MANASI CHAVAN

23104637

https://github.com/manasichavan08/svm-kernels-make-moons

Machine Learning Tutorial: Understanding Support Vector Machines with Kernel Functions: A Visual Guide Using make_moons

Contents

Introduction

One of the strongest and most reliable algorithms for resolving classification issues is Support Vector Machines (SVM). Their strong mathematical underpinnings and capacity to handle both linear and non-linear data have made them useful in a variety of domains, ranging from bioinformatics and text recognition to image classification and finance.

SVM sets itself apart from other classification algorithms with its method for determining the optimal class boundary. Instead of just drawing a line between the categories, SVM looks for the best hyperplane to maximize the margin, which is the distance between the boundary and the nearest data points from each class. These points that define boundaries are called support vectors. A higher margin frequently results in improved generalization, increasing the likelihood that the model will function effectively on unknown data.

The fact that SVM performs effectively in high-dimensional domains is another factor contributing to its popularity. This implies that even with a huge number of features in the dataset, which frequently leads to overfitting or difficulties for other models, it may still function effectively.
Real-world datasets are rarely straightforward, though. No flat surface or straight line can precisely divide the classes, hence they frequently involve patterns and class distributions that are not linearly separable. In these situations, SVM employs a technique known as the kernel trick. By using this method, the program can project the data into a higher-dimensional space without having to do the change itself. Even though the original data appears convoluted, it is easy to construct a separating border in that new area.

I decided to use the scikit-learn-provided make_moons synthetic dataset for this lesson. Its purpose is to show off a classifier's ability to handle input that is not linearly separable. Because the dataset produces two interlocking half-circles, it is impossible to use a straight line to divide the classes, which is the perfect situation to test the behavior of various SVM kernels.

This project's primary objective is to investigate the ways in which these various kernel functions impact the SVM's decision boundaries and evaluate their effectiveness using visual plots and accuracy ratings. You'll have a better grasp of how kernel selection is important for SVM classification by the end of this course, and why it's not a one-size-fits-all choice.

What is SVM & the Role of Kernels

Support Vector Machines (SVM) are based on the idea of finding the most optimal boundary that separates data points belonging to different classes. But rather than just aiming for any boundary that separates the data, SVM specifically looks for the **one that maximizes the margin**—the distance between the decision boundary and the closest data points from each class. These critical points are called **support vectors**, and the final model depends entirely on them. That's actually where the name "Support Vector Machine" comes from.

The margin plays a huge role in how well the model performs, especially when applied to unseen data. A larger margin usually means the model is more confident and less likely to overfit.

This is effective when the data can be neatly divided by a hyperplane or straight line, but it isn't always the case. In actuality, the majority of facts in the real world cannot be precisely separated in a straight line. Kernels enter the picture at this point. The idea of finding the optimal border between data points that belong to different classes forms the basis of Support Vector Machines (SVM). However, SVM specifically looks for the border that maximizes the margin—that is, the distance between the decision boundary and the closest data points from each class—instead of looking for any border that separates the data. These pivotal locations are known as support vectors, and the final model is entirely dependent upon them. The phrase "Support Vector Machine" actually comes from that.

The Role of Kernels

SVM's flexibility comes from its kernels. By projecting the data into a higher-dimensional space, they enable the method to handle data that isn't linearly separable. Even if the data appeared tangled or circular in the original space, it becomes simpler to construct a straight-line (or flat) decision boundary in that area.

The fact that SVM doesn't compute this transition directly is quite intelligent. Rather, it calculates the inner product between two data points in that higher-dimensional space using a kernel function, a mathematical shortcut. One of the reasons SVM can scale successfully, even to very high-dimensional data, is the kernel trick.

