Project # 4: Object Detection Part 2: Classification

Instructor: V. Paúl Pauca

Jianqiu Xu (Tony)

1. Introduction

In last project, the goal is to experiment on characterization of images in terms of representative features extracted using a variety of global and local descriptors. I focused on the palm tree datasets to detect features of palm trees using HOG and LBP. In this project, I will focus on developing and testing classification approaches that use the image features I studied in Project #3 to solve the recognition problem of palm trees.

2. Methodology

2.1 Datasets

In the palm tree datasets, we have some images of canopies among which palm trees can be identified at various scales. In this project, I took three image as datasets, which were taken at equally high altitude shows much more canopy with palm trees spread out but still identifiable. Since we found that smaller patches have clearer and more accurate feature patterns, I manually cut out 110 small patches for both palm and non-palm features from these three images.

Figure 1. Three Image Data took at high altitude

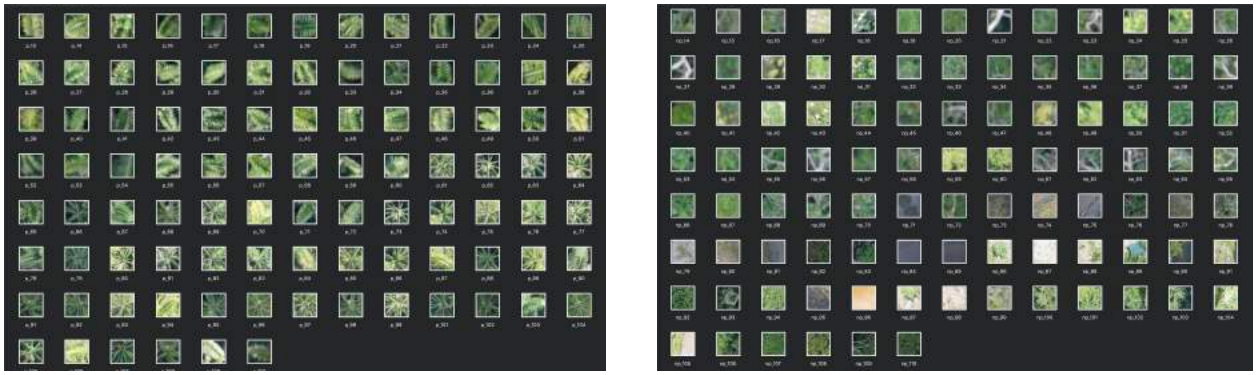


Figure 2. 110 small patches of palm feature and non-palm feature datasets

2.2 Histogram of Oriented Gradients (HOG)[1]

Histogram of Oriented Gradients is a feature descriptor used to do object detection. When we implement the HOG descriptor algorithm, we first divide the image into small connected regions called cells, and for each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell. Then, we discretize each cell into angular bins according to the gradient orientation. Each cell's pixel contributes weighted gradient to its corresponding angular bin. Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms. We take these blocks and contrast normalizes their overall responses before passing to next stage. Normalization introduces better invariance to illumination, shadowing, and edge contrast. Finally, normalized group of histograms represents the block histogram. The set of these block histograms represents the descriptor.

The computation of the HOG descriptor requires parameters including masks to compute derivatives and gradients, geometry of splitting an image into cells and grouping cells into a block, block overlapping, and normalization parameter.

