Scaler Companion beta

Support Vector Machines (SVM) - Comprehensive Notes

Support Vector Machines (SVM) is a supervised machine learning model used for classification and regression tasks. This document covers the detailed concepts explained during the class session, focusing primarily on the SVM formulation, dual form, kernel methods, hinge loss, and more.

Table of Contents

Introduction to SVM

Support Vector Machines are used to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. The algorithm aims to maximize the margin between the data points of different classes. It has several unique features such as hard margin and soft margin classifiers which are used depending on the dataset nature【4:14†source】.

Understanding Margins and Support Vectors

Margins: The margin is the distance between the separating hyperplane (decision boundary) and the nearest sample from either class. SVMs aim to maximize this margin. This is known as the maximum-margin classifier【4:13†source】.

position and orientation of the hyperplane【4:8†source】. A point can be a support vector if:

- It lies exactly on the separating hyperplane.
- It lies within the margin but on the wrong side of the hyperplane.
- It is misclassified but contributes maximally to defining the margin 【4:11†source】.

SVM Primal and Dual Form

SVM can be expressed in two formulations:

- **Primal Form**: This involves a straightforward optimization problem; however, it is usually computationally expensive due to the large number of constraints.
- **Dual Form**: The dual problem reduces the constraints to the number of support vectors, making it more computationally feasible. The main aim in dual form is to solve the Lagrange multipliers with respect to constraints 【4:10†source】【4:13†source】.

Kernel Functions

Kernel functions help transform data into a higher-dimensional space which can make it easier to classify. The kernel trick applies the kernel function to project the original data into a higher-dimensional space. The types of kernel functions discussed include:

- **Polynomial Kernel**: Useful when the data is not linearly separable in the original space. It includes an 'n' degree polynomial component to handle non-linearity【4:15†source】.
- **RBF (Radial Basis Function) Kernel**: An excellent choice for pattern recognition, mapping inputs into infinite-dimensional space 【4:18†source】.

Hinge Loss in SVM

Hinge loss is used to handle the classification errors. If a data point is correctly classified with a good margin, hinge loss is zero, but if it

Scaler Companion beta

Polynomial and RBF Kernels

- **Polynomial Kernel**: Presented as $(c + x_1^T x_2)^n$, where 'c' is usually set to one and 'n' is the polynomial degree. It can capture interactions up to degree 'n'【4:19†source】.
- **RBF Kernel**: Defined as $e^{-\gamma \|x_1 - x_2\|^2}$. It maps inputs onto a unit circle, efficiently handling high dimensionality【4:18†source】.

Analogies and Example Walkthroughs

- **Separability through Polynomial Features**: An example highlighting how a dataset not separable in one dimension becomes separable when squared to a plane【4:0†source】.
- **Kernel Usage**: Analogies stating how kernels help in data mapping beyond linear separability were explored, making complex decision boundaries effortlessly【4:15†source】【4:19†source】.

Conclusion

SVM is a versatile model capable of handling linear and non-linear data through smart use of kernels. Despite being computationally intensive, it remains a go-to model for classification problems, particularly when the dataset is small to medium-sized. The detailed exploration of the dual form and kernel functions highlights how SVM aims to effectively solve big data classification problems by optimizing margin separation【4:13†source】【4:11†source】.

This concludes the recap of the SVM class. For learning more about the implementation and advanced study, it's advisable to follow the structured syllabus and practice coding examples in a suitable programming environment such as Python with libraries like `scikit-learn`.