Types of Kernels and When to Use Them:

- **Linear Kernel**: When the classes are roughly linearly separable, the linear kernel works well. It is easy to use, quick, and frequently works well when there are many features relative to the amount of samples (as in text categorization).
- **Polynomial Kernel**: Gives the model more adaptability by permitting curved decision boundaries. The complexity of the curve is determined by the polynomial's degree. It can fit more complicated data, but if not adjusted properly, it is also more likely to overfit.
- **RBF (Radial Basis Function) Kernel**: One of the most commonly used kernels. It works well for a wide range of data types, especially when the decision boundary is not obvious. It forms circular decision regions and adapts to local variations in the data. The performance is heavily influenced by the gamma parameter, which determines how tightly the boundary fits around the data points.
- **Sigmoid Kernel**: Motivated by neural networks' activation function. Although it is less frequently employed and frequently exhibits inconsistent performance, it could be helpful in some situations requiring custom or hybrid models.

Summary

The performance of an SVM model is significantly impacted by the kernel selection. Underfitting or overfitting may result from using the incorrect kernel. Because of this, it's typical practice to begin with a linear or RBF kernel and then try out different ones depending on the behavior of the data.

SVM, in other words, is only as strong as the kernel that powers it. Building precise and trustworthy categorization models requires an understanding of how each one functions and when to apply it.

Mathematical Explanation with Examples

The goal of SVM is to solve this optimization problem:

Minimize:
$\mathbf{(1/2)\|w\|^2 + C \sum \xi_i}$

Subject to:
$\mathbf{y_i(w \cdot x_i + b) \geq 1 - \xi_i}$, and $\mathbf{\xi_i \geq 0}$

- **w**: weight vector (defines the direction of the hyperplane)
- **b**: bias (offset from the origin)
- **ξ_i**: slack variable (for points that are inside the margin or misclassified)
- **C**: regularization parameter (controls the trade-off between margin size and classification errors)

The Kernel Trick

Instead of calculating transformations manually, we use a kernel function:

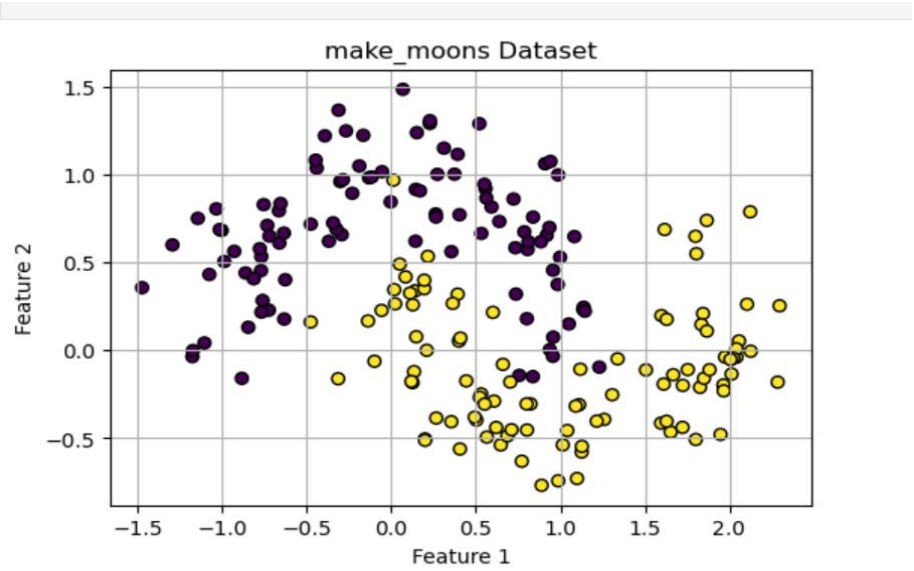
$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$

Here are some examples of kernels and how they work:

