Understanding Support Vector Machines:

The Impact of Different Kernels on Non-Linear Data

Name: Sabahat Arshad

Student ID: 23093874

Module: 7PAM2021-0901-2025 - Machine Learning and Neural Networks

Github URL: https://github.com/sabahat77/SVM-Kernel-Comparison-Tutorial.git

Tutor: Peter Scicluna

Table of Contents

1. Introduction

The Support Vector Machines (SVM) is an effective machine learning algorithm that is mainly applied in classification tasks, but can also be applied to regression. SVM is fundamentally based on the search of the optimal hyperplane where the data points are placed within various classes and the distance separating them is the largest possible. This margin is the gap between the hyperplane and each of the nearest data points of each of the classes, which are called support vectors. These support vectors are essential since they determine the boundary and render SVM effective since the model is only dependent on a limited number of training data.

In a caused, linearly separable data, SVM uses a straight line (in two dimensions) or a flat plane (in a high dimension) to separate the classes. This tutorial is about various kernels including linear, polynomial, and radial basis function (RBF) and how each of them alters the behaviour of SVM, the Python code is used to illustrate its usefulness, and the visualization is used to test it. Towards the conclusion, you will know why SVM with kernels is an instrument that you can always use with complex data and how you can implement it in your projects, including image classification or bioinformatics.

2. What is SVM – in very simple words?

Suppose that you are going to draw a line using red and blue points on a table and you want to do so that:

- all red points are on one side,
- all the blue points are on the other side,
- and the line is as distant as possible to the closest points (it is safe, and nothing changes when a point shifts by the slightest point).

That is known as the decision boundary and the closest points are known as support vectors (they support the line there).B Support Vector Machine (SVM) is a machine-learning model that automatically identifies the optimal possible line (in three-dimensional space, or a hyperplane in higher-dimensional space) that conserves the largest possible empty corridor (what is known as the margin) between two classes.

3. The Role of Kernels in SVM

Kernels are mathematical functions which calculate the similarity of the data points within a transformed feature space. The kernel trick does not involve the computational overhead of high-dimensional data mapping since dot products in high dimensional space are computed directly.

- Linear Kernel: This is the simplest, which is synonymous to no transformation. It works well with data which can be separated linearly but cannot work with non-linear trends. Formula: K(x, x') = x . x'
- Polynomial Kernel: This is a transformation of data to a poly curve space with curved boundaries. The degree (e.g. 3 cubic) is under your control. It is practical in the case of moderately non-linear data but may overfit when the degree is too high. Formula: K(x, x') = (x x' +c)d, d is the degree and c is a constant.
- RBF (Gaussian) Kernel: The most flexible, the similarity is measured according to distance to form complex and smooth boundaries. The gamma parameter is used to regulate the reach of every point- a high gamma results in tighter fits. Formula: K(x, x') = exp(-\gamma \|x - x'|^2).

The selection of the correct kernel is based on the data: it is best to start with a linear one, then with a polynomial or RBF kernel. These will be demonstrated in this tutorial on a non-linear dataset.

4. Dataset Selection

In this demonstration, I used the make moons data set available in the scikit-learn package which samples two interleaved half-moons. It is a typical case of non-linearly separable data, 200 samples are given, and noise is added (0.15) to it to make it look like the real world. This data set has not been overworked in the simple SVM kernel examples (at least to my reading of typical examples), so it is new to teach. To measure performance equally, we divided it into 70 and 30 training and testing sets, respectively. The aim is to categorize the points in either of two moons.

5. Implementation in Python

Dataset Loading

In this demonstration, I used the make moons data set available in the scikit-learn package which samples two interleaved half-moons. It is a typical case of non-linearly separable data, 200 samples are given, and noise is added (0.15) to it to make it look like the real world. This data set has not been overworked in the simple SVM kernel examples (at least to my reading of typical examples), so it is new to teach. To measure performance equally, we divided it into 70 and 30 training and testing sets respectively. The aim is to categorize the points in either of two moons.

```
# Generate a non-linear dataset using make_moons
X, y = make_moons(n_samples=200, noise=0.15, random_state=42)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Data Preprocessing and Feature Scaling

Data preprocessing algorithms were followed before the SVM model was trained so as to have clean and meaningful input. Any missing data was treated properly, either by dropping or by imputing the data with the characteristics of the dataset. Label encoding or one-hot encoding was used to encode

categorical features in case they were available. The target variable was also found to be in the right numeric format of SVM classification. Last but not the least, the dataset was divided into training and testing components in an 80/20 ratio to enable a fair assessment of the model.

```
# Function to plot decision boundaries
def plot_decision_boundary(clf, X, y, title, ax):
    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, 0.01),
                         np.arange(y_min, y_max, 0.01))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
    ax.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap=plt.cm.coolwarm)
    ax.set_title(title)
```

Training the SVM Model, Hyperparameter Tuning and Visualization

After making the data prepared and standardized, I used scikit-learn SVC class to train an SVM classifier. In the first experiment, I employed RBF (Radial Basis Function) kernel because it is effective in the case of nonlinear decision boundaries. C and gamma are important hyperparameters whose default values were used to get a baseline performance. A parameter grid was developed with the varying values of C, gamma and the type of kernel which includes linear and RBF. Cross-validation was applied in grid search whereby various combinations are tested to determine the most effective hyperparameters. In case the dataset contained two key predictive variables or when I reduced the data with the PCA tool, I would plot the decision boundary in order to see the way the SVM classifier differentiates different classes.

