Classification of Pixel Data using a Support Vector Machine

Laura McCrackin, Alex Scigajlo, Jamie Turner
Electrical & Computer Engineering Department, McMaster University

January 31, 2013

Contents

List of Figures

Chapter 1

Operational Information

1.1 Overview

Classifying the picture accurately and efficiently means generating a basis upon which to relate the output of the neural network with that of a human-based classification. To generate such a basis, we need to construct three major parts:

- Class Mask
- Training Point Pool
- Kernel & Hyperplane Selection

The class mask will designate to the neural network which part of the training image belongs to which class. The training point pool is used to allow certain parts of the mask to be weighted more than other parts when training the neural network for classification. The kernel and hyperplane functions are chosen to maximize the accuracy at similar point pool sizes.

1.2 Class Masking

To classify the picture as closely as possible to the desired result, it was necessary to review the details of the image for any features that could be extracted at an elementary level. The elementary level in this case is the raw image data available through individual pixel values. The support vector machine implementation in Matlab allowed for as many classification parameters to be passed for generating the classification structure as requested.

In our case, we narrowed the classification parameters down to a subset of the following:

- Cartesian Coordinates (X,Y)
- L*a*b* pixel values
- Gaussian radial blur L*a*b*
- Sobel edge-detected L*a*b*

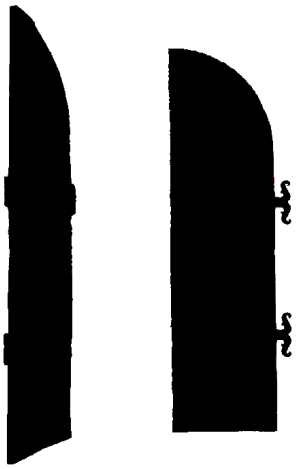


Figure 1.1: Classification Mask used to determine class of training points

When using the class mask, as seen in figure 1.1, it was necessary to generate a set of points to apply to the mask which maximized the positive effect on the support vector machine while reducing the requirement for an overly high number of required points.

The support vector machine internally tries to maximize the distance between the points it is currently training against and the desired hyperplane(Haykin, 2008b). The pixel data at image points are known, while the hyperplane is unknown. Randomly generating points from the entire images with no concern for position doesn't benefit the areas in the image where the distinction between both classes is quite complex. This is the reason for generating points specifically in relation to the class mask.

1.3 Point Generation and Pooling

A method for promoting certain parts of the image while also allowing for automation of point generation is shown below. By applying a Gaussian blur filter to the previous hard-edged mask, a new mask where the point densities increase relative to the distance to the mask boundary is generated. By edge-detecting and Gaussian blur filtering the class mask, the image can be extracted with ease from the class mask.

The training point generator works by sampling points through probability specified by the pixel values on this density mask. Visually the colormap, as shown in 1.3, is an example in which the points will be more likely to be generated from. For small training sizes, there is no guarantee that points will exist from those levels in the produced training set as the pool itself is sampled and culled to bring forth the final training set.

Once we have a pool of training points with higher density point nearest the edges of the mask, we can choose a kernel function to execute the neural network with.

Figure 1.2: Mask relating desired point density to position

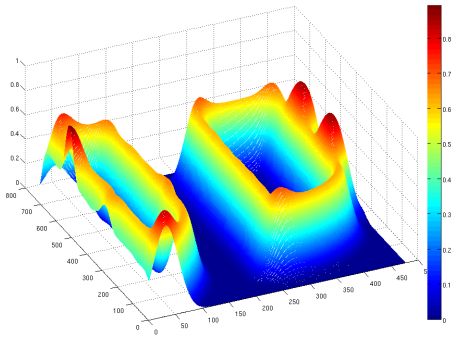


Figure 1.3: Colormap demonstrating point densities at different pixel locations

1.4 Kernel Function

For selecting the kernel to be used in the support vector machine, it was necessary to test those available to us from Matlab. The kernel functions available to us were as follows:

- Linear
- Quadratic
- Polynomial
- Radial Basis Function (RBF)
- Multilayer Perceptron (MLP)

We decided to use the RBF kernel as it worked well in conjunction with the image processing done during training. To compare the kernels, we can examine the accuracy at sampling 1000 points from the image (2/3rds for training, 1/3rd for testing). When using the MLP kernel, we noticed the accuracy dropped almost 40 percentile points for a similar calculation time to the RBF. Interestingly enough, the polynomial function was not only faster than the RBF and MLP but also

