### SUNSET SCENE DETECTOR USING SUPPORT VECTOR MACHINE

*Yuankai Wang, Chen Yin*

Department of Computer Science, Rose-Hulman Institute of Technology   
{wangy16, yinc}@rose-hulman.edu

#### Abstract

Getting features from pictures and classifying them is an important and common way of recognizing scenes. With a color-features based system using Support Vector Machine (SVM), we can distinguish sunset from non-sunset scenes. With a large training set and tuned hyper-parameters for training functions, we increased the performance of our machine and achieved 89.8% accuracy. We changed the threshold to getting different groups of True Positive Rate and False Positive Rate according to our needs.

**1. Introduction**

Recognizing sunset in a picture is a difficult and interesting problem. We were given an image, and we needed to figure out whether there was a sunset scene in the image. It is a classic problem on how to recognize a certain thing in a picture. This requires us to get information more than the how each pixel look like. We needed to know further than that to figure out what these things imply.

Sunset is somehow obvious when we look at the image. It has its unique color, it has a certain shape that almost never changes, and in most cases, it will have few noises around it on a picture. Whether the picture is a sunset scene is easy for a human to answer, because we know what it looks like and we can use our experience to get the correct answer. However, for computer, it does not have that experience. We need to train it to let it know what true sunsets are and what non-sunsets are.

To let a computer “fully” understand it, we need hundreds or thousands of pictures to cover most of the situations. However, some of the pictures can still be confusing, like the pictures shown in Figure 1. They both have a similar color set for the whole picture. It is extremely hard for computer to figure it out especially when we all feel that the second picture is a scene of sunset. The halo around the picture is what a sunset should have. Nevertheless, for the first one, we assume that we may be able to make improvement and let the computer realize it is not a sunset.



**Fig 1**: Confusing sunset images for computers

Meanwhile, detecting sunset is also quite interesting. We will need to find the features that can show sunset. We need to think deeper than our usual way. When we see a sunset, we not only should know it is a sunset, but also need to understand the reason that we think it is a sunset. We need to think and try to figure out the information by the data we get. This turns a sensitive understanding to a rational analysis on the problem.

We worked on the project is a typical way of doing image recognition, which are extracting features, managing data, training with inputs, and testing using test cases. We needed to find proper features for it and obtained it from every image. The features had to be uniform for every image so that we would have the same format for each one, and would be able to compare them. Since we had many features to compare, it was impossible to work on them manually. We needed to depend on machine learning techniques to find it out.

Recognition depends heavily on the color feature of each image. When there is a sunset, the color of the whole picture is quite likely to be affected by it. There will be a shiny point, and the pixels will be shiny around it. We get information from color to find it out. The same technique can also be applied to other recognitions to an object whose color is a dominant feature.

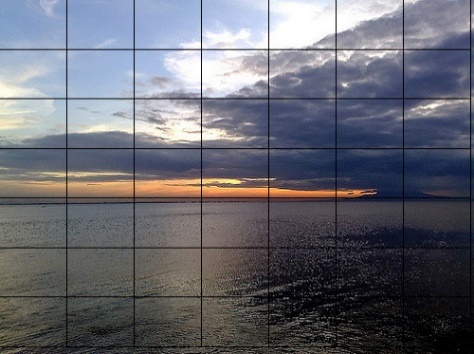
We trained the data with a large set of sunset and non-sunset pictures. We got 294 features for each picture and labeled it to indicate it was a sunset or not. Then, we trained with the features and determined another picture was a sunset or not with the result of our training.

**2. methodology**

**2.1 Feature Extraction**

As the color is an important feature we intend to inspect, we try to get color information from it. Even though we usually thought the sunset color was mainly just a combination of red, orange, and yellow, the actual color could differ a lot under different circumstances. Their color features are not universal, but their luminance almost never changes and has larger differences with other scenes. Therefore, we decided to use LST color space to get color information. L means luminance, S means red versus blue, and T means green versus magenta. The unscaled conversion equation to LST color space is; ; . We did not have to scale them, because we compared these values among images, not on the same image. Therefore, the scales of the values are the same at the corresponding position.