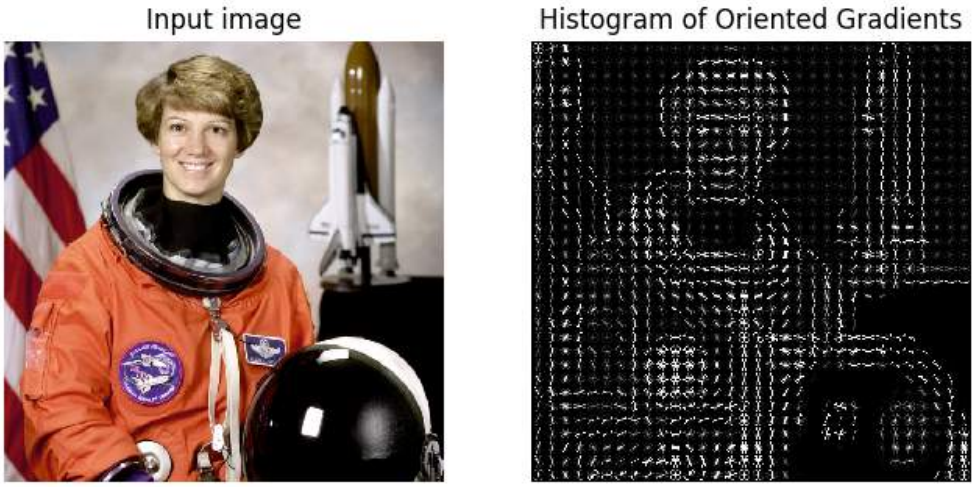


Figure 3. Example of input image and Histogram of Oriented Gradients

Parameters[2]*:*

- **pixels_pre_cell:** Size of HOG cell, specified in pixels as a 2-element vector. To capture large-scale spatial information, increase the cell size. When you increase the cell size, you may lose small-scale detail.
- **cells_pre_block:** Number of cells in a block, specified as a 2-element vector. A large block size value reduces the ability to suppress local illumination changes. Because of the number of

[1] Source: https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor

[2] Source: https://www.mathworks.com/help/vision/ref/extracthogfeatures.html?searchHighlight=HOG&s_tid=doc_srchtitle

pixels in a large block, these changes may get lost with averaging. Reducing the block size helps to capture the significance of local pixels. Smaller block size can help suppress illumination changes of HOG features.

- **orientation:** Number of orientation histogram bins, specified as positive scalar. To encode finer orientation details, increase the number of bins. Increasing this value increases the size of the feature vector, which requires more time to process.

```
Xhog = []
for image in X:
    fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1), visualize=True, multichannel=True)
    Xhog.append(fd)

Xhog = np.array(Xhog)
```

Figure 4. Implementation of HOG in the code

2.3 Support Vector Machines

Support vector machines are a set of supervised learning methods used for classification, regression and outliers detection. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N—the number of features) that distinctly classifies the data points[3].

SVM algorithms use a set of mathematical functions that are defined as the kernel. One of the advantages of support vector machine is its versatility, since we can implement different Kernel functions for the decision function, including linear, rbf, polynomial, sigmoid, and etc. The function of kernel is to take data as input and transform it into the required form.

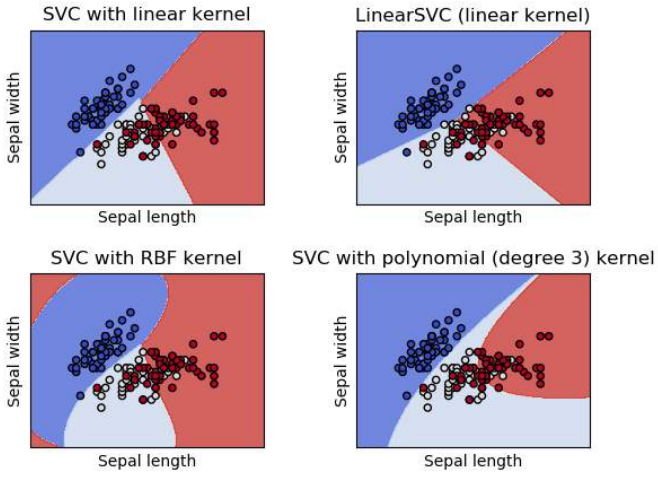


Figure 5. Example of Support Vector Machine with different kernels

[3] Source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

2.4 Confusion Matrix[4]

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

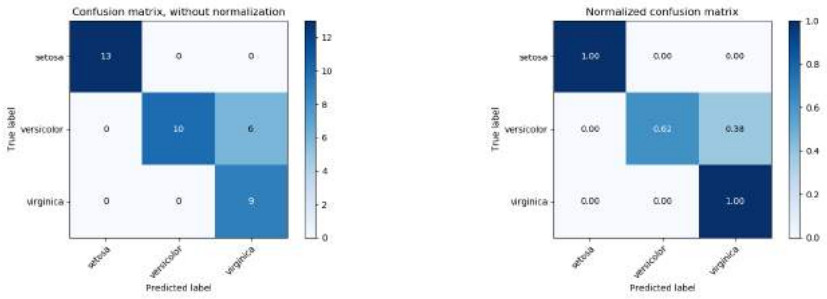


Figure 6. Example of Confusion Matrix

3. Experimental Results

In this project, I first converted all my 220 images into a X dataset and placed their labels of "palm" and "non-palm" into a y dataset. After splitting the training and test sets, I input my training sets to train the SVM and see the performance of the SVM using the test sets. I experimented the performance of SVM with different kernels for a fixed choice of HOG features.

