Cyberthreat detection using ML

Support Vector Machine (SVM)

SVM is a powerful supervised learning algorithm used for both classification and regression tasks. It works by finding the optimal hyperplane that best separates different classes in a high-dimensional space. SVM is widely used due to its effectiveness in high-dimensional spaces and its ability to handle both linear and non-linear classification problems through kernel tricks.

Mathematical Formulation

For a binary classification problem:
Given training data

$$\{(x_i, y_i)\}_{i=1}^{N}$$

where

$$x_i \in \mathbb{R}^d$$

represents feature vectors and

$$y_i \in \{-1, 1\}$$

indicates class labels, the objective is to find a hyperplane:

$$w^T x + b = 0$$

such that it maximizes the margin:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

subject to the constraint:

$$y_i(w^T x_i + b) \geq 1, \quad \forall i$$

If the data is not linearly separable, slack variables (\xi_i) are introduced:

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

subject to:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

Evaluation Metrics

1. Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- (TP) = True Positives
- (TN) = True Negatives
- (FP) = False Positives
- (FN) = False Negatives

2. Precision Score

$$\text{Precision} = \frac{TP}{TP + FP}$$

It measures the proportion of correctly predicted positive instances out of all predicted positive instances.

3. Recall (Sensitivity)

$$\text{Recall} = \frac{TP}{TP + FN}$$

It measures the ability to correctly identify all relevant positive instances.

4. F1 Score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is the harmonic mean of precision and recall, useful when the dataset is imbalanced.

Optimizers

An **optimizer** is an algorithm that adjusts model parameters to minimize the loss function. It updates weights during training to improve model performance.

SVM Optimizer (Sequential Minimal Optimization - SMO)

The dual formulation of SVM optimization involves solving:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to:

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

SMO decomposes this large quadratic optimization problem into smaller subproblems, making it computationally efficient.

Adam vs. Non-Adam Optimizers

Adam (Adaptive Moment Estimation) is a popular gradient-based optimizer used in deep learning models.

Adam Optimizer

Adam updates weights using first ((m_t)) and second ((v_t)) moment estimates:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Bias-corrected estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Parameter update rule:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Adam combines the benefits of both momentum and adaptive learning rates, making it highly effective for complex problems.

Non-Adam Optimizers

1. **SGD (Stochastic Gradient Descent)**

$$\theta_t = \theta_{t-1} - \eta \nabla L(\theta_{t-1})$$

2. **Momentum-Based Optimization**

$$v_t = \gamma v_{t-1} + \eta \nabla L(\theta_{t-1})$$

$$\theta_t = \theta_{t-1} - v_t$$

Momentum helps accelerate SGD in relevant directions, reducing oscillations. Adam further improves upon this by adapting the learning rate for each parameter.