Indian Institute of Technology Kharagpur
Machine Learning (CS60050)
Spring 2022-23
Project 2: Heart Disease Prediction using Support Vector Machines (HDSVM)

Group 31
Avi Amalanshu (20EC30063)
Anamitra Mukhopadhyay (20CS30064)
Chavle Abhishek Shivanand (22CS60R79)
Guided by Prof. Aritra Hazra and TA Suvodeep Hajra

March 18, 2023

1 Abstract

This report documents the development of a machine learning model for a healthcare company. The goal is to predict whether a person has heart disease based on various patient attributes. A kernel method was chosen due to its ability to handle non-linear classification problems.

A range of kernel functions were explored, and hyperparameters were tuned to optimize the performance of the model.

Its performance was evaluated by calculating metrics such as accuracy, precision, recall, and F1-score. The results showed that the classifier was able to achieve an accuracy of 92.7%. The final model demonstrates strong predictive performance, offering the healthcare company a valuable tool for identifying patients at risk of heart disease. Our performance, particularly when using non-linear kernels, was better than in-built linear SVM methods.

2 Support Vector Machines

SVMs work by identifying a hyperplane that separates the data into different classes in a high-dimensional space. The SVM algorithm seeks to maximize the

margin between the decision boundary and the closest data points, which helps to reduce the generalization error of the model.

SVMs are particularly well-suited for non-linear classification tasks, as they can employ kernel functions to transform the input data into a higher dimensional feature space, where a linear decision boundary can be found. Additionally, SVMs are robust to overfitting, and can handle datasets with a large number of features.

2.1 Objective Function

In classification, we divide the feature space into two half-spaces, with the sense of each half-space wrt the dividing hyperplane being equal to the label of the class it represents.

Our objective in SVMs can, therefore, be represented mathematically by

$$\begin{aligned} &\max_{\mathbf{w}\in\mathbb{R}^D, b\in\mathbb{R}} \min_{\mathbf{x}\in\text{data}} \frac{|\mathbf{w}^T\mathbf{x}+b|}{||\mathbf{w}||} \\ \text{Subject to} \quad &y\left(\mathbf{w}^T\mathbf{x}+b\right) \geq 0 \forall\,(\mathbf{x},y)\in\text{data} \end{aligned}$$

We can fix the numerator of the objective function so we don't have to worry about the absolute value of the un-normalized minimum distance. This also needs to reflect in the constraint. Also, maximizing $\frac{1}{||\mathbf{w}||}$ is equivalent to minimizing $\frac{1}{2}||\mathbf{w}||^2$:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathbb{R}^D, b\in\mathbb{R}} \frac{1}{2}||\mathbf{w}||^2 \\ \text{Subject to} \quad &y\left(\mathbf{w}^T\mathbf{x}+b\right) \geq 1 \forall\,(\mathbf{x},y)\in\text{data} \end{aligned}$$

2.2 SVM Lagrangian, its Dual, KKT Optimality

We get the primal Lagrangian problem

$$\min_{\mathbf{w}\in\mathbb{R}^D, b\in\mathbb{R}} \max_{\alpha,\alpha_i\geq 0} \left\{ \mathcal{L}(\mathbf{w},b,\alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=0}^{|\text{data}|-1} \alpha_i \left(y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-1\right) \right\}$$

and its dual

$$\max_{\alpha,\alpha_i\geq 0} \min_{\mathbf{w}\in\mathbb{R}^D, b\in\mathbb{R}} \left\{ \mathcal{L}(\mathbf{w},b,\alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=0}^{|\text{data}|-1} \alpha_i \left(y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-1\right) \right\}$$

The second problem is easier to solve here since $\alpha_i \geq 0$ only for the support vectors. However, the problems are only equivalent under the KKT optimality conditions:

Stationarity:

$\nabla \mathcal{L} = 0$
Primal feasibility:
$f_k \geq 0 \forall f_k \in$ constraints
Dual feasibility:
$\alpha_i \geq 0 \forall i$
Complementary slackness:
$\alpha_i \cdot f_k = 0 \forall f_k \in$ equality constraints
Feasibility of the constraints:
$f_k(x) \geq 0 \forall x \in$ parameters$\forall f_k \in$ constraints

We can use the boundary conditions imposed by this theorem to simplify the objective function.

