Python人工智能-机器学习系列课程

项目一 人脸识别系统

课程目录

•第一章 人脸识别中的基本算法

基本概念——人工智能（AI）

- 人工智能（Artificial Intelligence），英文缩写为AI。它是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。
- 人工智能是计算机科学的一个分支，它企图了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器，该领域的研究包括机器人、语言识别、图像识别、自然语言处理和专家系统等。人工智能从诞生以来，理论和技术日益成熟，应用领域也不断扩大，可以设想，未来人工智能带来的科技产品，将会是人类智慧的“容器”。人工智能可以对人的意识、思维的信息过程的模拟。人工智能不是人的智能，但能像人那样思考、也可能超过人的智能。
- 人工智能是一门极富挑战性的科学，从事这项工作的人必须懂得计算机知识，心理学和哲学。人工智能是包括十分广泛的科学，它由不同的领域组成，如机器学习，计算机视觉等等，总的说来，人工智能研究的一个主要目标是使机器能够胜任一些通常需要人类智能才能完成的复杂工作。但不同的时代、不同的人对这种“复杂工作”的理解是不同的。 2017年12月，人工智能入选“2017年度中国媒体十大流行语”。

人工智能就业分析

- 我们都说就业难，选择对口专业更难，毕业就等于失业，再而言之，市场上很多职位都处于饱和状态。但是有一个行业却恰恰相反，在2017年受到人工智能辐射的相关岗位需求却呈现爆发式的增长。

- 自从阿尔法狗一战成名，整个人工智能也似开了挂一般，发展非常迅速，人工智能人才缺失，企业出高薪也很难聘请到合适的人才。人工智能就业前景一片大好，紧缺人才

- 人工智能的发展现状处于成长期；国家发布相关政策促进人工智能的发展；一些省份也比较重视人工智能的发展，并提出了相应的规划；人工智能的人才市场处于空缺，严重的供不应求。所以，趁早搭上人工智能的快车，未来无限可能。

基本概念——机器学习

- 机器学习(Machine Learning, ML)是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。

- 数十年前,在20世纪60年代,我们就有了机器学习的应用----光学字符识别(OCR); 90年代, 垃圾邮件过滤, 作为机器学习的第一个应用, 具有里程碑意义.

- 机器学习, 是指“ 使计算机模拟或实现人类的学习行为，以从数据中获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能” 这样一种过程. 机器学习是数据科学的核心, 是现代人工智能的本质.机器学习是人工智能之核心，是使计算机的行为更接近人类的根本途径; 机器学习的应用遍及人工智能的各个领域，它主要使用归纳、综合方法. 类比, 小孩学习任何一个新概念.

机器学习基本概念

- 训练集:用于训练机器学习算法的数据样本集合
- 验证集：用来确定网络结构或者控制模型复杂程度的参数
- 测试集：检验最终选择最优的模型的性能如何
- 一个典型的划分是训练集占总样本的50%，而其它各占25%，三部分都是从样本中随机抽取

- 为了测试机器学习算法的效果,我们通常用到两套独立的样本集: 训练集和测试集. 机器学习先使用训练集作为算法的输入,训练完成之后输入测试集.

数据集的表示

- X, 用来表示特征. 如在邮件中，X=(“有无乱码”,”有无标题”,”有无敏感词汇”,) 第一封邮件: X=(1,0,1)
- Y, 用来表示标签. 如 Y = 0表示垃圾邮件，

机器学习的分类

- 监督学习: 训练样本中同时包含有特征和标签信息的机器学习算法. 例如,邮件过滤器中,作为训练集的邮件自带”正常邮件”或”垃圾邮件”标签.

- 无监督学习: 其样本只含有特征,不包含标签信息. 最典型的算法是聚类算法(k均值算法,期望最大化算法(EM)等)和降维算法(如主成分分析算法（PCA),线性判别式分析(LDA),小波分析,拉普拉斯特征分析,...）.