- **Linear**:
  $K(x, x') = x \cdot x'$
  Example: $[1, 2] \cdot [2, 1] = 1\times2 + 2\times1 =$ **4**
- **Polynomial**:
  $K(x, x') = (x \cdot x' + 1)^2$
  Example: $(4 + 1)^2 =$ **25**
- **RBF**:
  $K(x, x') = \exp(-\gamma\|x - x'\|^2)$
  Example: If distance = 2, $\gamma = 1 \rightarrow \exp(-4) =$ **0.018**
- **Sigmoid**:
  $K(x, x') = \tanh(\alpha(x \cdot x') + c)$
  Example: $\tanh(3) \approx$ **0.995**

Dataset & Preprocessing

For this tutorial, I used the built-in make_moons dataset from scikit-learn. It generates two interlocking half circles, making it perfect to demonstrate how SVM kernels work on non-linearly separable data.



- **Samples**: 200
- **Noise**: 0.2 (adds randomness to make it more realistic)
- **Split**: 70% training, 30% testing
- **Scaling**: StandardScaler was used to standardize feature values

---

Model Implementation

I implemented four different SVM models using the SVC class from sklearn.svm. Each model used a different kernel:

- Linear
- Polynomial (degree = 3)
- RBF
- Sigmoid

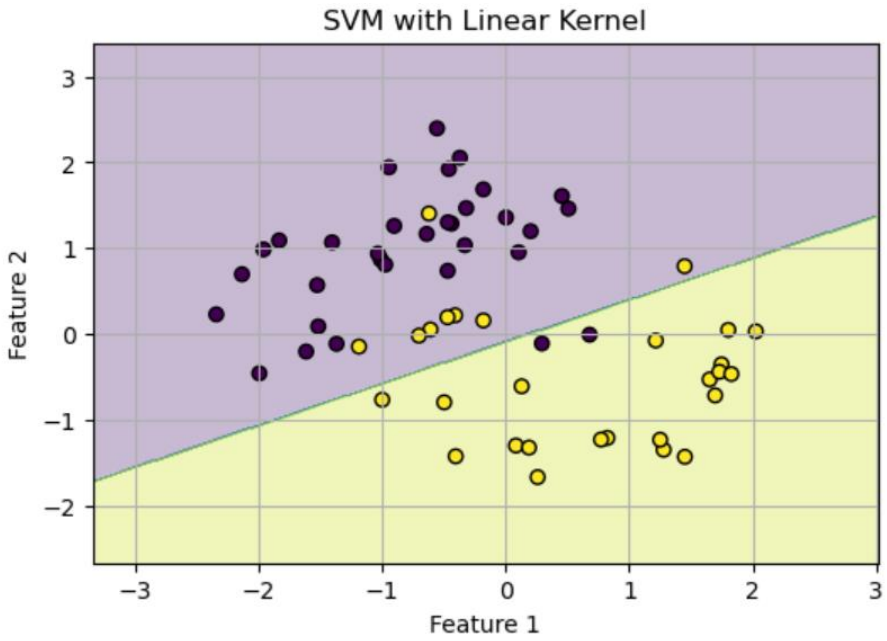
To visualize the decision boundaries, I created a mesh grid over the feature space and used the trained model to predict class labels across that grid. These predictions were then plotted, along with the test data, to clearly show how each kernel separates the classes

