Machine Learning

Lab 5 - Support Vector Machines (SVMs)
Spring 2025

Introduction

In this lab you will implement a Support Vector Machine (SVM) for the purpose of binary classification.

You may **not** use any functions from an ML library in your code. And as always your code should work on any dataset that has the same general form as the provided one.

Your task will be to write a *single script* such so that we can run it in the command line and it output (and/or displays) the requested information/figures.

Grading

- +1pt Can run script properly.
- +1pt Parses dataset correctly.
- +1pt Creates some weights and/or alpha values.
- +2pts Creates correct weights and/or alpha values.
- +1pt Reports statistics.
- +2pt Better than class prior results.
- +2pts Close to target results ($Accuracy \approx 90\%$).

Datasets

Spambase Dataset (spambase.data) This dataset consists of 4601 instances of data, each with 57 features and a class label designating if the sample is spam or not. The features are *real valued* and are described in much detail here:

https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names

Data obtained from: https://archive.ics.uci.edu/ml/datasets/Spambase

1 Support Vector Machine Classifier

Download the dataset *spambase.data* from Blackboard. As mentioned in the Datasets area, this dataset contains 4601 rows of data, each with 57 continuous valued features followed by a binary class label (0=not-spam, 1=spam). Your code should work on any dataset that lacks header information and has several comma-separated continuous-valued features followed by a class id $\in 0, 1$.

Write a script that:

1. Reads in the data.
2. Randomizes the data.
3. Selects the first 2/3 (round up) of the data for training and the remaining for validation.
4. Uses gradient assent to compute the value of α and the weights w (if not doing the kernel trick), using the training data.
5. Uses the learned weights (or alpha values) to determine the accuracy of the *accuracy* of the training and validation data and the *precision, recall, and f-measure* for the validation set.

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. Remember to z-score your data and add a bias feature!
3. It is recommended that you use numpy's *diagflat* function for obtaining your diagonal matrices.
4. While you aren't required to do so, you might want to write your code in such a way that you can use the *kernel trick* and experiment with the kernel to use.

Your script should output the following to the command line:

1. The class priors.
2. Requested classification statistics