Exploring SVM and Decision Tree in Multiclass classification of Iris Dataset

Problem Defination

This project explores and compares two fundamentally different classification algorithms: **Support Vector Machine (SVM)** and **Decision Tree** by implementing entirely from scratch, without relying on high-level machine learning libraries.

Objectivve :-

- Implement both models from first principles
- Apply them to the multiclass Iris classification problem
- Analyze their geometric behavior
- Study bias–variance tradeoff
- Examine regularization effects
- Investigate linear separability
- Prevent and demonstrate train–test leakage
- Compare model performance analytically and experimentally

Iris Dataset

Iris dataset is the collection of 150 samples from 3 different types of iris flower. Each sample contains 4 traits of flower namely :

1. **Sepal Length:** The length of the outer leaf-like part of the flower.
2. **Sepal Width:** The width of that outer part.
3. **Petal Length:** The length of the inner, usually more colorful petal.
4. **Petal Width:** The width of the inner petal.

There are exactly 50 samples for each of the following class:

1. Iris Setosa
2. Iris Versicolor
3. Iris Virginica

Algorithms Implemented

We implemented following two ML algorithms

1. Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a supervised machine learning algorithm used primarily for classification.It is a type od Marginal classifier. It works by finding the best possible line or plane that separates different classes of data with the widest possible gap. Mathemetically it is desiigned for binary classification but with some work around we can use it for multiclass classification too.

Marginal Classifier

It is a type of classifier that classifies the data points by drawing the margin between the two classes. The margin is the distance from hyperplane.

Hyperplane

Hyperplane is $(n -1)D$ for nD data.

for $n-dimesional$ space, equation of hyperplane is, $$w^T+b=0$$ where,

- w is weight vvector
- b is bias

for any data point x_i marginal classifier predicts based on which side of plane does the datatpoint falls

- if $w^T+b \geq 1$ we label it with $+1$
- if $w^T+b \leq -1$ we label it with -1

Functional Margin

We want the classifier to not just be `"right",` but to be `"right by a lot"`. We define the functional margin of a point (x_i,y_i) as: $$\hat{\gamma _i} = y_i (w^T + b)$$

An SVM is confident when $\hat{\gamma_i}$ is a large positive number. If $\hat{\gamma_i} > 0$, the point is correctly classified.

Support vector

Those point that fall exactly on the marrgins ie $w^T + b = 1$ and $w^T + b = -1$. Moving the support vectors affect the margin but moving any other points do not affect the margin.

Soft Margin

In real-world scenarios, data is rarely perfectly linearly separable. If we used a "Hard Margin," a single outlier could make it impossible to find a solution. To handle overlapping classes or noisy data, we use a Soft Margin, which allows some points to violate the margin or even be misclassified.

The slack variable ξ_i We introduce a non-negative slack variable, $\xi_i >0$, for every training point x_i. This variable measures the `degree of misclassification`:

- $\xi_i >0$: The point is correctly classified and lies on or outside the margin.
- $0 < \xi_i \leq 1$: The point is correctly classified but lies inside the margin.
- $\xi_i \geq 1$: The point is outside the margin (misclassified).

The Objective Function

The goal is to maximize the margin while simultaneously minimizing the classification error. We solve the following optimization problem:

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

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

Objective Function aims to minmize two things :

- $\frac{1}{2} |w|^2$: to get the widest possible margin.
- $ C \sum_{i=1}^{n} \xi_i$: to reduce the number of points falling on the wrong side of the boundary.

Regularization parameter C

C is the hyper parameter that balance the above two competing gooals.

- If the value of C is small the model focuses on the making margin wide even though more datapoins are prone to misclassification. `(High bias, Low varience)`
- If the value of C is large than model focuses on correce classification even though the margin may be small `(Low bias, High variance)`

Hinge Loss

In coding we don't explicitely use ξ_i insted we use hinge loss $$L = \max(0, 1 - y_i(w^T x_i + b))$$

This comes from our constrain $$ \xi_i \geq 1 - y_i (w^T x_i + b), \quad \forall i$$

so our final objective becomes $$\min_{w, b} \frac{1}{2} |w|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b))$$

2. Decision tree

A Decision Tree is a **non-parametric** supervised learning algorithm used for both classification and regression. It mimics the way humans make decisions by following a flowchart-like structure of "if-else" questions.