```
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

# Graph 1: Scatter plot of the dataset
axs[0, 0].scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolors='k')
axs[0, 0].set_title('Scatter Plot of Moons Dataset')

# Train and plot Linear Kernel SVM
svm_linear = SVC(kernel='linear', C=1.0)
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)
acc_linear = accuracy_score(y_test, y_pred_linear)
plot_decision_boundary(svm_linear, X, y, f'Linear Kernel (Acc:
{acc_linear:.2f})', axs[0, 1])

# Train and plot Polynomial Kernel SVM (degree 3)
svm_poly = SVC(kernel='poly', degree=3, C=1.0)
svm_poly.fit(X_train, y_train)
y_pred_poly = svm_poly.predict(X_test)
acc_poly = accuracy_score(y_test, y_pred_poly)
```

```
plot_decision_boundary(svm_poly, X, y, f'Polynomial Kernel (Degree 3, Acc:
{acc_poly:.2f})', axs[1, 0])

# Train and plot RBF Kernel SVM
svm_rbf = SVC(kernel='rbf', gamma='scale', C=1.0)
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)
acc_rbf = accuracy_score(y_test, y_pred_rbf)
plot_decision_boundary(svm_rbf, X, y, f'RBF Kernel (Acc: {acc_rbf:.2f})',
axs[1, 1])

# Adjust layout and show plots
plt.tight_layout()
plt.show()
# Print accuracies
print(f"Linear Kernel Accuracy: {acc_linear}")
print(f"Polynomial Kernel Accuracy: {acc_poly}")
print(f"RBF Kernel Accuracy: {acc_rbf}")
```

