Homework 4: Support Vector Machines

Jeremy Hintz

April 2016

1 Introduction

For this assignment, we revisit the problem of classifying handwritten numerical digits from the MNIST database. In Project 1, we approached the problem through Principal Component Analysis. In this project, we will instead use Support Vector Machines.

Classification of images is among the most difficult applications of computer visual recognition. As such many different methods for building and training classifiers have been developed, and many techniques and their success in classifying digits can be seen on the MNIST website. According to the website, we should expect a well-trained SVM to reach sub-1.5% error rates.

The MNIST data set that we are using is a set of 60,000 training images and 10,000 test images. The images in the dataset are represented as 28×28 matrices, with each cell of the matrix being a value from 0 to 255 representing the gray-scale pixel value (where 0 is black and 255 is white). Examples of the images can be seen in Figure 1. While we recognize immediately from the MNIST website that lower error rates are attainable by first deskewing the images, for our purposes, we will look elsewhere for areas to seek higher accuracy. The techniques used are discussed below.

Support Vector Machines are able to produce very good results with a relatively simple setup. Volumes of research exist on Support Vector Machines in their various forms. As such there is ample literature for us to work from. The SVM training algorithm I chose for this project is the Sequential Minimal Optimization (SMO) algorithm, described by Microsoft Research (http://research.microsoft.com/pubs/69644/tr-98-14.pdf). A drawback of using an SVM in general is that its computational cost is large when training

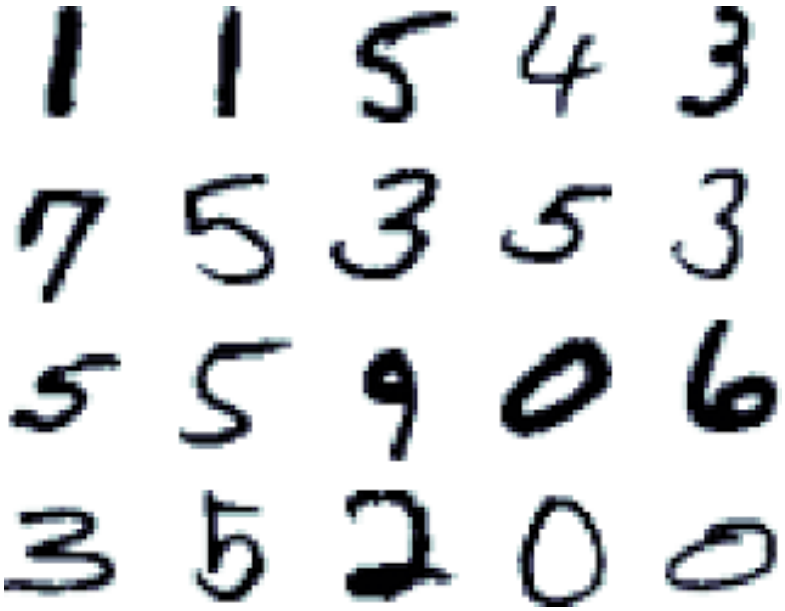

Figure 1: Samples of hand-written digits from the MNIST dataset

with a large data set. SMO, however, is generally faster, is easier to implement, and generally has better scaling properties than other SVM training algorithms so it is likely a good pick for classifying our digits. It will be a little computationally intensive to operate on all 60,000 28×28 images from the MNIST dataset, but nevertheless this cost is acceptable for testing our algorithm with segments of the dataset.

The implementation of these algorithms is done in MATLAB. The main focus in my approach was looking at the algorithms themselves. Many who have tried to classify this dataset previously have used pre-processing techniques such as edge blurring, pixel shifting, noise removal, deskewing/deslanting, and Haar features. While some of these methods have proven valuable by past attempts, no pre-processing was performed in advance of the algorithms described in this report.

2 Approach

In the following section, I attempt to provide an overview of the SMO approach to Support Vector Machines.

SMO was developed by John Platt at Microsoft Research. Platt's work was published in 1998. At that time, Support Vector Machines were not nearly as extensively researched as they are today.

For the mathematical intuition behing SMO, consider a classification problem (i.e. handwritten digit classification) with data set $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where x_i is an input vector (in our case representing an image) and y_i is the sample's corresponding class label. SMO is an algorithm for solving a problem that arises during the training of Support Vector Machines, the quadratic programming (QP) problem. This takes the dual form:

$$max_\alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=i}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j$$

where

$$0 \leq \alpha \leq C$$

for

$$i = 1, 2, ..., n$$

and

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

.

$K(x_i, x_j)$ is the kernel function supplied, C is a hyperparameter, and the variables α_i are Lagrange multipliers.

SMO breaks the QP problem into several smaller sub-problems and solves them iteratively. Because of the linear equality constraint involving the Lagrange multipliers α_i, the smallest possible problem involves two such multipliers. Thus, the above restraints are reduced to:

$$0 \leq \alpha_1, \alpha_2 \leq C$$

and

$$y_1 \alpha_1 + y_2 \alpha_2 = k$$

which is simply finding a minimum of a one-dimensional quadratic function.

