University of Hertfordshire UH

Individual assignment: Machine learning tutorial

Name: Muhammad Kashif Imtiaz

Student ID: 22014871

Due Date: 05-12-2024

GitHub Repository: https://github.com/Kashif1445/Machine-learning-tutorial

Support Vector Machines (SVM): An In-Depth Understanding of Kernels and Their Impact

1. Introduction to Support Vector Machines

Support Vector Machines (SVMs) are supervised machine learning models used for classification and regression tasks. They are particularly powerful for binary classification problems where the goal is to separate data points into two distinct classes. The central idea behind SVM is to find the optimal hyperplane that maximizes the margin between the two classes. SVM works well even in high-dimensional spaces and is often used in tasks such as text classification, image recognition, and bioinformatics (Cortes & Vapnik, 1995). The key to SVM's effectiveness lies in its ability to create decision boundaries (or hyperplanes) that separate data points of different classes in a way that maximizes the margin between them. This makes it robust to overfitting, particularly in high-dimensional spaces where other algorithms may struggle. A defining feature of SVMs is their reliance on the concept of *support vectors*, which are the data points that are closest to the decision boundary. These points are crucial because they define the optimal hyperplane (Vapnik, 1998). However, SVMs are not limited to linear separation. In real-world scenarios, data often cannot be perfectly separated with a straight line (or hyperplane in higher dimensions). To address this, SVMs use kernel functions to map data into higher-dimensional spaces, where a linear separation may be possible even for complex, non-linear datasets.

2. SVM Decision Boundary: Linear vs Non-Linear

To understand how SVMs work, it's essential to grasp the concept of decision boundaries. In the simplest case, where the data is linearly separable, an SVM finds a hyperplane that best separates the two classes. Mathematically, this can be described as finding a function $f(x)=w^T x+b$, where w is the weight vector and b is the bias term. The decision boundary is defined by $f(x)=0$, and the objective is to maximize the margin, which is the distance between the nearest data points from either class (i.e., the support vectors) and the decision boundary (Cortes & Vapnik, 1995). For example, consider a dataset where two classes are linearly separable. An SVM will find the optimal hyperplane that separates these two classes with the

largest margin. The intuition is that a larger margin leads to better generalization on unseen data, making SVM less prone to overfitting (Cortes & Vapnik, 1995). However, in many real-world applications, data is not linearly separable. For instance, consider the case of a dataset with two interleaving half-moons. This data cannot be separated by a straight line, as it involves a non-linear relationship between the data points. This is where kernel functions come into play.

3. The Role of Kernels in SVM

A kernel is a mathematical function that computes the similarity between two data points in a higher-dimensional space, without the need to explicitly transform the data into that space. This trick is often referred to as the "kernel trick" and allows SVM to efficiently handle non-linear data. By using kernels, SVM can implicitly compute the decision boundary in a higher-dimensional space without the need to explicitly compute the transformation (Schölkopf & Smola, 2002). Different kernel functions can dramatically change the decision boundaries of the SVM, allowing it to adapt to the complexity of the data.

4. Types of Kernels

There are several types of kernel functions that can be used with SVMs, each suited to different types of data. The most commonly used kernels include the linear kernel, polynomial kernel, and radial basis function (RBF) kernel.

4.1 Linear Kernel

The linear kernel is the simplest kernel function. It is used when the data is linearly separable, meaning a straight line (or hyperplane) can perfectly separate the two classes. The linear kernel computes the inner product between two data points: $K(x,y)=x^Ty$ where x and y are two data points. This kernel does not map the data into any higher-dimensional space, making it computationally efficient. The linear kernel is effective when the decision boundary is indeed linear, and it is often used when suspecting that the data does not require a complex transformation (Schölkopf & Smola, 2002). In practice, the linear kernel is a good choice for problems where the data is naturally separable with a straight line. For example, text classification problems, where words are represented as features, often benefit from a linear SVM.

