Devon Callanan
Intro ML
Dr. Huang

Project 2 – Support Vector Machines

Approach:

Data Set

The data was imported into a data set class and each vector was normalized by a factor of 128. The data set was split into two separate objects five times for five-fold verification. Each set contained 4/5ths of the data in the training set and 1/5th of the data in the testing set. The data was set up for one-versus-all classification.

Linear SVM

The linear style support vector machine (SVM) classification was completed first.

For training, matrices were assembled to represent the data and constraints of the soft margin SVM as a classic quadratic programming problem and the python package cvxopt was used to find the solution as a set of variables in vector *a*.

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Figure 1: SVM soft margin

$$\begin{aligned} \text{minimize} \quad & (1/2)x^T P x + q^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

Figure 2: Quadratic programming format

After *a* was discovered, w and b were recovered. Finally, all test data points were classified with the distance from the margin.

sign(*w dot z + b) = class label*

Eq 1: Classification formula

Polynomial SVM

A similar approach was taken here with the addition of a polynomial kernel. Instead of the dot product of the two training vectors, the transformation of the dot product was used.

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Figure 3: Dual problem with kernel (constraints not shown)

In this case, K(xi, xj) is simply the second order polynomial kernel (1 + xi·xj)^2. W is not recovered as the raw data is not converted to the kernel space phi(x), only the dot product has been. Instead, b is recovered and the discriminant function is used to classify the data.

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

Figure 4: Discriminant function

Parameters and tuning

The C parameter, used to define the acceptable error in the quadratic programming optimization for soft margins performed best at a values of size .1 or larger. Data normalization had a rather minor affect, helping the polynomial but inhibiting the linear.

Running

The program takes under 10 minutes to run, completing both training and classification of five fold verification for both the linear and polynomial kernel. Most time is spend in the computation of dot products prior to optimization and in the classification stage of the polynomial kernel.

The python dependencies are listed in the requirements.txt and can be installed using pip. Please consider using a virtual environment. Run using python svm.py to kick off both linear and then polynomial.

Results:

Type	Raw data	Normalized	C = .1	C = .001	C = 100
C param	1	1	.1	.001	100

Norm factor	N/A	128	128	128	128
Linear (acc)	0.98	0.9774999	0.97749999	0.8425	0.9774999
Poly (acc)	0.9875	0.99	0.99	0.99	0.99