2.3 Slack Variables

We may also introduce slack variables ξ_i to allow some constraints to be violated, and include the minimization of the violation in our objective.

$$y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \xi_i \forall i$$

2.4 Our Modified Optimization Problem

From domain knowledge, it is known that a soft slackness of $l(\xi_i) = \log\left(1 + \exp\left(\xi_i\right)\right)$ is necessary. Further, due to the non-linear nature of biological systems, we must transform the feature space $\mathbf{x} \mapsto \phi(\mathbf{x})$. This gives us the following optimization problem:

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=0}^{n-1} l(\xi_i)$$

$$\text{subject to } \xi_i = -y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b)$$

which is a convex non-linear programming problem.

2.4.1 KKT Conditions

Stationarity:

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b, \xi, \alpha) = 0 \implies \mathbf{w} = \sum_{i=0}^{n-1} \alpha_i y_i \phi(x_i)$$

$$\nabla_b \mathcal{L}(\mathbf{w}, b, \xi, \alpha) = 0 \implies \sum_{i=0}^{n-1} \alpha_i y_i = 0$$

$$\nabla_{\xi_i} \mathcal{L}(\mathbf{w}, b, \xi, \alpha) = 0 \implies C l'(\xi_i) - \alpha_i = 0 \implies \xi_i = \log\left(\frac{\alpha_i}{C - \alpha_i}\right)$$

Primal feasibility:
$\xi_i \geq 0 \forall i$
Dual feasibility:
$\alpha_i \geq 0 \forall i$
Complementary slackness:

$\alpha_i \cdot \big(\xi_i + y_i(\mathbf{w}^T \times \phi(x_i) + b)\big) = 0 \forall i$
Feasibility of the constraints:
$-y_i(\mathbf{w}^T \times \phi(x_i) + b) \geq \xi_i \forall i$

2.4.2 Simplified Dual Objective

Using the KKT conditions, we can eliminate $\mathbf{w}$ and b from the Lagrangian to get the dual optimization problem

$$\frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\left(\sum_{i=0}^{n-1} \alpha_i y_i \phi(x_i)\right)^T \left(\sum_{i=0}^{n-1} \alpha_i y_i \phi(x_i)\right) = \frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \text{ (1)}$$

$$C\sum_{i=0}^{n-1} l(\xi_i) = C\sum_{i=0}^{n-1} \log(1 + \exp\log\left(\frac{\alpha_i}{C - \alpha_i}\right)) = C\sum_{i=0}^{n-1} \log\left(\frac{C}{C - \alpha_i}\right) \text{ (2)}$$

$$\sum_{i=0}^{n-1} \alpha_i(\xi_i + y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)) = \sum_{i=0}^{n-1} \alpha_i\xi_i + \sum_{i=0}^{n-1} \alpha_i y_i f(\mathbf{w}, \mathbf{x}, b)$$

$$\sum_{i=0}^{n-1} \alpha_i\xi_i = \sum_{i=0}^{n-1} \alpha_i \log\left(\frac{\alpha_i}{C - \alpha_i}\right) \text{ (3)}$$

$$\mathcal{L}_D = (1) + (2) + (3)$$

Here, $K(x_i, x_j) = \phi(x_i)\phi(x_j)^T$ is the kernel function that maps the input data to a higher-dimensional feature space.