However, getting such information is too general. An image contains many information, and getting the overall data does not help much. Therefore, we decided to divide the image into several small sections and gained the data on each of them. There are two ways to accomplish this, dividing grid or dividing edges. Both have their advantages. Because we intended to get the data, which were easy to use, we chose to divide image with a matrix. Since the sizes of the picture are not necessarily divisible by 7. We began on the top-left corner and had to leave out some information on the bottom and the right side.



**Fig 2**: Sunset image with the grid overlaid on

After dividing, we had the data for each small part. However, the data were still just matrices. We wanted some more straightforward features that we could use to compare. Therefore, we calculated the first and second moment of each section on L, S, and T separately and finally get features. [1] The first moment is the mean value of the matrix. We could get a general idea of the color in this section by its mean value. The second moment is the standard deviation. We could know how the values in the matrix distributed by its standard deviation. The value shows whether the color in this part has a large or small difference.

Since the data we got were in different scales, we intended to normalize all the data with the minimum value to be zero and the maximum value to be one. We found the max value and min value in the matrix and use the equation for each data. We treated the output matrix as X, which we used in our training. As we wanted, our normalization applies to all the data together; we included every value from the images we got from feature extraction.

**2.2 Classifiers**

Support Vector Machines (SVM) is a common and useful way to classify things. It will receive a group of data; each data has a label to show which class it belongs. SVM usually handles binary classification. The label is either 1 or -1, which indicates yes or no for the thing we wanted to recognize. When it gets all the data, it will try to find the largest margin between the two kinds of data that differs them. It will allow some errors. However, with its default kernel, which is linear, it may not be able to handle some circumstances when the data have many intersections. Therefore, while classifying, we need to set the kernel to show the way of training. The most common used kernel is radial basis function (RBF) kernel, which is also the one we will use in our classification. We also changed its parameters to get an optimized way of classification.

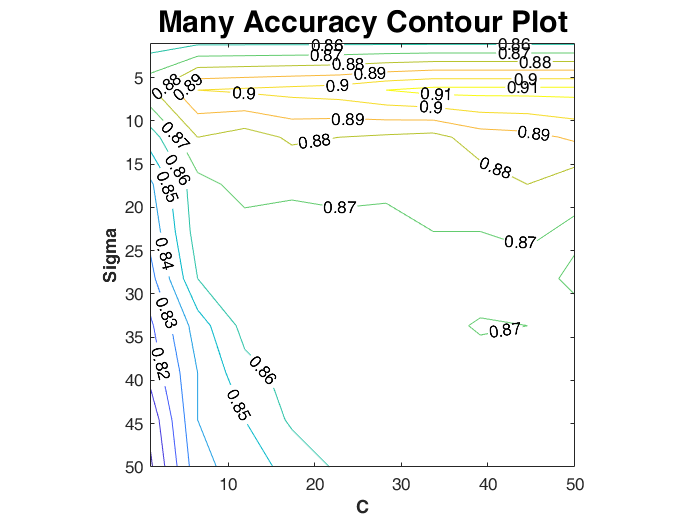
**2.3 Experimental setup**

After we prepared all the features, we used them to train the computer. We first selected the training set from its corresponding place in the matrix. We also extracted their labels in the same way. We collected 800 sunset pictures and 800 non-sunset pictures. We trained the computer with the 1600 lines of data. We also collected 300 sunset pictures and 300 non-sunset pictures for validation. There are two variables in the function sigma and c, which means kernel scale and box constraint. We used the validation set on the trained machine to test and record its accuracy.