监督学习——包含分类和回归两种算法

- 1.训练部分: (邮件, 小孩学习新概念)
- 获取数据 → 特征提取 → 监督学习 → 评估
- 2. 预测部分:
- 模型 → 预测

.分类算法和回归算法

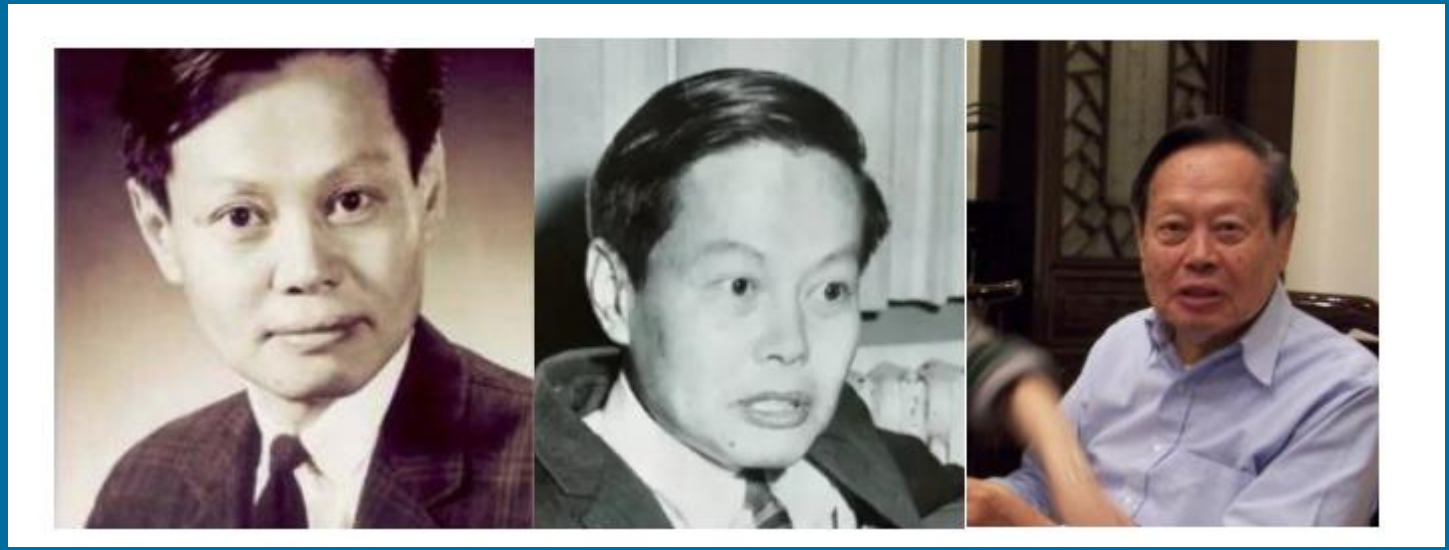
- 分类算法的目标是从(已有的)数个可能的分类标签中预测一个样本所属的分类标签. 例如,前面介绍的标记垃圾邮件的算法,就是一种分类算法.又如,识别一个网站所用的语言.识别结果可以是英语,汉语,日语等语言中的一种.语言之间没有连续性（即离散值）.{中文,日文,韩文,英文,德文,西班牙文,...}，好比一个一个的筐,一般物体不能同时放在两个筐中.)

- 回归算法的目标是预测一个连续的数(浮点型). 例如,由一个人的年龄,教育程度,住址来预测其收入. 由农作物的品种,种植的位置,降雨量等特征来预测该农作物的产量.这里的收入和农作物的产量的取值都可以是一定范围内的任何值.给计算机输入一套房的面积，位置，朝向，房间数目，计算机就可以自动给你算出它的价格（回归算法）；输入一个人的学历，住址，朋友数目，去过的地方，计算机也可以自动给你算出他/她的年收入(回归算法)；.....

人脸识别的步骤概述

- 输入一种植物的花萼数目，花瓣数目,花瓣长宽比，雄蕊数目,雌蕊数目,颜色,香味等，计算机就可以告诉我们这种植物的名称(分类算法). 这个识别花的问题都可以通过选择一个机器学习算法来解决: 给它数据集,训练出模型,然后等待输出结果. 一种花就好比一张脸.识别一张人脸,就好比对识别一种花;对花儿进行分类,就好比人脸识别.