3.1 Performance of SVM with different kernels

I tested the performance of SVM with different Kernel functions, including linear, rbf, and sigmoid, for a fixed choice of HOG features. I fixed my parameters for the hog feature classifier to orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1), which is the same setting as the last project. I set my test size to be 25% of my entire dataset. Noted that I have tested the performance of different test size, however, there is no obvious difference since I only have a relatively small dataset (Show in Figure 8).

```
# SVM
clf = svm.SVC(kernel='linear', gamma='scale') # or linear
clf = clf.fit(X_train, y_train)
y_fit = clf.predict(X_test)
```

Figure 7. SVM code (kernel is adjustable)

[4] Source: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

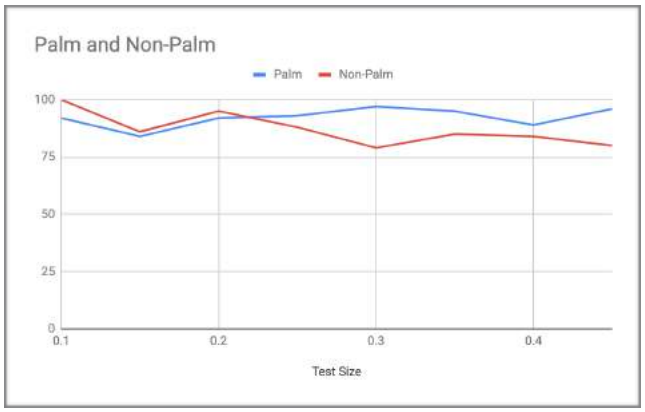
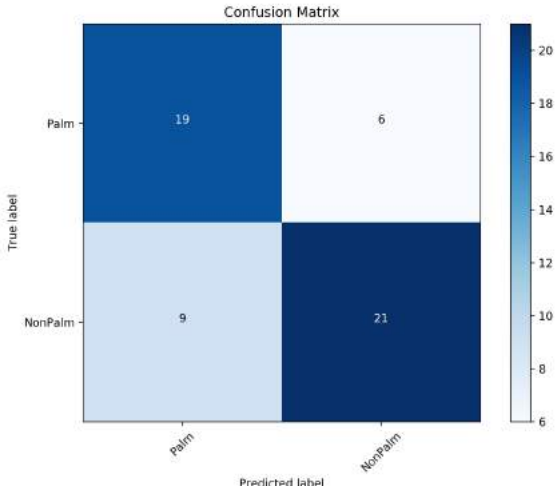


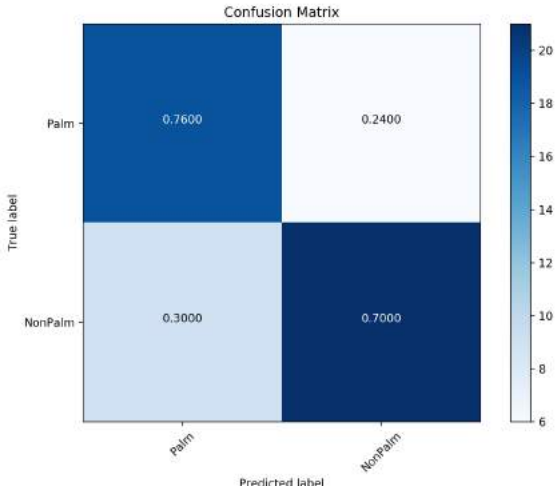
Figure 8. Performance of prediction accuracy with different test size

I tested performance of the SVM with linear, rbf, polynomial, and sigmoid kernels. I wrote code to print the classification report as well as the confusion matrix.

