Visión Artificial

Práctica 4: El clasificador de Bayes

David Martín Gómez
Laboratorio de Sistemas Inteligentes
Universidad Carlos III de Madrid

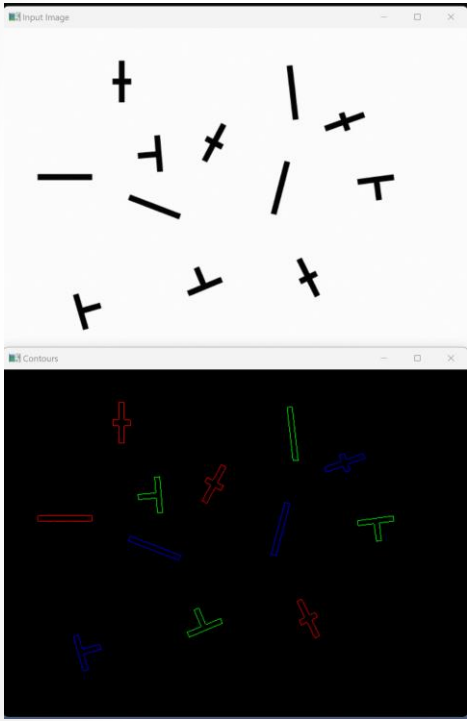
Índice

- Objetivos
- Etiquetado de objetos
- Obtención de descriptores
- El clasificador de Bayes

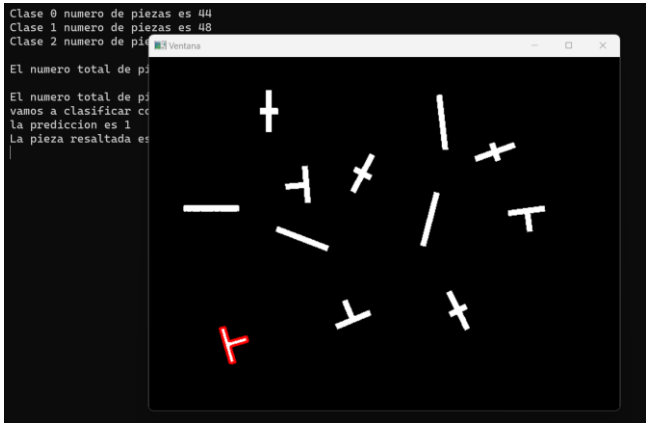
Índice

- Objetivos
- Etiquetado de objetos
- Obtención de descriptores
- El clasificador de Bayes

Objetivos



	A	B	C	D	E	F	G	H	I
1	perimetro	area	Hu1	Hu2	Hu3	Hu4	Hu5	Hu6	Hu7
2	164	540	0.862	0.714454	4.73E-05	3.97E-05	1.72E-09	3.35E-05	-9.62E-12
3	165	535	0.861	0.713894	6.41E-06	2.40E-06	8.56E-12	1.21E-06	-3.89E-12
4	168	533	0.873	0.733276	1.02E-05	2.26E-06	6.34E-12	3.11E-07	-8.85E-12
5	168	532	0.873	0.734509	6.56E-06	5.65E-06	3.44E-11	4.76E-06	5.20E-13
6	167	521	0.884	0.753504	1.70E-05	1.14E-05	1.59E-10	9.63E-06	9.88E-12
7	166	516	0.899	0.780266	3.78E-05	2.83E-05	9.22E-10	2.36E-05	-5.08E-11
8	168	520	0.888	0.760736	8.37E-06	1.26E-06	-1.47E-13	-4.05E-07	-4.09E-12
9	168	519	0.886	0.75758	2.45E-05	3.60E-06	-3.31E-12	-1.37E-06	3.36E-11
10	172	519	0.888	0.759855	2.48E-06	1.44E-06	2.70E-12	9.69E-07	4.05E-13
11	173	520	0.889	0.761395	1.37E-06	8.94E-07	9.84E-13	6.69E-07	-9.35E-14
12	174	518	0.889	0.761703	3.74E-05	2.88E-05	9.42E-10	2.37E-05	-4.57E-11



Etiquetado de objetos

- Imágenes
  - Pieza1.png, Pieza2.png, Pieza3.png, Piezas.png

```
using namespace cv;
using namespace std;

int main() {

    Mat image = imread("./imagenes/Piezas.png");

    if (!image.data) {
        cout << "Error!!\n";
        return 1;
    }
    imshow("Input Image", image);

    //Umbralizamos
    cvtColor(image, image, COLOR_BGR2GRAY);
    threshold(image, image, 128, 255, THRESH_BINARY_INV);

    //Encontrmaos los contornos
    vector<vector<Point> > contours;
    Mat contourOutput = image.clone();
    findContours(contourOutput, contours, RETR_LIST, CHAIN_APPROX_NONE);

    //Los dibujamos
    Mat contourImage(image.size(), CV_8UC3, Scalar(0, 0, 0));
    Scalar colors[3];
    colors[0] = Scalar(255, 0, 0);
    colors[1] = Scalar(0, 255, 0);
    colors[2] = Scalar(0, 0, 255);
    for (size_t idx = 0; idx < contours.size(); idx++)
            drawContours(contourImage, contours, idx, colors[idx % 3]);
    imshow("Contours", contourImage);

    waitKey(0);
    return 0;
}
```

Etiquetado de objetos

findContours() [1/2]

```
void cv::findContours ( InputArray image,
                        OutputArrayOfArrays contours,
                        OutputArray hierarchy,
                        int mode,
                        int method,
                        Point offset = Point()
                      )
```

Python:

```
cv.findContours( image, mode, method[, contours[, hierarchy[, offset]]] ) -> contours, hierarchy
```

`#include <opencv2/imgproc.hpp>`

Finds contours in a binary image.

