CMSC 678 Project 2

Paul Hudgins

Department of Computer Science
Virginia Commonwealth University
Richmond, VA, USA

***Abstract*—This report details the author's methods and results for CMSC 678 Project 2. A linear hard-margin SVM was tested on 2-dimensional linearly separable data. Linear-kernel, Gaussian-kernel and polynomial-kernel soft-margin SVMs were tested on 9-dimensional linearly inseparable data.**

Keywords-SVM; matlab; Gaussian kernel; Polynomial kernel;

I. Running the Project

Code for Part 1 can be executed by running Part_1.m. All results are printed. The code for Part 2 is configurable by changing the constant KERNELNO. Linear kernel is 1, Gaussian is 2, and Polynomial is 3. The default is 2: Gaussian.

II. Hard-margin SVM

A hard-margin SVM was developed and tested on the provided linearly seperable dataset, with 2 classes, 2 features, and 98 examples.

The alpha values of the support vectors were 0.4362, 0.8538, and 0.4177. The bias was 3.170, The full margin was 0.7652. For the test datapoints, [3 4] had a decision value of -3.0961 and [6 6] had a decision value of -7.8014. Graphical results are shown in Figure 1.

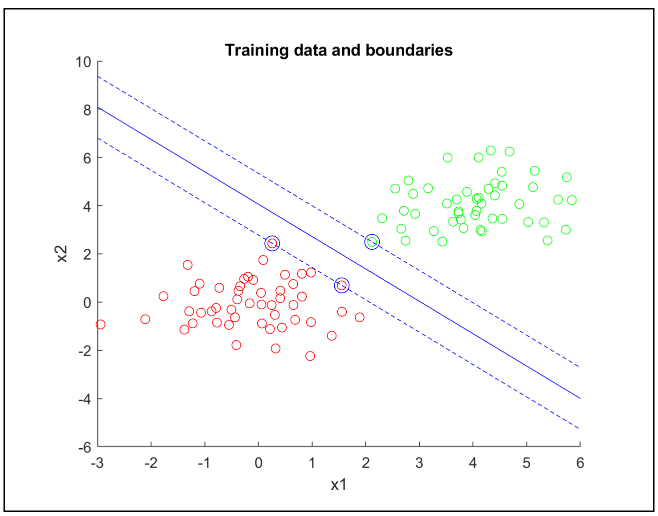

Figure 1. Training data, boundaries, and support vectors for Part 1.

III. Soft-margin SVM

A. Design and development

Soft-margin SVMs were tested on the provided dataset with 9 features, 6 classes, and 214 examples. First, a soft-margin linear SVM was developed to discriminate between Class 1 and all others. The project was then extended to predict class using six One-VS-All classifiers. The classifiers were then modified to take a kernel parameter. A linear kernel was used for testing purposes. Once this was functioning properly, other kernels were tested.

The code is organized into functions as follows, from bottom to top: Train_soft() trains a binary soft-margin SVM and returns values of v and b, as v is specified in equation 2.37 of the text. Decision() evaluates the decision function for an instance using these values. Score() tests a binary classifier on a dataset. Xval() performs cross-validation for a given kernel and parameters. Optimize() determines best parameters from a provided list using xval(). Build_classifiers() splits a multi-class problem into one-vs-all binary problems, and determines best values of v, b, C, and kernel parameters. Prediction() predicts the class for a single example using all classifiers and the MAX operator. Final_evaluation() tests the set of classifiers on the entire dataset.

B. Gaussian Kernel

The project can be run with a Gaussian kernel by running Part_2.m with KERNELNO set to 2. Runtime on my machine was 2 minutes and 42 seconds. Optimum values of C and kernel parameters are printed.

The accuracy of this classifier was 80.84% with 41 errors.

C. Polynomial Kernel

The project can be run with a Polynomial kernel by running Part_2.m with KERNELNO set to 3. Runtime on my machine was 4 minutes and 51 seconds. Optimum values of C and kernel parameters are printed.

The accuracy of this classifier was 82.71% with 37 errors.

I