# 使用逻辑回归实现黄色车道线识别

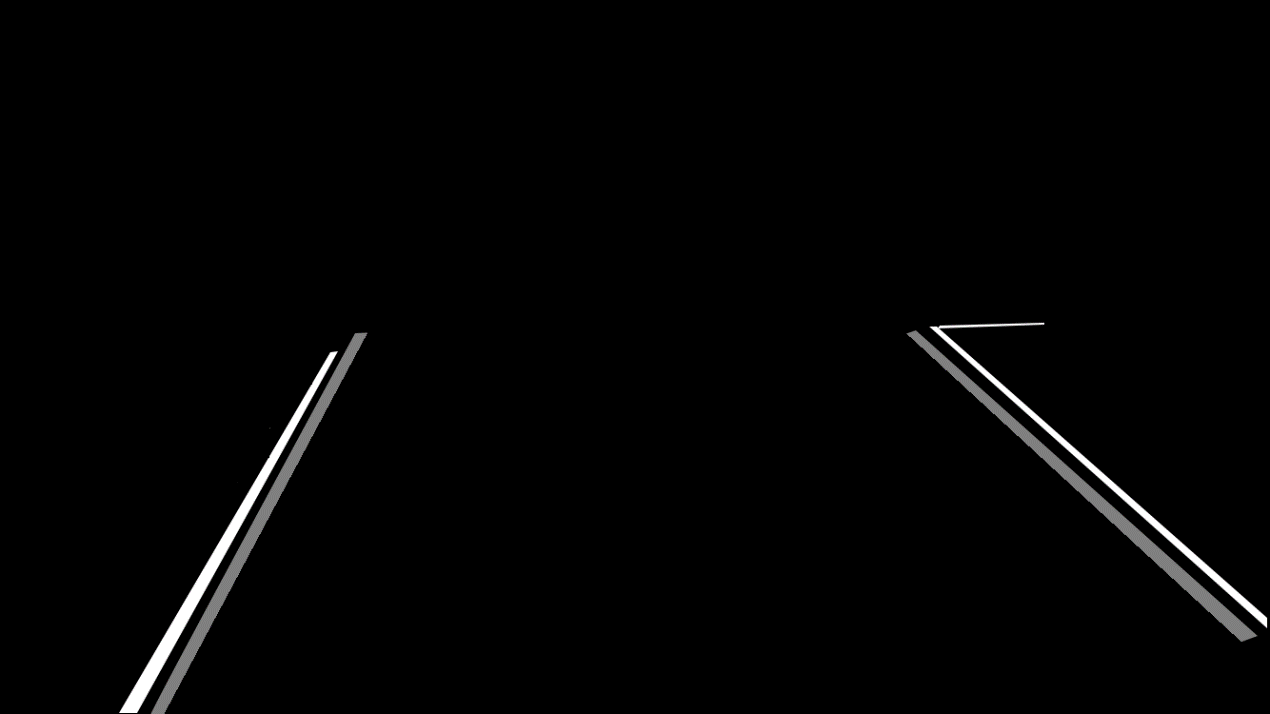
本实例主要演示python中数据的读取，sklearn的用法，以便在以后用的时候作参考。

有如下图像，需要识别图中的车道线，车道线为黄色，但由于地面反光的原因，车道线不易提取，为此，考虑使用简单的机器学习方法对车道线进行学习。



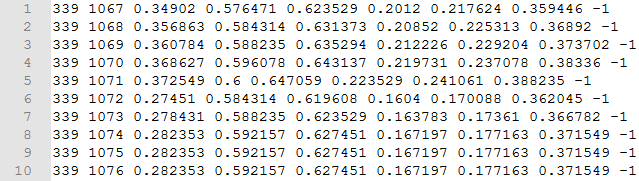
## 数据采集

首先，对车道线进行标记，收集车道线正负样本考虑收集正负样本数量大概为1：1。对车道线进行人为标记，如下图所示，正样本为白色，负样本为灰色，编写程序对像素值进行收集。考虑收集以下信息：B, G, R, B\*G, B\*R, G\*R，并对其归一化到0~1.0范围。