**3. Results**

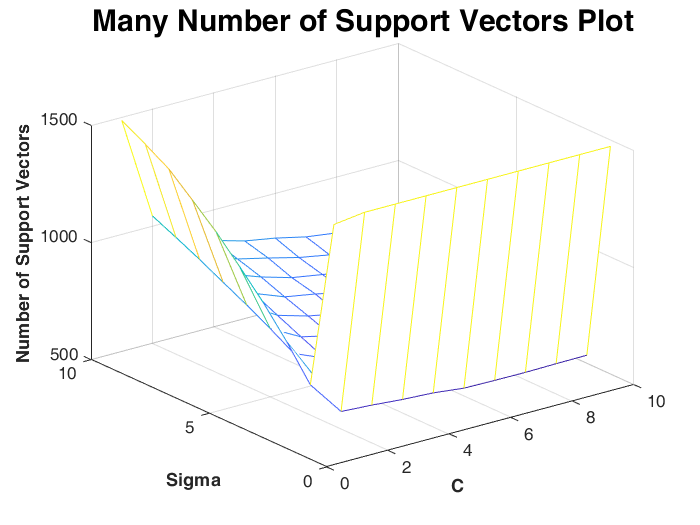
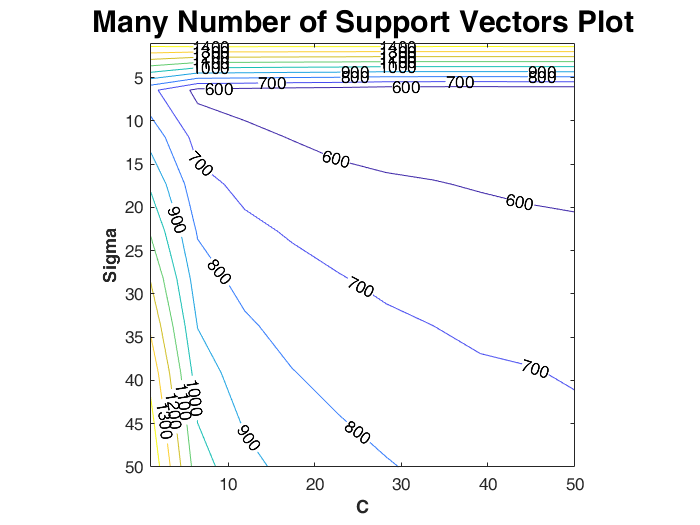
**3.1 Tune Hyper-parameters**

To achieve a higher accuracy, we had to tune our hyper-parameters. We looped over a broad range of values and recorded the accuracy of each group of values. Therefore, we finally got a matrix of accuracies with their corresponding sigma and c values. In this first step, we set the sigma from 1 to 50 with a step of 5, and c from 1 to 50 with a step of 5. We drew a picture of the matrix to give us a direct understanding of the performance. [2]



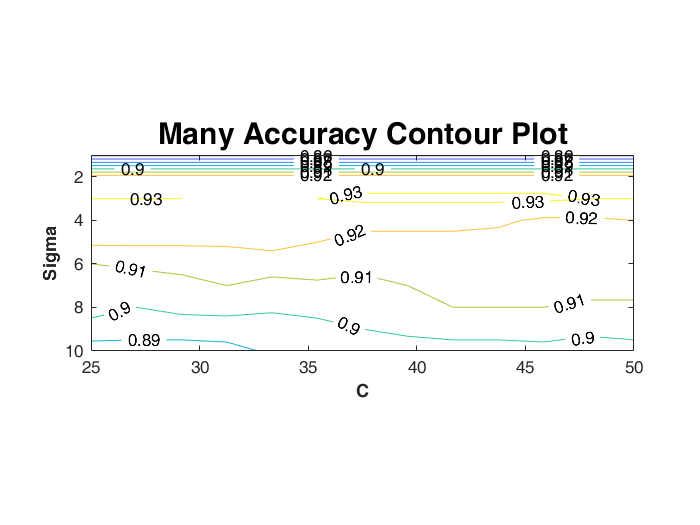
**Fig 3**: The contour plot of the accuracy for the grid search on sigma from 1 to 50 with step of 5, and c from 1 to 50 with step of 5.

