SVM

Karol Korszun

May 2024

1 Introduction

Support Vector Machine (SVM) is a supervised learning algorithm used for classification tasks. It's particularly effective in high-dimensional spaces and is widely used in applications like image classification, bioinformatics, and text categorization.

How SVM Works

1. **Separating Hyperplane**: SVM finds the optimal hyperplane that best separates the data points into different classes. This hyperplane is chosen such that the margin, i.e., the distance between the hyperplane and the closest data points (support vectors), is maximized.

2. **Kernel Trick**: SVM can efficiently handle non-linear decision boundaries by using the kernel trick. This involves transforming the input data into a higher- dimensional space, where a linear hyperplane can be used to separate the classes.

3. **Margin Maximization**: SVM aims to maximize the margin, which leads to better generalization against overfitting.

2 Impact of Hyperparameters on Model Performance

Different combinations of learning rates and γ parameters were used in the experiments. The kernel used was the regular kernel given by the equation:

$$K(x_i, x_j) = \langle x_i, x_j \rangle \tag{1}$$

where x_i and x_j are input data points. Lambda (λ) is a regularization parameter in the SVM model, controlling the balance between minimizing the training error and minimizing model complexity. It helps prevent overfitting by penalizing overly complex decision boundaries, promoting a simpler and more generalizable

solution. The learning rates and λ parameters were varied to observe their effect on the accuracy of the SVM model.

In the following Python code snippet, we can see where parameters γ and the learning rate of the SVM model where implemented in code.

```
for idx, x_i in enumerate(X):
    condition = class_labels[idx] * (self.kernel(self, x_i, self.w)
     - self.b) >= 1
    if condition:
        self.w -= self.learning_rate * (2 * self.lambda_param *
    self.w)
    else:
        self.w -= self.learning_rate * (2 * self.lambda_param *
    self.w - np.dot(x_i, class_labels[idx]))
        self.b -= self.learning_rate * class_labels[idx]
```

