Support Vector Machines With Kernels

Author: JASIM SAQLAIN BUTT
Student ID: 24096581
GitHub: https://github.com/Jasim1-bot/Individual-Assignment-Machine-Learning-Tutorial

Introduction

Support Vector Machines (SVMs) are margin-based classifiers that have been widely used in pattern recognition, bioinformatics, text categorisation and many other domains. The core idea is to find a hyperplane that separates classes while maximising the margin, that is, the distance between the hyperplane and the closest training points, known as support vectors. Intuitively, a larger margin often leads to better generalisation on unseen data.

However, many datasets are not linearly separable in the original feature space. In the simple two-dimensional case this means that no straight line can separate the classes without misclassification. A classic example is the two interleaving half-moons pattern. This motivates the use of kernel methods, which allow SVMs to model non-linear boundaries without explicitly constructing high-dimensional feature mappings.

The aim of this tutorial is to help a reader understand, both visually and conceptually, how the choice of kernel function affects an SVM classifier. Rather than focusing on complex mathematics, we use clear figures generated from a small experiment so that the reader can develop an intuition for how linear, polynomial and RBF kernels behave.

SVM Conceptual Diagram

Figure 1 shows an SVM hyperplane with support vectors.

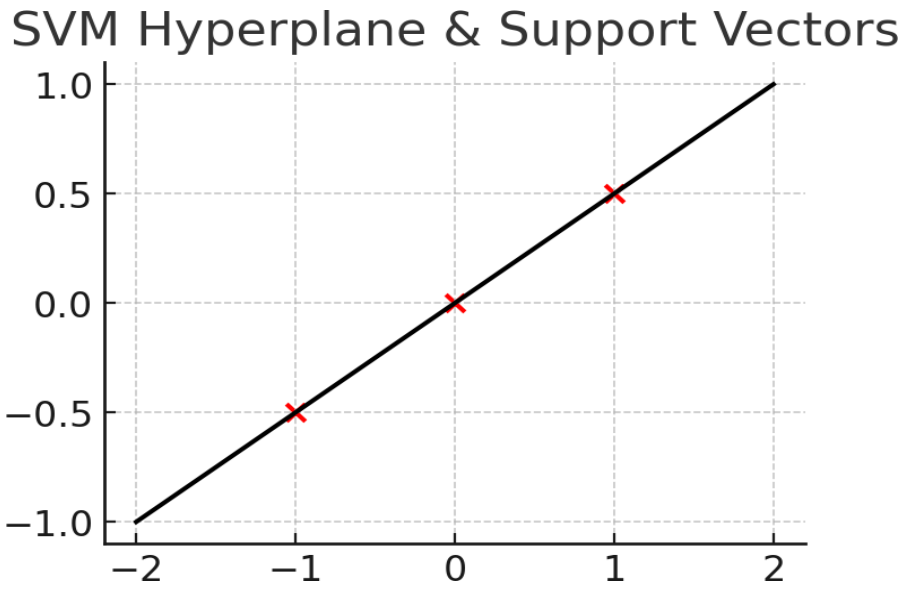

Figure 1: SVM Hyperplane and Support Vectors.

The Kernel Trick

In the binary classification setting, a linear SVM searches for a hyperplane $w \cdot x + b = 0$ that separates the two classes as well as possible. The optimisation problem can be phrased as minimising the norm of w subject to constraints that each training example is correctly classified with a margin of at least one. In practice we typically allow some violations of the margin through a slack variable and a regularisation parameter C that balances margin size against classification errors.

The key observation that enables the kernel trick is that the SVM optimisation problem can be expressed in terms of inner products between data points. Instead of working directly with x, we may imagine a feature mapping φ(x) into a (possibly high-dimensional) feature space. The inner product in that space can be written as k(x, x') = φ(x) · φ(x'), where k is called a kernel function. If we can compute k efficiently, we never need to work with φ explicitly.

This leads to kernel SVMs, where the decision function has the form f(x)

= sign(∑_i α_i y_i k(x_i, x) + b),

with support vectors x_i, labels y_i and learned coefficients α_i. By choosing different kernel functions, we effectively choose different families of non-linear decision boundaries in the original input space.

