Machine Learning - Assignment 4

Kaivan Wadia
kaivanwadia@gmail.com
kpw423

April 1st, 2016

1 Introduction

The objective of this assignment is to learn about Support Vector Machines (SVMs) and how it is used in the domain of classification problems. Support Vector Machines are used to solve a classification problems with two classes. Support Vector Machine is a supervised learning model that analyzes data and finds a hyperplane that separates the data points into two different classes with a given margin. Basic SVMs use linear classifiers to separate the data points. In cases where the data is not linearly separable we can use the kernel trick for SVMs. The kernel trick involves projecting the training data points on to a higher dimension in which the points can be linearly separated using a hyperplane. Thus, one can classify data points easier and to a better degree of accuracy in the higher dimensions.

Support Vector Machines are binary classifiers. Our assignment requires us to classify digits from 0 to 9. This involves using a classifier that can separate data points into more than two classes. Unfortunately, SVMs are binary classifiers. However, we can use multiple SVMs to solve the multi-class classification problem. We can reduce the single multi-class problem into multiple binary classification problems. In order to solve our problem we can use two methods of reduction. The two methods are one-vs-all and one-vs-one which are explained in further detail in Section 2 of the paper.

Our goal in this assignment is to use SVMs for the task of recognizing handwritten digits which is a classical classification problem. We were provided access to a training data set consisting of 60,000 training samples and a test data set consisting of 10,000 samples. Both these data sets had labels attached to each handwritten digit image. Each image in the data set was represented by a pixel map of dimensions 28x28 which is a total of 784 pixels per image. You can think of these 784 pixels as being the degrees of freedom or the features of each image.

To implement this task I decided to use Matlab. In the next section I describe in detail the steps involved in performing the classification task and then go on to describe my experiments and results. I finally conclude with some extra remarks and discussions in the final section of the paper.

2 Method

In this task we use Support Vectore Machines to solve the classical problem of handwritten digit recognition. As mentioned in the previous section SVMs are used in binary classification problems but can be modified to solve the multi-class classification problem. In order to discuss the method used to solve this problem I shall initially define some parameters that are used in the experiments and algorithm:

1. N - The number of training data points used to train the SVM classifier.
2. *CodingType* - The coding type of the multi-class SVM. It can either be *one-vs-all* or *one-vs-one*.

3. K - The kernel used in the SVM. The three different kernels used in this experiment are linear, polynomial and gaussian.
4. M - The number of features used to classify the training data points. The maximum number of features is 784 as that is the number of pixels. We can also use a smaller feature set using hog features.
5. D - The number of test data points used to test the classifier.

We can broadly classify the classification task into the Training Phase and Testing Phase. In the training phase we use the training data provided to train the SVM classifiers and in the testing phase we test our SVM against the testing data.

2.1 Training Phase

In the training phase we can change various parameters such as the Number of Training samples to use and the number of features to consider for each training sample. In the following subsections I describe the parameters that can be varied in the training process.

2.1.1 Number of Training Samples

The number of training samples used affects the classifier in two ways. Firstly, as the number of training samples used increases the time required to train the model also increases. However, using a larger number of training samples usually results in a more accurate classification model. Thus, it is important to choose the number of training samples such that the classification accuracy is high while keeping the time to train the model low.

2.1.2 Number of Features - HoG Features

The number of features in our data set is 784. It is not necessary to use all 784 features as that will lead to an increase in training time. One technique to reduce the feature set is to perform PCA on the data set to extract a smaller number of more relevant features. I decided to use HoG (Histogram of Gradients) features instead of running PCA. HoG features are used in computer vision and image for the purpose of object detection. We can think of each unique digit as an object and thus apply the same technique to this task.
I decided to use HoG features instead of PCA in order to learn something new and try a new process in the classification task as we already performed PCA in Assignment 1. By extracting the HoG features of the digit images we reduce the feature set from 784 features to 144 features. This not only reduces the time required to train the model, but also extracts the important features while ignoring features that are common in all the classes.

2.1.3 Kernel

The kernel trick is helpful when the SVM cannot separate the data into two classes in the dimension given. We can use different kernels. In this experiment I tried all three kernels available in Matlab for the SVM toolkit. The three kernels are *linear, gaussian and polynomial.*

2.1.4 Coding Type

As mentioned earlier SVMs can only be used for the binary classification problem and for the multi-class problem we use multiple SVMs. The multiple SVMs can either be coded as *one-vs-all* or *one-vs-one.* In the *one-vs-all* coding type each binary classifier assigns a positive label to one of the classes and a negative label to all the other classes. A test sample is classified by assigning it the class that has the highest output function. In the *one-vs-one* coding type each binary classifier assigns a positive and negative label to two unique class types and ignore the rest of the classes. A test data point is classified by a voting strategy where each positive label gets one vote. The class with the most votes is the class that is assigned to the test sample.