Listing 1: Updating weights

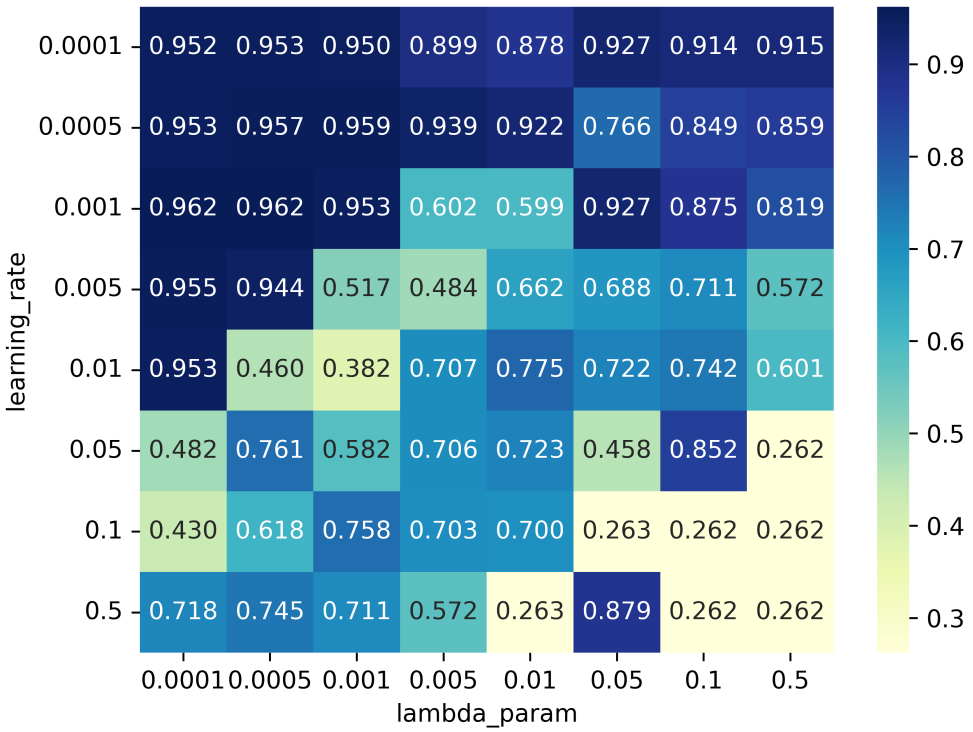

Figure 1: Effect of Hyperparameters on Model Accuracy

It can be observed that smaller parameters of learning rate and λ tend to

produce better results, however it is important to note that there exist some outliers for example a learning rate equalt to 0.5 and $\lambda = 0.05$

3 Different kernel comparison

In this section, the performance of two different kernels, namely the linear and polynomial kernels, is compared, and their impact on the accuracy of the SVM model is analyzed. Learning rate of 0.005 and $\lambda = 0.005$ were used for following experiments.

3.1 Linear Kernel

Linear kernel is given by the following equation

$$K(x, y) = \langle x, y \rangle + c \tag{2}$$

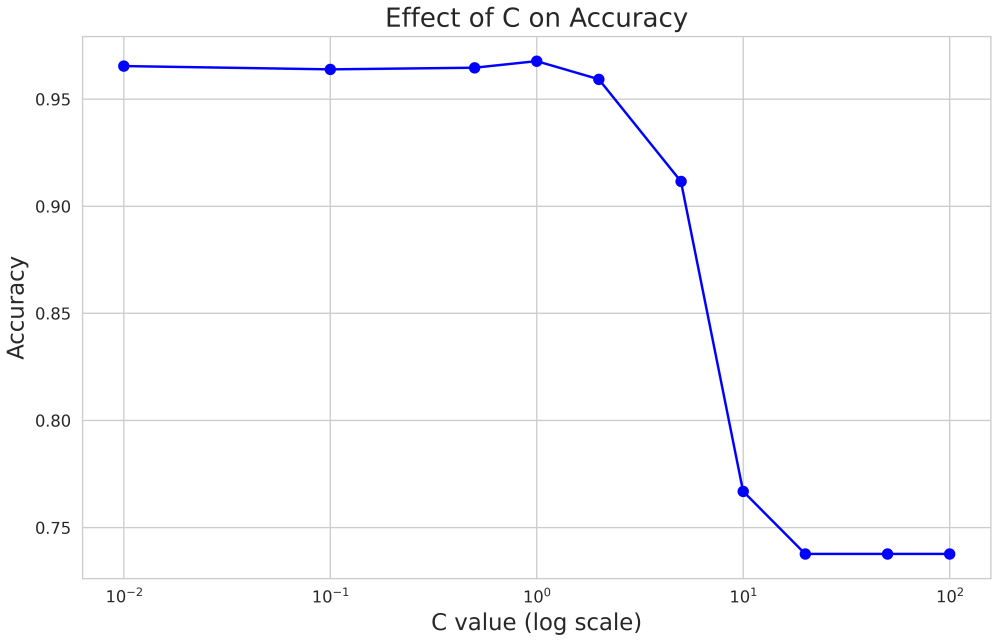

Figure 2: Effect of C on Model Accuracy

As demonstrated in the plot, increasing the C hyperparameter has a detrimental effect on the overall model accuracy, as it does not significantly improve accuracy beyond the baseline level

3.2 Polynomial Kernel

Polynomial kernel is given by the equation

$$K(x, y) = (\gamma\langle x, y\rangle + r)^d \tag{3}$$

In order to facilitate easy visualization r parameter has been set to 0.

