Handwritten digit recognition
- Moderate non-linearity problems

1.3 Radial Basis Function (RBF) Kernel (Gaussian Kernel)

Equation:

$$K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2)$$

Behavior:

- The RBF Kernel transforms data into infinite-dimensional space.
- It creates flexible, non-linear decision boundaries.
- The γ (gamma) parameter controls model complexity:
    - High $\gamma \rightarrow$ Focuses on individual points (overfitting),
    - Low $\gamma \rightarrow$ More generalized (underfitting).

Use Cases:

- Image classification
- Medical diagnosis

1.4 Sigmoid Kernel

Equation:

$$K(x_i, x_j) = \tanh(\alpha\, x_i \cdot x_j + c)$$

Behavior:

- The Sigmoid Kernel behaves similarly to a neural network activation function.
- It can capture complex relationships but may suffer from convergence issues.
- Often used in hybrid approaches with other kernels.

Use Cases:

- Biological sequence analysis
- Deep learning-inspired applications

2. Summary: Choosing the Right Kernel

Kernel	Best for	Computational Cost	Decision Boundary
Linear	Linearly separable data, text classification	Low	Straight line
Polynomial	Moderate non-linearity, feature interactions	Medium-High	Curved
RBF (Gaussian)	Highly non-linear problems, image processing	High	Flexible, highly curved
Sigmoid	Probabilistic classification, neural network-like tasks	Medium	Non-linear, complex

3. Conclusion

The choice of kernel depends on the distribution and problem complexity:

- Linear Kernel is simple and computationally efficient.
- Polynomial Kernel introduces curved decision boundaries but may overfit.
- RBF Kernel is powerful for capturing complex patterns but requires careful tuning.
- Sigmoid Kernel mimics neural networks but is less commonly used.

B. Case Study: Predicting PCOS Diagnosis using SVM with Different Kernels

1. Introduction

Polycystic Ovary Syndrome (PCOS) is a common endocrine disorder that affects people with ovaries, leading to menstrual irregularities, hormonal imbalances, and fertility challenges. Early diagnosis of PCOS is crucial for effective management and treatment.

Machine learning models, such as Support Vector Machines (SVM), can help classify whether a patient has PCOS based on biological and clinical features. This study explores how different SVM kernels (Linear, Polynomial, RBF, and Sigmoid) affect model performance in predicting PCOS.

2. Understanding the Dataset

The dataset contains six key features (PCOS Diagnosis Dataset, 2025):

- **Age** (years)
- **BMI** (Body Mass Index)
- **Menstrual Irregularity** (1 = Yes, 0 = No)
- **Testosterone Level (ng/dL)**
- **Antral Follicle Count** (Number of follicles in ovaries)
- **PCOS Diagnosis** (1 = PCOS, 0 = No PCOS) → Target Variable

The goal is to build SVM models using different kernels and compare their performance.

3. Python Implementation of SVM for PCOS Prediction

Step 1: Load and Preprocess Data

- Check for missing values
- Normalize numerical features using StandardScaler
- Split dataset into Training (80%) and Testing (20%) sets

```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay

# Load the dataset
file_path = "pcos_dataset.csv"
pcos_data = pd.read_csv(file_path)
pcos_data.head()
```

Figure 2 Importing libraries and Load Dataset

	Age	BMI	Menstrual_Irregularity	Testosterone_Level(ng/dL)	Antral_Follicle_Count	PCOS_Diagnosis
0	24	34.7	1	25.2	20	0
1	37	26.4	0	57.1	25	0
2	32	23.6	0	92.7	28	0
3	28	28.8	0	63.1	26	0
4	25	22.1	1	59.8	8	0

Figure 3 Sample Dataset

```python
# Display dataset information
print("Dataset Shape:", pcos_data.shape)
print("Missing Values:\n", pcos_data.isnull().sum())
```

```
Dataset Shape: (1000, 6)
Missing Values:
 Age                         0
BMI                         0
Menstrual_Irregularity      0
Testosterone_Level(ng/dL)   0
Antral_Follicle_Count       0
PCOS_Diagnosis              0
dtype: int64
```