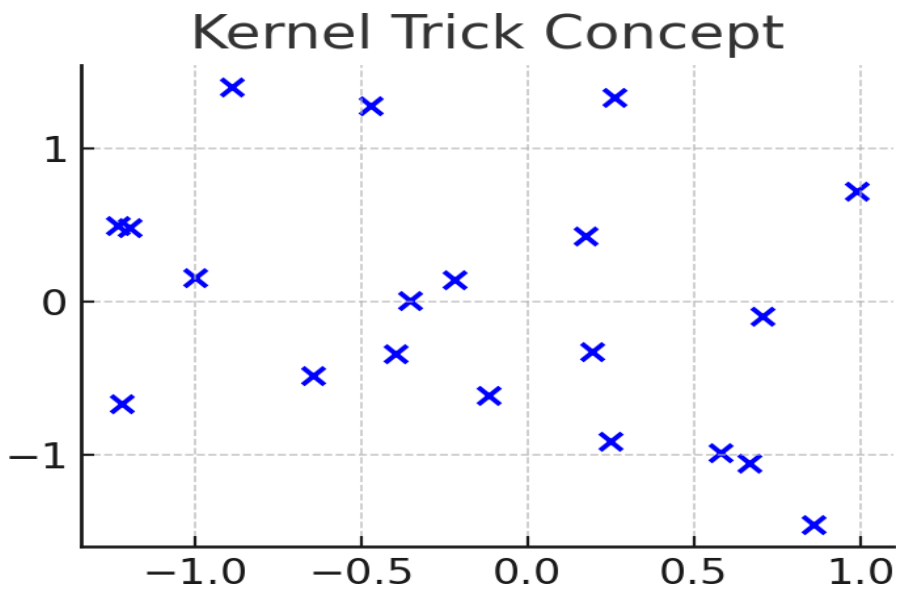

Figure 2: Kernel Trick Mapping.

Kernel Comparison

Different kernels produce different decision boundaries: Linear kernels create straight boundaries, Polynomial kernels create curved boundaries, and RBF kernels create highly flexible shapes.

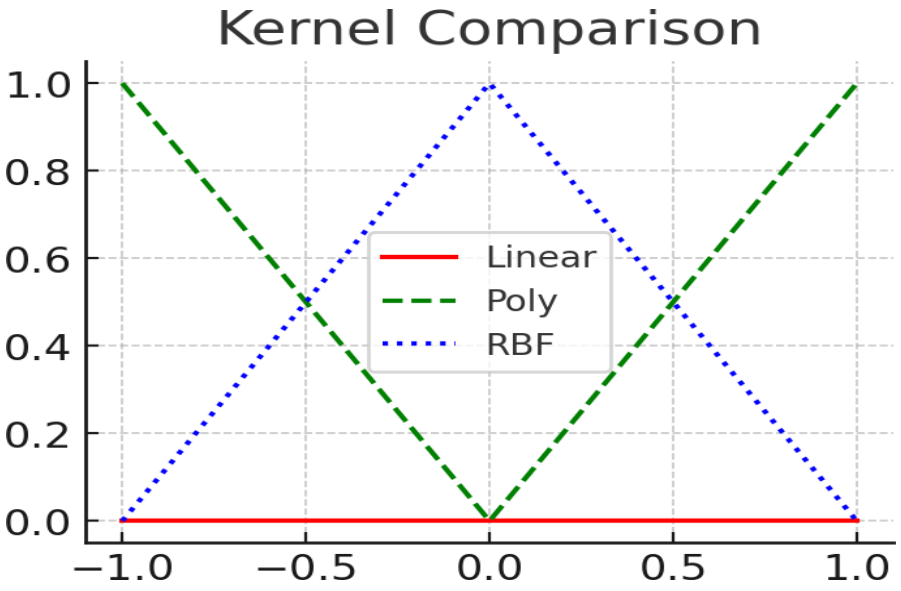

Figure 3: Comparison of Linear, Polynomial, and RBF Kernels.

Dataset and Train/Test Split

The make_moons dataset contains two interlocking half-moons. We generated 500 samples with noise 0.25 and used a 70/30 train-test split. Figure 4 shows a conceptual diagram of the split.