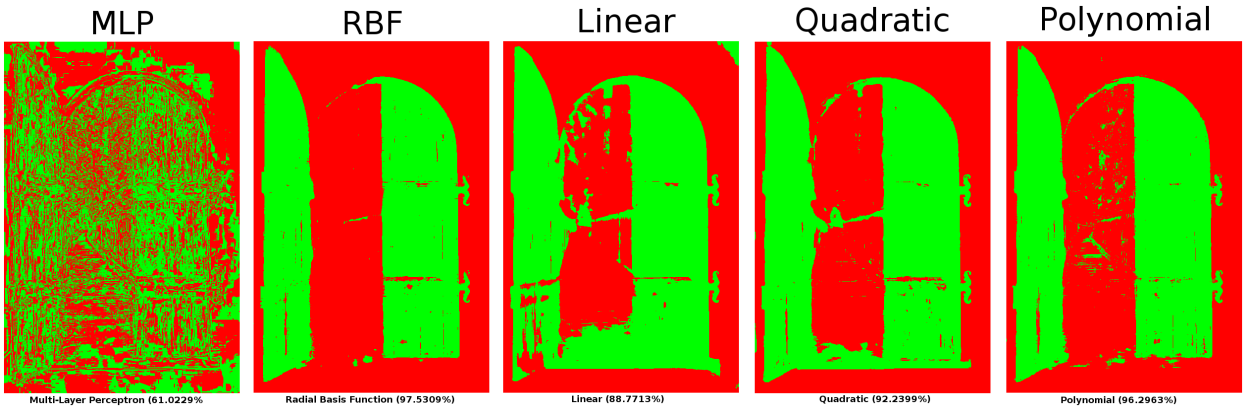


Figure 1.4: Comparison of the Different kernels using 1000 total points

had a comparable accuracy to the RBF. The quadratic kernel function had a time almost halfway between the linear and RBF kernels.

After deciding on a kernel function, it was necessary to determine a function to minimize for calculating the hyperplane.

1.5 Hyperplane Method

For the method of determining the appropriate hyperplane, we had a few choices available within Matlab. The choices available to us were:

- Least Squares (LS)
- Quadratic Programming (QP)
- Sequential Minimal Optimization (SMO)

Quadratic Programming was highly resource intensive for generating a classifier in respect to both Least Squares and Sequential Minimal Optimization. When using QP, Matlab failed to finish generating the hyperplane, thus we were unable to generate a classifier with the resources we had with QP. Due to this fact, we had to exclude it from our methods.

When comparing least squares versus sequential minimal optimization, there was no appreciable difference between the accuracies of the two. The differences between the images in Figure 1.5 are very minimal, to the degree of the sixth significant digit or more. In the end, we decided to use the "Least Squares" method.

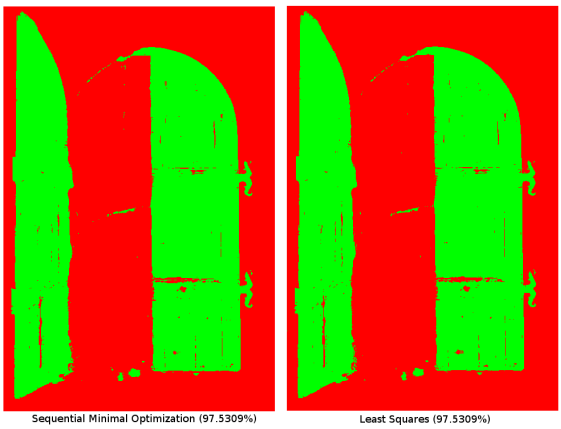


Figure 1.5: Comparison between methods of Least Squares and Sequential Minimal Optimization

Chapter 2

Parameter Selection

When choosing the parameters to train the support vector machine, we decided to expand on the data available from the image. By applying certain filters to the image, it is possible to expand on the features of the image without requiring manual editing of the image.

2.1 X and Y Coordinates

Perhaps the most obvious parameters to use are the X and Y-coordinates of each pixel. The door area is confined to two very specific, continuous areas of the image; the boundaries of this image, however, are not simple, and to satisfactorily represent them, a very high number of sample points would be required.