Besides getting high accuracy, we also intended to have a reasonable number of support vectors to avoid overfitting. Therefore, we also formed a matrix of the support vectors in the same way.

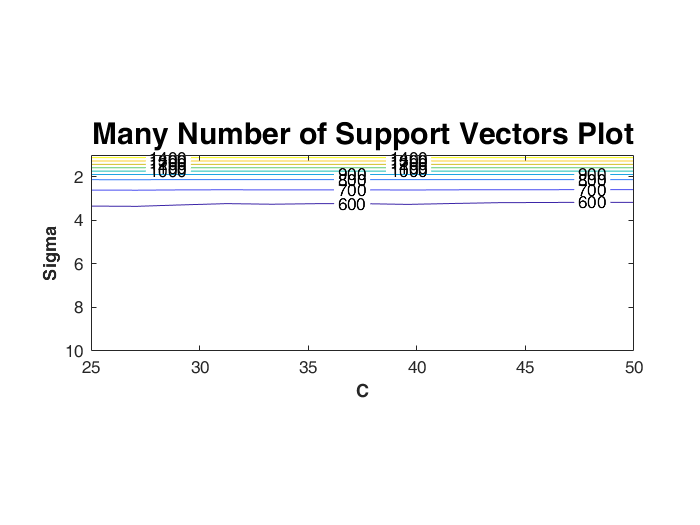


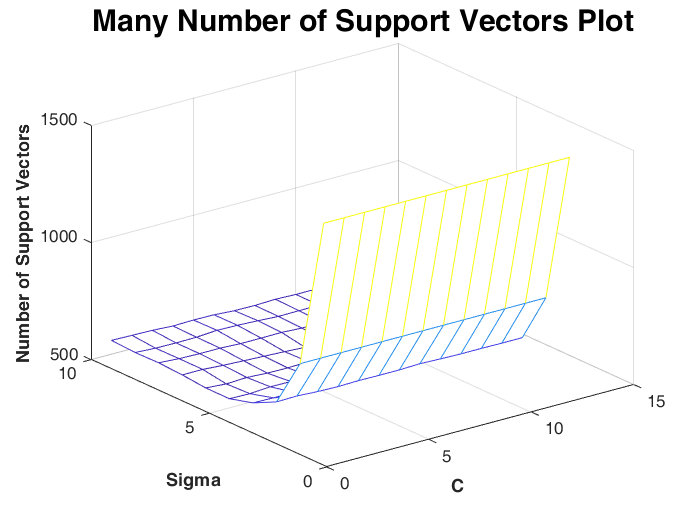
**Fig 4**: The contour plot and mesh plot of the support vectors for the grid search on sigma from 1 to 50 with a step of 5, and c from 1 to 50 with a step of 5.

From the picture about the accuracy, we can see that when sigma is in 1~10, c is in 25~50, the accuracy is obviously higher than other value of sigma and c. From the picture about number of support vectors, the number of support vectors grows dramatically when sigma or c is small. However, the accuracy does not raise much. We believed that it is better for us to choose the value that has less support vectors to get a higher efficiency with an acceptable reduction on accuracy. Therefore, we reduced the range and formed the new pictures.