---

Results & Decision Boundaries

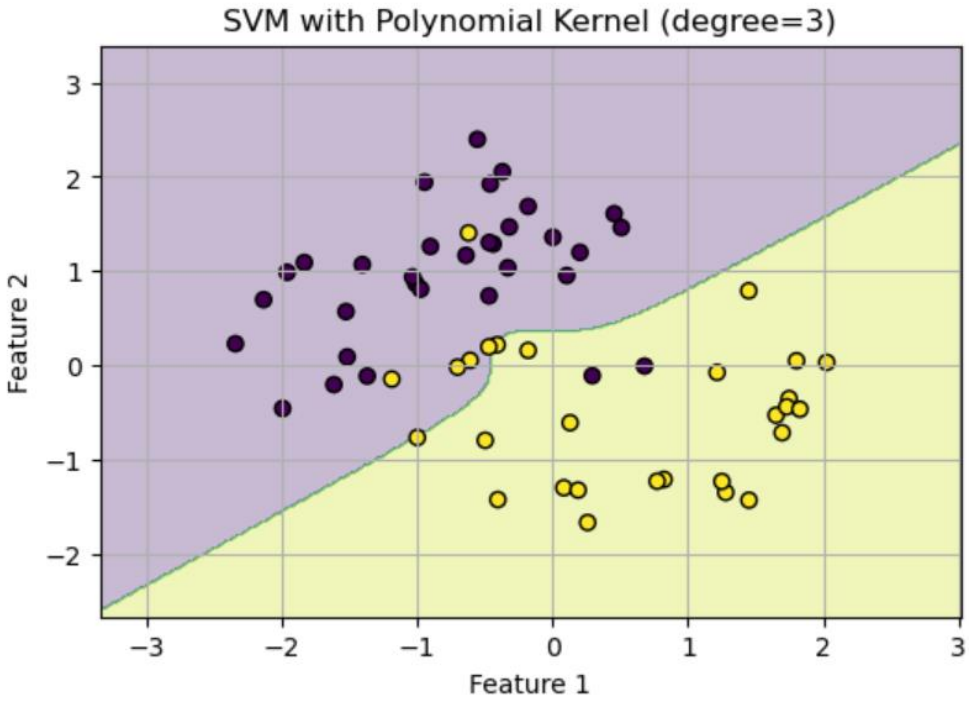
Here's a summary of how each kernel performed on the test set:

- **Linear Kernel**



  - **Accuracy**: 0.83
  - Decision boundary: straight line
  - Not flexible enough for the curved shape of the dataset

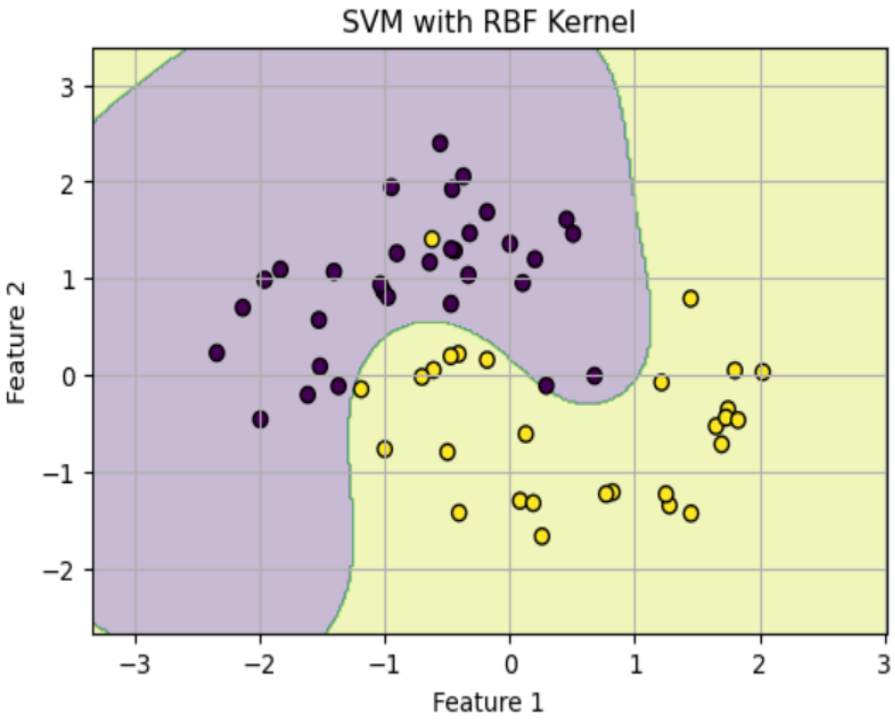
- **Polynomial Kernel (degree = 3)**



- **Accuracy**: ~0.87
- Decision boundary: curved and fits better
- Works well for moderately complex patterns