2.2 L*a*b* Colour Values

Another, simpler observation is that the door largely consists of a single colour: brown. There are, of course, other brown regions in the image, and there are considerable variations in the shades of brown throughout the door, but this provides a decent guess.

This observation can be further strengthened by representing the colours of image pixels in terms of luminance and chrominance channels, as specified by the CIELAB (L*a*b*) colour system, which is designed to closely model human visual perception. It consists of three channels: L*, the luminance channel (where 0 is black, and 100 is diffuse white); a*, the red/magenta-green channel (where positive values are red, and negative are green); and b*, the blue-yellow channel (where positive values are yellow, and negative values are blue) (Berns, 2000). The conversion from sRGB pixel values to L*a*b* values is well-known, and is performed in the following manner (McLaren, 1976).

First, the CIE XYZ tristimulus values are obtained from the sRGB values. sRGB component values R_{srgb}, G_{srgb}, and B_{srgb} are normalized by dividing by 255 to achieve values in the range 0 to 1. Then, for each C_{srgb}, where $C \in R, G, B$:

$$C_{linear} = \begin{cases} \frac{C_{\text{srgb}}}{12.92}, & C_{\text{srgb}} \leq 0.04045 \\ \left(\frac{C_{\text{srgb}}+a}{1+a}\right)^{2.4}, & C_{\text{srgb}} > 0.04045 \end{cases} \tag{2.1}$$

where a = 0.055.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R_{\text{linear}} \\ G_{\text{linear}} \\ B_{\text{linear}} \end{bmatrix} \tag{2.2}$$

Next, the tristimulus values are converted to CIELAB values:

$$\begin{aligned} L^\star &= 116 f(Y/Y_n) - 16 \\ a^\star &= 500\left[f(X/X_n) - f(Y/Y_n)\right] \\ b^\star &= 200\left[f(Y/Y_n) - f(Z/Z_n)\right] \end{aligned} \tag{2.3}$$

where

$$f(t) = \begin{cases} t^{1/3} & \text{if } t > (\frac{6}{29})^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases} \tag{2.4}$$

and Xn, Yn and Zn are the tristimulus values of the reference white point. In this case, as defined for the sRGB gamut:

$$\begin{aligned} X_n &= 0.3127 \\ Y_n &= 0.3290 \\ Z_n &= 0.3583 \end{aligned} \tag{2.5}$$

The L*, a*, and b* channels of the source image after it has been converted from sRGB in this manner may be observed in Figure 2.1. It is interesting to note that JPEG compression artifacts are much more visible on the a* and b* channels than on the L* channel; many image compression algorithms take advantage of the fact that L* colour information is the most perceptually important, and leave this channel relatively untouched while heavily compressing the remaining, less important colour information (Sayood, 2005).

Because the L*a*b* colour space is designed for perceptual uniformity, the perceived colour difference between any two points is fairly accurately defined by the Euclidian distance between them, as specified in the CIELAB ΔE^*_{ab} 1976 standard:

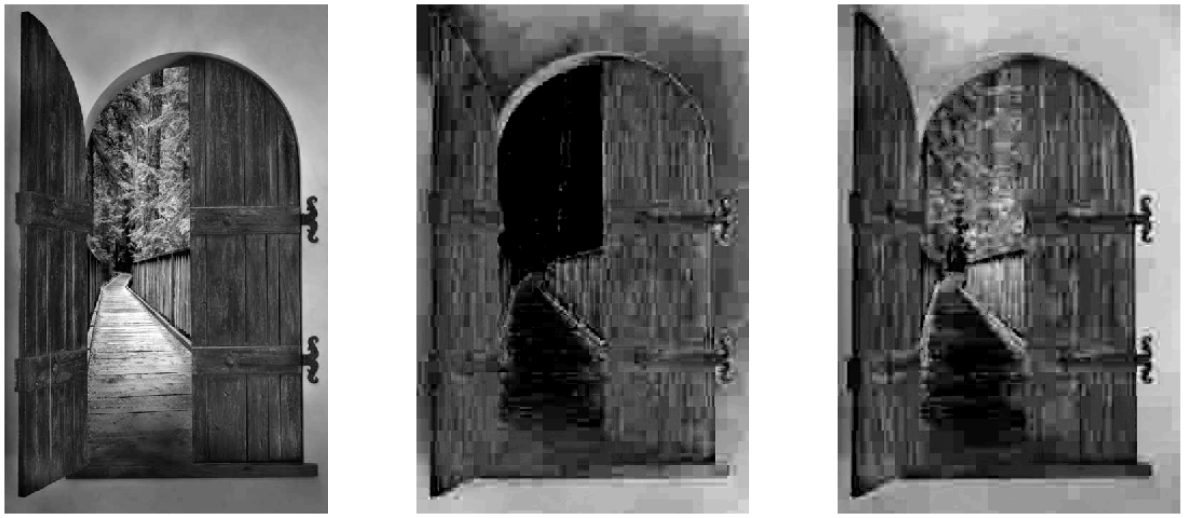
$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \tag{2.6}$$