- 人脸识别(face recognition)也是可以通过机器学习算法来做的，它是由一系列机器学习算法构成的．下图显示同一个人的三张照片



人脸识别的步骤

- 1. 在图上找出所有的人脸．
- 2. 对同一张脸，即使它旋转或者扭曲，让计算机要能知道这是同样一张脸．
- 3. 能找出这张脸独特的特征，以用来将这张脸与其他人的脸区分开（特征可以是：眼睛大小，鼻子长度等）．
- 4. 将该脸的独特特征与所有已经标识的人脸进行对照,以确定该人脸对应的人的姓名.

注意：我们的大脑非常善于人脸识别！能瞬间完成上面所有的步骤！

我们将分四个步骤来处理人脸识别. 对每一步, 我们将学习一个不同的机器学习算法．我们将会学习每一个算法背后的基本思想．以后你可以用Python去建立自己的人脸识别系统．可能用到的库有OpenFace和dlib, scikit-image,os,matplotlib,scikit-learn等.

•第二章　人脸探测算法

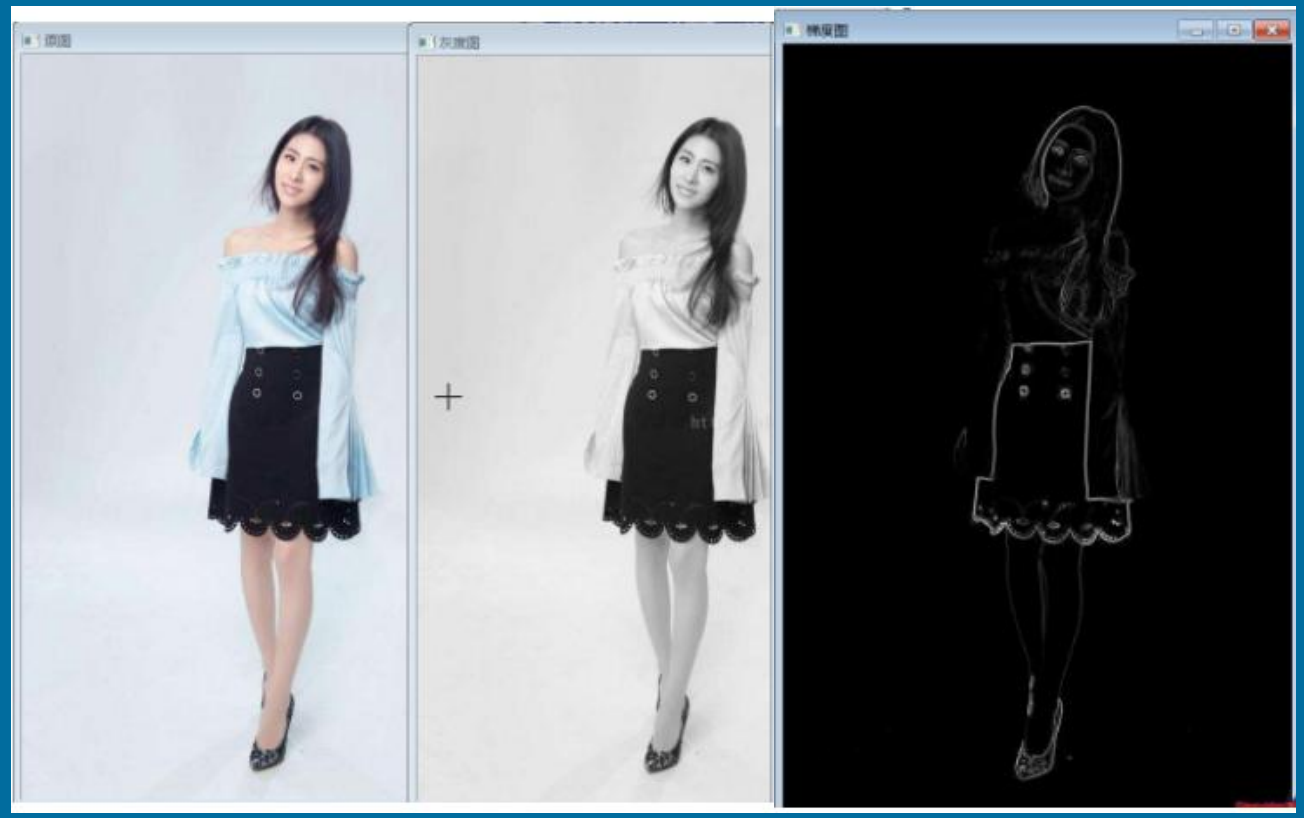
人脸探测

- 区分人脸之前，首先要找出人脸在照片中的位置!　近几年来的相机都有人脸探测的功能，它能找出人脸，以便可以对每张人脸对焦，从而得出更清晰的照片．我们这里用它来做识别人脸，而不是用来得到人脸的更清晰的照片

- 本世纪初,Paul Viola和Michael Jones 发明了能应用在一般照相机上的快速探测人脸的方法，人脸探测在2000到2010年间就成为主流技术．现在，人们有了更好的方法用于快速探测人脸．

人脸探测方法——HOG方向梯度直方图

- 我们将用的方法----方向梯度直方图(Histogram of Oriented Gradients, HOG, 2005).



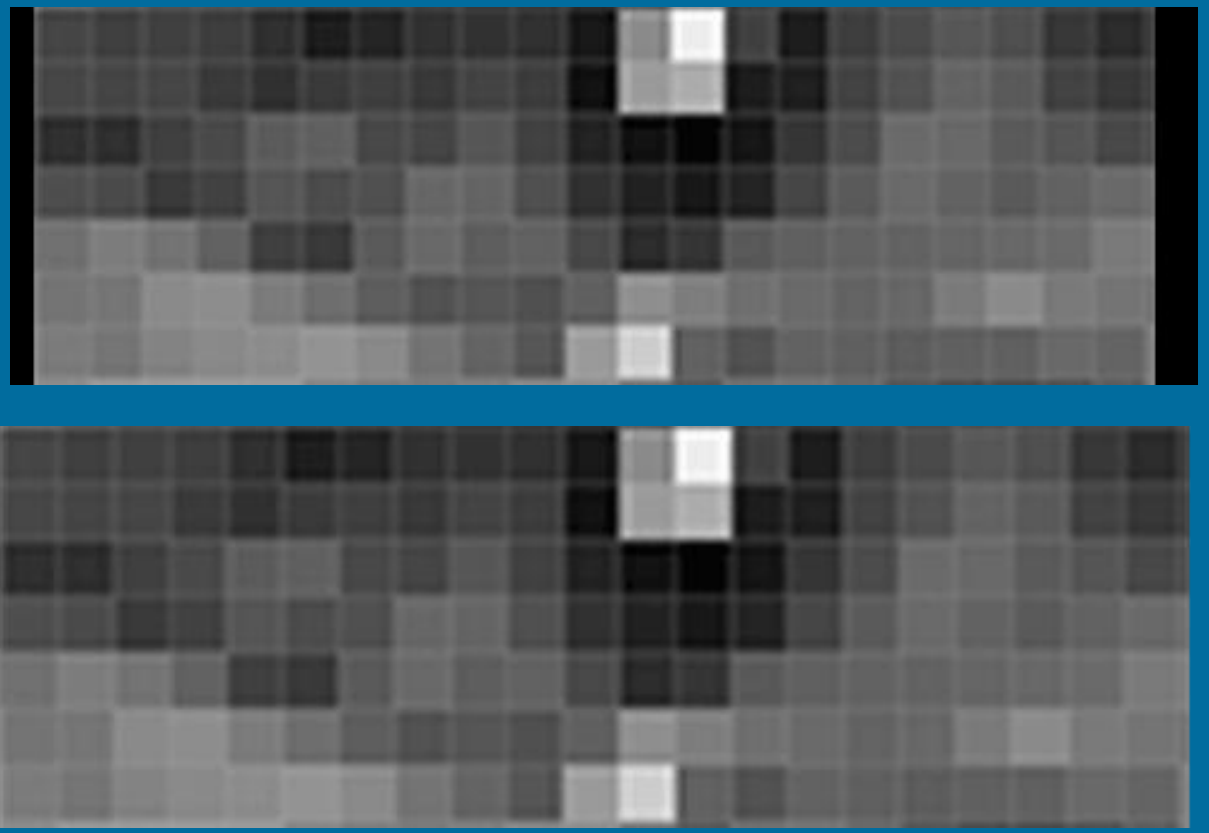
人脸探测

- 首先, 因为色彩的信息对我们做人脸识别时不需要,所以我们先将照片转成黑白照片 .

