Basic Image Recognition using Support Vector Machines

TsunSing Leung

May 7, 2016

1 Introduction

This project's purpose was to achieve basic image recognition. Sets of five different types of images were give, and the goal was to classify these images and be able to use what was learned to predict new images that are given. Given the nature of this project, I determined that the simplest thing to do was to use a Support Vector Machine to classify the features of the given images and predict the images.

2 Approach

I approached this project by first finding out what I types of machine learning I could use to do image recognition. There were options of using pixel to pixel matching, neural networks, SVMs. I eventually decided on using a Support Vector Machine with Linear Support Vector Classification. Using the Linear kernel, I processed the given images and trained the machine with supervised learning to classify them into types, 'Smile', 'Dollar', 'Hash', 'Heart', and 'Hat'. Before giving the training set to the SVM to process, I had to manually process the image first to prevent the SVM from classifying noise in the images as features. To do this, I turned the image into black and white by averaging the pixel color of the image, and used it as the threshold. Anything above the threshold was considered black, and anything below the threshold was considered white. Afterwards, I blurred the image using a Gaussian filter to reduce the noise in the image. Originally, I used the RBF kernel which was said to be the best kernel to use. However, doing so gave me 0% accuracy while the Linear kernel gave me the best results.

3 Accuracy

To test the accuracy of the machine, I used k-fold cross-validation to look through my data. I randomized the list of image bitmaps imported so when I trained the machine, using a smaller k causing a large part of a type the training set to be left out would be prevented. k-fold cross validation was used by splitting my data up to k section and training the machine with the rest of my data while leaving one section out to use as validation as I already knew what the images in that section were classified as. After testing that, I shift to the next section and train everything else aside for that section as well and I repeat until all the sections have been tested with the other sections as the training set. My final accuracy was then found calculating the total correct over the total predictions which ended up to be around 93%-94% as long as my k was high enough that most of the images were used in the training, but not higher than the amount of images there are.

With 10 folds, I had around 92.9% accuracy with 381 correct and 29 incorrect
With 20 folds, I had around 93.7% accuracy with 375 correct and 25 incorrect
With n folds, I had around 93.4% accuracy with 385 correct and 27 incorrect

4 Conclusion

Finally, I believe that despite getting great results of over 90%, the accuracy could be improved. This project has helped me learn a lot about machine learning and how it takes in real world data so the machine can process and understand it. There were other kernels such as the RBF kernel that may give me better results, but the low accuracy may have been caused by how I parsed my data.