Figure 4 Exploring data and missing values

```python
# Define Features (X) and Target Variable (y)
X = pcos_data.drop(columns=['PCOS_Diagnosis'])  # Independent variables
y = pcos_data['PCOS_Diagnosis']  # Target variable

# Standardize the features (SVM performs better with normalized data)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split dataset into Training (80%) and Testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Print dataset info
print(f"Training Data Shape: {X_train.shape}, Testing Data Shape: {X_test.shape}")
```

Figure 5 Normalization and Splitting Data

Step 2: Train SVM Models with Different Kernels

- **Linear Kernel**: Good for linearly separable data
- **Polynomial Kernel**: Captures moderate non-linearity
- **RBF (Gaussian) Kernel**: Best for highly non-linear data
- **Sigmoid Kernel**: Similar to neural networks, rarely effective for classification

```python
# Function to train and evaluate SVM with different kernels
def train_svm(kernel_type):
    # Train SVM classifier
    model = SVC(kernel=kernel_type, C=1.0, random_state=42)
    model.fit(X_train, y_train)

    # Predict on test data
    y_pred = model.predict(X_test)

    # Evaluate Performance
    acc = accuracy_score(y_test, y_pred)
    print(f"\nSVM with {kernel_type} kernel - Accuracy: {acc:.4f}")
    print("Classification Report:\n", classification_report(y_test, y_pred))

    return model, y_pred

# Train SVM with different kernels
svm_linear, y_pred_linear = train_svm('linear')
svm_poly, y_pred_poly = train_svm('poly')  # Default degree=3
svm_rbf, y_pred_rbf = train_svm('rbf')
svm_sigmoid, y_pred_sigmoid = train_svm('sigmoid')
```

Figure 6 Modelling and Evaluation

```
SVM with linear kernel - Accuracy: 0.8950
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       161
           1       0.74      0.72      0.73        39

    accuracy                           0.90       200
   macro avg       0.83      0.83      0.83       200
weighted avg       0.89      0.90      0.89       200
```

```
SVM with poly kernel - Accuracy: 0.9000
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       161
           1       0.81      0.64      0.71        39

    accuracy                           0.90       200
   macro avg       0.86      0.80      0.83       200
weighted avg       0.90      0.90      0.90       200
```

```
SVM with rbf kernel - Accuracy: 0.9550
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       161
           1       0.89      0.87      0.88        39

    accuracy                           0.95       200
   macro avg       0.93      0.92      0.93       200
weighted avg       0.95      0.95      0.95       200
```

```
SVM with sigmoid kernel - Accuracy: 0.8300
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89       161
           1       0.56      0.62      0.59        39

    accuracy                           0.83       200
   macro avg       0.73      0.75      0.74       200
weighted avg       0.84      0.83      0.83       200
```

Figure 7 Performance Metrics

Step 3: Evaluate Performance

- Compare models using Accuracy, Confusion Matrix, and Classification Report
- Plot confusion matrices for each kernel

```python
# Function to display confusion matrix
def plot_confusion_matrix(y_pred, kernel_type):
    cm = confusion_matrix(y_test, y_pred)

    plt.figure(figsize=(5, 4))
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap=plt.cm.Blues, values_format='d')
    plt.title(f"Confusion Matrix - {kernel_type} Kernel")
    plt.show()

# Plot confusion matrices for each kernel
plot_confusion_matrix(y_pred_linear, 'Linear')
plot_confusion_matrix(y_pred_poly, 'Polynomial')
plot_confusion_matrix(y_pred_rbf, 'RBF')
plot_confusion_matrix(y_pred_sigmoid, 'Sigmoid')
```