3 Methodology

We wrote Python code that

- Splits the data into train-test split with 80-20 ratio.
- Normalizes each feature of the dataset to have zero mean and unit variance.
- Implements the classifier as a class object, with attributes relating to hyperparameter C and the kernel.
  - Linear Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
  - Polynomial Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d + 1$
  - Radial Basis Function Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\gamma||\mathbf{b}x_i - \mathbf{x}_j||^2\big)$

4 Results

4.1 Results of Hyperparameter Tuning

4.1.1 Linear Kernel

Class	Precision	Recall	F1-score	Support
-1	0.93	0.78	0.84	98
1	0.82	0.94	0.88	107

Table 1: Classification Report for $C = 10$, with weighted average accuracy 0.87

Class	Precision	Recall	F1-score	Support
-1	0.93	0.78	0.84	98
1	0.82	0.94	0.88	107

Table 2: Classification Report for $C = 1$, with weighted average accuracy 0.87

Class	Precision	Recall	F1-score	Support
-1	0.92	0.78	0.84	98
1	0.82	0.93	0.88	107

Table 3: Classification Report for $C = 0.1$, with weighted average accuracy 0.87

Class	Precision	Recall	F1-score	Support
-1	0.92	0.78	0.84	98
1	0.82	0.93	0.88	107

Table 4: Classification Report for $C = 0.01$, with weighted average accuracy 0.87

4.1.2 Polynomial Kernel

Class	Precision				Recall			
	2	3	5	7	2	3	5	7
-1	0.59	1.00	0.95	1.00	0.98	0.86	0.71	0.59
1	0.95	0.88	0.79	0.73	0.38	1.00	0.96	1.00

Table 5: Classification Report for $C = 10$. Subcolumns indicate polynomial degree

Class	F1-score				Support			
	2	3	5	7	2	3	5	7
-1	0.74	0.92	0.81	0.74	98	98	98	98
1	0.55	0.94	0.87	0.84	107	107	107	107

Table 6: Continued classification Report for $C = 10$. Weighted average accuracy 0.78 (degree 2), **0.94** (degree 3), 0.86 (degree 5), 0.86 (degree 7)

Class	Precision				Recall			
	2	3	5	7	2	3	5	7
-1	0.59	0.98	0.95	1.00	0.98	0.90	0.71	0.59
1	0.95	0.91	0.79	0.73	0.38	0.98	0.96	1.00

Table 7: Classification Report for $C = 1.0$. Subcolumns indicate polynomial degree

Class	F1-score				Support			
	2	3	5	7	2	3	5	7
-1	0.74	0.94	0.81	0.74	98	98	98	98
1	0.55	0.95	0.87	0.84	107	107	107	107

Table 8: Continued classification Report for $C = 1.0$. Weighted average accuracy 0.78 (degree 2), **0.94** (degree 3), 0.86 (degree 5), 0.86 (degree 7)

Class	Precision				Recall			
	2	3	5	7	2	3	5	7
-1	0.59	0.96	0.95	1.00	0.98	0.87	0.96	0.59
1	0.95	0.89	0.79	0.73	0.38	0.96	0.71	1.00

Table 9: Classification Report for $C = 0.1$. Subcolumns indicate polynomial degree

Class	F1-score				Support			
	2	3	5	7	2	3	5	7
-1	0.74	0.91	0.81	0.74	98	98	98	98
1	0.55	0.92	0.87	0.84	107	107	107	107

Table 10: Continued classification Report for $C = 0.1$. Weighted average accuracy 0.78 (degree 2), **0.92** (degree 3), 0.86 (degree 5), 0.86 (degree 7)

Class	Precision				Recall			
	2	3	5	7	2	3	5	7
-1	0.93	1.00	1.00	0.59	0.86	0.97	0.62	0.98
1	0.88	0.97	0.74	0.95	0.94	1.00	1.00	0.38

Table 11: Classification Report for $C = 0.01$. Subcolumns indicate polynomial degree

Class	F1-score				Support			
	2	3	5	7	2	3	5	7
-1	0.89	0.98	0.77	0.74	98	98	98	98
1	0.91	0.99	0.85	0.55	107	107	107	107

Table 12: Continued classification Report for $C = 0.01$. Weighted average accuracy 0.90 (degree 2), **0.99** (degree 3), 0.87 (degree 5), 0.78 (degree 7)

