HW3: Support Vector Machines

Manu Agarwal
UTEID: ma53767

Abstract. In this paper, we analyze the performance of Support Vector Machines (SVMs) on the MNIST dataset. We use three different kinds of features as input to the SVM - image pixels, image pixels after PCA and Histogram of Oriented Gradients (HOG) features. We also vary different properties of the SVM kernel and see how it impacts the performance on the dataset.

Introduction Statistical Learning Theory aims at developing mathematical models that receive data as input and output a function that can be used to predict some features of the future data. Support Vector Machines are supervised learning models that analyze data used for various tasks including classification and regression analysis. SVMs are most useful in tasks where multiple data points need to be classified into different categories.

Given a set of training examples belonging to one of many categories, an SVM learns a model that can classify new examples into one of those categories. It is thus a no-probabilistic binary linear classifier. An SVM projects data points onto a higher dimensional space where data points between different categories are separated by a margin. New examples are then classified by projecting them onto this higher dimensional space and observing which side they fall on.

In addition to performing linear classification, SVMs can also be used to perform non-linear classification using the Kernel trick.

Support Vector Machines We are given a training dataset of n points (x_1, y_1), $(x_2, y_2), \ldots, (x_n, y_n)$. The y_i's can take values +1 or -1. The x_i's are m-dimensional vectors. Our goal is to learn the maximum margin hyperplane that divides the group of points x_i's into two groups - one for which $y_i = +1$ and the other for which $y_i = -1$. A hyperplane can be written as $w^T x + b = 0$ where w is the normal vector to the hyperplane. The parameter $\frac{b}{|w|}$ determines the offset of the hyperplane from the origin along the normal vector w. Learning the SVM can be formulated as an optimization problem:

$$\max_w \frac{2}{|w|} \; subject\ to\ w^T x_i + b \geq 1\ if\ y_i = 1\ and\ w^T x_i + b \leq 1\ if\ y_i = -1 for\ i = 1 \ldots n$$

This can be written as minimizing $\frac{1}{2} w^T w$ subject to $y_i(w^T x_i + b) \geq 1, i = 1 \ldots n$. The above optimization problem is solved by formulating it's dual problem and

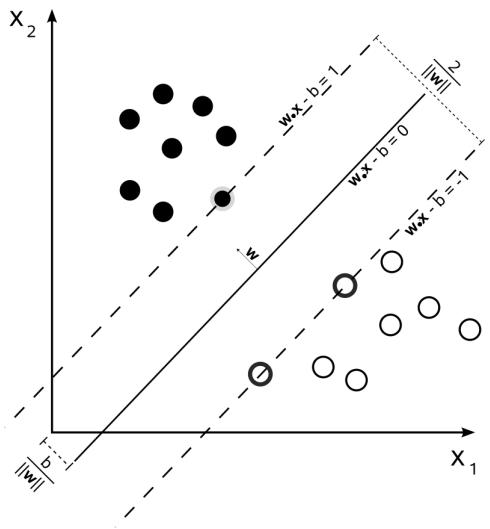

Fig. 1. Support Vector Machines

using standard optimization techniques to solve the dual.
We can also use $\Phi(x_i)$ as a function to map the points x_i onto a higher dimensional space so that the new hyperplane becomes $w^T\Phi(x_i)+b=0$. The function $\Phi_i(x)^T\Phi_j(x)$ is known as the kernel function and the choice of this function plays an important role in how well an SVM can classify data. Examples of kernel functions include polynomial kernel function $(x_i^T x_j + 1)^p$, radial basis function $\exp(\frac{|x_i - x_j|^2}{2\sigma^2})$ and linear kernel $(x_i^T x_j + 1)$.