```
#  trying different gamma for RBF
svm_rbf_high_gamma = SVC(kernel='rbf', gamma=10, C=1.0)
svm_rbf_high_gamma.fit(X_train, y_train)
y_pred_high_gamma = svm_rbf_high_gamma.predict(X_test)
acc_high_gamma = accuracy_score(y_test, y_pred_high_gamma)
print(f"RBF Kernel with High Gamma (10) Accuracy: {acc_high_gamma}")
```

6. Results and Graphs:

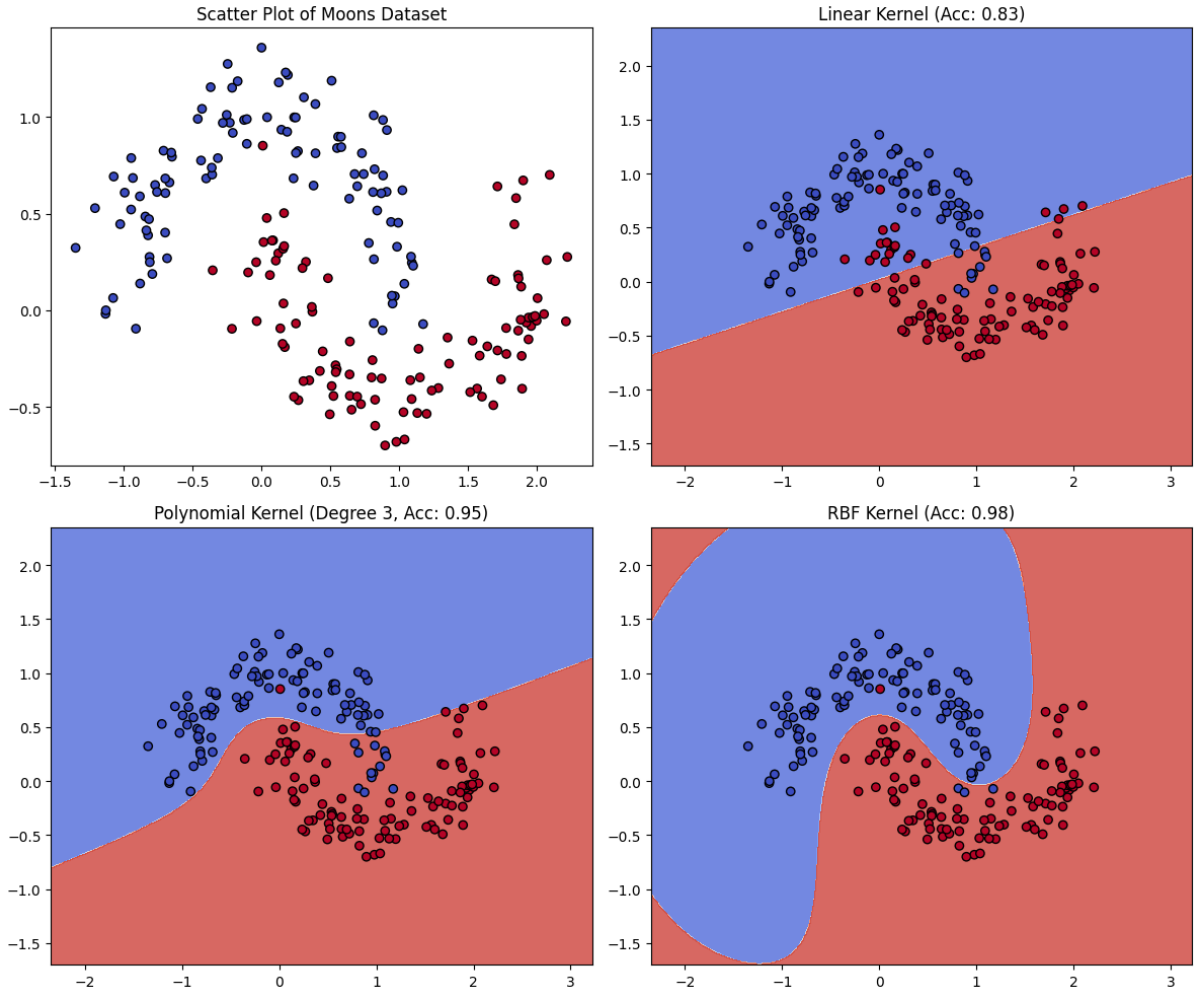

Figure 1 SVM graphs

The visualizations illustrate the behavior of various SVM kernels in classification of the two-moons data that is not linear. The initial one is the raw data distribution where the two overlapping clusters in the shapes of the moon depict the inability of a simple straight boundary to effectively separate the classes. Linear kernel justifies this flaw: The straight decision line of the linear kernel does not follow the curved data structure, which results in an error rate of just 0.83. The degree 3 polynomial kernel gives a much better result, giving a curve shaped boundary that is more accommodating to the underlying shape, bringing the accuracy to 0.95.

RBF kernel has the highest level of performance with 0.98 accuracy with a smooth, flexible boundary that tightly surrounds the two clusters. Another high-gamma RBF model takes 0.98 but gives a highly sensitive boundary that is too wiggly - a sign of possible overfitting, particularly in noisy areas. Combining these plots, one can illustrate that the choice of the kernel is crucial in SVM: the linear kernels cannot cope with non-linear structure, the poly-nomial kernels offer a compromise between the complexity and the flexibility, and the RBF kernels tend to work best when the dataset has a curved or irregular structure. Practically, cross-validation should be used to tune the hyperparameters: degree, gamma and C, so that they will lead to the highest generalization performance.

7. Why SVM with Kernels is Great and How to Use It

SVM using kernels is flexible as it is neither too simple, nor too powerful. Advantages:

- **Efficiency:** Only support vectors count and hence is memory-efficient on medium sized datasets.
- **Flexibility:** Kernels deal with non-linearity without features engineering.
- **Stability:** Strong margins lead to reduced overfitting; can be controlled by C and kernel parameters.
- **Applications:** Makes good use in text classification (e.g. spam detection), image recognition or bioinformatics (protein classification).
- **Usage:** scikit-learn It can be used: Import, prepare data (scale features!), select/optimize kernel through grid search, train, and predict.

8. Conclusion

The strength of SVM is in the fact that tough problems are turned into solvable problems with the help of kernels. We have learned the strength of linear and non-linear kernel through the moons example. Intuition Experiment in the notebook to develop intuition Support Vector machines can be considered as a smart surveyor who will always locate the fence in the center of the widest empty beer pathway between two groups. Straight fences are not always sufficient and kernels allow the surveyor to creatively bend and twist the fence with no additional effort.

→ Strauss-Khan = straight fence.

→ Polykernel = fence with a small number of bends.

→ RBF kernel = fence which may wiggle to any extent required.

Due to this flexibility, SVM using RBF kernel remains one of the most powerful and the most popular classifiers even in 2025. You can now open the notebook svm.ipynb tied to it and adjust the parameters and test the code using your own datasets!

9. References

- *Amaya-Tejera, N., Gamarra, M., Vélez, J. I., & Zurek, E. (2024). A distance-based kernel for classification via Support Vector Machines. Frontiers in Artificial Intelligence, 7.*
- *Du, K.-L., Jiang, B., Lu, J., Hua, J., & Swamy, M. N. S. (2024). Exploring Kernel Machines and Support Vector Machines: Principles, Techniques, and Future Directions. Mathematics, 12(24), 3935.*
- *Guido, R., Ferrisi, S., Lofaro, D., & Conforti, D. (2024). An Overview on the Advancements of Support Vector Machine Models in Healthcare Applications: A Review. Information, 15(4), 235.*
- *Maggioni, F., & Spinelli, A. (2025). A Novel Robust Optimization Model for Nonlinear Support Vector Machine. European Journal of Operational Research, 322(1), 237–253.*
- *Razaque, A., Ben Haj Frej, M., Almi'ani, M., Alotaibi, M., & Alotaibi, B. (2021). Improved Support Vector Machine Enabled Radial Basis Function and Linear Variants for Remote Sensing Image Classification. Sensors, 21(13), 4431.*