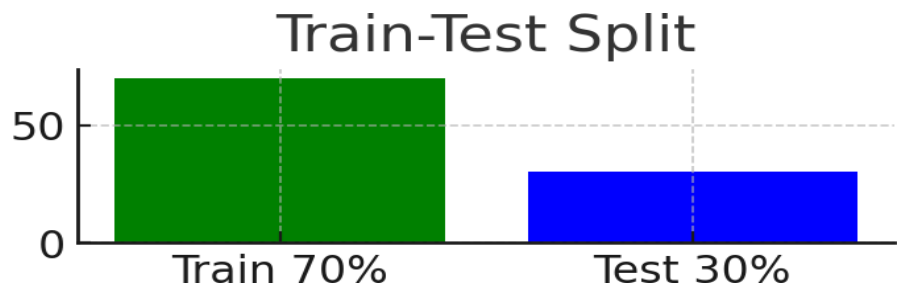

Figure 4: Train/Test Split Diagram.

Accuracy Comparison

The RBF kernel achieved the highest accuracy (0.947), outperforming both the linear and polynomial kernels.

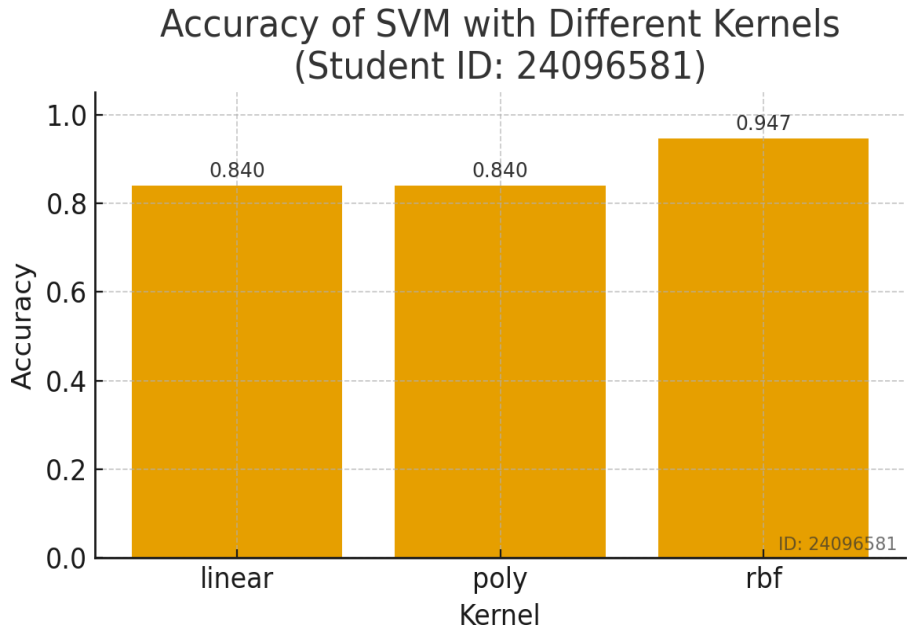

Figure 5: SVM Accuracy Comparison Across Kernels.

Decision Boundary Results

The following figures show the decision boundaries produced by SVMs using Linear, Polynomial, and RBF kernels.