- 然后，我们依次关注照片中的每一个像素点 . 先看一个例子.

如果我们直接分析像素, 同一个人的非常暗的照片和非常亮的照片将具有完全不同的像素值(如图).
一个很好的办法是: 考虑像素亮度的变化! 当考虑亮度改变的方向时，暗照片和亮照片将有几乎一样的表示! 所以我们不直接处理像素值信息，而是处理像素明暗的变化信息 .
对每个像素点， 我们想看看那些直接包围着它的像素点. 我们的目标：计算出当前像素相对于其周围的像素有多暗. 然后我们可以画出像素由明到暗变化最快的方向

图像放大



对照片中的每一个像素点重复上述操作, 最终每一个像素都被一个箭头取代了. 这些箭头称为梯度 ,它们显示整张照片由明到暗变化最快的方向 .

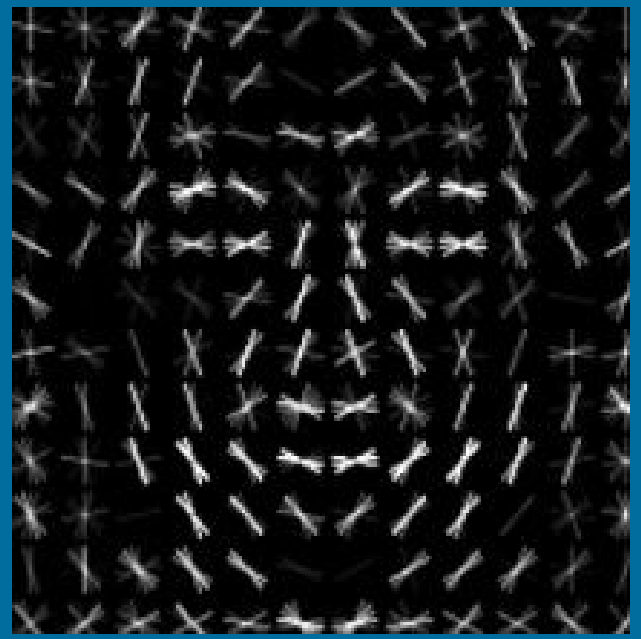
在计算HOG时, 如果将方向数目这个参数设置为9,那么圆周(360°)就被分成九个区间. 它们分别是:

[0, 40],[40, 80],[80,120],[120, 160],[160, 200],[200, 240],[240, 280],[280, 320],[320,360].单位为度(°).

怎么从HOG图中找到人脸?

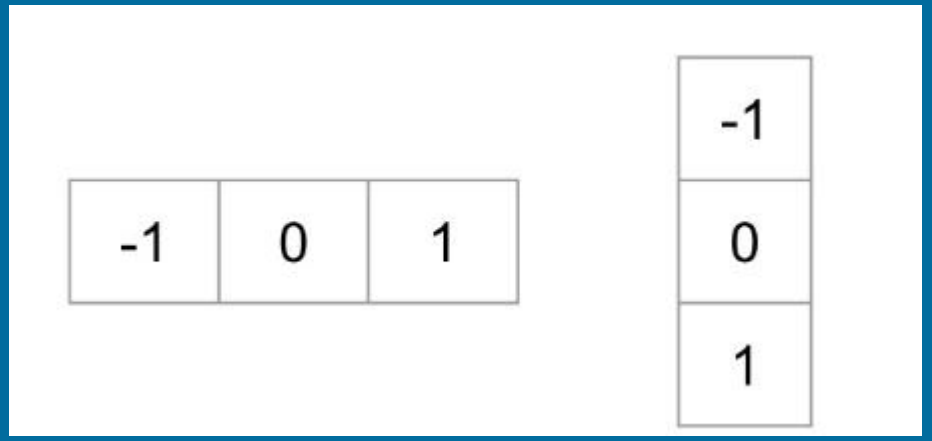
- 进一步简化：我们用梯度来看一些基本模式，而不是所有的细节！为了达到这个目的，我们将照片分成小方格，每个方格的大小为16*16像素（或其他），然后我们以最强的箭头方向去取代该小方格． 这样，我们将原图片转化成了非常简单的表示． 它只以简单的方式描述人脸的基本结构.
-

