Demystifying Support Vector Machines: Dealing with Non-Separable Data

Rishabh Iyer

March 19, 2024

1 Introduction

This is the third installment in our series on Support Vector Machines (SVMs). The initial article explored SVMs in the context of linearly separable data, while the second delved into the use of kernel methods to address non-linear yet separable datasets. In this piece, we turn our attention to the application of SVMs to non-separable datasets, going beyond the mere absence of linear separability to tackle datasets that defy separation by non-linear features without resorting to overfitting.

As highlighted in the preceding articles, SVMs address a critical limitation of Perceptrons by identifying the optimal classifier that offers superior generalization. Yet, Perceptrons face another significant challenge: their inability to function with non-separable datasets, coupled with a lack of convergence guarantees. In contrast, SVMs, with slight modifications, can gracefully manage non-separability.

In our first article on SVMs, we observed that their strategy centers on identifying a hyperplane that maximizes the margin between classes, a key factor in their durability and generalization strength. This method stands in stark contrast to Perceptrons, which aim for any separating hyperplane without regard for separation quality.

The second article revealed how SVMs leverage the kernel trick to map data into higher-dimensional spaces, facilitating the separation of classes that are inseparable in the original feature space. This capability enables SVMs to form intricate, non-linear decision boundaries without the need for direct computation of high-dimensional transformations.

This article will demonstrate how, through the introduction of slack variables, SVMs broaden their scope to encompass datasets where perfect separation is unattainable. Slack variables introduce a level of tolerance for misclassification or margin constraint violations, allowing SVMs to negotiate the balance between decision boundary complexity and misclassification extent. This modification renders SVMs exceptionally capable, even amidst noise and overlapping class distributions.

2 Deep Dive into SVMs for Non-Separable Data

When dealing with non-separable data, SVMs are adapted by introducing slack variables to allow for misclassification of training examples. This adaptation leads to the formulation of the hinge loss, which is central to the optimization of SVMs in non-separable cases. The Step 1 (obtaining the dataset and the features) are the same as the first two blog articles, so we directly start with Step 2 below.

2.1 Step 2: Linear Model and Polynomial Features

In non-separable datasets, the linear SVM model is extended by incorporating polynomial features or using kernel functions to project the data into a higher-dimensional space where a linear separation is possible. However, even with these extensions, perfect separation might not be achievable due to noise and overlaps in the data distribution. To accommodate this, slack variables (ξ_i) are introduced for each data point to measure the degree of misclassification.

2.2 Step 3: Slack Variables and Hinge Loss

The introduction of slack variables ξ_i in the SVM optimization problem allows for a certain degree of misclassification or violation of the margin constraints. This modification is crucial for handling non-separable data, where a strict separation between classes might not be possible without incurring some errors.

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M} \xi_i$$

subject to $y^{(i)}(w^T x^{(i)}+b) \geq 1-\xi_i$ and $\xi_i \geq 0$ for all i. Below are a few comments about the formulation:

- In the formulation above, C is a regularization parameter that controls the trade-off between maximizing the margin ($2/\|w\|$) and minimizing the misclassification ($\sum_{i=1}^{M} \xi_i$).
- This formulation aims to minimize the norm of the weight vector w (which maximizes the margin) while also minimizing the sum of the slack variables ξ_i, penalized by a regularization parameter C. The slack variables ξ_i measure the degree to which each data point $x^{(i)}$ is allowed to violate the margin constraint.
- Figure 1 illustrates the scenario very well. For any point where $\xi_i = 0$, it means the point is perfectly classified by the SVM and there is no penalty for those elements i. If ξ_i is between 0 and 1, it means that this is a margin violation, which means that it is on the correct side of the separating hyperplane (and correctly classified) but wrong side of the margin. If $\xi_i > 1$, it is misclassified. Whenever $\xi_i > 0$, the objective penalizes these misclassifications linearly.

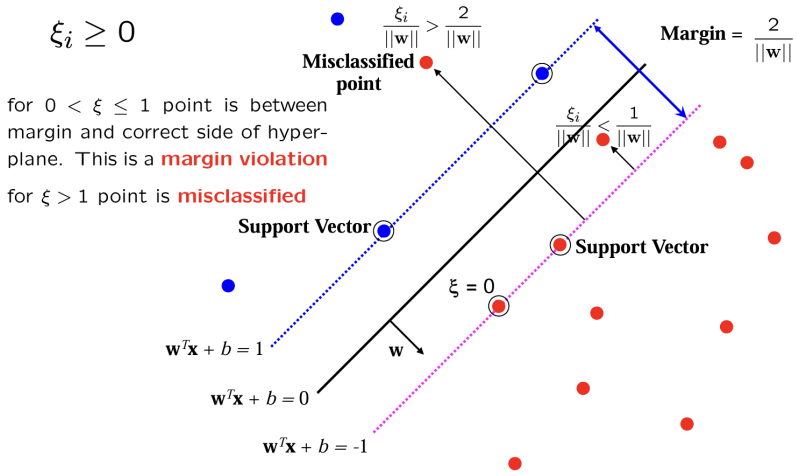

Figure 1: Illustrating SVMs with slack (see text for details)