It should be noted that the square of this function is of a very similar form to the RBF kernel function, which also features the Euclidian distance between samples:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||_2^2}{2\sigma^2}\right) \tag{2.7}$$

Therefore, with appropriate training data, the RBF kernel function will loosely approximate the ΔE^*_{ab} function.

Preserving the luminance of the image in a channel of its own is also beneficial for another reason. This L^* channel contains the most perceptually important colour information; it may be thought of as a greyscale version of the image, and indeed is a popular choice for simple greyscale conversion. For this image in particular, most regions of the door appear to have similar brightness levels, a feature that would be nearly impossible to infer by using the RGB colour values instead.



(a) L* channel (b) a* channel (c) b* channel

Figure 2.1: The three colour channels of the sample image in the L*a*b* colour space.

2.3 Gaussian Blurred L*a*b* Colour Values

The L*a*b* colour values of each pixel provide a good deal of useful information about the location of the pixel, as the colour value of neighbouring pixels are very likely to be similar. However, this is not always the case, as parts of the image may contain high-frequency detail. To compensate for these anomalies, the average colour values of a small surrounding pixel area may be used. Specifically, a Gaussian low-pass filter may be used on the image's L*a*b* pixel values.

A Gaussian filter of large radius size places a higher weighting on neighbouring pixels at a greater distance from the pixel of interest; a smaller radius better preserves the colour data of the pixel of interest at the expense of not including data from more distant neighbours. For this application, a Gaussian filter with a circular radius of 5 pixels was arbitrarily chosen as a suitable compromise between the two.

2.4 Sobel Values

Horizontal and vertical Sobel edge-detection filters were also used to pre-process pixel data. They are defined as follows:

$$\mathbf{G}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

where A represents the L^* channel of the image.

These filters assign each pixel a value corresponding to the amount of local pixel value change along the x and y directions, respectively. A high Sobel value indicates that a pixel occurs in an

area of the image that has pronounced colour change; under these circumstances, it may be better to place a higher weighting on the colour values of neighbouring pixels than to simply consider the colour values of the pixel itself during classification. Including Sobel values in the training data set, therefore, allows such a correlation to be inferred, and the classifier should then be able to weight the L*a*b* and Gaussian-blurred L*a*b* values accordingly.

Chapter 3

Testing & Results of Support Vector Classification

3.1 Classification of Testing Data

To determine optimal performance, the system was setup with different numbers of training points. A variable number of parameters were used to test the improvement in performance of the SVM classification. The testing data was gathered by hand, and later automated for ease of testing. After gathering the points they were classified into their respective categories with a premade mask selecting whether specific points were part of the door or not. The final accuracy was measured as a percentage of correctly classified points for the total image, and of points that were specifically near where the optimal hyperplane should be. As mentioned before, the main parameters tested are:
XY (Pixel Location)
L*a*b* Pixel value
L*a*b* Radial Blur
L*a*b* Sobel

3.2 Overall Effect

To evaluate the numerous types of parameters we could specify for the support vector machine to operate on, the order of effectiveness can be determined by the relative accuracy increase through its inclusion into the training data. The accuracies presented are from 33% of the total sample points used – where 66% are for training.

(a) L*a*b* Pixel data – 89.87% accuracy

(b) Gaussian Radial Blur (L*a*b* pixel data) – 90.16% accuracy

(c) XY Pixel Location – 82.53% accuracy

(d) Sobel information – 61.82% accuracy

(e) L*a*b*, Gaussian Radial Blur, and Sobel – 92.22% accuracy

(f) XY, L*a*b*, Gaussian Radial Blur, and Sobel – 94.71% accuracy

Figure 3.1: Results of Differing SVM information, individual testing – using 2000 training points. Accuracy values are from edge case testing, total value is higher

As seen in figure 3.1, it was determined that the inputs that yielded the most optimal results were the XY position, L*a*b* colour data, the Sobel filter data, and the Gaussian Radial Blur. In effect, the pixel position, the pixel colour, the derivation, and the integration of the pixel values. These have shown to be effective parameters for classification. The hardest parts for the SVM to properly classify were areas with both similar colour and texture to the door. In these areas, the SVM had a higher probability to return false positive results.