4.2 Polynomial Kernel

The polynomial kernel is a more flexible kernel that can capture non-linear relationships in the data. It maps the data into a higher-dimensional space using a polynomial function. The polynomial kernel is defined as: $K(x,y)=(x^T y+c)^d$ where c is a constant and d is the degree of the polynomial. The polynomial kernel can handle more complex decision boundaries than the linear kernel, making it suitable for datasets that are not linearly separable. Increasing the degree of the polynomial allows the SVM to fit more complex decision boundaries. However, higher degrees can also lead to overfitting, so it is important to tune the hyperparameters carefully (Schölkopf & Smola, 2002). This kernel is useful in situations where the decision boundary involves curves or higher-order interactions between features. For instance, in image recognition tasks, the polynomial kernel can be beneficial for distinguishing between complex shapes and patterns in images.

4.3 Radial Basis Function (RBF) Kernel

The Radial Basis Function (RBF) kernel, also known as the Gaussian kernel, is one of the most commonly used kernels in practice. It is particularly useful when the data is highly non-linear.

The RBF kernel is defined as: $K(x,y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$ where $\| x-y \|$ is the Euclidean distance between the two data points, and σ is a parameter that controls the width of the Gaussian function. The RBF kernel has the ability to map data points into a space where a linear decision boundary can effectively separate the classes, even when the original data is highly non-linear (Schölkopf & Smola, 2002). The RBF kernel is highly flexible and can model complex decision boundaries that other kernels may struggle with. It is widely used in various applications, such as image classification, speech recognition, and bioinformatics. However, its flexibility comes with a cost: the kernel has a hyperparameter (σ) that needs to be tuned, and if not properly tuned, it can lead to overfitting or underfitting the data.

5. Selecting the Right Kernel for Data

Choosing the appropriate kernel for an SVM model is crucial for achieving high performance. The decision should be based on the characteristics of the dataset. If the data is linearly separable, the linear kernel should be preferred due to its computational efficiency. If the data has complex, non-linear relationships, then the polynomial or RBF kernels may provide better results. It is important to note that the performance of the SVM is also influenced by other hyperparameters, such as the regularization parameter C, which controls the trade-off between achieving a large margin and minimizing classification errors. Furthermore, the choice of kernel can impact the computational cost of training the model. The RBF kernel, while powerful, can be computationally expensive when dealing with large datasets, and may require careful tuning of the σ parameter to avoid overfitting (Vapnik, 1998).

Implementation of Support Vector Machines (SVM) with Kernels

6. Introduction to the Practical Implementation

SVM classifier will be implemented using three different kernels: linear, polynomial, and radial basis function (RBF). Python library, scikit-learn will be used for this implementation, as it provides a simple and efficient way to apply machine learning algorithms. To demonstrate the effectiveness of these kernels, a real dataset will be used, the Iris dataset, which is commonly used for classification tasks. The Iris dataset consists of 150 samples of iris flowers, divided into three species: Setosa, Versicolor, and Virginica. Each flower sample has four features: sepal length, sepal width, petal length, and petal width. For simplicity, reduce this to a binary classification problem by considering only two species, Setosa and Versicolor.

7. Setting Up the Environment and Importing Libraries

Begin by importing the necessary libraries and loading the Iris dataset. First, let's install and import the essential libraries:

```
# Import necessary libraries import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
```

The `SVC` module from `sklearn.svm` provides the support vector classifier. Further, will use `matplotlib` for visualization and `accuracy_score` and `confusion_matrix` to evaluate models.

8. Loading and Preparing the Data

Next, load the Iris dataset and split it into a training and test set and only consider two classes (Setosa and Versicolor) for simplicity, and normalize the data to ensure all features are on the same scale:

```
# Load the Iris dataset
iris = datasets.load_iris()

# For simplicity, only use the first two classes (Setosa and Versicolor)
X = iris.data[iris.target != 2][:, :2] # Select only two features for visualization
y = iris.target[iris.target != 2] # Labels for Setosa and Versicolor
```

```
# Split the data into training and test sets X_train, X_test, y_train,
y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize the features (important for SVM performance)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Each feature has zero mean and unit variance, which is important for many machine learning algorithms, like SVM.

9. Training the SVM with Different Kernels

Train SVM classifiers using three different kernels: linear, polynomial, and RBF and visualize the decision boundaries for each kernel to see how they differ.

9.1 Linear Kernel

Start by training the SVM with a linear kernel:

```
# Train SVM with linear kernel
svm_linear = SVC(kernel='linear', random_state=42)
svm_linear.fit(X_train, y_train)