(a) SVC with linear kernel

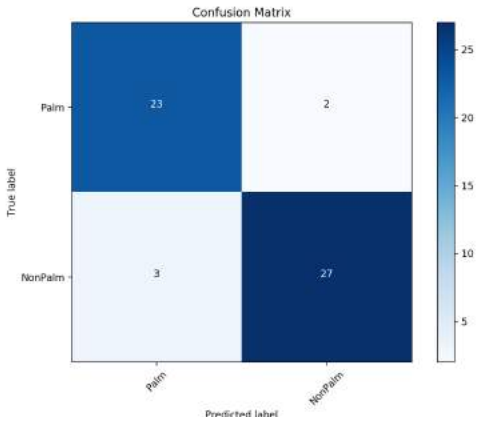
	precision	recall	f1-score	support
NonPalm	0.68	0.76	0.72	25
Palm	0.78	0.70	0.74	30
micro avg	0.73	0.73	0.73	55
macro avg	0.73	0.73	0.73	55
weighted avg	0.73	0.73	0.73	55

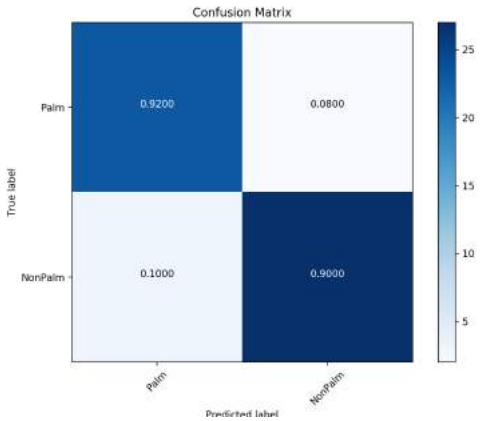




(b) SVC with rbf kernel

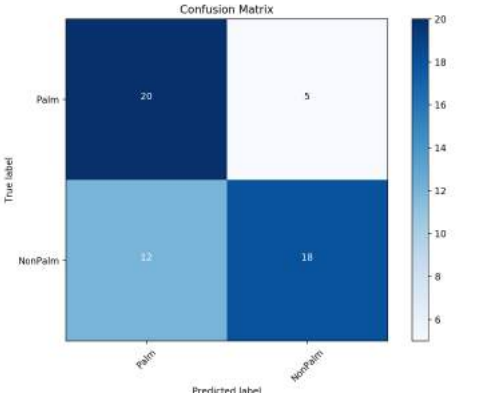
	precision	recall	f1-score	support
NonPalm	0.88	0.92	0.90	25
Palm	0.93	0.90	0.92	30
micro avg	0.91	0.91	0.91	55
macro avg	0.91	0.91	0.91	55
weighted avg	0.91	0.91	0.91	55