2.2 Testing Phase

The testing phase is used to determine the accuracy of the trained classification model. Our testing data consists of 10,000 handwritten digit samples. The selection of the test set and the accuracy calculations are described below.

2.2.1 Data Set

The test data set has 5,000 easy data samples and 5,000 hard data samples. While one could choose only the easy test set to achieve a higher accuracy I prefer using all 10,000 samples or a random uniform distribution of samples from both data sets in order to simulate realistic conditions. I feel there is no point in testing on just one of the data sets as that is not a true depiction of real life scenarios.

2.2.2 Accuracy

To measure the accuracy we initially get the predicted labels for our test data set based on the trained classification model. We then calculate the number of test samples that were classified incorrectly. This is the number of errors in the test set. The ratio of the number of errors to the number of test samples is the error rate. Finally, one minus the error rate is the accuracy of the classification model. I wrote a script which implements the above mentioned algorithm.

3 Experiments

I conducted various experiments to test the effect of changing various parameters of the SVMs on the accuracy of the resulting model. All the classification model in my experiment were tested against the complete data set of 10,000 sample handwritten digits. The number of training samples used varied for different experiments and was actually one of the parameters that was examined for its effect on the accuracy.

The paramters I varied were the N - *Number of training samples*, *CodingType*, *Kernel* and the *Number of features* for each training sample. Initially I determined the best kernel to use for my experiments and then kept the kernel constant for most of my future experiments. After determining the kernel I decided to investigate the effects of varying the number of training samples on the accuracy of the resulting classification model. Finally, my last experiments involve changing the number of features and the coding type used in training the SVMs.

For the number of features experiment I decided to use HoG features instead of using PCA as suggester by the assignment notes. HoG features are useful in extrapolating common features among images. It divides the images into blocks and for each block a histogram of gradient directions is compiled. The descriptor is then the concatenation of these histogram. The resulting concatenated description of the image has lesser number of features than the original 784 features of the image.

I wrote a script to perform the experiments wherein I could specify which parameter to vary and the value of the parameter for different iterations of the experiment. The script could then run each test with the appropriate parameters. After conducting each test I store the accuracy achieved by the classification model. Once all the tests were conducted for a certain experiment we can plot the effect of varying a certain parameter on the accuracy of the classification model. The results of the experiments are provided in the next section.

4 Results

As mentioned in the previous section I conducted various experiments to determine the effect of changing various parameters to the task proposed in this assignment. The first experiment I conducted was to determine which kernel is the best kernel. The three kernel options were *Linear, Gaussian and Polynomial.* I tested all the kernels while keeping the number of training samples used constant at 300. However I changed the number of features to use both the complete feature set of 784 pixels and the reduced feature set generated using the HoG features algorithm. Figure 1 shows the results of this experiment.

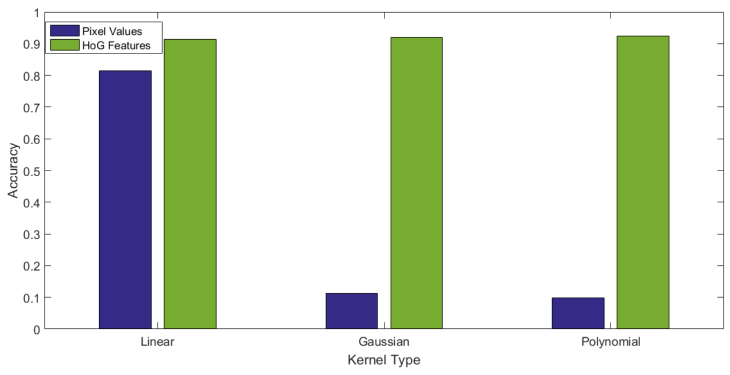

Figure 1: Plot of Kernel Type vs Accuracy.

As can be seen in Figure 1 the best performance occurs when using the HoG features. When using HoG features the Polynomial kernel outperforms the Gaussian kernel which in turn has higher accuracy than the linear kernel. In comparison when using all the pixel values as the feature set the linear kernel performs far better than the other two kernels as they basically average the values across all the features which usually is the same for all the digit types.

The next experiment I conducted was to determine how changing the number of training samples used affects the accuracy of the classification model. I decided to use the HoG features of the handwritten digit images along with a polynomial kernel for this experiment. I ran two iterations of the experiment using different Coding Types. Figure 2 shows the results of this experiment.