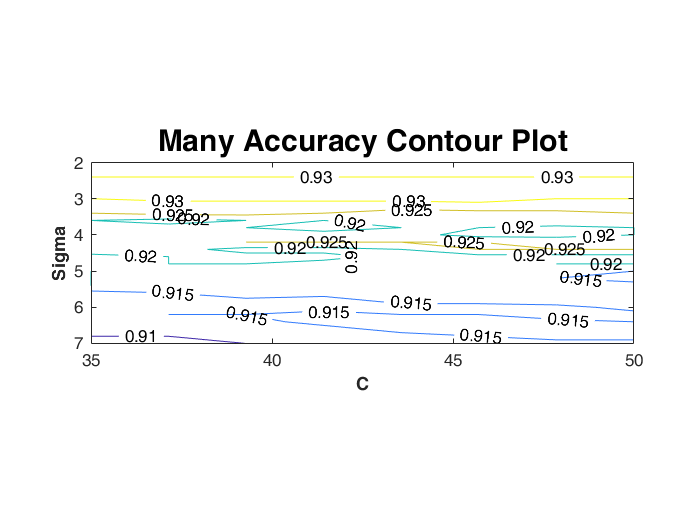
**Fig 5**: The contour plot of the accuracy for the grid search on sigma from 1 to 10 with a step of 1 and c from 25 to 50 with a step of 2.



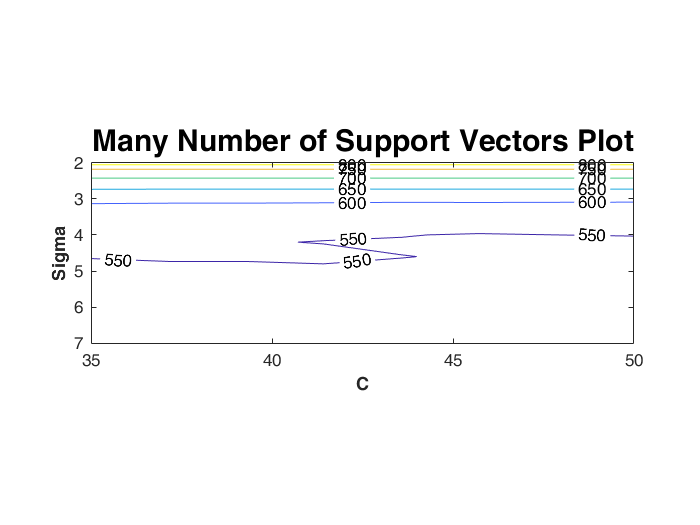


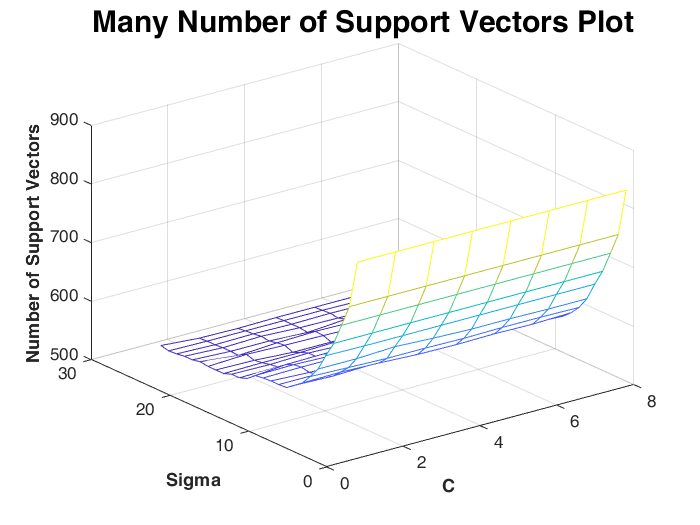
**Fig 6**: The contour plot and mesh plot of the support vectors for the grid search on sigma from 1 to 10 with step of 1 and c from 25 to 50 with step of 2.

From the accuracy plot, we found that when sigma is in 2~4, the accuracy is higher than it is in other locations. From the number of support vectors plots, we found that when sigma is in 4~10, the number of support vectors is less than 600. So we decided to constrain sigma to 2~7 and c to 35~50.