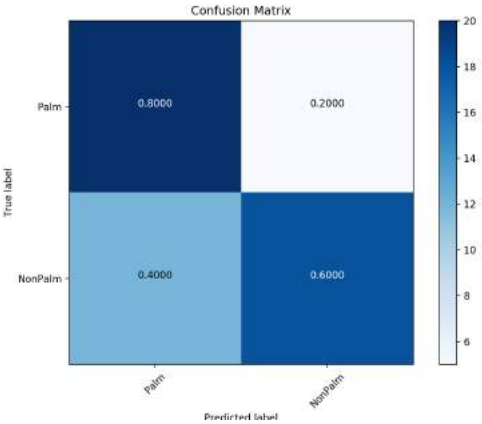




(c) SVC with polynomial kernel

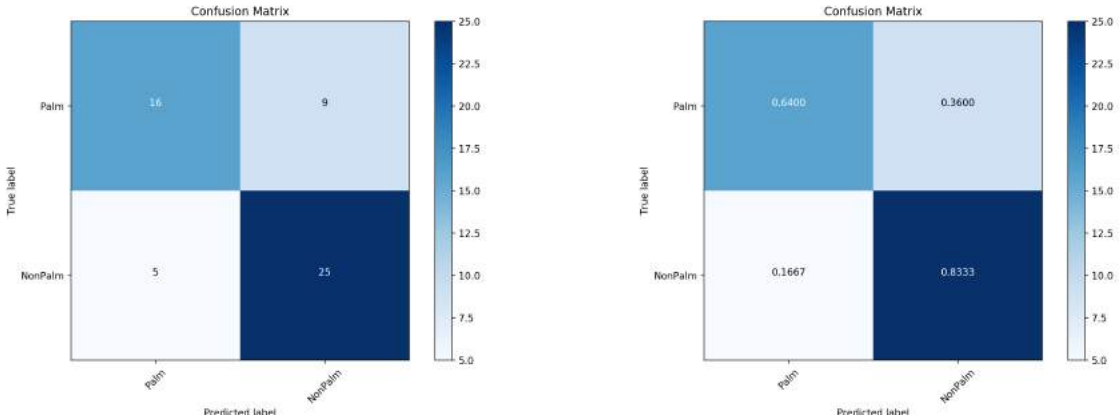
	precision	recall	f1-score	support
NonPalm	0.62	0.80	0.70	25
Palm	0.78	0.60	0.68	30
micro avg	0.69	0.69	0.69	55
macro avg	0.70	0.70	0.69	55
weighted avg	0.71	0.69	0.69	55





(d) SVC with sigmoid kernel

	precision	recall	f1-score	support
NonPalm	0.76	0.64	0.70	25
Palm	0.74	0.83	0.78	30
micro avg	0.75	0.75	0.75	55
macro avg	0.75	0.74	0.74	55
weighted avg	0.75	0.75	0.74	55



We can see that the Radial Basis Function (RBF) kernel has a much higher accuracy of prediction, comparing to the other three kernels, with a precision of 93% for palm features and 88% for non-palm features. The accuracy for non-palm feature prediction is usually lower than prediction for palm is might because that non-palm features are more diverse since I included not only non-palm trees patterns but also river and grass patterns, which makes it difficult to identify a specific feature of non-palm in the prediction. I believe this problem can be solved by training the SVM with a larger dataset.

When training an SVM with the Radical Basis Function (RBF) kernel, two parameters must be considered: C and gamma. I experimented on several C and gamma values to try to get a better prediction performance for SVM with rbf kernel.

3.2 Performance of SVM (rbf kernel) with different C and gamma value

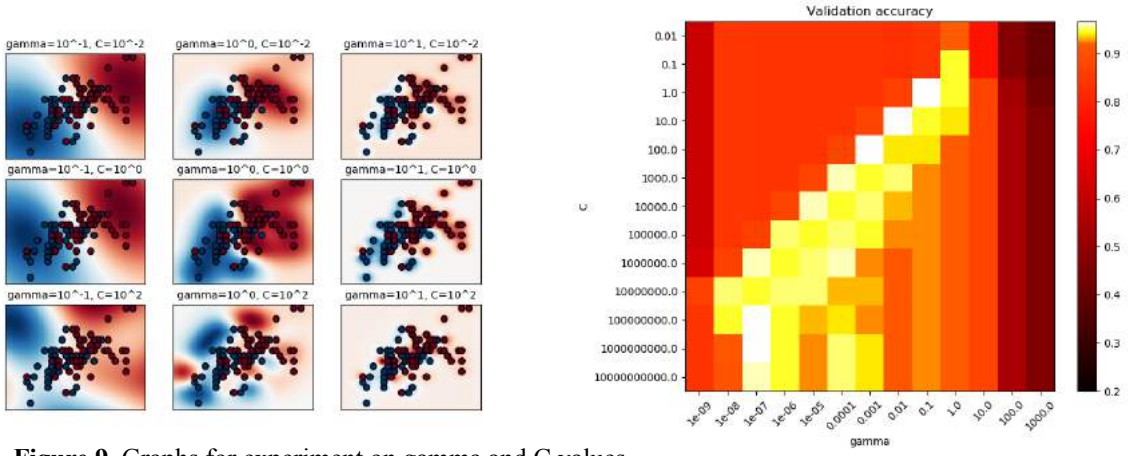
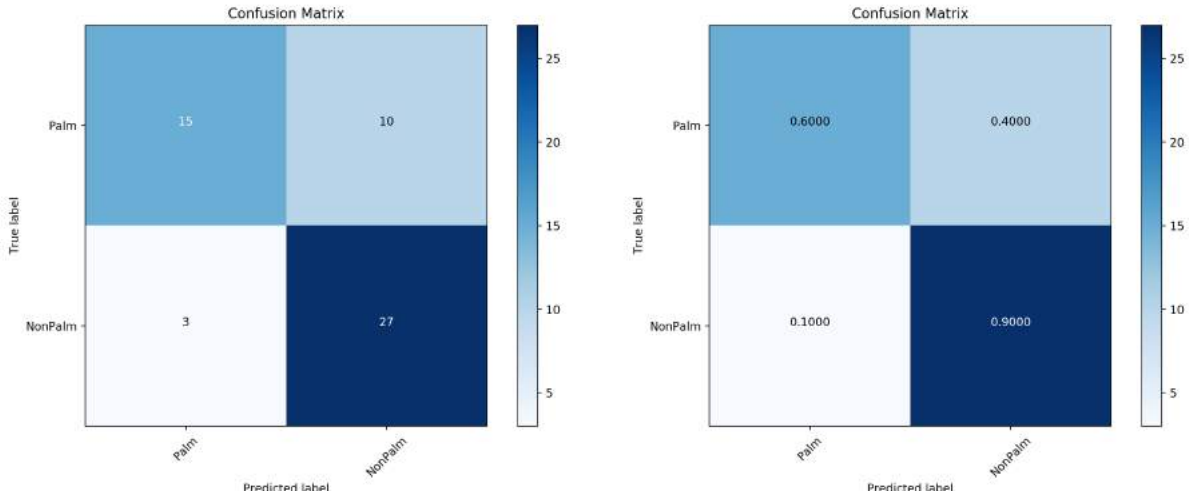


