1. 课程介绍

1.人脸探测和人脸标记.

2.人脸探测与人脸标记的相关算法.

3.学习 sklearn, dilb 等库函数的安装和使用.

2. 库函数介绍

2.1. Dlib 的 Python 接口

Dlib 是一个现代的 C++工具库. 它的特征:它包含很多机器学习算法,以及用来创建复杂软件以解决真实问题的很多工具. Dlib 的应用领域很广, 包括嵌入式设备,机器人技术, 移动电话,和超级计算机的计算环境. Dlib 是免费的. (http://dlib.net/)

使用方法:

在 python 文件或者交互环境中调用

```
import dlib
```

安装 dlib 库的 Python 接口的方法:

你可以直接用命令

```
pip install dlib
```

你也可以自己编译 dlib, 进入 dlib 的根目录,运行

```
python setup.py install
```

注意: 只要操作系统安装了合适的 CMake,就可以完成 dlib 的编译.

参考: Python API: http://dlib.net/python/index.html

2.2. skimage 库

skimage 是很多图像处理算法汇集在一起构成的 Python 库.

安装 skimage 的方法:

```
pip install scikit-image
```

在课程中,我们会使用 skimage 中的 io 模块.

```
from skimage import io
```

3. 人脸探测的原理

将原图片转换成 **HOG** 图片以后，结合其他辅助技术，如**线性分类器**，**影像金字塔**，和**滑动窗口** (sliding window)检查机制，就可以生成人脸探测器．

3.1. 线性分类器

我们通过举例来了解线性分类算法．

假设有一个$N = 1000$(1000 张**图片**)的训练集，每个图像有$D = 32 \times 32 = 1024$个像素，图片被分成了$K = 3$类(猫，人脸，汽车)．如何将原始图像分到各个类别中呢？可以定义一个操作f

$$f: R^D \rightarrow R^K$$

这个操作f将原始像素值(**1024 个数值**)映射到各类别的**得分**(**3 个数**)．

一个线性分类器，就是一个线性映射：

$$f(x_i, W, b) = Wx_i + b$$

图像数据x_i可以看成一个向量，它的长度为D． 输入值x_i包含第i个图像的所有像素的值．本例中，x_i的长度为 1024． 它可以写成一列数．例如：

$(20, 20, 200, 203, \ldots, 255)^T$， 符号$m^T$ 表示将m转置．

x_i的取值： $x_1, x_2, .., x_N$中的任何一个．

矩阵W的尺寸为$K \times D$． 本例中，W为3×1024． W称为**权重**．

参数b也是一个向量，它的长度为K． 参数b称为偏差向量． 参数b会影响输出值．

总之，这个操作的作用：输入 1024 个值(一张图片)，输出 3 个数值(也就是 3 个不同分类，得分)．

注意：训练数据用来学习参数W和b，学习完成后，我们可以舍弃训练集．

我们将每张图看成是**高维空间中的一个点** (**一列数**)．本例中，每张图是 1024 维空间中的一个点(1024 行 1 列)．整个数据集就是所有 1000 个点的集合． 每个点都带有一个分类标签． 分类的得分是权重W和图像x_i的矩阵乘积，每个分类就是这个高维空间中的一个线性**函数的值**．

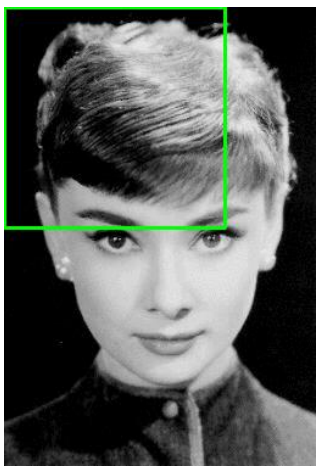
例．矩阵乘法举例：$W x_i + b$

```
0.5*50 + (-0.2*200) + 0.2 *25 +(-4* 2) +1  = -17
1.2*50 +  2 * 200  +  4 * 25  +  0*2  +   5   = 575
0*50 + 0.2*200  + (-1.6*25) + (-0.5 * 2) -1 = -2
```