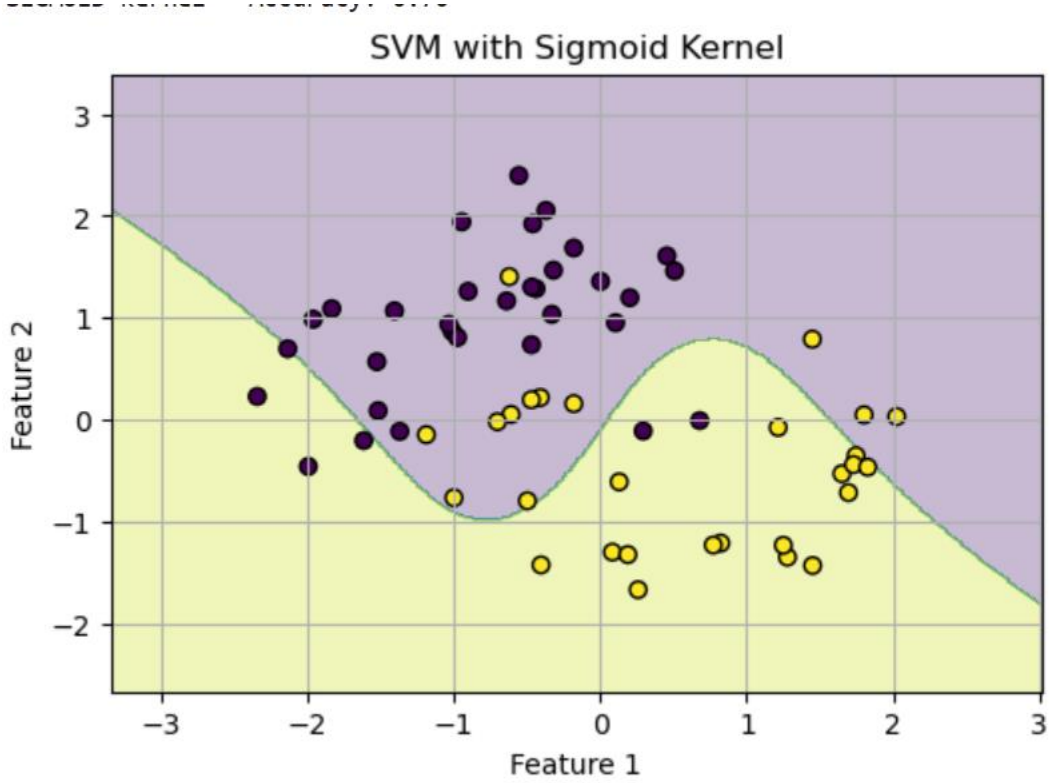
🔹 RBF Kernel

```
RBF kernel - Accuracy: 0.98
```



- **Accuracy**: 0.98
- Decision boundary: very flexible and smooth
- Best performance on this dataset