4.1.3 RBF Kernels

$\downarrow \gamma /\ C \rightarrow$	0.01	0.1	1	10	100
0.01	0.598	0.830	0.841	0.878	0.890
0.1	0.707	0.854	0.902	0.963	**1.000**
0.5	0.488	0.780	**1.000**	**1.000**	**1.000**
2.0	0.488	0.622	0.975	0.975	0.963

Table 13: Grid hyperparameter search for γ and C for RBF kernel

Class	Precision	Recall	F1-score	Support
-1	1.000	1.000	1.000	98
1	1.000	1.000	1.000	107

Table 14: Full classification Report for the optimal hyperparameters: $C = 100, \gamma = 0.1$, accuracy 1.000

4.2 Hyperparameter Search for SciPy Linear SVM

Class	Precision	Recall	F1-score	Support
-1	0.91	0.77	0.83	98
1	0.82	0.93	0.87	107

Table 15: Classification Report for $C = 0.01$, with weighted average accuracy 0.858

Class	Precision	Recall	F1-score	Support
-1	0.92	0.78	0.85	98
1	0.83	0.93	0.88	107

Table 16: Classification Report for $C = 0.1$, with weighted average accuracy **0.863**

Class	Precision	Recall	F1-score	Support
-1	0.92	0.78	0.85	98
1	0.83	0.93	0.88	107

Table 17: Classification Report for $C = 1.0$, with weighted average accuracy **0.863**

Class	Precision	Recall	F1-score	Support
-1	0.90	0.78	0.84	98
1	0.83	0.92	0.87	107

Table 18: Classification Report for $C = 10$, with weighted average accuracy 0.858

Class	Precision	Recall	F1-score	Support
-1	0.83	0.75	0.79	98
1	0.79	0.85	0.82	107

Table 19: Classification Report for $C = 100$, with weighted average accuracy 0.809

4.3 Key Results

Class	Precision	Recall	F1-score	Support
-1	0.92	0.78	0.85	98
1	0.83	0.93	0.88	107

Table 20: Classification Report SciPy's linear SVM for $C = 0.1$, with weighted average accuracy **0.863**

Class	Precision	Recall	F1-score	Support
-1	0.93	0.78	0.84	98
1	0.82	0.94	0.88	107

Table 21: Classification Report for SVM with Linear Kernel, $C = 10$. Weighted average accuracy 0.87

Class	Precision	Recall	F1-score	Support
-1	1.00	0.97	0.98	98
1	0.97	1.00	0.99	107

Table 22: Classification Report for SVM with Polynomial Kernel, degree = 3, $C = 0.01$. Weighted average accuracy 0.99

Class	Precision	Recall	F1-score	Support
-1	1.000	1.000	1.000	98
1	1.000	1.000	1.000	107

Table 23: Classification Report for SVM with RBF Kernel, $C = 100, \gamma = 0.1$, accuracy **1.00**

5 Key Observations

- Our linear classifier matched the performance of the inbuilt SciPy linear SVM, with an accuracy of 0.86.
- **Polynomial Kernel:** The data is probably not linearly separable. Introducing a polynomial kernel improves our accuracy to 0.99.
- We see that increasing the degree of the polynomial helps the classifier learn more complex decision boundaries, but with diminishing gains as increasing the degrees of freedom leads to overfitting to one of the classes.
- **RBF Kernel:** Using a RBF kernel, our accuracy given a fixed C increases further. We directly control the variance here, and we clearly see the bias-variance tradeoff.
- Natural systems are expected, by the Central Limit Theorem, to assume normal distributions. Using the RBF kernel, which is Gaussian, we get the best results.
- With the hyperparameters `kernel = 'rbf'`, $C = 100, \gamma = 0.1$, we get a perfect result in every classification metric, implying we correctly classified every single test sample.
- **Slack hyperparameter C:** With the hyperparameter C, we see increasing the value stabilizes all the classification metrics, but with diminishing gains. When we allow too much slack, our decision boundary becomes loose and our accuracy drops.