Figure 9. Graphs for experiment on gamma and C values

The parameter C, common to all SVM kernels, trades off misclassification of training examples against simplicity of the decision surface, while gamma defines how much influence a single training example has.

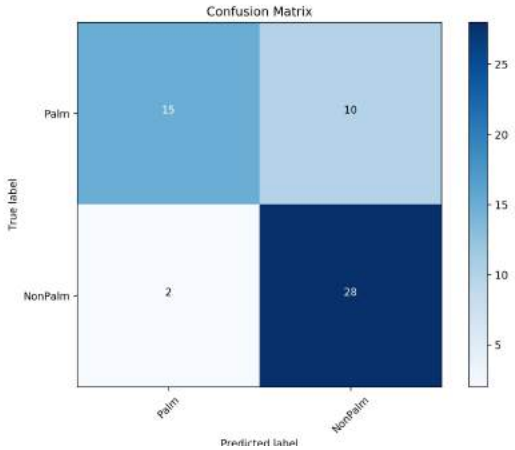
(a) C = 1000, gamma = 0.0001

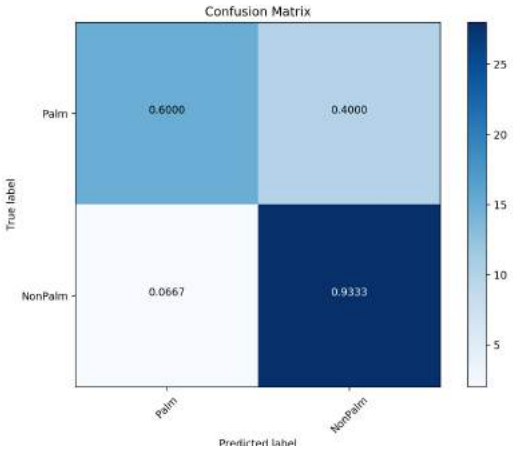
	precision	recall	f1-score	support
NonPalm	0.83	0.60	0.70	25
Palm	0.73	0.90	0.81	30
micro avg	0.76	0.76	0.76	55
macro avg	0.78	0.75	0.75	55
weighted avg	0.78	0.76	0.76	55