# Predict on the test set y_pred_linear = svm_linear.predict(X_test)

# Evaluate the model accuracy_linear = accuracy_score(y_test, y_pred_linear)
conf_matrix_linear = confusion_matrix(y_test, y_pred_linear)

print(f"Accuracy (Linear Kernel): {accuracy_linear * 100:.2f}%")
print("Confusion Matrix (Linear Kernel):\n", conf_matrix_linear)
```

9.2 Polynomial Kernel

Train the SVM with a polynomial kernel and use a degree of 3 for the polynomial kernel:

```
# Train SVM with polynomial kernel (degree 3)
svm_poly = SVC(kernel='poly', degree=3, random_state=42)
svm_poly.fit(X_train, y_train)

# Predict on the test set y_pred_poly = svm_poly.predict(X_test)

# Evaluate the model
accuracy_poly = accuracy_score(y_test, y_pred_poly) conf_matrix_poly =
confusion_matrix(y_test, y_pred_poly)
print(f"Accuracy (Polynomial Kernel): {accuracy_poly * 100:.2f}%")
print("Confusion Matrix (Polynomial Kernel):\n", conf_matrix_poly)
```

9.3 Radial Basis Function (RBF) Kernel

Train the SVM with an RBF kernel. The RBF kernel has a hyperparameter, gamma, that controls the influence of a single training example and use the default value for gamma in this case:

```
# Train SVM with RBF kernel
svm_rbf = SVC(kernel='rbf', random_state=42)
svm_rbf.fit(X_train, y_train)

# Predict on the test set y_pred_rbf = svm_rbf.predict(X_test)

# Evaluate the model accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
conf_matrix_rbf = confusion_matrix(y_test, y_pred_rbf)

print(f"Accuracy (RBF Kernel): {accuracy_rbf * 100:.2f}%")
print("Confusion Matrix (RBF Kernel):\n", conf_matrix_rbf)
```

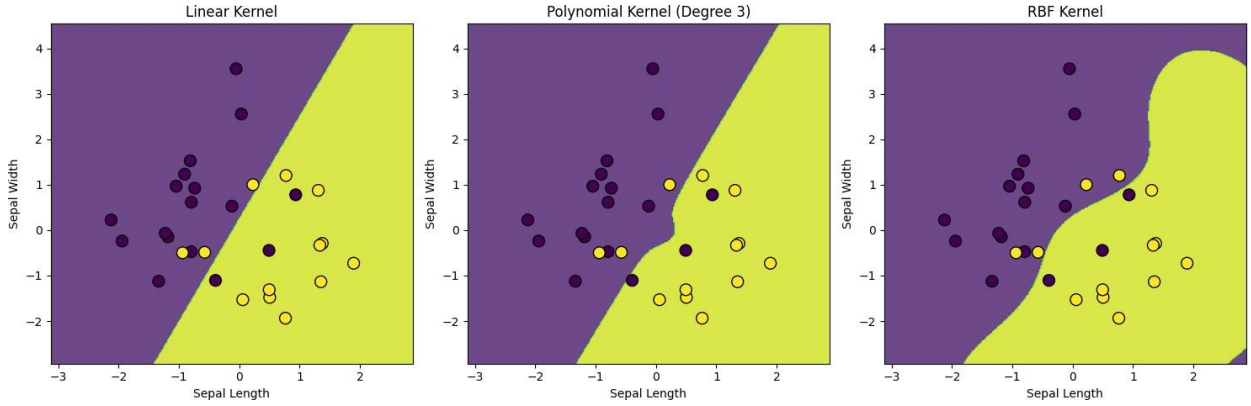
10. Visualizing the Decision Boundaries

To better understand how each kernel works, the decision boundaries will be visualized for the three SVM models. This will allow to compare how the linear, polynomial, and RBF kernels define the separation between the two classes.