|  |
| --- |
| #include "stdafx.h"  #include <opencv2\opencv.hpp>  #include <fstream>  #include <iostream>  using namespace std;  using namespace cv;  int \_tmain(int argc, \_TCHAR\* argv[])  {  Mat mask = imread("mask.bmp", 0);  Mat src = imread("src.jpg", 1);  Mat bgr[3];  split(src, bgr);  ofstream file("data.txt");  for (int i = 0; i < src.rows; i++)  {  for (int j = 0; j < src.cols; j++)  {  int label = 0;  if (mask.at<uchar>(Point(j, i)) > 50 && mask.at<uchar>(Point(j, i)) < 200)  {  //负样本  label = -1;  }  else if (mask.at<uchar>(Point(j, i)) > 200)  {  //正样本  label = 1;  }  if (label == 0)continue;  float b = bgr[0].at<uchar>(Point(j, i));  float g = bgr[1].at<uchar>(Point(j, i));  float r = bgr[2].at<uchar>(Point(j, i));  b /= 255.0;  g /= 255.0;  r /= 255.0;  float bg = b\*g;  float br = b\*r;  float gr = g\*r;  file << i <<" " << j << " " << b << " " << g << " " << r  << " " << bg << " " << br << " " << gr << " " << label << endl;  }  }  file.close();  return 0;  } |

收集到的数据如图所示：



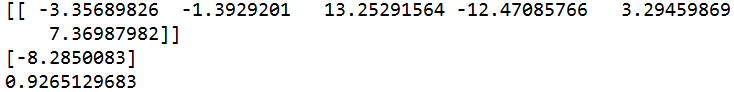
## 训练

使用python的sklearn包进行训练，这里主要是为了演示数据的读取和训练：

|  |
| --- |
| *#coding = utf-8* **import** numpy **as** np **from** sklearn.linear\_model.logistic **import** LogisticRegression **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.metrics **import** accuracy\_score  **def** loadData(filename):  numpy\_list=[]  **with** open(filename) **as** f:  line = f.readline()  **while**(line):  line.strip(**'\n'**)  line = line.split(**' '**)  line = [float(item) **for** item **in** line]  numpy\_list.append(line)  line = f.readline()  numpy\_list = np.array(numpy\_list)  dataSet = numpy\_list[:, 2:-1]  label = numpy\_list[:, -1]  label = label.astype(np.int32)  **return** dataSet, label   X, Y = loadData(**"data.txt"**) trainX, testX, trainY, testY = train\_test\_split(X, Y, test\_size=0.1) clf = LogisticRegression() clf.fit(trainX, trainY) print(clf.coef\_) print(clf.intercept\_)  prediction = clf.predict(testX) print(accuracy\_score(testY, prediction)) |

这里还要注意的是，逻辑回归得学习到的参数在程序中是clf.coef和clf.intercept\_，前者为各变量系数，后者为偏置项。

输出：



## 测试

将训练得到的变量参数和偏置代到测试程序中：

|  |
| --- |
| #include "stdafx.h"  #include <opencv2\opencv.hpp>  #include <fstream>  #include <iostream>  using namespace std;  using namespace cv;  void main()  {  Mat src = imread("src.jpg", 1);  Mat bgr[3];  split(src, bgr);  float coef[6] = { -3.35689826, - 1.3929201, 13.25291564, -12.47085766, 3.29459869, 7.36987982 };  float intercept = -8.2850083;  Mat lane(Size(src.cols, src.rows), CV\_8UC1, Scalar::all(0));  Mat b, g, r, bg, br, gr;  bgr[0].convertTo(b, CV\_32FC1, 1 / 255.0);  bgr[1].convertTo(g, CV\_32FC1, 1 / 255.0);  bgr[2].convertTo(r, CV\_32FC1, 1 / 255.0);  multiply(b, g, bg);  multiply(b, r, br);  multiply(g, r, gr);  Mat resultMat = coef[0] \* b + coef[1] \* g + coef[2] \* r + coef[3] \* bg + coef[4] \* br + coef[5] \* gr + intercept;  threshold(resultMat, resultMat, 0, 255, CV\_THRESH\_BINARY);  resultMat.convertTo(resultMat, CV\_8UC1);  imshow("resultMat", resultMat);  waitKey(0);  } |