(b) C = 10, gamma = 0.01

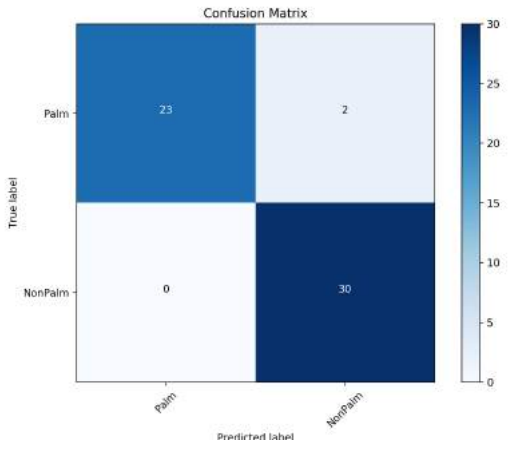
	precision	recall	f1-score	support
NonPalm	0.88	0.60	0.71	25
Palm	0.74	0.93	0.82	30
micro avg	0.78	0.78	0.78	55
macro avg	0.81	0.77	0.77	55
weighted avg	0.80	0.78	0.77	55

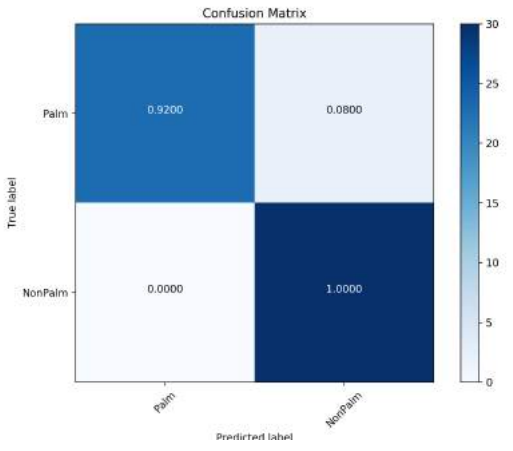




(c) C = 1, gamma = 1

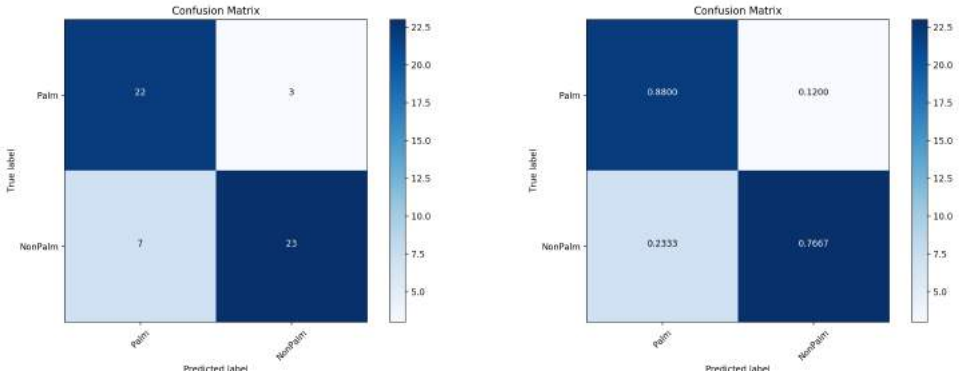
	precision	recall	f1-score	support
NonPalm	1.00	0.92	0.96	25
Palm	0.94	1.00	0.97	30
micro avg	0.96	0.96	0.96	55
macro avg	0.97	0.96	0.96	55
weighted avg	0.97	0.96	0.96	55





(d) C = 0.1, gamma = 10

	precision	recall	f1-score	support
NonPalm	0.76	0.88	0.81	25
Palm	0.88	0.77	0.82	30
micro avg	0.82	0.82	0.82	55
macro avg	0.82	0.82	0.82	55
weighted avg	0.83	0.82	0.82	55



After experimenting C value ranged from 1000 to 0.1 and gamma value ranged from 0.0001 to 10, the precision of prediction first increased and then decreased. The best parameters are {C: 1.0, gamma: 1.0} with a score of 100% accuracy on non-palm and 94% accuracy on palm features, which is much higher compared to the previous accuracy when we did not adjust C and gamma.

As we said above, the parameter C, common to all SVM kernels, trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. Gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.

This explains the increase in accuracy of prediction when I decreased C and increased gamma. The decrease in precision when I further decreased C to 0.1 and increased gamma to 10 might be because that the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting, since gamma is too large. The behavior of the model is very sensitive to the gamma parameter. When gamma is very small, the model is too constrained and cannot capture the complexity or “shape” of the data.