Kernel Magic: Unveiling the Secrets of Support Vector Machines

Minor In AI, IIT Ropar

24th April, 2025

Welcome Note

Welcome, budding AI wizards! Get ready to embark on an exciting journey into the world of Artificial Intelligence, specifically focusing on a powerful tool of Support Vector Machines, or SVMs for short. This document will be your guide, turning complex concepts into understandable insights, starting with a real-world scenario we all can relate to.

1 The Shopping Spree Dilemma - An Intuitive Introduction

Imagine you are trying to understand your friends' shopping habits. You've noticed some friends spend practically all their money, while others window-shop mostly.

You have a spreadsheet with each friend's information: their age and how much they spend each month (including everything: food, entertainment, bills, even potential shopping). You also have a column indicating whether they tend to shop more than needed (a "1") or stick to essentials (a "0").

Figure 1: Spend more for a cheerful today or spend less for a cheerful next day? :)

Now, you stare at the spreadsheet, filled with ages from teenagers to adults, and spending ranging from a few hundred to several thousand. Can you easily draw a line on a graph and say, "Everyone above this line loves to shop, and everyone below doesn't?" Probably not! The data points are scattered everywhere.

Apples are red, oranges are orange - easy to classify. But this? This is where SVMs come to the rescue. It takes messy data and creates groups based on common traits.

That's what we're going to build in this document! We'll learn how to make such groups even in complex situations and see that it is very difficult to visualize the graph.

2 Data Collection & Preparation - Setting the Stage

Before we dive into the technical stuff, let's talk about the data we will be using. In our imaginary shopping scenario, you'd collect data like this:

- **Gender:** (Male/Female)
- **Age:** (e.g., 18, 25, 35)
- **Cost:** Monthly expenditure (e.g., 15000, 45000)
- **Purchase:** 0 or 1 (0 for minimal shopping, 1 for more than needed)

The lecture included a shared spreadsheet for collecting this data from its participants. Each participant was instructed to enter their information (or a hypothetical representation) to generate a dataset to be utilized for the rest of the lecture.

Key Notes

- Keep data clean. (No spaces.)
- Use small letters when entering the data.

3 Diving into Code - SVM with Python

Time to get our hands dirty with some code! We'll be using Google Colab, a fantastic environment for running Python code in the cloud.

3.1 Step 1: Importing Libraries and Loading Data

First, we import the necessary libraries and load the data from our CSV file.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from matplotlib.colors import ListedColormap

# Load the data
data = pd.read_csv('batch_data.csv')

print(data.head())
```

This code snippet:

1. Imports the `pandas` library for handling data frames.
2. Imports `numpy` for numerical operations.
3. Imports `matplotlib.pyplot` for making visualizations.
4. Imports `train_test_split` from `sklearn.model_selection` for splitting data into train and test datasets.
5. Imports `StandardScaler` for standardizing the data.
6. Imports `SVC` (Support Vector Classifier) from the `sklearn.svm` module.
7. Imports `ListedColormap` from `matplotlib.colors` to define the colors for visualizations.

Important

Make sure you upload your `batch_data.csv` file to your Google Colab environment.

3.2 Step 2: Preparing the Data for SVM

Next, we define our independent variables (X) and dependent variable (Y). Remember, we use age and cost to predict purchase behavior.

```
# Define X and Y variables
X_columns = ['Age', 'Cost']
Y_columns = 'Purchase'
X = data[X_columns].values
Y = data[Y_columns].values.flatten()
```

3.3 Step 3: Scaling our Data

The values of the cost column and age column are on different scales. We must standardize this.

```
# Scaling Data
sc = StandardScaler()
X = sc.fit_transform(X)
```

3.4 Step 4: Training and Visualizing the SVM

Here's the core of our SVM implementation:

```
# Train the model
classifier = SVC(kernel = 'rbf', random_state = 0, gamma='auto')
classifier.fit(X, Y)

