Introduction to Image Processing and Computer Vision

Laboratory Project 2

PLANT SPECIES RECOGNITION

Łukasz Niedziałek
niedzialekl@student.mini.pw.edu.pl

Contents

1 Problem description

Image recognition is the process of identifying and detecting an object or a feature in a digital image or video.[1]

Our goal was to assign leaves into different species based on photos of them. We had to create a classifier, train and then test it on photos provided by the teacher.

Figure 1 Leaves without labels.

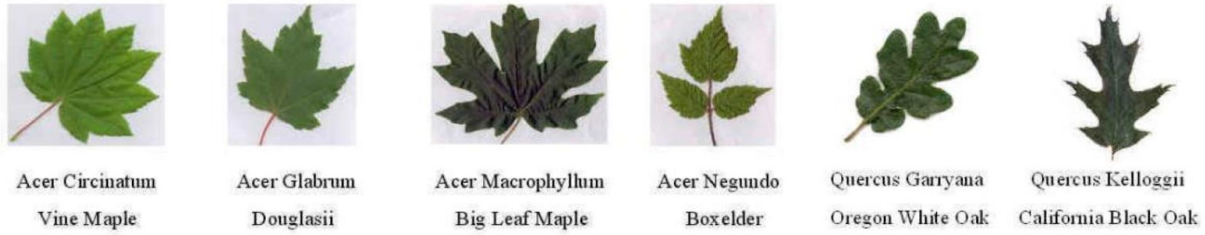


Figure 2 Leaves with assigned labels.

[1] https://www.mathworks.com/discovery/image-recognition.html

2 Solution

2.1 Pre-processing

Firstly I have prepared a simple architecture for the whole training and testing program. It consists of three components:

- class responsible for calculating probabilities based on Naive Bayes
- class responsible for reading and preparing of images and collecting information for a single type of leaves
- classes storing certain information about images of the same type

In the process of reading and preparing of images I have created functions that:

- apply simple color thresholding
- detect and separate the biggest contour inside the image
- return image with green color mask applied
- return image with biggest contour mask applied
- rotate given contour

2.2 Training

I have created fours factors that take part in the process of recognition:

- leaf's roundness
- leaf's slimness
- leaf's dispersion
- leaf's average green color value

Roundness is a feature describing a ratio of leaf's area compared to its perimeter.

```
def roundness(contour):
    perimeter = cv2.arcLength(contour, True)
    area = cv2.contourArea(contour)
    return (4 * 3.14159 * area) / (perimeter ** 2)
```

Slimness is a ratio of leaf's width compared to its height.

```
def slimness(contour):
    x, y, w, h = cv2.boundingRect(contour)
    ratio = float(w) / float(h)
    return ratio
```

Dispersion is a value describing leaf's irregularity.

```
def dispersion(contour):
    M = cv2.moments(contour)
    cx = int(M['m10'] / M['m00'])
```

```
    cy = int(M['m01'] / M['m00'])
    cx = float(cx)
    cy = float(cy)
    max_val = 0
    min_val = 1000000
    for t in contour:
        (x, y) = tuple(t[0])
        val = np.math.sqrt((float(x)-cx) ** 2 + (float(y) - cy) ** 2)
        if val > max_val:
            max_val = val
        if val < min_val:
            min_val = val
    return float(max_val / min_val)
```

For each of those features there exists a class containing all of the information from one class and being able to calculate a mean of this information as well as a variance.

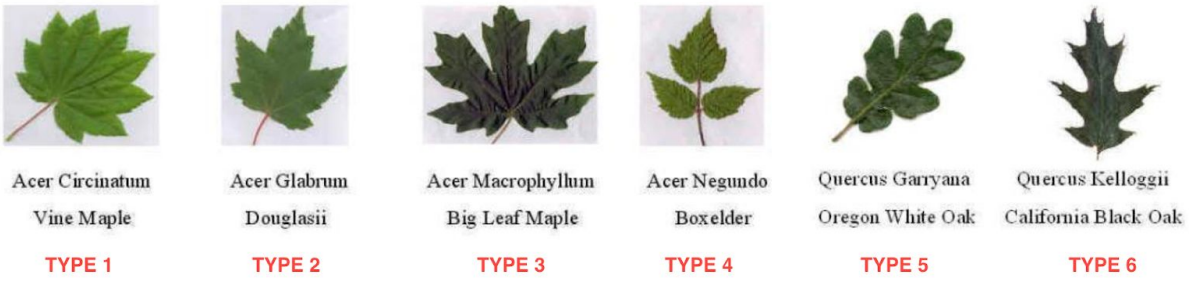
```
def get_mean(self):
      instances_num = len(self.values)
      sum_val = 0
      for r in self.values:
            sum_val += r
      return float(sum_val) / float(instances_num)

def get_variance(self):
      mean = self.get_mean()
      sum = 0
      for val in self.values:
            sum += (val - mean) ** 2
      return sum / len(self.values)
```

2.3 Testing

During the test phase I go through each of the pictures, that needs to be recognized, and perform exactly the same actions as in preparing phase. Then I compare obtained values with those assigned to each class, and calculate the probability (using Naive Bayes) that given photo belongs to the given class, and label it by placing in the corresponding folder. For the sake of simplicity of calculations, I assume that values are spread according to gaussian distribution.

```
  def get_prob(self, some_val):
        variance = self.get_variance()
        mean = self.get_mean()
        first = 1 / (np.sqrt(2 * 3.14159 * variance))
        second = np.exp(-1 *((some_val - mean) ** 2) / (2 * variance))
        return first * second
```

3 Results and conclusions



Achieved accuracy: (16.6% should be a result in case of complete random placement)

- 26.6% when applied only **color**
- 30.0% when applied only **slimness**
- 40.0% when applied only **roundness**
- 56.6% when applied only **dispersion**
- 65% when applied **slimness, roundness** and **dispersion**
- 68.3% when applied all four features (**slimness, roundness, dispersion and color**)
- 70.0% when applied **roundness, dispersion** and **color**
- 70.0% when applied **roundness** and **dispersion** (result is different than in previous case)

In both best cases the biggest number of leaves is assigned to type number three.

Roundness, dispersion and color:	**Roundness and dispersion:**
class: 1, proper: 9	class: 1, proper: 9
class: 2, proper: 5	class: 2, proper: 4
class: 3, proper: 9	class: 3, proper: 9
class: 4, proper: 8	class: 4, proper: 7
class: 5, proper: 5	class: 5, proper: 6
class: 6, proper: 6	class: 6, proper: 7
sum proper: 70.0%	sum proper: 70.0%

Dispersion seems to be the most valuable of the features in this comparison.

Roundness is a nice feature, but in the case of this dataset, leaves from the first four classes have very similar width/height ratios, and Naive Bayes does not attach weights to the features, so in case of this simple classifier it is not working so well.