Support Vector Machines with Kernels: An Illustrated

Student ID: 24096581
Date: 2025-12-11

Module: Advanced Machine Learning / Neural Networks
GitHub repository: https://github.com/Jasim1-bot/Individual-Assignment-Machine-Learning-Tutorial

Abstract

Support Vector Machines (SVMs) are powerful supervised learning models that construct a decision boundary with maximum margin between classes. While linear SVMs work well when classes are separable in the original feature space, many real-world problems are not linearly separable. Kernel methods address this limitation by implicitly mapping the data into a higher-dimensional feature space in which a linear separator may exist.

This tutorial introduces SVMs and explains how different kernel functions change the geometry of the decision boundary. Using the synthetic `make_moons` dataset, we compare three kernels: linear, polynomial and radial basis function (RBF). We visualise decision boundaries, report classification accuracy, and discuss when each kernel is appropriate. We also briefly reflect on ethical and practical considerations when deploying kernel SVMs in real applications.

1 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.

2 Support Vector Machines and 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 $\varphi(x)$ into a (possibly high-dimensional) feature space. The inner product in that space can be written as $k(x, x') = \varphi(x) \cdot \varphi(x')$, 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

$$f(x) = \mathrm{sign}\left(\sum_i \alpha_i y_i k(x_i, x) + b\right),$$

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.

3 Kernel Functions Considered

We focus on three widely used kernels that are implemented in scikit-learn.

Linear kernel

The linear kernel is simply $k(x, x') = x \cdot x'$. It corresponds to no additional feature mapping and therefore produces a straight decision boundary in the original space. Linear SVMs are often effective when the number of features is large compared to the number of samples, such as in text classification.

Polynomial kernel

The polynomial kernel has the form

$$k(x, x') = (\gamma x \cdot x' + r)^d,$$

where d is the degree, γ controls the influence of individual features and r is a constant term. For degree $d > 1$, this kernel allows curved decision boundaries that can model interactions between features. In our experiment we use a cubic polynomial kernel with degree 3.

Radial basis function (RBF) kernel

The RBF kernel, also called Gaussian kernel, is defined as

$$k(x, x') = \exp(-\gamma ||x - x'||^2),$$

where $\gamma > 0$ controls the width of the Gaussian. This kernel creates a flexible, smooth decision boundary that can adapt to complex shapes. For many problems the RBF kernel is a strong default choice.

4 Dataset and Experimental Setup

To illustrate the effect of kernel choice we use the synthetic `make_moons` dataset from scikit-learn. This function generates two interleaving half-moon shapes in two dimensions, with a configurable amount of noise. In our experiment we sample 500 points with noise level 0.25 and split them into 70% training and 30% test data. The resulting classes are not linearly separable, making them an ideal test case for non-linear kernels.

Before training, we standardise the features using `StandardScaler` so that each dimension has zero mean and unit variance. This is important because SVMs are sensitive to the scale of the input features, particularly when using RBF or polynomial kernels.

We train three SVM classifiers using scikit-learn's `SVC` implementation:

- Linear kernel (`kernel='linear'`)
- Polynomial kernel of degree 3 (`kernel='poly', degree=3`)
- RBF kernel (`kernel='rbf'`)

All models use the default regularisation parameter C and `gamma='scale'` where applicable. For each model we compute the classification accuracy on the test set, print the confusion matrix, and generate a decision boundary plot using a dense grid over the feature space. The code for this experiment is provided in the accompanying Jupyter notebook `svm_kernels.ipynb`.

5 Results

Figure 1 shows the decision boundary learned by the linear kernel SVM. As expected, the model produces a straight separating line in the feature space. Because the `make_moons` classes are arranged in curved shapes, this line cannot separate the classes cleanly and several points are misclassified on both sides. The test accuracy for the linear kernel is therefore relatively modest.

Figure 2 displays the decision boundary for the polynomial kernel with degree 3. The resulting curve bends to follow the arcs of the moons more closely. Visually, the boundary now wraps around one class and excludes most of the other, reducing the number of misclassifications. The test accuracy increases compared to the linear case, illustrating how even a relatively low-degree polynomial can capture important non-linear structure.

Figure 3 shows the decision boundary for the RBF kernel. Here the boundary is smooth but highly flexible, closely tracing the interface between the two moon shapes. In our run, the RBF SVM achieves the highest test accuracy of all three kernels. Figure 4 summarises the accuracies in a bar chart for direct comparison. On this dataset, the ranking of kernels by accuracy is typically linear < polynomial < RBF, although the exact values may vary slightly with the random train/test split.

6 Discussion and Practical Guidance

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.

7 Ethical and Societal 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.

8 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.

9 **References**

Cortes, C. and Vapnik, V. (1995) 'Support-vector networks', Machine Learning, 20(3), pp. 273–297.

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine Learning in Python', Journal of Machine Learning Research, 12, pp. 2825–2830.

Schölkopf, B., Smola, A. and Müller, K.-R. (1998) 'Nonlinear component analysis as a kernel eigenvalue problem', Neural Computation, 10(5), pp. 1299–1319.

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