using System.Drawing;

using Emgu.CV.Structure;

using Emgu.CV.ML;

using Emgu.CV.ML.Structure;

...

int trainSampleCount = 150;

int sigma = 60;

#region Generate the training data and classes

Matrix<float> trainData = new Matrix<float>(trainSampleCount, 2);

Matrix<float> trainClasses = new Matrix<float>(trainSampleCount, 1);

Image<Bgr, Byte> img = new Image<Bgr, byte>(500, 500);

Matrix<float> sample = new Matrix<float>(1, 2);

Matrix<float> trainData1 = trainData.GetRows(0, trainSampleCount / 3, 1);

trainData1.GetCols(0, 1).SetRandNormal(new MCvScalar(100), new MCvScalar(sigma));

trainData1.GetCols(1, 2).SetRandNormal(new MCvScalar(300), new MCvScalar(sigma));

Matrix<float> trainData2 = trainData.GetRows(trainSampleCount / 3, 2 \* trainSampleCount / 3, 1);

trainData2.SetRandNormal(new MCvScalar(400), new MCvScalar(sigma));

Matrix<float> trainData3 = trainData.GetRows(2 \* trainSampleCount / 3, trainSampleCount, 1);

trainData3.GetCols(0, 1).SetRandNormal(new MCvScalar(300), new MCvScalar(sigma));

trainData3.GetCols(1, 2).SetRandNormal(new MCvScalar(100), new MCvScalar(sigma));

Matrix<float> trainClasses1 = trainClasses.GetRows(0, trainSampleCount / 3, 1);

trainClasses1.SetValue(1);

Matrix<float> trainClasses2 = trainClasses.GetRows(trainSampleCount / 3, 2 \* trainSampleCount / 3, 1);

trainClasses2.SetValue(2);

Matrix<float> trainClasses3 = trainClasses.GetRows(2 \* trainSampleCount / 3, trainSampleCount, 1);

trainClasses3.SetValue(3);

#endregion

using (SVM model = new SVM())

{

SVMParams p = new SVMParams();

p.KernelType = Emgu.CV.ML.MlEnum.SVM\_KERNEL\_TYPE.LINEAR;

p.SVMType = Emgu.CV.ML.MlEnum.SVM\_TYPE.C\_SVC;

p.C = 1;

p.TermCrit = new MCvTermCriteria(100, 0.00001);

//bool trained = model.Train(trainData, trainClasses, null, null, p);

bool trained = model.TrainAuto(trainData, trainClasses, null, null, p.MCvSVMParams, 5);

for (int i = 0; i < img.Height; i++)

{

for (int j = 0; j < img.Width; j++)

{

sample.Data[0, 0] = j;

sample.Data[0, 1] = i;

float response = model.Predict(sample);

img[i, j] =

response == 1 ? new Bgr(90, 0, 0) :

response == 2 ? new Bgr(0, 90, 0) :

new Bgr(0, 0, 90);

}

}

int c = model.GetSupportVectorCount();

for (int i = 0; i < c; i++)

{

float[] v = model.GetSupportVector(i);

PointF p1 = new PointF(v[0], v[1]);

img.Draw(new CircleF(p1, 4), new Bgr(128, 128, 128), 2);

}

}

// display the original training samples

for (int i = 0; i < (trainSampleCount / 3); i++)

{

PointF p1 = new PointF(trainData1[i, 0], trainData1[i, 1]);

img.Draw(new CircleF(p1, 2.0f), new Bgr(255, 100, 100), -1);

PointF p2 = new PointF(trainData2[i, 0], trainData2[i, 1]);

img.Draw(new CircleF(p2, 2.0f), new Bgr(100, 255, 100), -1);

PointF p3 = new PointF(trainData3[i, 0], trainData3[i, 1]);

img.Draw(new CircleF(p3, 2.0f), new Bgr(100, 100, 255), -1);

}

Emgu.CV.UI.ImageViewer.Show(img);