Figure 8 Plot Confusion Matrix

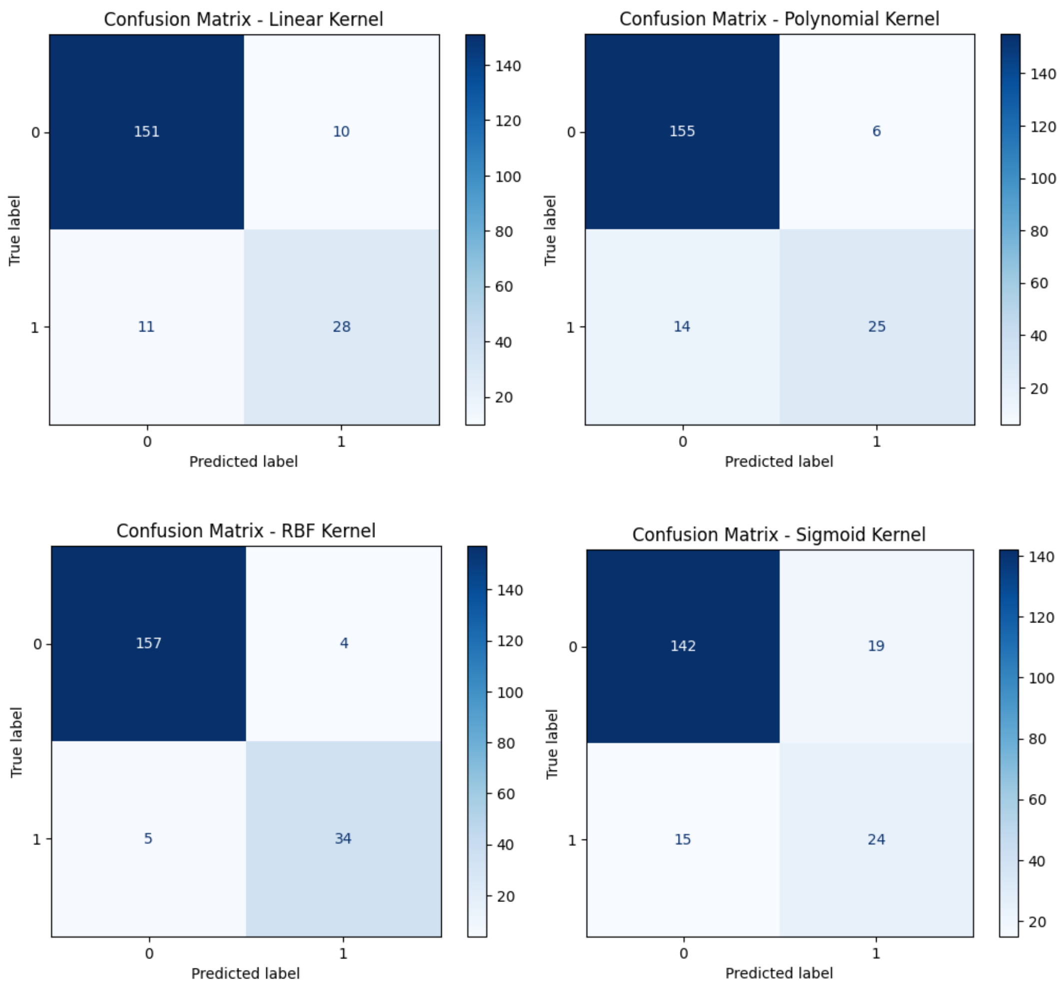

Figure 9 Confusion Matrix

The confusion matrices illustrate the classification performance of SVM models using four different kernel functions: Linear, Polynomial, RBF, and Sigmoid. Each matrix presents the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) counts, allowing for a detailed evaluation of the model's effectiveness in distinguishing between PCOS-positive (class 1) and PCOS-negative (class 0) individuals.

From the matrices:

- The RBF kernel performed the best, with just 4 false positives and 5 false negatives, making it the most accurate model for categorizing PCOS and non-PCOS cases together.

- While the linear kernel performed well, it misclassified 10 non-PCOS cases as PCOS (FP) and 11 PCOS cases as FN, indicating a modest trade-off in decision boundaries.

- Six false positives and 14 false negatives indicated that the polynomial kernel had more trouble detecting PCOS cases than linear.

- The sigmoid kernel has the highest misclassification rate (19 false positives, 15 false negatives), making it the least effective for this dataset.

RBF SVM has good sensitivity and specificity because it minimizes Type I and Type II errors. Linear kernels work better than polynomial and sigmoid kernels, especially for PCOS classification, which may involve non-linear patterns best captured by RBF. SVM models trained using Linear, Polynomial, RBF, and Sigmoid kernel functions perform well on PCOS datasets. The essential metrics are accuracy, precision, recall, and F1-score.

1. Linear Kernel Performance

Linear separability in the data is indicated by an 89.5% linear kernel SVM accuracy. Precision and recall were 0.74 and 0.72 for Class 1 (PCOS-positive cases), leading to a 0.73 F1-score. The model detects PCOS cases well, although it sometimes misclassifies positive cases as negative. The model can detect class 0 (non-PCOS cases) with high precision (0.93).