猫	0.5	-0.2	0.2	-4		50		1	-17
人脸	1.2	2	4	0		200	+	5	= 575
车	0	0.2	-1.6	-0.5		25		-1	-2
						2			
	W					x_i		**b**	$Wx_i + b$

3.2. 滑动窗口检测

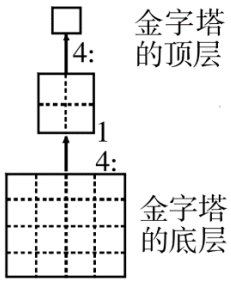
滑动窗口检测机制是机器学习中的一个概念， 滑动窗口是一个具有固定宽和高的矩形区域，我们用它滑过图片，以求探测到有趣的**模式**． 滑动窗口如下图绿色矩形框所示．

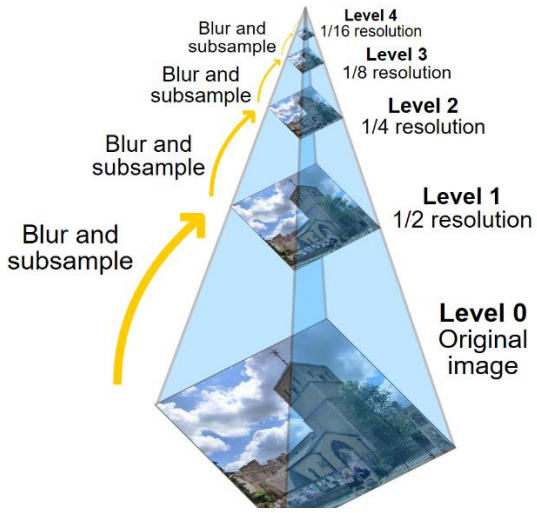


3.3. 影像金字塔

影像金字塔由原始影像按一定规则生成的由**细到粗**不同分辨率的影像集． 第一层是原始图片，每上一

层,图片就按照一定的比例缩小(变得更模糊). 像素;3*3=9 像素(cell); (3*3cells,区间)





4. 人脸探测的步骤

S1. 获取图片名称; 将图片保存进一个数组

用 io.imread()读取图片文件.

S2. 建立一个 HOG 人脸探测器 (工具: dlib)

dlib.get_frontal_face_detector()

返回一个人脸探测器**对象**.

S3. 运行 HOG 人脸探测器于图片数据

返回结果: 人脸的边界盒子.

实例化对象.

打印出探测到的人脸的位置.

S4．建立窗口,显示图片

dlib.image_window()

S5．把照片中的每一张人脸都画出一个边界盒子.

利用 for 循环,分别对[A, B]中的 A,B 进行处理.

Enumerate()函数.

窗口对象的 add_overlay()方法.

5. 人脸标识的步骤

S1.获取图片文件名；并加载图片

io.imread(file_name)

S2.用 dlib 内置类创建 HOG 人脸探测器

dlib.get_frontal_face_detector()

S3 对加载的图片运行 HOG 人脸探测器,得到探测出的人脸

上面的类 dlib.get_frontal_face_detector()的实例化.

打印出发现的人脸,和其所在的图片文件：print()

S4．利用 dlib 的"68 点-特征预测器"，进行"68 点-特征"提取

dlib.shape_predictor(人脸预测模型)

#5.通过如下链接下载预先训练的人脸探测模型：

http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

人脸预测模型 = "shape_predictor_68_face_landmarks.dat"

S6.显示出带图片的窗口

S7.遍历图片中的每一张人脸

For 循环：

画出人脸边界盒子

得到人脸标识对象,并画出(显示)

定义 landmarks

```
win.add_overlay(landmarks)
```

6. 扩展

1. 更多地了解 dlib,参考：http://dlib.net/
2. Dlib 的人脸探测： face recognition
3. Dlib 中的人脸标识探测： face landmark detection