The hinge loss emerges naturally from this formulation when considering the objective of minimizing the slack variables. Specifically, the slack variable ξ_i for each data point can be interpreted as the loss incurred by that point, leading to the hinge loss expression:

$$\xi_i = \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$$

Thus, minimizing the sum of the slack variables ξ_i is equivalent to minimizing the sum of the hinge loss across all data points. Incorporating the hinge loss directly into the optimization problem, we obtain the full loss function formulation for the SVM, which combines the regularization term (related to the margin size) and the hinge loss (related to classification errors):

$$\min_{w,b} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{M} \max(0, 1 - y^{(i)}(w^T x^{(i)} + b)) \right)$$

This loss function consists of two parts:

- The regularization term $\frac{1}{2}\|w\|^2$, which controls the complexity of the model by penalizing large weights and thereby encourages a larger margin.
- The hinge loss term $C \sum_{i=1}^{M} \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$, which penalizes misclassifications and margin violations, with the regularization parameter C balancing the trade-off between margin size and classification error.

The Hinge loss is very similar to the perceptron loss (**CITE Blog**). Figure 2 illustrates the difference between the 0/1 Loss, the Perceptron loss and the Hinge loss.

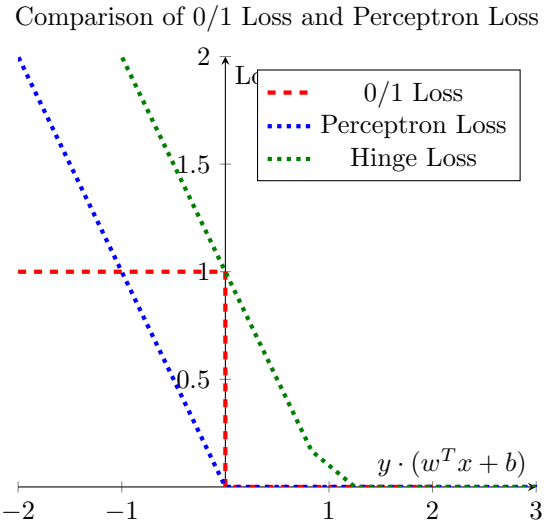

Figure 2: Comparison of 0/1 Loss, Perceptron Loss, and Hinge Loss.

2.3 Step 4: Optimization with Hinge Loss

The hinge loss function for SVMs is defined as:

$$L_{\text{hinge}}(w, b) = \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$$

This loss function penalizes misclassified points and points within the margin, encouraging the model to correctly classify all points with a margin greater than 1. The optimization of the hinge loss involves minimizing this function with respect to the model parameters w and b.

2.3.1 Sub-Gradients of the Hinge Loss

The hinge loss is piecewise linear and not differentiable at $1-y^{(i)}(w^T x^{(i)}+b) = 0$, which necessitates the use of subgradients for optimization. The subgradient of the hinge loss with respect to w can be expressed as follows:

- For correctly classified points where $1 - y^{(i)}(w^T x^{(i)} + b) \leq 0$, the function is flat, and the subgradient with respect to w is 0.
- For misclassified points or points within the margin ($1-y^{(i)}(w^T x^{(i)}+b) > 0$), the subgradient with respect to w is $-y^{(i)}x^{(i)}$, similar to the Perceptron loss case.

The subgradient with respect to b follows a similar logic, being 0 for correctly classified points outside the margin and $-y^{(i)}$ for points within the margin or

misclassified. The derivation of the sub-gradcients is similar to the Perceptron Loss using the chain rule (**TODO: CITE Perceptron Blog** so we do not go over the precise derivation. The readers are encouraged to derive the sub-gradients from scratch using the chain rule.

2.3.2 Subgradient Descent Algorithm for Hinge Loss

To minimize the hinge loss using subgradient descent, the algorithm iteratively updates the weights and bias based on the subgradient of the loss function at the current parameters. The update rules are:

1. Initialize w and b to zeros or small random values. 2. For each iteration, compute the subgradient of the hinge loss for each training example. Then, update w and b as follows:

$$w := w - \alpha \cdot g_w$$
$$b := b - \alpha \cdot g_b$$

where g_w and g_b are the subgradients of the hinge loss with respect to w and b, respectively, and α is the learning rate.