3.3 Outcome

By limiting the data sent to the training set for the support vector machine, it was possible to observe a performance of 90% and above for even small amounts of training data. The position of these points had a greater effect on the results than when the number of points was higher. The variance in the accuracy with differing test sets decreased when the total number of points increased. As expected, the results were very sensitive to the position of the training points; more points will cause the support vectors to be further away from the optimal hyperplane

Using X and Y as the additional input information was good at a higher number of training points, do not show as good of a performance alone, or when using smaller training sets. The

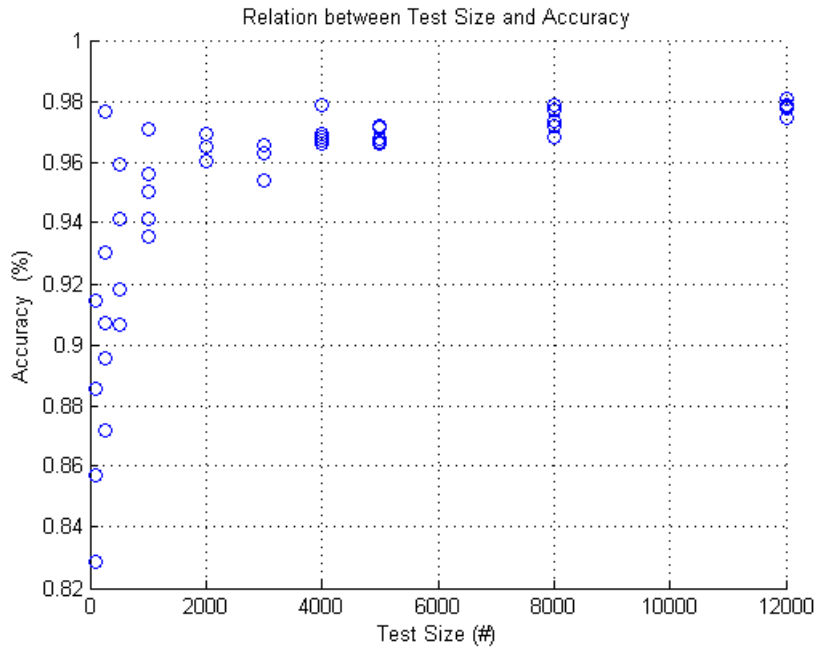


Figure 3.2: Changes in accuracy in the SVM from variance in the training set size.

L*a*b* Gaussian Radial blur information was by far the best for contributing to overall accuracy.

Total Points	**Training Set size**	**Testing set accuracy**	**Overall accuracy**
100	66	80%	90.33%
3000	1980	96.08%	97.53%
6000	3960	96.72%	98.11%
12000	7920	97.48%	98.57%

As seen in figure 3.3 there is a obvious diminishing returns effect that is hit quite quickly when increasing the number of points that are used to train the SVM. While it is possible to get good results at low number of sample points, the performance depends far more on the position of the points.

(a) 100 Sample points

(b) 3000 Sample points

(c) 6000 Sample points

(d) 12000 Sample points

Figure 3.3: Results of Differing sample points

Bibliography

Berns, R. (2000). *Billmeyer and Saltzman's Principles of Color Technology*. Wiley, New York.

Haykin, S. (2008a). *Section 4.13 Cross-Validation*, pages 171–172. Prentice Hall.

Haykin, S. (2008b). *Section 6.1 Introduction*, pages 268–269. Prentice Hall.

Haykin, S. (2008c). *Section 6.2 Optimal Hyperplane for Linearly Separable Patterns*, pages 269–280. Prentice Hall.

McLaren, K. (1976). XIII - The Development of the CIE 1976 (L* a* b*) Uniform Colour Space and Colour-difference Formula. *Journal of the Society of Dyers and Colourists*, **92**(9), 338–341.

Sayood, K. (2005). *Introduction to Data Compression (Third Edition)*. Morgan Kaufmann.