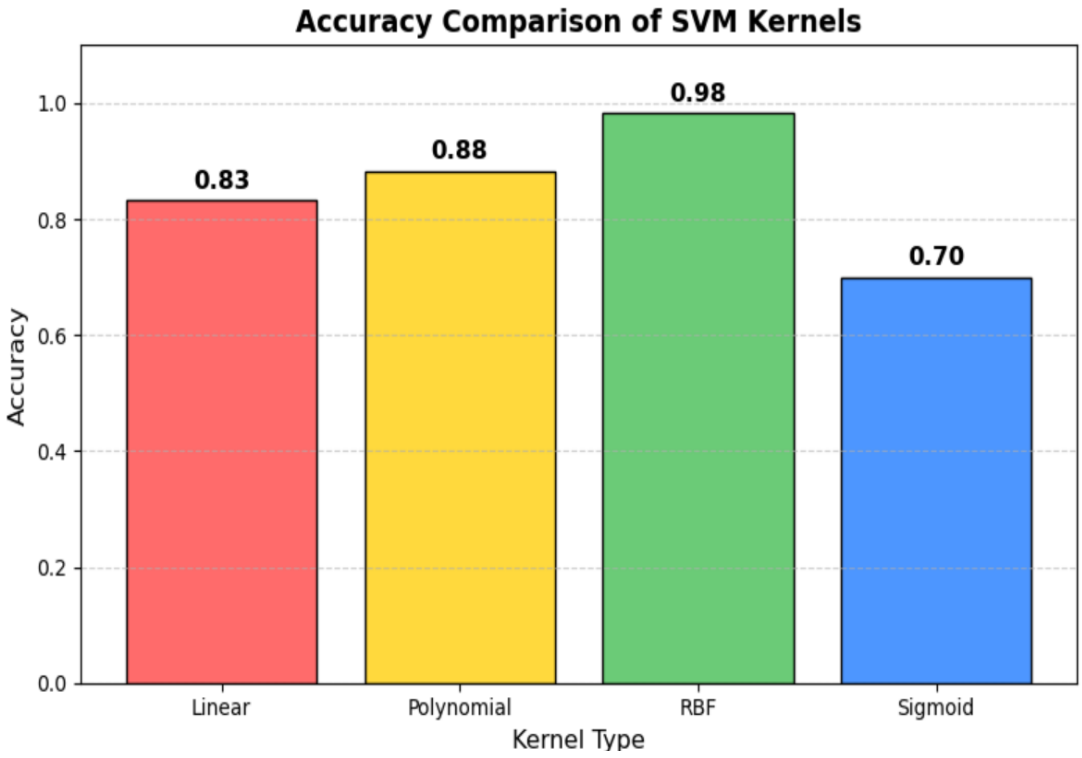
🔹 **Sigmoid Kernel**



- **Accuracy**: 0.70
- Decision boundary: irregular and inconsistent
- Not ideal for this problem

Accuracy Comparison

To better visualize the performance difference, I created a bar chart comparing the accuracy of each model:



- RBF performed the best
- Polynomial was good
- Linear was okay
- Sigmoid underperformed

Choosing the Right Kernel

The choice of kernel can make a big difference in how well your SVM performs.

- Use **Linear** when your data is clearly separable with a straight line
- Use **Polynomial** when the separation involves curves but isn't overly complex
- Use **RBF** when you're not sure—it works well in most cases
- Use **Sigmoid** if you're doing experiments related to neural networks

Don't forget to tune **C** (regularization) and **gamma** (for RBF) to get the best results. You can use tools like GridSearchCV to automate this.

Conclusion

This tutorial showed how different SVM kernels behave on a simple but non-linear dataset. The results make it clear that the **RBF kernel** is a great choice when you're dealing with complex patterns.

Understanding how each kernel works and being able to visualize their effects helps not just in choosing the right model—but in building better machine learning intuition.

References

1. Cortes, C. & Vapnik, V. (1995). *Support-vector networks.*
2. Cristianini, N. & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines.*
3. Jakkula, V. (2006). *Tutorial on SVM.*
4. Scikit-learn Documentation: https://scikit-learn.org/stable/modules/svm.html
5. Tharwat, A. (2019). *SVM Parameter Investigation.*

GitHub Link

https://github.com/manasichavan08/svm-kernels-make-moons

Includes:

- Jupyter notebook
- Code for all four kernels
- Visualizations and plots
- README and License

Accessibility Notes

This tutorial includes the following accessibility features:

- Colorblind-safe color palette (viridis) used in all plots
- Clear font sizes and high contrast text
- Headings used for screen reader compatibility
- Alt-text recommended for plots when exporting to PDF
- Code is modular and commented for readability
- GitHub version is compatible with online renderers like Colab and Binder