3. Repeat the update process until convergence criteria are met, such as a small change in the loss between iterations or reaching a maximum number of iterations.

The learning rate α can be fixed or adaptively changed over iterations. A common approach is to start with a larger α and decrease it over time to allow finer adjustments as the optimization process converges.

The optimization of SVMs with the hinge loss through subgradient descent provides a robust method for finding the optimal separating hyperplane, even in the presence of non-linearly separable data. By iteratively adjusting the model parameters in the direction that minimizes the hinge loss, SVMs effectively balance the trade-off between maximizing the margin and minimizing classification errors, leading to models with strong generalization capabilities.

2.4 Step 5: Evaluation and Inference

The inference is exactly the same as the Linear SVM case. Given the learnt model (w, b), and given a test data instance x_{test}, the predicted label is simply:

$$y_{\text{pred}} = \text{Sign}(w^T x_{\text{test}} + b) \tag{1}$$

After computing the predicted labels for every test data instance, we compare it to the true labels using the evaluation metrics Accuracy, Precision/Recall, and F1 Score (as outlined in the Linear SVM case).

3 The Role of C in SVM Hinge Loss Formulation

In the context of Support Vector Machines (SVMs) employing the hinge loss, the regularization parameter C plays a crucial role in determining the trade-off

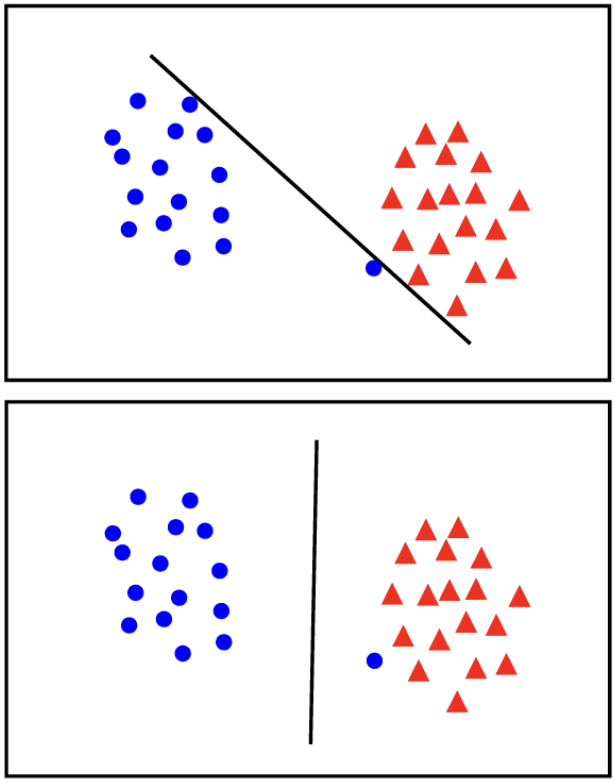

Figure 3: Illustrating the tradeoff of C. The top figure is obtained with a large value of C resulting in perfect classification with a small margin. The bottom figure is a model obtained with a small value of C resulting in lesser penalty for misclassification and a larger margin (better generalization).

between maximizing the margin and minimizing the classification error. Specifically, C controls the penalty imposed on observations that violate the margin constraint, thereby influencing the flexibility and complexity of the decision boundary.

With a *small* C, the model is less penalized for misclassifications, leading to a wider margin and a more simplistic model that may underfit the data. This setting prioritizes the maximization of the margin over the minimization of the classification error, potentially resulting in a model with higher bias but lower variance, favoring generalization overfitting to the training data. The bottom plot in Figure 3 illustrates a decision boundary obtained with a small C (larger margin and non-zero training error).

Conversely, a *large* C places a higher penalty on misclassifications, com-

pelling the model to fit the training data more closely. This can result in a narrower margin and a more complex decision boundary that intricately separates the classes, potentially at the expense of overfitting. A larger C value thus emphasizes the importance of correctly classifying all training observations, even if it means compromising the margin width, leading to a model with lower bias but potentially higher variance. The top plot in Figure 3 is obtained with a large value of C resulting in a very small margin and zero training error.

In summary, the choice of C in SVMs with hinge loss formulation is pivotal in balancing the bias-variance trade-off. Selecting an appropriate C value is essential for achieving a model that generalizes well to unseen data, necessitating careful consideration and possibly hyperparameter tuning based on cross-validation performance.

4 Summary of SVMs with Non-Separable Data

SVMs for non-separable data leverage slack variables and the hinge loss to create a flexible model that can handle overlaps and noise in the data. By optimizing the hinge loss, SVMs find a decision boundary that effectively balances the trade-off between the complexity of the model (as controlled by the margin size) and its accuracy on the training data. This makes SVMs a powerful tool for classification tasks, even in challenging scenarios where data is not linearly separable.