# Visualizing the Training set results
X_set, y_set = X, Y
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,
↪  0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,
                     ↪  1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
↪  X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', 'red')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('blue', 'red'))(i), label = j)
plt.title('SVM (RBF Kernel)')
plt.xlabel('Age')
plt.ylabel('Cost')
plt.legend()
plt.show()
```

In this snippet:

1. We create an `SVC` object with an RBF (Radial Basis Function) kernel. The kernel is what helps the SVM separate complex data. We set `random_state` for reproducibility. `gamma='auto'` allows the model to automatically determine the best value.

2. We train the classifier using our scaled data `X` and the `Y` variable.

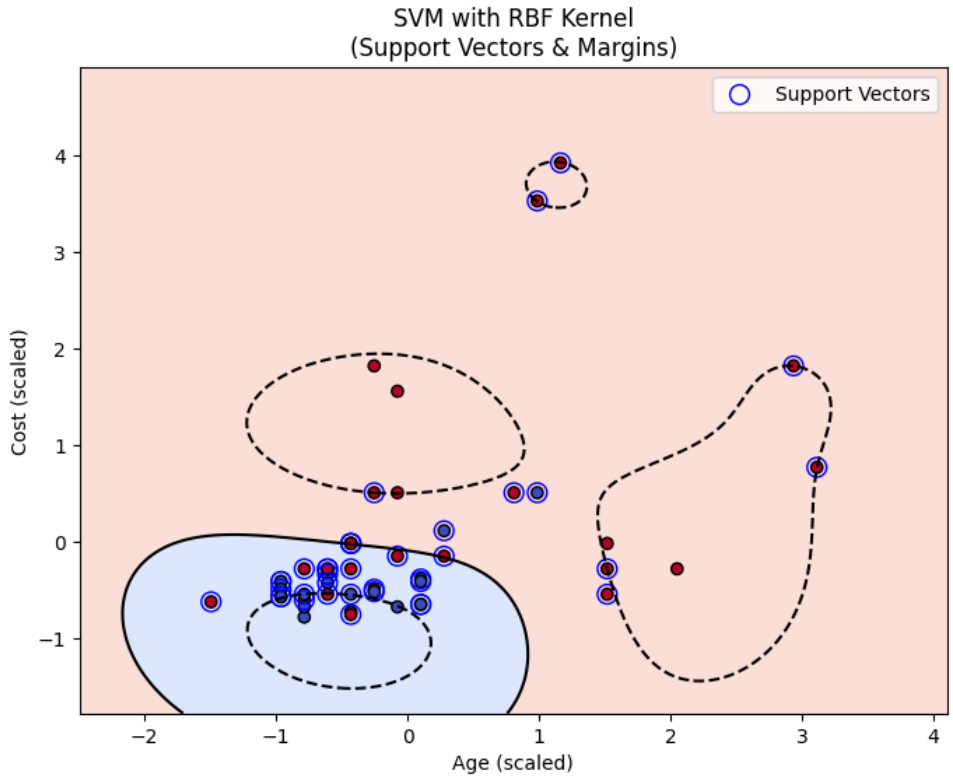

Figure 2: [PS: Don't worry if you don't fully understand the graph for the RBF Kernel, nobody does LOL] SVM with RBF Kernel: Age and Cost classification with decision boundary showing shoppers (red) and minimal shoppers (blue)

This graph visualizes how the SVM has separated the data into two groups: potential shoppers (red) and minimal shoppers (blue). The support vectors are the data points closest to the boundary. The decision boundary is a multi-dimensional map that takes the input and assigns it to one of the two groups.

3.5 Step 5: Making New Predictions

Now, let's see if we can predict the shopping behavior of a new individual.

```
# Predicting a new result
new_tuple = np.array([[38, 40000]]) # Age and Cost
new_tuple = sc.transform(new_tuple)
prediction = classifier.predict(new_tuple)

plt.scatter(new_tuple[:, 0], new_tuple[:, 1], color = 'green', marker = 'x', s =
↪  100, label = 'New Data Point')
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
↪  X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', 'red')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
```

```
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('blue', 'red'))(i), label = j)
plt.title('SVM (RBF Kernel)')
plt.xlabel('Age')
plt.ylabel('Cost')
plt.legend()
plt.show()

print("Prediction:", prediction[0])
```

This code:

1. Creates a new data point representing someone with age 38 and a monthly cost of 40000.
2. Scales the new data using the `sc.transform` method.
3. Uses the trained classifier to predict the shopping behavior.
4. Prints the classification prediction.

Understanding the Output:
If the output is "1", the model predicts that the person is likely to shop more than needed. If the output is "0", it predicts they'll stick to essentials. The green X plotted on the graph will show in which region this prediction lies.

4 Kernels - The Secret Sauce

The `kernel` parameter in our `SVC` function is what enables SVMs to handle complex data. Here are the most popular options:

- **Linear Kernel:** Suitable for linearly separable data. (Apples and Oranges.)
- **Polynomial Kernel:** Handles more complex data, but can be prone to overfitting.
- **RBF (Radial Basis Function) Kernel:** A general-purpose kernel that often performs well on a variety of datasets.

5 Interpreting the Graph - A Guided Tour

The SVM graph can be a bit intimidating, but don't worry! Here's how to make sense of it:

- **Data Points:** Each point represents an individual from your data. The color of the point corresponds to the class (e.g., red for shoppers, blue for minimal shoppers).
- **Decision Boundary:** The shaded region represents the decision boundary. It separates the data into different classes.

- **Support Vectors:** The data points closest to the decision boundary. They are crucial in defining the boundary.

The data may often be divided messy, but by using RBF kernel, we can get a decision boundary that allows us to classify the dataset.

Congratulations! You've taken a few more steps into the world of Support Vector Machines.

However, there's still much to explore:

- **Hyperparameter Tuning:** Experiment with different values for parameters like `C` and `gamma` to improve model performance.
- **Other Kernels:** Try using different kernels to see which one works best for your data.

Final Thoughts

Remember, the world of AI is vast and exciting. Keep experimenting, keep learning, and who knows? You might just discover the next breakthrough innovation!