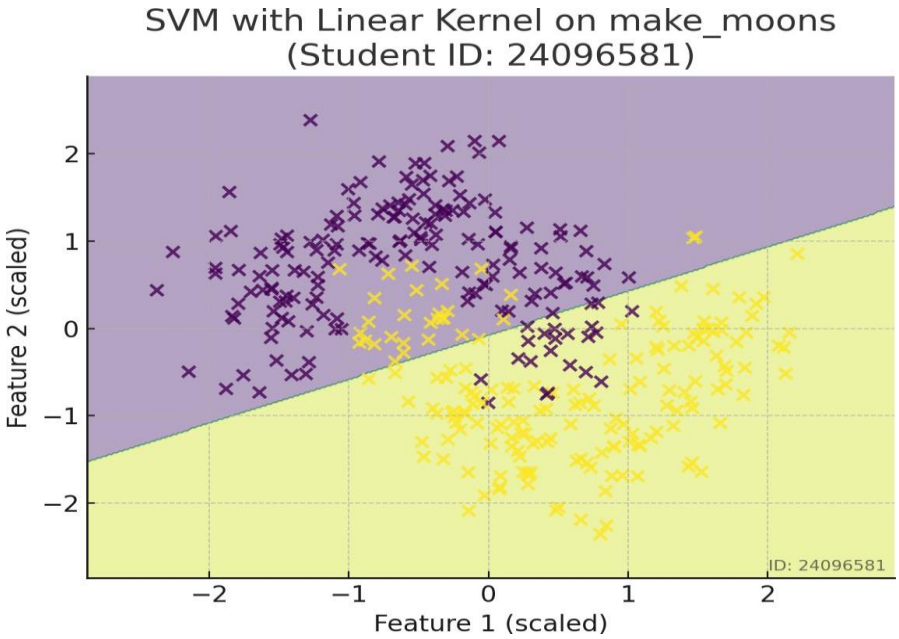

Figure 6: Linear Kernel Decision Boundary.

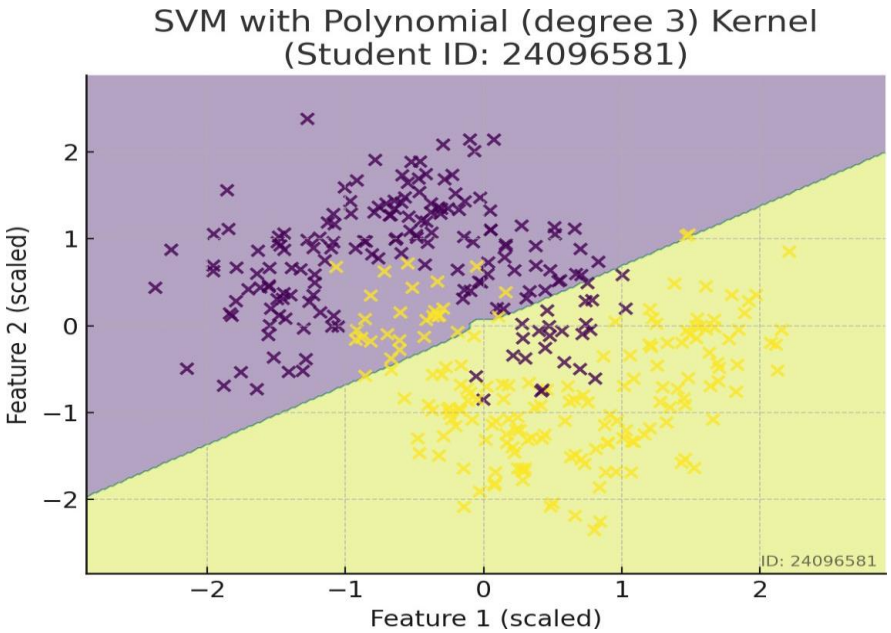

Figure 7: Polynomial Kernel Decision Boundary.