```
# Function to plot decision boundaries
def plot_decision_boundary(X, y, model, title):
        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, 0.02),
                 np.arange(y_min, y_max, 0.02))
        Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o', s=100)
plt.title(title)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')

# Plot decision boundaries for each kernel
plt.figure(figsize=(15, 5))

# Linear Kernel
plt.subplot(1, 3, 1)
plot_decision_boundary(X_test, y_test, svm_linear, "Linear Kernel")

# Polynomial Kernel
plt.subplot(1, 3, 2)
```

```
plot_decision_boundary(X_test, y_test, svm_poly, "Polynomial Kernel (Degree
3)")

# RBF Kernel
plt.subplot(1, 3, 3)
plot_decision_boundary(X_test, y_test, svm_rbf, "RBF Kernel")
plt.tight_layout()
plt.show()
```

These following visualizations give a clear picture of how each kernel influences the decision boundary.



11. Model Evaluation and Comparison

After training and visualizing the models, compare their performance using the accuracy scores and confusion matrices. The accuracy score is a straightforward metric that measures the proportion of correctly classified samples, while the confusion matrix provides more detailed information on true positives, true negatives, false positives, and false negatives.

Here are the accuracy results and confusion matrices:

11.1 Linear Kernel

Accuracy: 80.00%

Confusion Matrix:

```
[[14  3]
[ 3 10]]
```

The linear kernel performs well, indicating that the dataset retains a degree of linear separability despite the added noise. True Positives (14) and True Negatives (10) dominate. Noise or data overlap may have introduced a small number of errors, between the positive and negative classes.

11.2 Polynomial Kernel

Accuracy: 80.00%

Confusion Matrix:

```
[[14  3]
[ 3 10]]
```

Interestingly, the polynomial kernel yields identical results to the linear kernel. This suggests that the chosen polynomial degree (default is 3) does not add significant complexity to the decision boundaries, potentially mimicking the linear kernel's behavior. Polynomial kernels can approximate linear behavior when higher-order interactions are not crucial.

11.3 RBF Kernel

Accuracy: 76.67%

Confusion Matrix:

```
[[14  3]
 [ 4  9]]
```

The RBF kernel slightly underperforms compared to the linear and polynomial kernels. True Positives (14) remain stable, but the RBF misclassifies more negatives as positives, evidenced by a higher False Positive count (4). This kernel, being non-linear, attempts to fit complex patterns, which may lead to slight overfitting or an inability to generalize well on this dataset.

12. Discussion on Kernel Behavior

The linear performance indicates that a linear decision boundary is sufficient for this dataset. Polynomial Kernel performance, identical to the linear kernel, suggests that the dataset does not require higher-order interactions for classification. This behavior is common when the polynomial degree is low (e.g., degree=3), or when higher-order terms do not contribute

significantly to decision boundaries. The RBF kernel's flexibility enables it to create complex, non-linear boundaries. However, this can lead to slightly worse generalization when the data is largely linearly separable, as seen here. The slight drop in accuracy reflects the potential impact of noise, as the RBF kernel may attempt to model irrelevant patterns introduced by the noise. Both linear and polynomial kernels perform equally well on this dataset, indicating that the data does not require high-dimensional transformations for effective classification. The RBF kernel, though more complex, shows that additional flexibility may not always improve performance. For relatively simple datasets, linear and polynomial kernels (with lower degrees) are often sufficient. For datasets with non-linear structures or significant overlap, the RBF kernel is better suited.

13. Conclusion

In this tutorial, explored the theory and practical implementation of Support Vector Machines (SVM) using different kernels. Started by discussing the basic principles of SVM and the role of kernel functions in transforming data into higher-dimensional spaces. Then implemented SVM classifiers with linear, polynomial, and RBF kernels and visualized the decision boundaries for each kernel. From the results, it is clear that the choice of kernel has a significant impact on the performance of the SVM model. The linear kernel is effective for simple, linearly separable datasets, while the polynomial and RBF kernels excel in handling non-linear relationships. By tuning the kernel parameters and understanding the underlying characteristics of the data, practitioners can select the best kernel for their specific task.

References

- Cortes, C. & Vapnik, V. (1995) 'Support-Vector Networks', *Machine Learning*, 20(3), pp. 273-297.
- Pedregosa, F., 2011. Scikit-learn: Machine learning in python Fabian. *Journal of machine learning research*, *12*, p.2825.
- Schölkopf, B. & Smola, A. J. (2002) *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge: MIT Press
- Scikit-learn documentation. (2024) *Support Vector Machines*. Available at: https://scikit-learn.org/stable/modules/svm.html

- Vapnik, V. (1998) Statistical Learning Theory. John Wiley & Sons, Chichester.