Demystifying Support Vector Machines: Kernel Machines

Rishabh Iyer

March 19, 2024

1 Introduction

This is the second blog article in the Support Vector Machine series. In the previous blog article (**CITE**), we introduced support vector machines, and argued that SVMs focus on finding the hyperplane that maximizes the margin between the classes, which is fundamental to their robustness and generalization capability. We also contrasted support vector machines with perceptrons, which merely seek any separating hyperplane without considering the quality of the separation.

In this blog article, we will highlight one more very powerful feature of support vector machines, which is it enables us to use kernels. SVMs employ the kernel trick to project data into higher-dimensional spaces, enabling the separation of classes that are not linearly separable in the original feature space. This feature allows SVMs to construct complex, non-linear decision boundaries without explicitly computing the high-dimensional transformations.

2 Deep Dive into SVMs: Kernel SVMs

Kernel SVMs extend the linear SVM framework to handle non-linearly separable data by mapping input features into a higher-dimensional space where a linear separation is possible. This section explores the model's choice of kernels, the formulation of the dual problem, and the optimization challenges. Since the data preperation (step 1) is the same, we do not repeat it here.

2.1 Step 2: Model – Choice of Kernels

The kernel function in SVMs allows the algorithm to operate in a high-dimensional feature space without explicitly computing the coordinates in that space, thus avoiding the curse of dimensionality. Common choices for kernel functions include:

- **Linear Kernel:** $K(x, x') = x^T x'$. This kernel does not actually transform the data and is equivalent to the standard linear SVM.

- **Polynomial Kernel:** $K(x, x') = (\gamma x^T x' + r)^d$, where d is the degree of the polynomial, γ is a scale factor, and r is a constant term.
- **Radial Basis Function (RBF) Kernel:** $K(x, x') = \exp(-\gamma \|x - x'\|^2)$, where γ is a scale factor. The RBF kernel is particularly popular due to its flexibility in handling non-linear relationships.
- **Sigmoid Kernel:** $K(x, x') = \tanh(\gamma x^T x' + r)$, which resembles the activation function used in neural networks.

The choice of kernel depends on the data and the specific problem. The RBF kernel is often a good first choice due to its versatility.

2.2 Step 3: Formulation – Dual Problem and Kernel Extension

Recall that the original SVM optimization problem (see this **CITE** blog) is given as follows:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq 1 \text{ for all } i = 1, \cdots, M \end{aligned} \tag{1}$$

Using the concept of Dualuty (**CITE**), the original SVM can be reformulated into its dual form, which is as follows.

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)}.x^{(j)} \\ &\text{subject to } 0 \leq \alpha_i \leq C \text{ for all } i, \text{ and } \sum_{i=1}^{M} \alpha_i y^{(i)} = 0. \end{aligned} \tag{2}$$

where α_i are the Lagrange multipliers, C is the penalty parameter for the soft margin. For more details on the concept of duality and why the above is the dual formulation of SVMs, see this blog article on duality **CITE**.

The above dual expression, can be easily extended if we replace $x^{(i)}.x^{(j)}$ with a more general kernel function: $K(x^{(i)}, x^{(j)})$. The formulation is:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)}) \\ &\text{subject to } 0 \leq \alpha_i \leq C \text{ for all } i, \text{ and } \sum_{i=1}^{M} \alpha_i y^{(i)} = 0. \end{aligned} \tag{3}$$

Equation (3) is the same as Equation (w) whe K is the linear kernel. This dual formulation allows the SVM to incorporate the kernel trick, enabling the algorithm to learn non-linear decision boundaries by implicitly mapping inputs into a high-dimensional space.

2.3 Step 4: Optimization

The optimization problem in the dual formulation involves finding the set of α_i that maximizes the dual objective function. This is typically achieved using optimization techniques suited for constrained problems, such as Sequential Minimal Optimization (SMO), which breaks the problem into smaller, more manageable optimization problems that can be solved analytically.

2.4 Step 5: Inference and Evaluation

The evaluation is mostly the same for Kernal SVMs and Linear SVMs. The main difference is the inference step which we detail below.

Once a Kernel SVM model has been trained, the learned parameters consist of the Lagrange multipliers $(\alpha_i, i = 1, \cdots, M)$. These elements are used to construct the decision function that predicts the class of new test examples. Here's how the inference process works:

The test prediction for a Kernel SVM is given by:

$$y_{\text{pred}} = \text{sign}\left(\sum_{i=1}^{M} \alpha_i y^{(i)} K(x^{(i)}, x_{\text{test}}) + b\right)$$

where:

- x_{test} is a new test example for which we want to predict the label.
- $x^{(i)}$ are the support vectors, which are the training examples that have non-zero α_i.
- $y^{(i)}$ are the labels of the support vectors.
- $K(x^{(i)}, x_{\text{test}})$ is the kernel function applied between the support vector $x^{(i)}$ and the test example x_{test}.
- b is the bias term, which is determined using complementary slackness (**TODO: CITE Blog on Duality**. To recap, we take the points i which are support vectors: in that case, $\alpha_i > 0$. Using complementary slackness, $b = y^{(i)} - w^T x^{(i)} = b = y^{(i)} - \sum_{j=1}^{M} \alpha_i y^{(i)} K(x^{(i)}, x^{(j)})$. To obtain a more robust estimate of b, it is common practice to average b over all support vectors on the margin:

$$b = \frac{1}{|S|} \sum_{i \in S} \left(y^{(i)} - \sum_{j=1}^{M} \alpha_j y^{(j)} K(x^{(i)}, x^{(j)})\right)$$

where S denotes the set of indices of support vectors on the margin.