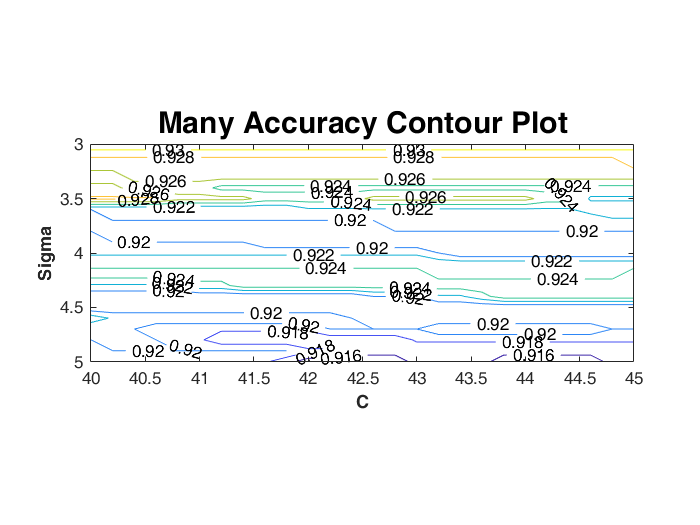
**Fig 7**: The contour plot of the accuracy for the grid search on sigma from 2 to 7 with step of 0.2 and c from 35 to 50 with step of 2.



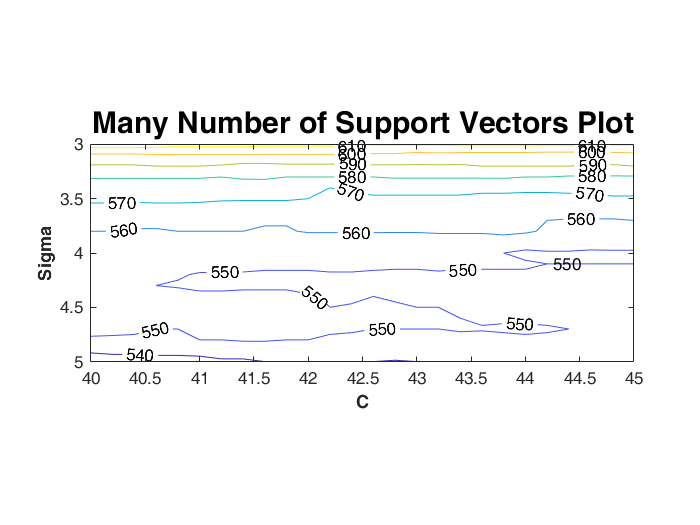


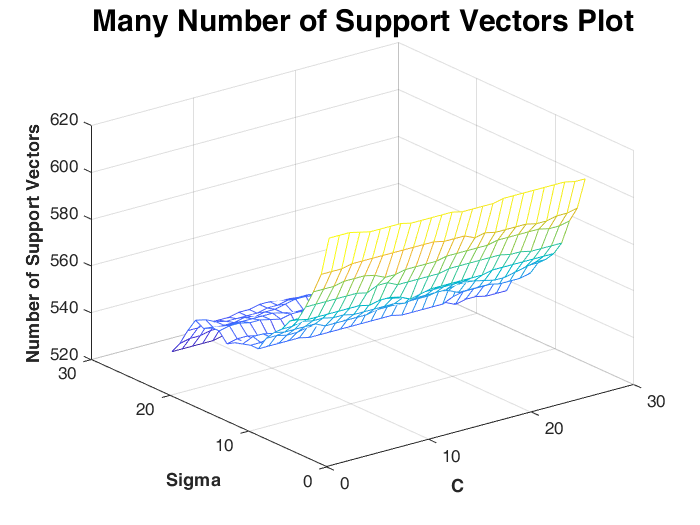
**Fig 8**: The contour plot and mesh plot of the support vectors for the grid search on sigma from 2 to 7 with step of 0.2 and c from 35 to 50 with step of 2.

To find reasonable number of support vectors and high accuracy at the same time, when sigma is in 3 to 5, accuracy can achieve 92% or higher. When c is in 40 to 45, the number of support vectors are around 550, which is fairly reasonable.