The function retrieves contours from the binary image using the algorithm [239] . The contours are a useful tool for shape analysis and object detection and recognition. See squares.cpp in the OpenCV sample directory.

Note

Since opencv 3.2 source image is not modified by this function.

Parameters

- **image** Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary . You can use compare, inRange, threshold , adaptiveThreshold, Canny, and others to create a binary image out of a grayscale or color one. If mode equals to RETR_CCOMP or RETR_FLOODFILL, the input can also be a 32-bit integer image of labels (CV_32SC1).
- **contours** Detected contours. Each contour is stored as a vector of points (e.g. std::vector<std::vector<cv::Point> >).
- **hierarchy** Optional output vector (e.g. std::vector<cv::Vec4i>), containing information about the image topology. It has as many elements as the number of contours. For each i-th contour contours[i], the elements hierarchy[i][0] , hierarchy[i][1] , hierarchy[i][2] , and hierarchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.

RetrievalModes

enum cv::RetrievalModes

`#include <opencv2/imgproc.hpp>`

mode of the contour retrieval algorithm

Enumerator	
RETR_EXTERNAL Python: cv.RETR_EXTERNAL	retrieves only the extreme outer contours. It sets `hierarchy[i][2]=hierarchy[i][3]=-1` for all the contours.
RETR_LIST Python: cv.RETR_LIST	retrieves all of the contours without establishing any hierarchical relationships.
RETR_CCOMP Python: cv.RETR_CCOMP	retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.
RETR_TREE Python: cv.RETR_TREE	retrieves all of the contours and reconstructs a full hierarchy of nested contours.
RETR_FLOODFILL Python: cv.RETR_FLOODFILL	

ContourApproximationModes

enum cv::ContourApproximationModes

`#include <opencv2/imgproc.hpp>`

the contour approximation algorithm

Enumerator	
CHAIN_APPROX_NONE Python: cv.CHAIN_APPROX_NONE	stores absolutely all the will be either horizontal,
CHAIN_APPROX_SIMPLE Python: cv.CHAIN_APPROX_SIMPLE	compresses horizontal, up-right rectangular con
CHAIN_APPROX_TC89_L1 Python: cv.CHAIN_APPROX_TC89_L1	applies one of the flavo
CHAIN_APPROX_TC89_KCOS Python: cv.CHAIN_APPROX_TC89_KCOS	applies one of the flavo

Índice

- Objetivos
- Etiquetado de objetos
- Obtención de descriptores
- El clasificador de Bayes

Obtención de descriptores

- cv::arcLength
- cv::contourArea
- cv::convexHull
- cv::HuMoments
- cv::minAreaRect
- cv::minEnclosingCircle
- cv::minEnclosingTriangle
- cv::moments

Perímetro y área

arcLength()

```
double cv::arcLength ( InputArray  curve,
                       bool        closed
                     )
```

Python:

```
cv.arcLength( curve, closed ) -> retval
```

contourArea()

```
double cv::contourArea ( InputArray  contour,
                         bool        oriented = false
                       )
```

Python:

```
cv.contourArea( contour[, oriented] ) -> retval
```

Momentos

moments()

```
Moments cv::moments ( InputArray  array,
                      bool        binaryImage = false
                    )
```

Python:

```
cv.moments( array[, binaryImage] ) -> retval
```

HuMoments() [1/2]

```
void cv::HuMoments ( const Moments &  moments,
                     double           hu[7]
                   )
```

Python:

```
cv.HuMoments( m[, hu] ) -> hu
```

..y más

convexHull()

void cv::convexHull (Inpu
Outp
bool
bool
)

minAreaRect()

RotatedRect cv::minAreaRect (**InputA**

Python:

cv.minAreaRect(points) -> retval

minEnclosingCircle()

void cv::minEnclosingCircle (**InputArray** points,
Point2f & center,
float & radius

minEnclosingTriangle()

double cv::minEnclosingTriangle (**InputArray**
OutputArray
)

Python:

cv.minEnclosingTriangle(points[, triangle]) ->

boxPoints()

void cv::boxPoints (**RotatedRect** box,
OutputArray points
)

Python:

cv.boxPoints(box[, points]) -> points

`#include <opencv2/imgproc.hpp>`

Finds the four vertices of a rotated rect. Useful to draw the rotated rectangle.