Input Features to the SVM Selecting what features to pass to the SVM as input is as important than setting any other parameter of the SVM. In this experiment, we try out three different features - image pixels, image pixels after PCA and Histogram of Oriented Gradients (HOG) features. Each image in the dataset is 28x28 which makes for an input vector of size 784 if we take all image pixels as the input feature vector. We try reducing the dimensionality of this feature vector using Principal Component Analysis(PCA). We tried out different values of the projected feature vector and found 300 to work reasonably well, so we used 300 in all the subsequent experiments involving PCA.

Finally, we also used Histogram of Oriented Gradients (HOG) features. HOG features count occurrences of gradient orientation in localized portions of an image. HOG features have a number of advantages over other features - they are invariant geometric and photometric transformations.

Algorithm-The basic pipeline

Experiments The MNIST dataset consists of 60000 training images of size 28x28 along with their labels and 10000 test images of size 28x28 out of which

Algorithm 1 SVM for digit classification

Input: Multiple hand-written digit images and their labels
Output: Classification of test data set into different digits

1: Extract image pixels into a linear vector
2: Image Pixels: Use image pixels as training features
3: PCA: Project the image pixels onto a lower dimensional feature space and use them as training features
4: HOG: Extract HOG features from the image pixels and use them as training features
5: Train an SVM using the training features and their labels from the MNIST dataset.
6: Use the SVM trained above to classify images from the test set.
7: Compute the accuracy on the test set

5000 are easy and 5000 are hard. The test sample we used consists of 500 images chosen randomly at runtime from the 10000 test images in the MNIST dataset. The complete statistics along with results of the experiments are available here.

Using Images Pixels as the training features We first use image pixels as the training features. Figure 2 shows the gradation in accuracy as the number of training images are increased. We see that there is a sharp increase in accuracy in the initial stages and slows down as sufficient training images are supplied to the SVM for training. This is true for both the hard images and the easy ones. We used one vs all SVM for this case.

Performance of different kernel functions Next, we tried varying the kernel function of the SVM. We used linear, Gaussian and polynomial kernel functions along with one vs all SVM classifier. Figure 3 shows the performance of the three kernels on easy and hard samples from the test set. The curves suggest that Gaussian and polynomial kernels outperform linear kernel on both the easy and the hard test set. The performance of the Gaussian and polynomial kernels is quite similar with no clear distinction between the two in terms of performance accuracy.

Performance of One vs All SVM classifier as compared to One vs One We then went on to see how one vs all and one vs one SVM classifiers differ. One vs one classifier trains $\frac{n(n-1)}{2}$ classifiers and classifies test examples using a max-win voting strategy while the one vs all classifier trains n classifiers and the classifier with the highest output function is used to classify the test exmaple. Figure 4 shows the performance of the one vs one and one vs all SVM classifiers. The curves tell us that one vs all SVM classifier outperforms the one vs one SVM classifier when testing on hard samples while there is no clear distinction between the two when tesing on easy samples. One thing to note here is that one vs one SVM classifier takes a significantly longer time to train than the one vs all SVM classifier. This is possibly because the one vs one classifier trains $\frac{n(n-1)}{2}$ classifiers while the one vs all trains n classifiers.

