**Answers to Theoretical Questions**

**Q1. What is the mathematical formula for a linear SVM?**

The mathematical formula for a linear SVM can be expressed as:

f(x)=w⋅x+bf(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b

where:

* x\mathbf{x} is the input vector,
* w\mathbf{w} is the weight vector,
* bb is the bias term,
* f(x)f(\mathbf{x}) is the decision function.

**Q2. What is the objective function of a linear SVM?**

The objective function for a linear SVM is to minimize the cost function, which balances the margin maximization and classification errors:

min⁡w,b12∥w∥2+C∑i=1nξi\min\_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum\_{i=1}^n \xi\_i

subject to the constraints:

yi(w⋅xi+b)≥1−ξi,ξi≥0y\_i (\mathbf{w} \cdot \mathbf{x}\_i + b) \geq 1 - \xi\_i, \quad \xi\_i \geq 0

where:

* ∥w∥2\|\mathbf{w}\|^2 represents the margin width (to be maximized),
* CC is the regularization parameter,
* ξi\xi\_i are slack variables to allow misclassification for soft margins,
* yiy\_i is the label for xi\mathbf{x}\_i.

**Q3. What is the kernel trick in SVM?**

The kernel trick allows SVM to perform classification in a high-dimensional space without explicitly computing the transformation. Instead, it computes the inner products in the feature space using a kernel function K(xi,xj)K(\mathbf{x}\_i, \mathbf{x}\_j). Common kernel functions include:

* Linear: K(xi,xj)=xi⋅xjK(\mathbf{x}\_i, \mathbf{x}\_j) = \mathbf{x}\_i \cdot \mathbf{x}\_j,
* Polynomial: K(xi,xj)=(xi⋅xj+c)dK(\mathbf{x}\_i, \mathbf{x}\_j) = (\mathbf{x}\_i \cdot \mathbf{x}\_j + c)^d,
* Gaussian (RBF): K(xi,xj)=exp⁡(−∥xi−xj∥22σ2)K(\mathbf{x}\_i, \mathbf{x}\_j) = \exp\left(-\frac{\|\mathbf{x}\_i - \mathbf{x}\_j\|^2}{2\sigma^2}\right).

**Q4. What is the role of support vectors in SVM?**

Support vectors are the data points that lie closest to the decision boundary (hyperplane). They determine the position and orientation of the hyperplane. Removing these points would change the boundary, making them critical to the SVM model.

For example, in a two-class dataset, the support vectors are the points from each class that define the margin. If we visualize the data in 2D, the support vectors are the points closest to the boundary line, which are equidistant from it.

**Q5. Illustrate with examples and graphs of Hyperplane, Marginal plane, Soft margin, and Hard margin in SVM.**

* **Hyperplane**: A decision boundary that separates data points of different classes.
* **Marginal Plane**: The planes parallel to the hyperplane at a distance equal to the margin from it.
* **Hard Margin**: SVM assumes perfectly separable data and aims for zero classification error.
* **Soft Margin**: Allows some misclassification for non-linearly separable data, balancing accuracy and margin.

Let me generate visual examples for these concepts.

The two graphs illustrate the concepts of **Hard Margin** and **Soft Margin** in SVM:

1. **Hard Margin SVM**: The hyperplane separates the two classes perfectly with no allowance for misclassified points. This is feasible only if the data is linearly separable.
2. **Soft Margin SVM**: The hyperplane allows some points to fall within the margin or even on the wrong side, balancing classification accuracy and generalization.

Next, I'll proceed with implementing an SVM classifier using the Iris dataset and the bonus task of building an SVM from scratch.

The linear SVM classifier achieved an accuracy of **100%** on the test set. The decision boundary graph shows how the classifier separates the two classes using a hyperplane with the first two features of the Iris dataset.

Next, I’ll implement a linear SVM classifier from scratch and compare its performance with this implementation.

The custom linear SVM classifier from scratch achieved an accuracy of **100%** on the test set, matching the scikit-learn implementation.

* **Weights**: [1.5075,−2.3627][ 1.5075, -2.3627 ]
* **Bias**: 0.7550.755

This demonstrates that a simple implementation of an SVM with gradient descent can yield comparable results to optimized libraries like scikit-learn for basic datasets.

Let me know if you'd like to explore the effects of varying the regularization parameter CC or delve deeper into any part of the implementation.