From there, we try to find a Lagrange multiplier α_k that violates the KarushKuhnTucker (KKT) conditions for the optimization problem. Next pick a second arbitrary multiplier α_v and optimize for the pair (α_k, α_v). Following the above steps iteratively until convergence is reached. When all the Lagrange multipliers satisfy the KKT conditions (within a user-defined tolerance), the problem has been solved.

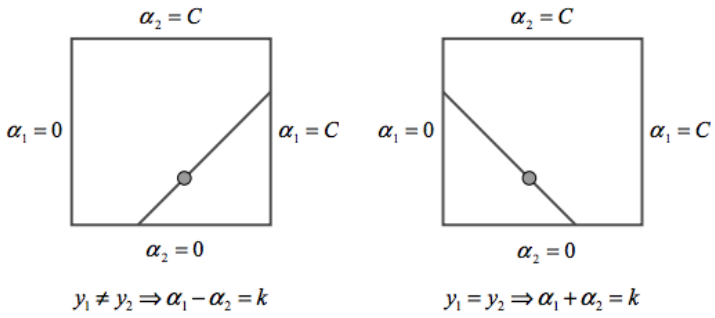

Figure 2: This figure is taken from John Platt's original paper on SMO Support Vector Machines. It illustrates the way in which the Lagrange multipliers must fit the constraints outlined in the description of the algorithm. One step of SMO must find an optimum of the objective function on a diagonal line segment.

3 Model Evaluation

The above technique was used to classify the MNIST images. The data set has 60,000 training samples and 10,000 test samples. We want to evaluate our models on how much input data they need to achieve their threshold of accuracy, as well as what that threshold is. Clearly as with most machine learning algorithms, the more training data our model consumes, the better the predictions. In order to judge our results for our SVM approach, we will benchmark using the results we obtained from KNN and Max-likelihood in Project 1.

Figure 3 shows our results for the algorithms, including multiple values for k that were used for the number of nearest neighbors to consider. We see that our SVM implementation performs better than either of our implementations from Project 1. This is likely because of the fact that KNN is a clustering algorithm, and therefore has some trouble with the boundary cases between classes. With hand written digits, it is not hard for a 4 to look like a 9 or a 5 to look like an 8. SVM, on the other hand, handled this ambiguity relatively nicely.

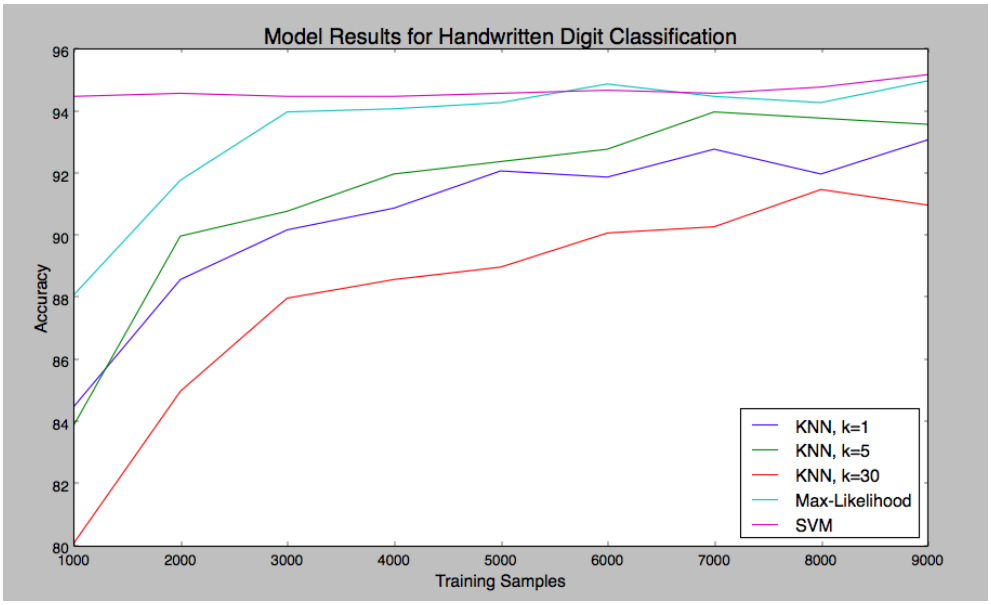

Figure 3: Model Results for SVM, compared to K-Nearest Neighbors Max Likelihood

4 Conclusion

PCA often requires reducing the dimensionality of data while simultaneously minimizing the amount of useful information that we trim away. For this reason, we used eigenvectors and eigenvalues of the covariance matrix.

There are many ways of classifying the MNIST data set. The technique of max likelihood works very well because of its assumption that the data follows a Gaussian distribution, which more or less panned out for this case.