3 Decision Boundaries in Kernel SVMs

Kernel Support Vector Machines (SVMs) extend the linear SVM framework to handle non-linearly separable data by employing kernel functions. These functions implicitly map input features into a higher-dimensional space where a linear separation is possible. The choice of kernel significantly influences the shape and complexity of the decision boundary that SVMs can model. Below, we discuss several common kernels and the characteristics of the decision boundaries they produce.

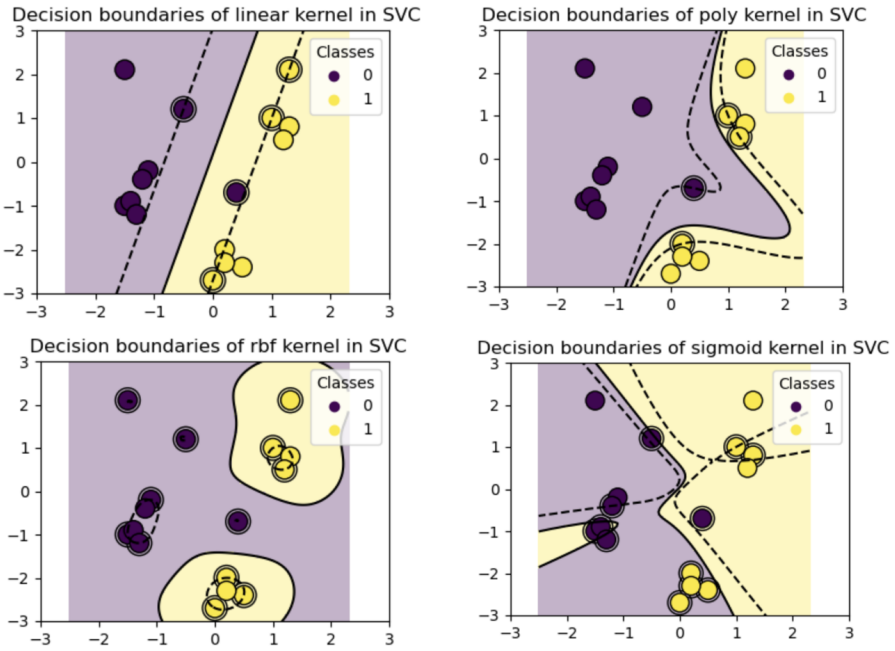

Figure 1: Illustrating the behavior of different kernels on a simple binary classification dataset which is not linearly separable. The figure is from `https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html`

Linear Kernel: The linear kernel is the simplest form, where no explicit mapping to a higher-dimensional space is performed. It is suitable for linearly separable data. The top right of Figure 1 shows the linear SVM model decision boundary. For the dataset considered in Figure 1, it does not perfectly separate the classes since the dataset is not linearly separable.

Polynomial Kernel: The polynomial kernel allows for the modeling of curved decision boundaries by raising the input features to a specified power.

The degree of the polynomial determines the flexibility of the decision boundary. Using a polynomial kernel in SVM gives a decision boundary shown in the top right of Figure 1. As expected, polynomial kernel is the same as doing a linear SVM with polynomial feature transformations (though this would be much more computationally expensive). The figure shows a degree 3 polynomial which perfectly separates the two classes.

Radial Basis Function (RBF) Kernel: The RBF kernel, also known as the Gaussian kernel, is highly flexible and can model complex decision boundaries by considering the distance between feature points in the input space. It is particularly effective for datasets where the separation between classes is not linear or polynomial. The RBF kernel is $K(x_1, x_2) = \exp(-\gamma\|x_1 - x_2\|^2$ and γ controls the influence of each individual training sample on the decision boundary. The larger the euclidean distance between two points, the closer the kernel function is to zero. Figure 1 bottom right shows the decision boundary obtained using the RBF kernel. In the plot we can see how the decision boundaries tend to contract around data points that are close to each other.

Sigmoid Kernel: The sigmoid kernel, inspired by the activation function used in neural networks, can produce complex, non-linear decision boundaries. However, its use is less common due to the potential for non-convex decision regions. The bottom right of Figure 1 shows the Sigmoid kernel. We can see that the decision boundaries obtained with the sigmoid kernel appear curved and irregular. The decision boundary tries to separate the classes by fitting a sigmoid-shaped curve, resulting in a complex boundary that may not generalize well to unseen data. From this example it becomes obvious, that the sigmoid kernel has very specific use cases, when dealing with data that exhibits a sigmoidal shape. In this example, careful fine tuning might find more generalizable decision boundaries. Because of it's specificity, the sigmoid kernel is less commonly used in practice compared to other kernels.

4 Summary of Kernel SVMs

Kernel SVMs represent a powerful extension of the linear SVM model, capable of handling complex, non-linearly separable datasets by leveraging the kernel trick. The choice of kernel function plays a critical role in the model's ability to capture the underlying patterns in the data, while the dual formulation and optimization techniques ensure that the model can be efficiently trained, even in high-dimensional feature spaces.

The decision boundary of a Kernel SVM is profoundly influenced by the choice of kernel function. Linear kernels are suited for linearly separable data, while polynomial and RBF kernels offer greater flexibility for complex datasets. The visualization of these decision boundaries provides intuitive insights into the behavior of Kernel SVMs across different types of data distributions.