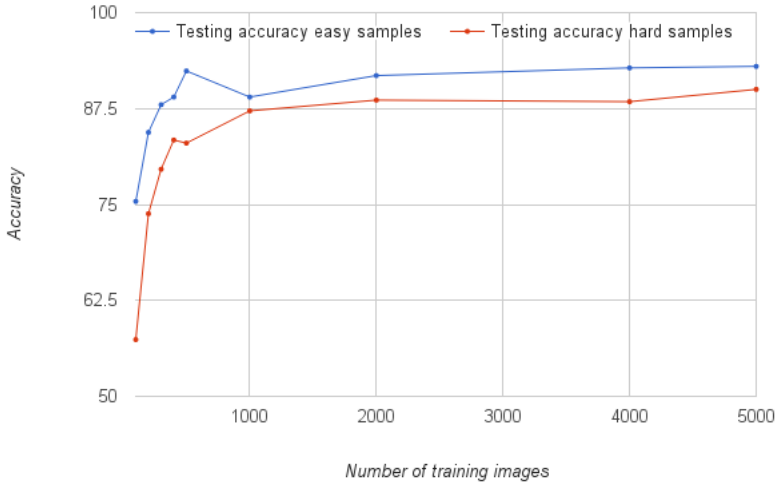

Fig. 2. Change in accuracy as a function of the number of training images

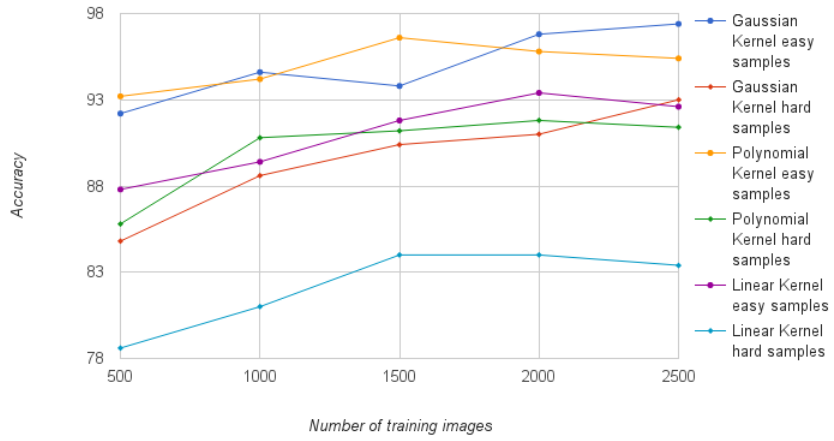

Fig. 3. Performance of different kernels as a function of the number of training images

Fig. 4. Performance of one vs one SVM classifier and one vs all SVM classifier as a function of the number of training images

Using PCA to reduce the dimensionailty of input features We now use Principal Component Analysis(PCA) to reduce the dimensionality of the training features input to the SVM. We used the code from homework 1 for the purposes of this experiment. 300 eigenvectors were found to perform the best and hence, we used 300 eigenvectors for this experiment. Figure 5 shows the performance of the easy and hard samples for this case.

Performance of different kernel functions We again compared the performance of different kernel functions and found the linear kernel to perform worse than the Gaussian and polynomial kernels. Figure 6 shows the performance of the different kernels on both the easy and hard data sets.

One vs One as compared to One vs all SVM Classifier We further tested the performance of one vs one as compared to one vs all in this setting. Figure 7 tells us that one vs all performs better on most instances than the one vs one classifier.One thing to observe is that the difference between the performance of the two classifiers has gone down as compared to the original case of using image pixels.

Using HOG features We finally used HOG features as input to the SVM. Figure 8 shows that HOG features perform considerably well, better than the above cases of using image pixels. This is possibly due to the fact that HOG features are very low level features that capture orientations in images. These orientations could be very useful features when training an SVM for hand-written digits as a digit is usually identified by how the gradient of the pixel intensity changes in the image.

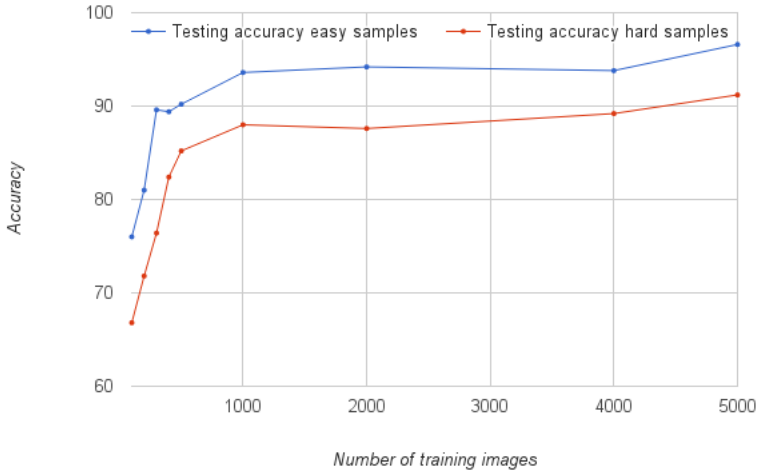

Fig. 5. Change in accuracy as a function of the number of training images when using PCA