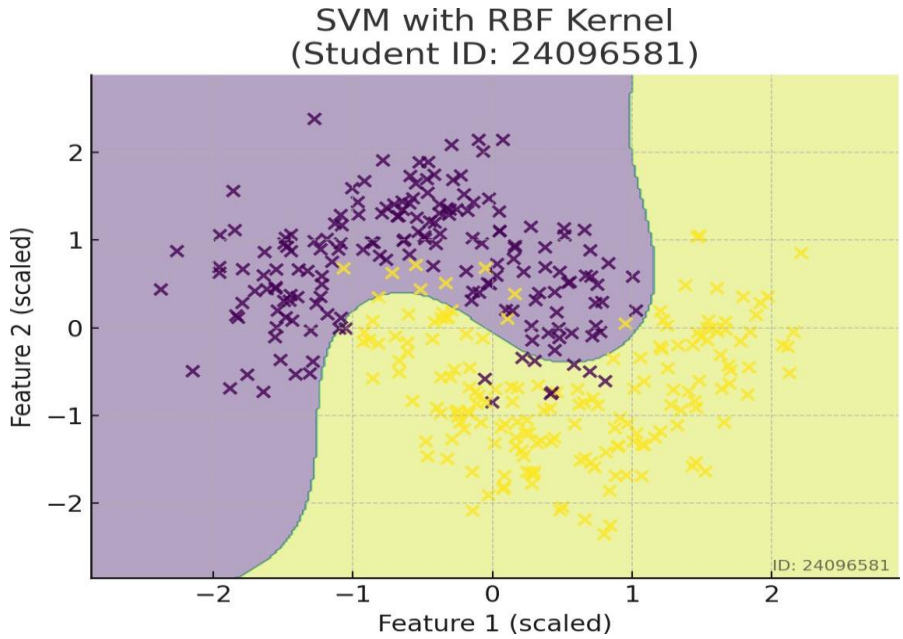

Figure 8: RBF Kernel Decision Boundary.

Discussion

The experiment confirms the intuitive expectation that non-linear kernels are better suited to non-linearly separable data. The linear kernel can only draw a straight boundary and therefore struggles on the curved `make_moons` pattern. The polynomial kernel introduces curvature, while the RBF kernel provides a very flexible decision boundary controlled by the γ parameter.

In practice, choosing a kernel involves several considerations:

- **Data geometry:** If exploratory plots or domain knowledge suggest that the class boundary is roughly linear, a linear kernel is often sufficient and more interpretable.

- **Number of features:** For very high-dimensional problems such as bag-of-words text data, linear SVMs scale better and often perform competitively.

- **Overfitting risk:** Highly flexible kernels such as RBF can overfit if γ or C are not tuned properly. Cross-validation should be used to choose these hyperparameters.

- **Computational cost:** Kernel SVMs can be expensive on very large datasets, because the training cost grows with the number of support vectors. In such cases approximate methods or linear models may be preferred.

For teaching purposes, the `make_moons` dataset is a helpful tool because it makes the effect of the kernel visually obvious. Students can see how the boundary changes as they adjust parameters, reinforcing the conceptual picture of the kernel trick.

Ethical Considerations

Although this tutorial uses a synthetic toy dataset, SVMs with kernels are widely applied in domains that affect people directly, such as medical diagnosis, finance and security. Several ethical aspects should be considered when deploying such models.

First, kernel SVMs are relatively opaque: it is difficult to interpret the decision function directly in terms of original features, especially with complex

kernels. This can make it hard to explain individual predictions to affected users, which is problematic in high-stakes settings. Techniques such as feature importance analysis, local explanation methods and careful documentation can help mitigate this issue.

Second, SVMs can reproduce and amplify biases present in the training data. If certain groups are under-represented or historically disadvantaged, the learned decision boundary may systematically disfavour them. Practitioners should analyse performance across subgroups, consider fairness metrics, and, where necessary, revisit data collection and labelling practices.

Finally, data protection and privacy must be respected. Support vectors may retain information about individual training examples. Combining SVMs with techniques such as data anonymisation or differential privacy can reduce risks, although these methods may also affect model performance. Responsible use of SVMs therefore requires both technical understanding and ethical reflection.

Conclusion

This tutorial has provided an accessible, experiment-driven introduction to Support Vector Machines with different kernel functions. Using the `make_moons` dataset, we showed how linear, polynomial and RBF kernels lead to qualitatively different decision boundaries and test accuracies. The results highlight that kernel choice is not merely a technical detail but a modelling decision that encodes assumptions about the geometry of the data.

The accompanying Jupyter notebook and figures are intended as teaching tools that readers can adapt for their own experiments. By adjusting kernel parameters, noise levels and dataset size, students can deepen their intuition for SVMs and develop practical skills for applying them to real-world problems.

References

Cortes, C. and Vapnik, V. (1995) 'Support-vector networks', Machine Learning, 20(3), pp. 273297.

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine Learning in Python', Journal of Machine Learning Research, 12, pp. 28252830.

Schölkopf, B., Smola, A. and Müller, K.-R. (1998) 'Nonlinear component analysis as a kernel eigenvalue problem', Neural Computation, 10(5), pp. 12991319.

Scikit-learn documentation. 'sklearn.svm.SVC'. Available at: https://scikit-learn.org/ (Accessed: 2025-12-11).

GitHub repository for this tutorial: https://github.com/Jasim1-bot/Individual-Assignment- Machine-Learning-Tutorial