**Fig 9**: The contour plot of the accuracy for the grid search on sigma from 3 to 5 with step of 0.1 and c from 40 to 45 with step of 0.2.





**Fig 10**: The contour plot and mesh plot of the support vectors for the grid search on sigma from 3 to 5 with step of 0.1 and c from 40 to 45 with step of 0.2.

With the new range, accuracy and number of support vectors is stable at 92% and 550, which is pretty reasonable and acceptable result for us.

Finally, we decide our value for sigma and c to be 4.2 and 40.

**3.2 Result with hyper-parameters**

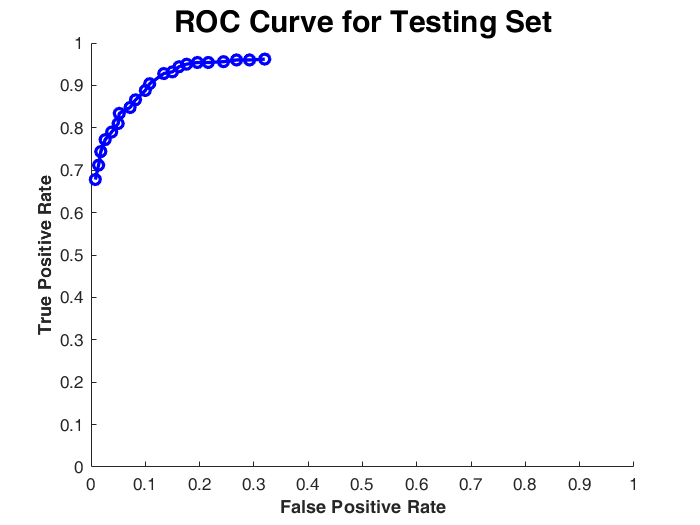
Using this two values for our hyper-parameters for the SVM training, we got 279(true positive), 276(true negative), 24(false positive), 21(false negative) in the validation set. The true positive rate is 93%, false positive rate is 8% and the accuracy is 92.5%. The number of support vectors used is 552. The accuracy is rather high and the number of support vectors is reasonable.

We had 500 sunset pictures and 500 non-sunset pictures for testing. With the same value for hyper-parameters, we got 452(true positive), 446(true negative), 54(false positive), 48(false negative) in the validation set. The true positive rate is 90.4%, false positive rate is 10.8% and the accuracy is 89.8%. The number of support vectors used is 552. The result does not have much difference with the one we got from the validation set, which is satisfying.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Image set | TP | TN | FP | FN | TPR | FPR | ACC |
| validation | 279 | 276 | 24 | 21 | 93% | 8% | 92.5% |
| Test set | 452 | 446 | 54 | 48 | 90.4% | 10.8% | 89.8% |

**Table 1**: Results from validation set and test set with 552 support vectors

We tried to change the threshold of our classification. Its default value is 0. When we changed it, it will affect the decision of an image. Therefore, we got several groups of different truth positive rates and false positive rates. When we increased the threshold, some false positives were identified as true negatives, but also some true positives were identified as false negatives. Therefore, both true positive rate and false positive rate decreased. Similarly, if we decrease the threshold, the true positive rate and false positive rate both increased. We went over the threshold from -1 to 1 with 0.1 to be the interval.



**Fig 11**: ROC curve for the testing set with threshold from -1 to 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TPR | 0.962 | 0.96 | 0.96 | 0.956 | 0.954 | 0.954 | 0.95 |
| FPR | 0.32 | 0.292 | 0.268 | 0.244 | 0.216 | 0.196 | 0.176 |
| TPR | 0.944 | 0.932 | 0.928 | 0.904 | 0.888 | 0.866 | 0.848 |
| FPR | 0.162 | 0.15 | 0.134 | 0.108 | 0.1 | 0.082 | 0.072 |
| TPR | 0.834 | 0.81 | 0.79 | 0.772 | 0.744 | 0.712 | 0.678 |
| FPR | 0.052 | 0.05 | 0.038 | 0.026 | 0.018 | 0.014 | 0.008 |

**Table 2**: Many TPR and FPR we used to plot ROC curve.

We calculate the distance of between each point and top-left corner (0,1), and found out the smallest value. When the value of threshold is 0, we had the smallest distance. Therefore, we would use 0 to classify our data.