Understanding the tree

A tree is composed of three main types of components:

- **Root Node:** The very first node at the top. It represents the entire dataset and the first feature used to split it.
- **Internal (Decision) Nodes:** These nodes represent a `test` on an attribute (e.g., `Is Petal Width > 0.8?`). Each branch represents the outcome of that test.
- **Leaf Nodes:** The terminal nodes at the bottom. These do not split further and represent the final class label or predicted value.

Recursive Partitioning

A Decision Tree performs `Recursive Binary Splitting`. At every node, it searches for a feature j and a threshold t that splits the data into two regions:

- $R1(j,t) = { X | X_j \geq t}$
- $R2(j,t) = { X | X_j < t}$

The goal is to choose j and t such that the resulting child nodes are as `pure` as possible.

Measuring `Purity` (Gini Impurity)

This is the measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

For a node m with K classes, the Gini Impurity G is: $$G = 1 - \sum_{k=1}^{K} p_k^2$$

Where p_k is the proportion of observations in node m belonging to class k.

- $G=0$: Perfect purity (all samples in the node belong to one class).
- $G=0$.5 (for 2 classes): Maximum impurity (50/50 split).

The Objective Function (Information Gain)

To decide on the best split, the tree calculates the Weighted Gini Impurity of the children. It wants to minimize this value (or maximize the Gain):

$$\text{Loss}(j, t) = \frac{n_{\text{left}}}{n} G_{\text{left}} + \frac{n_{\text{right}}}{n} G_{\text{right}}$$

The tree iterates through every feature and every possible threshold value to find the pair (j,t) that results in the lowest combined Loss.

Stopping Criteria (Regularization)

A Decision Tree is a `greedy algorithm`. It will keep splitting until every leaf is perfectly pure $(G=0)$. This leads to massive overfitting. To prevent this, we use hyperparameters:

- **Max Depth :** Limits how many levels the tree can have.
- **Min Samples Split :** A node must have at least N samples before it's allowed to split.
- **Min Impurity Decrease :** A split is only made if it improves purity by a certain amount

SVM vs Decision Trees

	SVM	Decision Tree
Foundation	Geometry based	Logic based
Boundry	slanted hyperplane	Axis based Boxes
Multiclass	Needs external adjustments like `OvR` or `OvO`	Foundational Support
Scaling	Extremly Sensetive	Doesn't Affect
Optimization	Global (Looks at all datapoint to decide hyperplane)	Local (Once split is made it doesn't look back)
Outliars	If away from hyperplane it doesn't affect	Extremly sensetive

Experiment Results

1. Iris dataset with SVM

As multiclass is not supported matthemetically in SVM we had to use `OvR` approach. Here we trained 3 seperate SVM models

1. **SVM1 :** class-0 (Setosa) vs. All others
2. **SVM2 :** class-1 (Versicolor) vs. All others
3. **SVM3 :** class-2 (Virginica) vs. All others

`Note : all models were initially trainied with C = 1`

`SVM1` peformed extremly good as setosa lies quive far away in vector space so it was **Linearly seperable**. We got **accuracy of 100%** with average training loss of **0.0978**

`SVM2` was complete different story. Versicolor and Virginica are mixed in vector space and are not **Linearly seperable**. so when we trained with $C = 1$ the model gave accuracy of 66.67% with average training loss of ~11 which means it missed 100% of the datapoints as Versicolor. In order to fix tat we used $C=100$ which boosted the accuracy to **73.33%** with average training loss of 56.

`SVM3` did similar to `SVM1` and it being mixed with versicolor got slightly lower accuracy of **86.67%** with average training loss of 0.42.

Finally for the multiiclass classification we predictted the datapoint with all 3 models and classified based on which SVM gave highest probablity. we got final accuracy of **76.67%**

2. Iris dataset with Decision Tree

Decision Tree was pretty straight forward. The important features were **Petal Leangth** and **Petal width**. Model got accuracy of **96.67%**. We got the maximum depth of 3 with 4 splits.