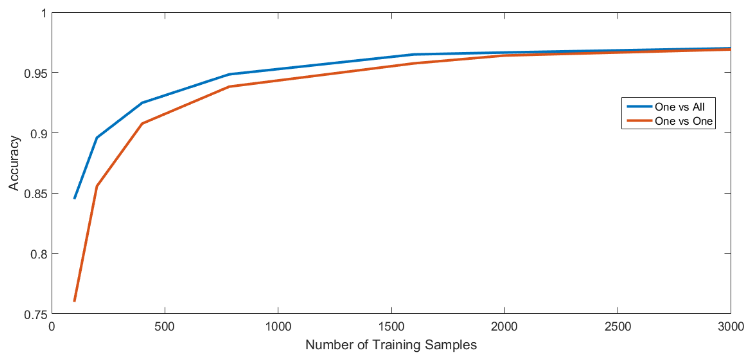

Figure 2: Plot of Number of Training Samples vs Accuracy.

As we increase the number of training samples used we improve the accuracy of the classification model.

However, the time taken to train the classification model also increases significantly. One needs to consider the time taken in relation to the accuracy increase when deciding the number of training samples. It seems that after a point increasing the training samples used does not significantly improve the accuracy of the resultant model. The *one-vs-all* coding type seems to outperform the *one-vs-one* coding type in all cases and hence might be the better coding type to use.

My final experiment involved changing the cell size in the HoG features algorithm to change the number of features that we consider for the classification model. The cell size is the number of pixels that make up each cell of the algorithm. This is a 2D vector which specifies the height and width of the cell. I decided to try square cells of varying sizes. As the cell size decreases the number of features increases. Figure 3 shows the result of this experiment.

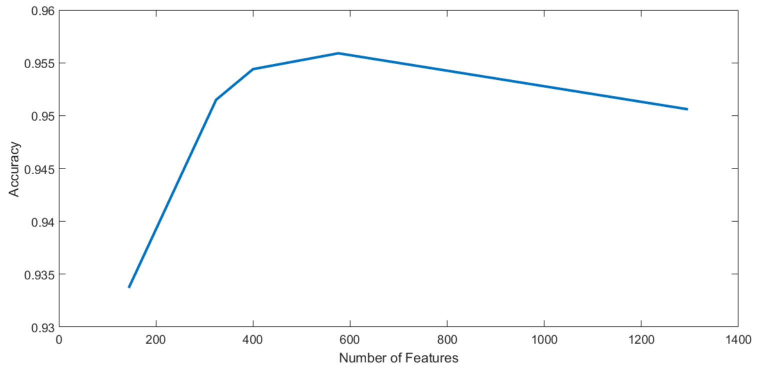

Figure 3: Plot of Number of Features vs Accuracy.

As we decrease the cell size and increase the number of features the accuracy achieved increases. This is because we are calculating finer gradient changes between each digit image. This results in a better approximation of each digit class by extracting the important and distinctive features of each class. However, if we decrease the cell size to be very small such as (2,2) then that results in a large number of features beyond the original size of the image itself. The resulting accuracy of the trained model also decreases since now we have basically extracted too many features and each digit by itself is unique from another handwritten digit in the same class and very few common characteristics can be found between different training samples.

Finally I tried various combinations to train a classification model that achieved the best possible accuracy. The highest accuracy I achieved was 98.26% when I used 5,000 training samples to train the SVM and used a cell size of (5,5) to extract the HoG features. I used a polynomial kernel and a o*one-vs-all* coding type.

5 Conclusion

I gained a deeper understanding on how SVMs work especially in the multi-class classification problem through this assignment. The assignment was fairly straightforward to implement, but it was challenging to come up with ways in which to improve the accuracy while keeping the number of training samples reasonably low in order to ensure that the training time was not very long.

Few of the other tests I would have wished to run is to test some more types of kernels rather than just the three provided within Matlab's API. I would also have liked to check how the accuracy is affected or rather the comparison between using soft and hard margins for the Support Vector Machine.

As mentioned previously I decided to use HoG features instead of performing Principal Component Analysis to reduce the feature set of the images. There are various parameters which one can change for extracting the HoG features. I used the default parameters and would like to see the effect of changing these parameters to extract the best possible features and achieve a higher accuracy. I did experiment with changing the cell size, but did not analyze the effect of changing the other parameters such as BlockSize, NumBins, BlockOVerlap etc.

Overall, I do feel that the assignment was a good learning experience. I understand the practical functionality of SVMs much better and also what goes into tuning them to achieve a better accuracy. Figuring out how to use SVMs for the multi-class problem was initially difficult, but once I read up on it in detail I understood how we could use Matlabs in-built functions to tackle the problem rather than setting up multiple SVMs myself.