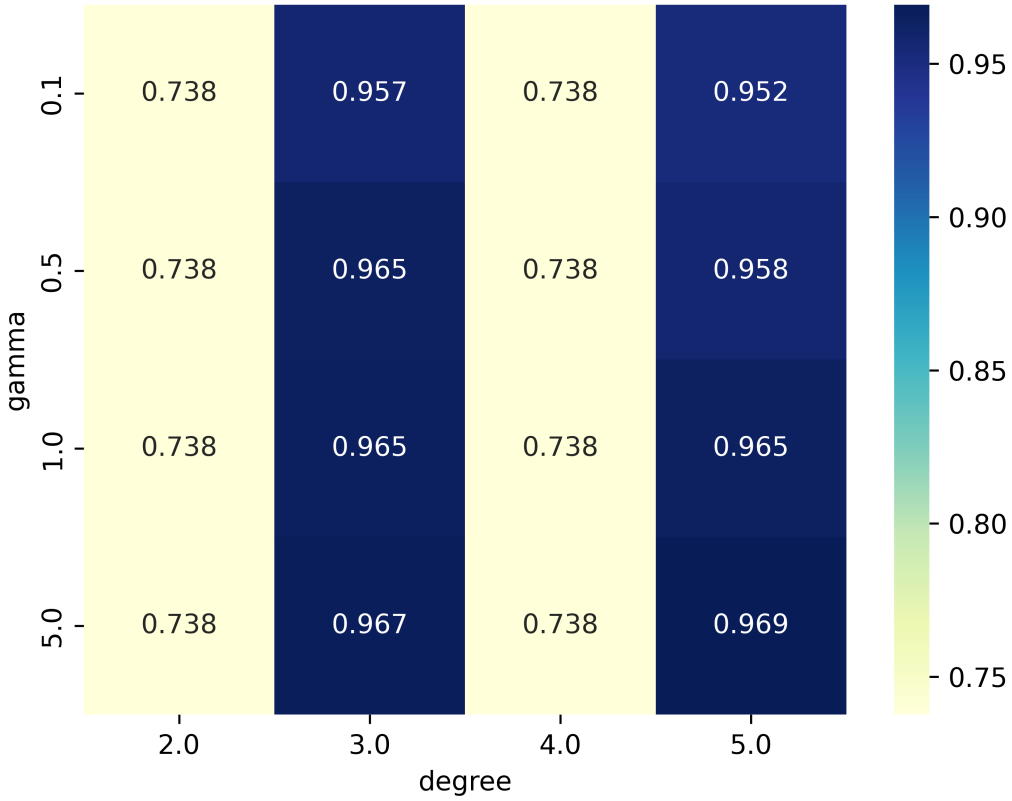

Figure 3: Effect of C on Model Accuracy

The results from the polynomial kernel experiments, demonstrated that the model's accuracy was optimal for polynomial degrees 3 and 5. However, for other polynomial degrees, the accuracy significantly declined. However, it is noteworthy that even with the optimal polynomial degrees of 3 and 5, the performance observed did not notably surpass the baseline accuracy.

4 Comparison with Logistic Regression

Metric	Logistic Regression	SVM
Accuracy	0.9903498190591074	0.9758745476477684
Precision	0.9757575757575757	0.9607843137254902
Recall	0.9615384615384616	0.9423076923076923
F1 Score	0.9803921568627451	0.9514563106796117

Table 1: Performance metrics for the Logistic Model and SVM

A look into a confusion matrix for both models may provide a helpful insight as well. A confusion matrix is a performance evaluation tool used in machine learning to visualize the performance of a classification model by presenting the counts of true positive, true negative, false positive, and false negative predictions across different classes. The confusion matrix for the SVM model reveals