2. Polynomial Kernel Performance

A polynomial kernel captures minor dataset non-linearity with 90.0% accuracy. PCOS kernel precision increased to 0.81, but recall reduced to 0.64, resulting in a lower F1-score (0.71) than linear kernel. This suggests the polynomial kernel overfits to the majority class (non-PCOS cases), leading to less accurate PCOS predictions. However, its great accuracy suggests good performance.

3. RBF (Gaussian) Kernel Performance

The best model was the RBF kernel, with 95.5% accuracy. A class 1 F1-score of 0.88 detects 98% of non-PCOS cases and 87% of PCOS cases. This proves the RBF kernel handles complex, non-linear data well. The balance between precision (0.89) and recall (0.87) suggests that this model generalizes well to unseen data and misclassifies positive and negative cases best.

4. Sigmoid Kernel Performance

The worst accuracy is 83.0% for the sigmoid kernel. Class 1 had a precision of 0.56 and a recall of 0.62, resulting in an F1 score of 0.59. The sigmoid kernel may have problems separating PCOS from non-PCOS cases due to poor margin optimization and characterization. The model misclassifies cases more often, making it unsuitable for this dataset.

4. Conclusion

- The RBF kernel is the best choice, as it handles complex, non-linear patterns effectively and provides the highest accuracy and recall for PCOS detection.

- The linear kernel is a strong contender, especially if computational efficiency is a concern, as it performs well and provides good interpretability.

- The polynomial kernel can be useful for datasets with moderate non-linearity, but it needs careful tuning to avoid overfitting.
- The sigmoid kernel should not be used for this dataset, as it performs significantly worse than the other kernels.

5. Recommendation

RBF kernel in SVM is optimal for PCOS classification. It has the best accuracy, precision, and recall for PCOS diagnosis and classification error reduction. RBF kernels outperform Linear (89.5%), Polynomial (90.0%), and Sigmoid (83.0%) kernels with 95.5% accuracy. In PCOS-related measures including BMI, testosterone levels, and follicle count, RBF kernel captures non-linear relationships, making it effective. Medical diagnosis benefits from RBF kernel's low false positives and negatives. PCOS cases missing may delay or hinder therapy. Thus, with the best generalization and reliable performance, RBF SVM is the optimal model for accurate, sensitive, and reliable PCOS detection in healthcare applications.

References

Chen, X., Yin, Z. and Tian, H. (2024) 'Support vector machines principles and actually example,' *Procedia Computer Science*, 243, pp. 2–11. https://doi.org/10.1016/j.procs.2024.09.002.

Kamran, K.V. *et al.* (2021) 'A comparative approach of support vector machine kernel functions for GIS-based landslide susceptibility mapping,' *Applied Geomatics*, 13(4), pp. 837–851. https://doi.org/10.1007/s12518-021-00393-0.

López, O.A.M., López, A.M. and Crossa, J. (2022) 'Support vector machines and support vector regression,' in *Springer eBooks*, pp. 337–378. https://doi.org/10.1007/978-3-030-89010-0_9.

PCOS Diagnosis Dataset (2025). https://www.kaggle.com/datasets/samikshadalvi/pcos-diagnosis-dataset.

Reza, Md.S. *et al.* (2023) 'Improving SVM performance for type II diabetes prediction with an improved non-linear kernel: Insights from the PIMA dataset,' *Computer Methods and Programs in Biomedicine Update*, 4, p. 100118. https://doi.org/10.1016/j.cmpbup.2023.100118.

Sepahvand, K.K. (2021) 'Structural damage detection using supervised nonlinear support vector machine,' *Journal of Composites Science*, 5(11), p. 303. https://doi.org/10.3390/jcs5110303.

Sun, S., Yuan, J. and Yang, Y. (2024) 'Research on Effectiveness Evaluation and Optimization of baseball teaching method based on Machine Learning,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2411.15721.

Virmani, D. and Pandey, H. (2022) 'Comparative analysis on effect of different SVM kernel functions for classification,' in *Lecture notes in networks and systems*, pp. 657–670. https://doi.org/10.1007/978-981-19-3679-1_56.

Xu, S. *et al.* (2021) 'A High-Precision implementation of the Sigmoid activation function for Computing-in-Memory architecture,' *Micromachines*, 12(10), p. 1183. https://doi.org/10.3390/mi12101183.