2. Programming Write-Up

Introduction

The first step in creating any SVM algorithm is to determine the Gram Matrix by evaluating the kernel function. Due to the high dimensionality of the data (784 pixels per image), the Gaussian Kernel Function is implemented for the MNIST dataset. The Gaussian Kernel Function allows for projections from infinitely many dimensions to draw a maximum hyperplane. The equation for the Gaussian Kernel function is given by Equation 6.25 (Bishop Pattern Recognition). This kernel function is simplified by making use of the length formula to give the equation below (Lecture Notes).

$$k(x, x_0) = \exp(\frac{-||k_x - x_0||^2}{2\sigma^2})$$

In this implementation, the kernel function's variance parameter is set to the sum of the variances in pixels across each image. The Gram Matrix yields an n x n matrix where n is the number of images in the dataset or datapoints.

With the Gram Matrix, a dual optimization problem to find the weights of the maximum hyperplane is solved. This dual problem is given in Equation 7.10 with constraints in Equations 7.11 and 7.12 respectively (Bishop). Solving the dual problem yields an optimal dual variable which is then used to find the bias term of the maximum hyperplane given by Equation 7.18. Using the dataset targets, the dual optimized variable, the Gaussian Kernel function and the bias parameter; the maximum hyperplane is deterministically calculated using Equation 7.13.

A maximum hyperplane is used to separate two classes of data. In the MNIST dataset there are 10 classes to separate creating a Multi-Class SVM. Three methods of solving Multi-Class SVMs are explored below. Two of the methods are one versus the rest and one versus one which vary in the way of defining target variables and a voting procedure. The third method, DAGSVM, uses the same outputs as the one versus one algorithm but utilizes a different voting algorithm.

One Versus the Rest (One Versus All)

Analysis:

In the one versus the rest algorithm a separate SVM is created for each of the N classes during training. For the MNIST dataset of hand written digits, this corresponds to 10 SVMs. The Nth SVM is constructed by mutating the labeled dataset so any value that is equal to the Nth class variable is set to 1 and every other variable is set to -1. This mutated labeled dataset serves as the training target vector which is used along with the Gram Matrix to solve the dual optimization problem. Since the training vector values consists of all non-zero values, each SVM takes a relatively long time to train in this implantation. Using the maximum hyperplanes from the N SVMs, N predictions for the values of the datapoints are made by choosing the mode value at each datapoint as the predicted output. This voting scheme requires evaluating N SVMs

Results:

The confusion matrix for the SVM one vs. all algorithm is:

84	0	0	0	0	0	1	0	0	1
0	121	0	0	0	0	0	0	1	0
0	0	107	0	0	0	0	3	3	0
0	0	1	105	0	4	2	1	1	1
0	0	1	0	100	0	1	0	2	4
1	0	0	2	1	85	1	0	2	0
2	0	0	0	0	2	82	0	1	0
0	1	1	1	2	0	0	94	0	0
1	0	1	2	1	1	0	1	78	1
0	0	0	0	1	0	0	0	1	90

The accuracy for the SVM one vs. all algorithm is:

0.9460

Strengths:

- The most accurate method for this dataset
- Only N SVMs for training and predicting
- Relatively easy to implement
- Can use a soft margin SVMs

Weaknesses:

- SVMs grow linearly with N
- Uses dense training target vectors
- Requires the most time to train

One Versus One

Analysis:

The one versus all algorithm creates an SVM for every two-class classifier. This implementation results in $\frac{N(N-1)}{2}$ SVMs which is 45 SVMs for hand written digits classification. Each two class SVM is constructed by setting all labels in the dataset equal to 1 for the positive class and -1 for the negative class. All labels that do not belong to one of the two classes are set to 0. These mutated dataset labels are used as training target vectors to solve the dual optimization problem as in the One Versus All method. However, in this implementation the training target vectors are sparse resulting in a reduction in training time per SVM in comparison to the One Versus All method. A voting scheme then compares the 45 classifications at each datapoint and picks the most commonly chosen class as the prediction output. This voting scheme requires evaluating $\frac{N(N-1)}{2}$ SVMs.

Results:

The confusion matrix for the SVM one vs. one algorithm is:

84	0	0	0	0	1	1	0	0	0
0	122	0	0	0	0	0	0	0	0
0	0	107	0	0	0	0	1	4	1
0	0	3	105	0	2	0	3	1	1
0	0	1	0	92	0	3	1	0	11
2	0	2	3	0	82	0	0	1	2
2	0	2	0	0	1	82	0	0	0
0	1	1	0	3	0	0	91	0	3
1	0	3	3	2	2	1	0	68	6
0	1	1	1	0	0	0	0	2	87

The accuracy for the SVM one vs. one algorithm is:

0.9200

Strength:

- Uses sparse target vectors

Weaknesses:

- Requires $\frac{N(N-1)}{2}$ target vectors for training and predicting
- Least accurate method for this dataset

DAGSVM

Analysis:

The DAGSVM algorithm implements a different voting scheme on the maximum hyperplanes that are determined from the standard One Versus One method. This voting scheme uses the process of elimination to decide on a predictive class each datapoint. The process of elimination method works by determining the most least probable class of the two outermost SVM class-pair at a given datapoint and eliminating that class from the set of possible prediction outputs. Iterating the algorithm N-1 times for each datapoint results in a single class remaining for each point which is chosen as the predictive output.

Results:

The confusion matrix for the DAGSVM algorithm is:

84	0	1	0	0	0	1	0	0	0
0	122	0	0	0	0	0	0	0	0
0	0	107	0	0	0	0	2	4	0
0	0	5	104	0	2	0	2	1	1
0	0	3	0	95	0	1	0	0	9
2	0	2	3	0	82	0	0	1	2
1	0	4	0	0	1	81	0	0	0
0	1	1	0	2	1	0	92	0	2
1	0	4	4	3	2	1	0	67	4
0	1	1	1	0	0	0	0	2	87

The accuracy for the DAGSVM algorithm is:

0.9210

Strengths:

- Better accuracy than One Versus One standard implementation
- Only N-1 SVMs evaluated for predicting
- Fastest processing time for this dataset

Weaknesses:

- $\frac{N(N-1)}{2}$ SVMs to train
- Relatively difficult to implement

Summary

The below table summarizes the implementation and results for the three SVM algorithms implemented in this exercise.

	One vs Rest	**One vs One**	**DAGSVM**
Number of SVMs Trained	N	N(N-1)/2	N(N-1)/2
Number of SVMs for Predictions	N	N(N-1)/2	N-1
Training Target Vector Type	Sparse	Dense	Dense
Optimization Upper Bound (C)	Unbounded	1.4	1.2
Kernel Used	Gaussian	Gaussian	Gaussian
Kernel Parameter (σ)	0.77	0.240	0.230
Accuracy	94.60%	92.00%	92.10%