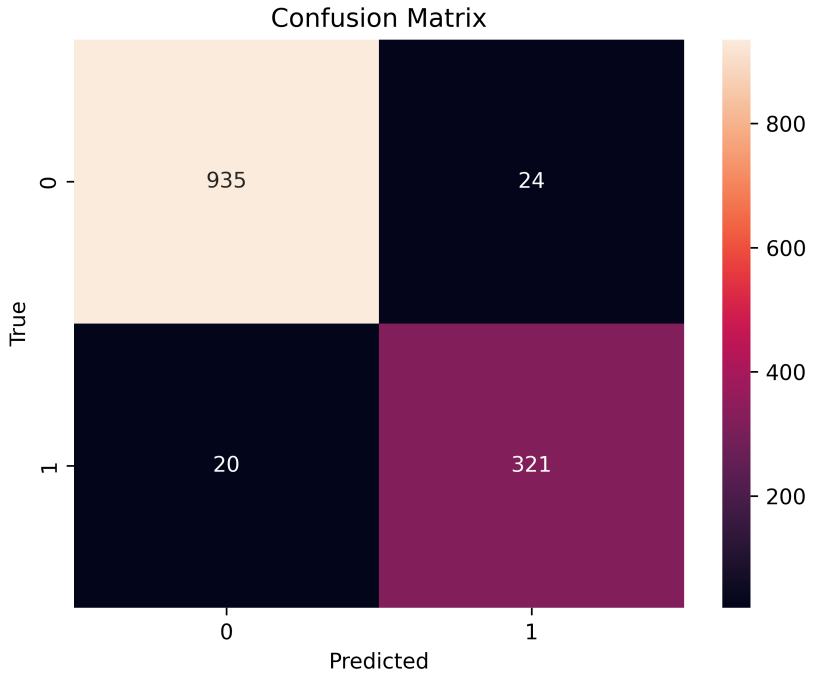

Figure 4: SVM confusion matrix

a relatively balanced distribution of true positives and true negatives across the classes. In the confusion matrix for logistic regression, the number of false negative is slightly higher than the number of false positives, indicating that the

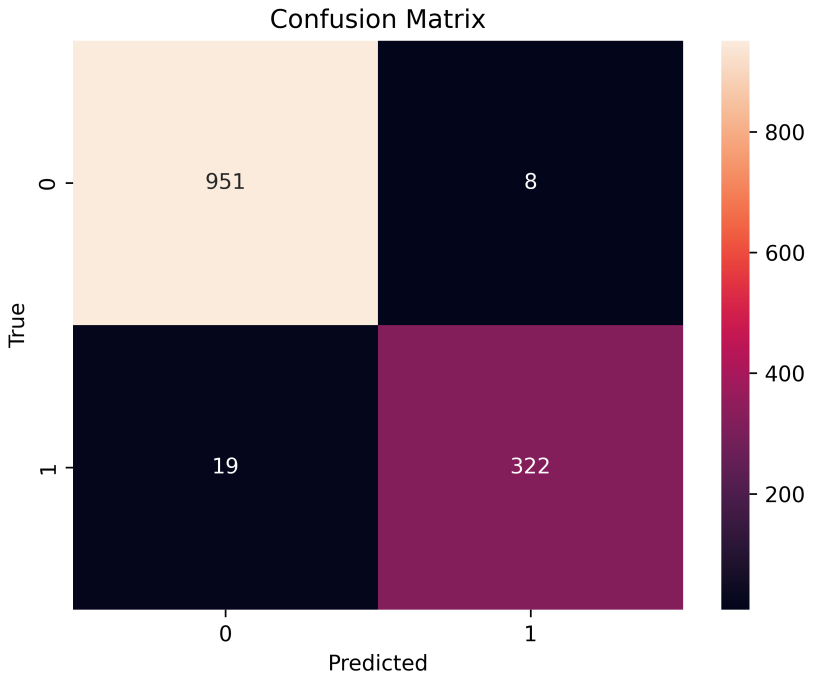

Figure 5: Logistic Regression confusion matrix

model is more prone to incorrectly predicting negative instances as positive.

5 Conclusions

Despite extensive experimentation with different hyperparameters, the impact on accuracy was challenging to observe due to the high baseline performance level of SVM model.

Moreover, the comparison between different kernels highlighted nuanced performance differences, with the polynomial kernel showing optimal accuracy at degrees 3 and 5. However, even with these optimal degrees, the models failed to notably surpass the baseline accuracy.