The function finds the four vertices of a rotated rectangle. This function is useful to draw the rectangle. In C++, instead of using this function, you can directly use **RotatedRect::points** method. Please visit the **tutorial on Creating Bounding rotated boxes and ellipses for contours** for more information.

Parameters

box The input rotated rectangle. It may be the output of

points The output array of four vertices of rectangles.

Índice

- Objetivos
- Etiquetado de objetos
- Obtención de descriptores
- El clasificador de Bayes

El clasificador de Bayes

- Hay que crear las estructuras del clasificador.
  - Primero los datos

```
// creación de las matrices para almacenar la imformación
Mat train_data(nentre, NUMDESCRIPTORES, CV_32FC1);
Mat response_data(nentre, NUMDESCRIPTORES, CV_32FC1);
```

```
//Convertir los datos de una matriz al tipo TrainData
static Ptr<ml::TrainData> prepare_train_data(const Mat& data, const Mat& responses, int ntrain_samples)
{
    Mat sample_idx = Mat::zeros(1, data.rows, CV_8U);
    Mat train_samples = sample_idx.colRange(0, ntrain_samples);
    train_samples.setTo(Scalar::all(1));

    int nvars = data.cols;
    Mat var_type(nvars + 1, 1, CV_8U);
    var_type.setTo(Scalar::all(ml::VAR_ORDERED));
    var_type.at<uchar>(nvars) = ml::VAR_CATEGORICAL;

    return ml::TrainData::create(data, ml::ROW_SAMPLE, responses,
        noArray(), sample_idx, noArray(), var_type);
}
```

El clasificador de Bayes

- Hay que crear las estructuras del clasificador.
  - Luego entrenamos

```
//ponemos la informacion en el formato que le gusta a Opencv
Ptr<ml::TrainData> tdata = prepare_train_data(train_data, response_data, nentre);

//Creacion del clasificador de Bayes
Ptr<ml::NormalBayesClassifier> bayes = ml::NormalBayesClassifier::create();
// Entrenamos
bayes->train(tdata);
```

Member Function Documentation

◆ create()

static Ptr<NormalBayesClassifier> cv::ml::NormalBayesClassifier::create ( ) static

Python:

cv.ml.NormalBayesClassifier.create( ) -> retval

cv.ml.NormalBayesClassifier_create( ) -> retval

Creates empty model Use **StatModel::train** to train the model after creation.

◆ load()

static Ptr<NormalBayesClassifier> cv::ml::NormalBayesClassifier::load (const String & filepath,
const String & nodeName = String()
) static

Python:

cv.ml.NormalBayesClassifier.load(filepath[, nodeName]) -> retval

cv.ml.NormalBayesClassifier_load(filepath[, nodeName]) -> retval

Loads and creates a serialized **NormalBayesClassifier** from a file.

Use **NormalBayesClassifier::save** to serialize and store an **NormalBayesClassifier** to disk. Load the **NormalBayesClassifier** from this file again, by calling this function with the path to the file. Optionally specify the node for the file containing the classifier

Parameters

filepath path to serialized **NormalBayesClassifier**

nodeName name of node containing the classifier

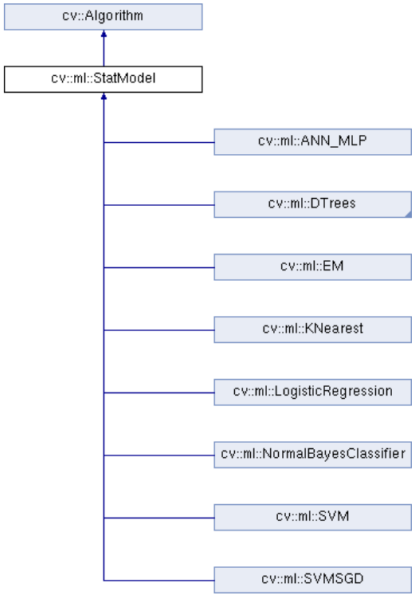
El clasificador de Bayes

cv::ml::StatModel Class Reference

Base class for statistical models in OpenCV ML. More...

`#include <opencv2/ml.hpp>`

Inheritance diagram for cv::ml::StatModel:



Public Member Functions

virtual float	**calcError** (const **Ptr**< **TrainData** > &data, bool test, **OutputArray** resp) const Computes error on the training or test dataset. More...
virtual bool	**empty** () const **CV_OVERRIDE** Returns true if the **Algorithm** is empty (e.g. in the very beginning or after unsuccessful read. More...
virtual int	**getVarCount** () const =0 Returns the number of variables in training samples. More...
virtual bool	**isClassifier** () const =0 Returns true if the model is classifier. More...
virtual bool	**isTrained** () const =0 Returns true if the model is trained. More...
virtual float	**predict** (**InputArray** samples, **OutputArray** results=**noArray**(), int flags=0) const =0 Predicts response(s) for the provided sample(s) More...
virtual bool	**train** (const **Ptr**< **TrainData** > &trainData, int flags=0) Trains the statistical model. More...
virtual bool	**train** (**InputArray** samples, int layout, **InputArray** responses) Trains the statistical model. More...