Fig. 6. Performance of different kernels as a function of the number of training images when using PCA

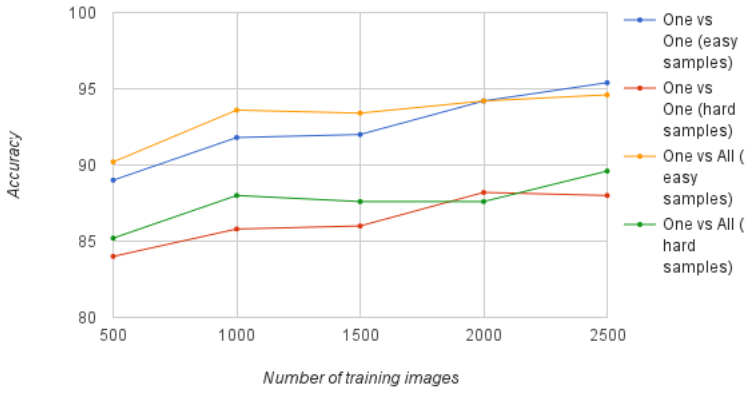

Fig. 7. Performance of one vs one as compared to one vs all SVM classifier as a function of the number of training images when using PCA

Fig. 8. Performance of HOG features as a function of the number of training images

Imposing box constraints We also investigated how imposing additional box constraints impact the performance of the SVM. It is a parameter that controls the maximum penalty imposed on margin-violating observations and is a form of regularization. If we increase the box constraint, the SVM classifier assigns fewer support vectors. Figure 9 shows the performance of SVM as we vary the value of box constraint parameter from 10^{-3} to 10^{3}. We see that the performance is maximum at a box constraint of 1 which is also the default for Matlab fitcecoc function.

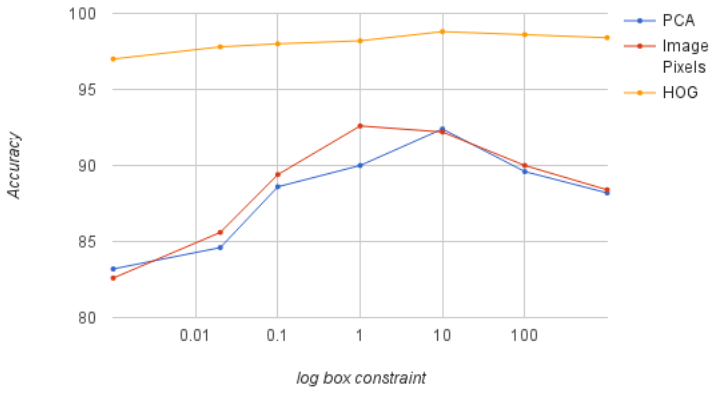

Fig. 9. Varying the parameter box constraint while training the SVM

Conclusion The experiments suggest that SVMs can be a very powerful tool for real-world classification problems. We saw that using HOG features as input to the SVM increased the performance significantly. This implies that the choice of features we input to the SVM impacts it's performance to a great extent. We also saw that introducing non-linearity in the form of Gaussian and Polynomial kernels helps improve the performance. However, the choice of features still outweighs model parameters of the SVM such as choice of kernel, box constraints etc.

References

1. Pattern Recognition and Machine Learning by Christopher Bishop
2. Resources on the Machine Learning course homepage
   http://www.cs.utexas.edu/ dana/MLClass/446outline.html
3. Wikipedia