Fruit Classification

5/13/2022

1. Overview

We are going to do fruit classification using Apples, Bananas, and Oranges classes. Each of these classes have 10 images.

2. Design and Feature Set

1. **File Reading:** First, we read the image using the imread() command
2. **Image Resizing:** Then, we do the resizing using the `imresize()`

3. **File Preprocessing:** Then, we have done the noise removal, binarization using OTSU method, then we calculate the wavelet coefficients using the `2D Wavelet Transform.`

4. **Feature Matrix:** Then, we will calculate the vertical, horizontal and diagonal coefficients, and subsequently, their max values. We concatenate these values and store in a csv file. The total number of features are 4.

5. **Dataset:** Then, this csv file is appended with the labels in the last column. This represents our dataset. We have two csv files, namely, *training.csv* and *test.csv*. The training file has

3. Specifications of Model

In order to train our model, we turn a linear SVM into a Multi Label one.

3.1. Pseudocode for SVM

```
Input Dataset[X,Y]; X(array with n features), Y(array of labels)
for learning_rate in number_of_runs:
        error=0;
        for value in X:
            if(Y[values]*(X[value]*w))<1
              then
                w+=learning_rate*(Y[value]*(X[value])*(-2*(1/number_of_runs)*w)
            else
```

```
            w+=learning_rate*(-2*(1/number_of_runs)*w)
        end if
        end
end
```

3.2. Mathematical Functions

Let's discuss the equations behind SVM. Firstly, dot product is intensely used like

$$\overrightarrow{X}.\overrightarrow{w} = c \quad (the\ point\ lies\ on\ the\ decision\ boundary)$$

$$\overrightarrow{X}.\overrightarrow{w} > c\ (positive\ samples)$$

$$\overrightarrow{X}.\overrightarrow{w} < c\ (negative\ samples)$$

We take an arbitrary value of c, and if X dot c is less than 0, X belong to negative class. If X.c>0, then X belongs to positive class. If X.w=0, then, the point is on decision boundary.

Similarly, we also make use of margins. To classify objects, we need a margin that has the most distance between two classes.

To classify images as negative and positive, we make use of the following rules:

$$\overrightarrow{X}.\overrightarrow{w} - \mathrm{c} \geq 0$$

$$\text{putting } -\mathrm{c} \text{ as } \mathrm{b}, \text{we get}$$

$$\overrightarrow{X}.\overrightarrow{w} + \mathrm{b} \geq 0$$

$$\text{hence}$$

$$\mathrm{y} = \begin{cases} +1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + \mathrm{b} \geq 0 \\ -1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + \mathrm{b} < 0 \end{cases}$$

Optimization Function:

$$\mathrm{argmax}(\mathrm{w}^*, \mathrm{b}^*) \frac{2}{\|\mathrm{w}\|} \text{ such that } \mathrm{y}_\mathrm{i}\left(\overrightarrow{w}.\overrightarrow{X} + b\right) \geq 1$$

3.2.1. SVM and its kernels:

SVM supports different types of kernels, let's discuss the sigmoid one.

It is defined as follows:

```
f(x, y) = tanh(alpha * x^T * y + C)
```

So, we are just taking a point a mapping it to 0 or 1, so it can be separated by a line. With our data, rbf kernel was performing the best. The equation for it is as follows:

```
f(x1, x2) = exp(-gamma * ||x1 - x2||^2)
```

Gamma signifies how much effect a single training instance has on the other points around it.

4. Implementation and Testing

4.1. Manual for Users:

1. In order to generate the data, and preprocess it, use the file preprocess_images.m. It will convert your data into an xls format that you can easily feed to your SVM model, or any other model. You have to provide the function the absolute path of the directory where the data is, and also the class label to which it belongs.
2. Then, for training and testing, we use the main.m file. You have to simply replace the training and testing data files names in the code to your own, and provide class labels using an array. Then, without passing any argument, run the main.m file. It will also generate classification report.

4.2. Table of training and testing files.

In the Data\Apples directory:

`

Training	Testing
1.jpeg	8.jpeg
2.jpeg	9.jpeg
3.jpeg	10.jpeg
4.jepg	
5.jpeg	

6.jpeg	
7.jpeg	

In the Data\Oranges directory:

`

Training	Testing
1.jpeg	8.jpeg
2.jpeg	9.jpeg
3.jpeg	10.jpeg
4.jepg	
5.jpeg	
6.jpeg	
7.jpeg	

In the Data\Bananas directory:

`

Training	Testing
1.jpeg	8.jpeg
2.jpeg	9.jpeg
3.jpeg	10.jpeg
4.jepg	
5.jpeg	
6.jpeg	
7.jpeg	

4.3. Classification Report

4.3.1. Confusion Matrix:

Multi-Class Confusion Matrix Output

	TruePositive	FalsePositive	FalseNegative	TrueNegative
Actual_class1	3	0	1	5
Actual_class2	1	2	0	6
Actual_class3	3	0	1	5

4.3.2. Performance Metrics Per Class:

AccuracyOfSingle	ErrorOfSingle	AccuracyInTotal	ErrorInTotal	Sensitivity	Specificity	Precision	FalsePositiveRate	F1_score
0.75	0.25	0.33333	0	0.75	1	1	0	0.85714
1	0	0.11111	0.22222	1	0.75	0.33333	0.25	0.5
0.75	0.25	0.33333	0	0.75	1	1	0	0.85714

MatthewsCorrelationCoefficient	Kappa	TruePositive	FalsePositive	FalseNegative	TrueNegative	Class
0.79057	0.27778	3	0	1	5	'class1==>1'
0.5	0.58333	1	2	0	6	'class2==>2'
0.79057	0.27778	3	0	1	5	'class3==>3'

4.3.3. Overall Metrics:

Over all values

Accuracy:	0.7778
Error:	0.2222
Sensitivity:	0.8333
Specificity:	0.9167
Precision:	0.7778
FalsePositiveRate:	0.0833
F1_score:	0.7381
MatthewsCorrelationCoefficient:	0.6937
Kappa:	0.5000

4.3.4. Summary of the Results:

Our results show us that our model has some difficulty in classifying the orange and apple classes, which are classes one and three. That can be attributed to the similarity of orange and apple shape. This can be minimized by used more robust feature extraction techniques and increasing the training instances.