**3.3 Example Pictures Analysis**

After getting the result, we want to step back to the original image to see what kind of image it can correctly classify or not.

Here are the two images that are true positive.



**Fig 12**: Sunset images correctly were labeled as sunsets.

Obviously, these two pictures are two types of typical sunset scenes. The first has a center point, which is the Sun. The colors around it change according to their distance to the center point. The overall hue is about red, orange, and yellow. There is also a horizontal line at the bottom, which color is much darker.

The second image presents a cloudy scene. We can hardly find a center point that shows the Sun. However, the overall color keeps the style of sunset scene. It can still be recognized as a sunset.

Here is the images that are true negative.



**Fig 13**: Non-sunset images were correctly labeled as non-sunsets.

The first picture is blue-sky scene. There is no sun in this image. At the same time, the main color of this picture is not red, yellow, or orange. The cloud makes a diagonal in the image and the sunset will not distribute like this in the sky. It is easy for the computer to label this picture as non-sunset.

The second picture is a purple flower and green leaves with a pink background. The spatial distribution of the flower is vertical which will not confuse the computer. The color also has a large difference with what a sunset scene should have.

Here is the images that are false negative.



**Fig 14**: Sunset images incorrectly were labeled as non-sunsets.

The first image is a picture of sunlight reflected from water. From the overall color of the picture, we can know that it is the light of a sunset scene. However, it is concerning that the information about sunset is vague here. It does not have a center point of light. The color is not distinguishable enough for a sunset scene. The animals in the middle may affect the data of the color because they are almost black. Usually, we will have black at the bottom to be considered as the horizontal line. However, in this picture, it has black at the top of the image, which may be a reason of misclassifying.

The second image displays the sun that is surrounded by noises. Most of the pictures do not have noises around the sun. This one is much more difficult to be recognized as sunset scene because the overall color is blue. Although it has a shining center point, the pixels around it are mainly noises with black color. It does not have the information of the changing colors around it.

Here are the images that are false positive.



**Fig 15**: Non-sunset image were incorrectly labeled as sunset.

The first picture is a tree with yellow leaves all over the place. The color is quite similar to the sunset scene. The distribution of the color is broad. The computer can hardly tell it is the light of the Sun or the original color of the object. This picture even has some smoothly changing colors on the side.

The second picture is an orange flower. It also has a similar color and a wide spread of that color. We believe the reason is similar to the first picture. Since we depend heavily on colors, similar colors is the main reason that causes misclassification.

**4. conclusions**

If we were granted another two weeks, we would like to try to use some different kernels. They may be the polynomial kernel, which is also in the Matlab or some other useful kernels from the Internet. Use several different validation sets of images to get more general and accurate hyper-parameters. We also want to process the image first in some different ways to improve the result or just see what difference the method will bring.

If we have another year, we would like to include some other useful features, like the edge of the sun. We may also try some different neural networks. We can also add some other things to classify together. For example, we can train the computer to recognize a scene is a sunset or a sunrise or neither of them.

With our sunset detector using SVM, we can recognize whether an image is a scene of sunset or not. We increased its accuracy and made a trade-off with efficiency. The concept can also be applied to recognize other objects that processes obvious features and a large number of training sets.

**5. References**

[1] Boutell, M., Jiebo Luo, and Robert T. Gray. "Sunset scene classification using simulated image recomposition." *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*. Vol. 1. IEEE, 2003.

[2] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003): 1-16.