怎么从HOG图中找到人脸? 为了在这张HOG图片中找到人脸, 我们只需找到我们的图片中最像已知HOG模式的那部分．已知HOG模式由大量其他照片训练并提取出来，如图：



计算梯度图像

- 为了计算一个HOG描述器, 我们首先需要计算水平梯度和竖直梯度;然后可求得方向梯度直方图. 这可以由如下的核过滤该图片而得:



我们也可用OpenCV中的Sobel算符(with kernel size 1)来达到同样的目的 .

```
1.  # Python gradient calculation
2.  # Read image
3.  im = cv2.imread('E:\open_courses\eg.png')
4.  im = np.float32(im) / 255.0
5.
6.  # Calculate gradient
7.  gx = cv2.Sobel(im, cv2.CV_32F, 1, 0, ksize=1)
8.  gy = cv2.Sobel(im, cv2.CV_32F, 0, 1, ksize=1)
```

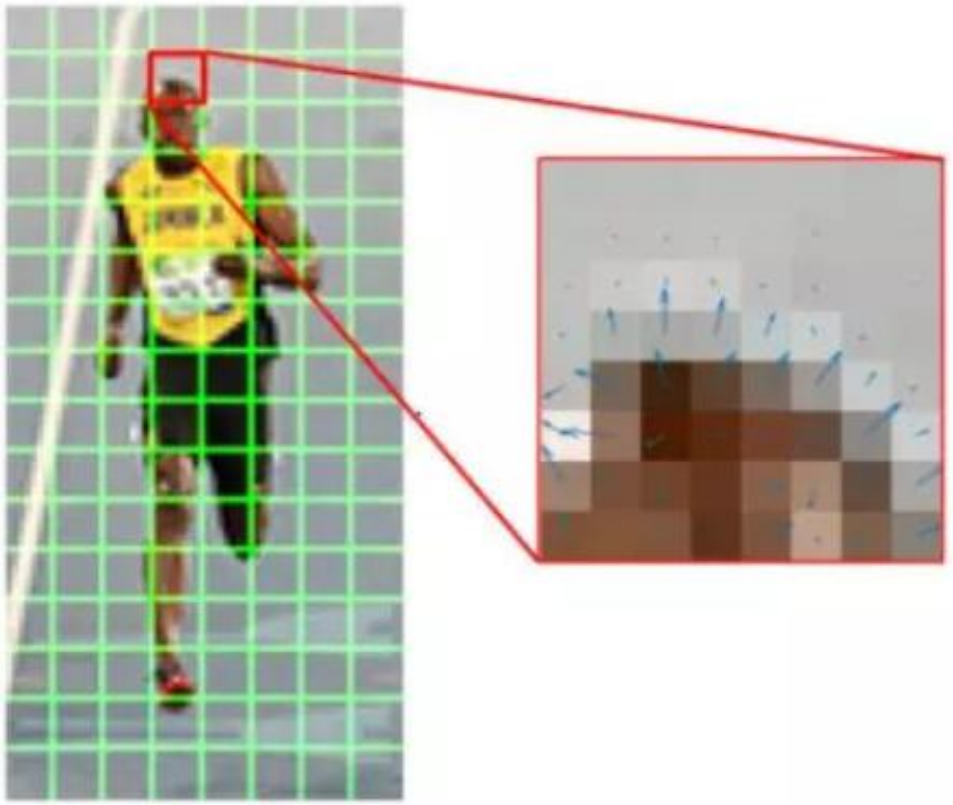
计算梯度图像

- 然后，我们可以用g_x,g_y来计算梯度的大小和方向：

$$g^2 = (g_x^2 + g_y^2)$$

$$\text{angle} = \arctan(\frac{g_y}{g_x})$$

计算梯度图像



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

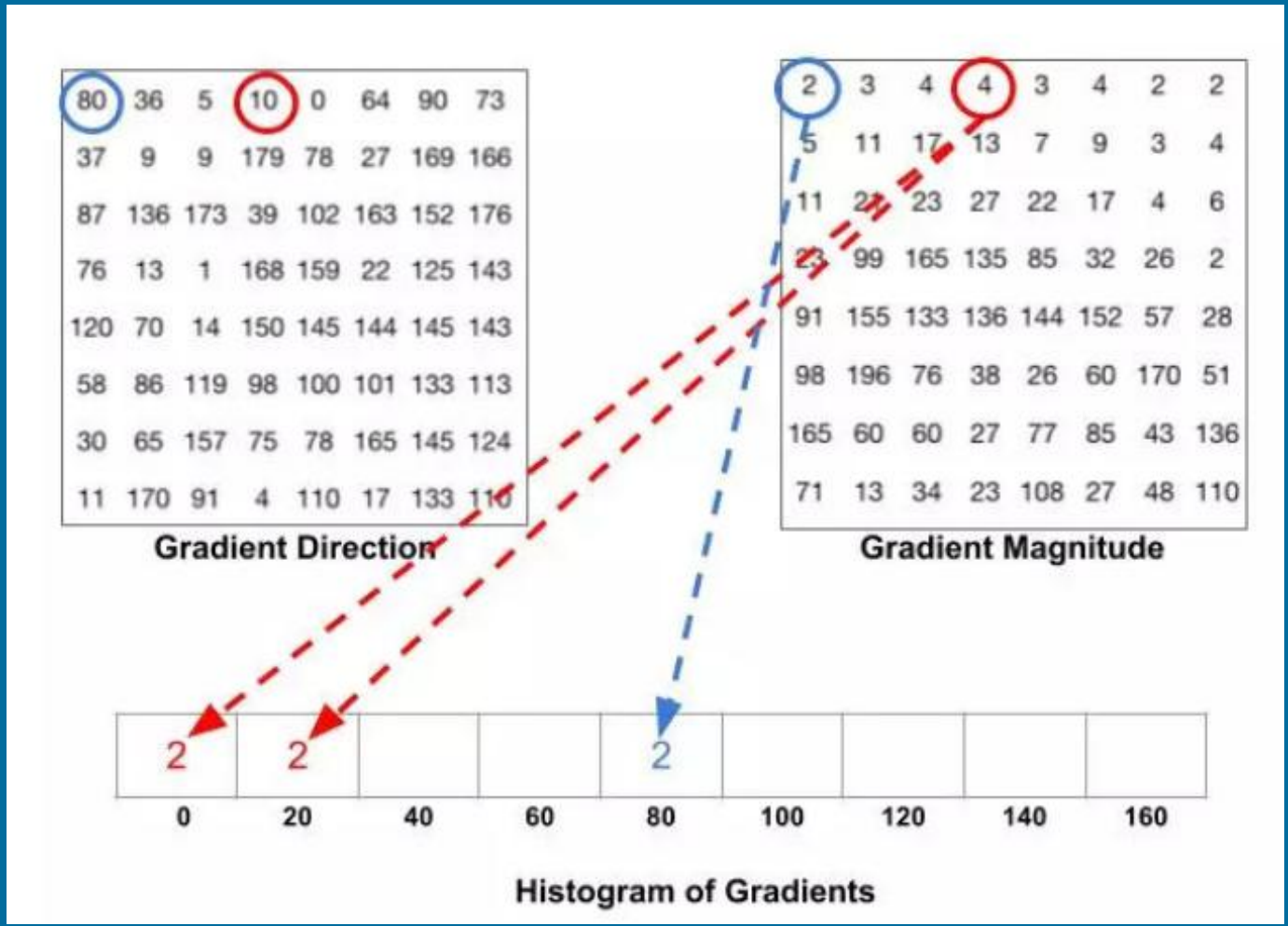
Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

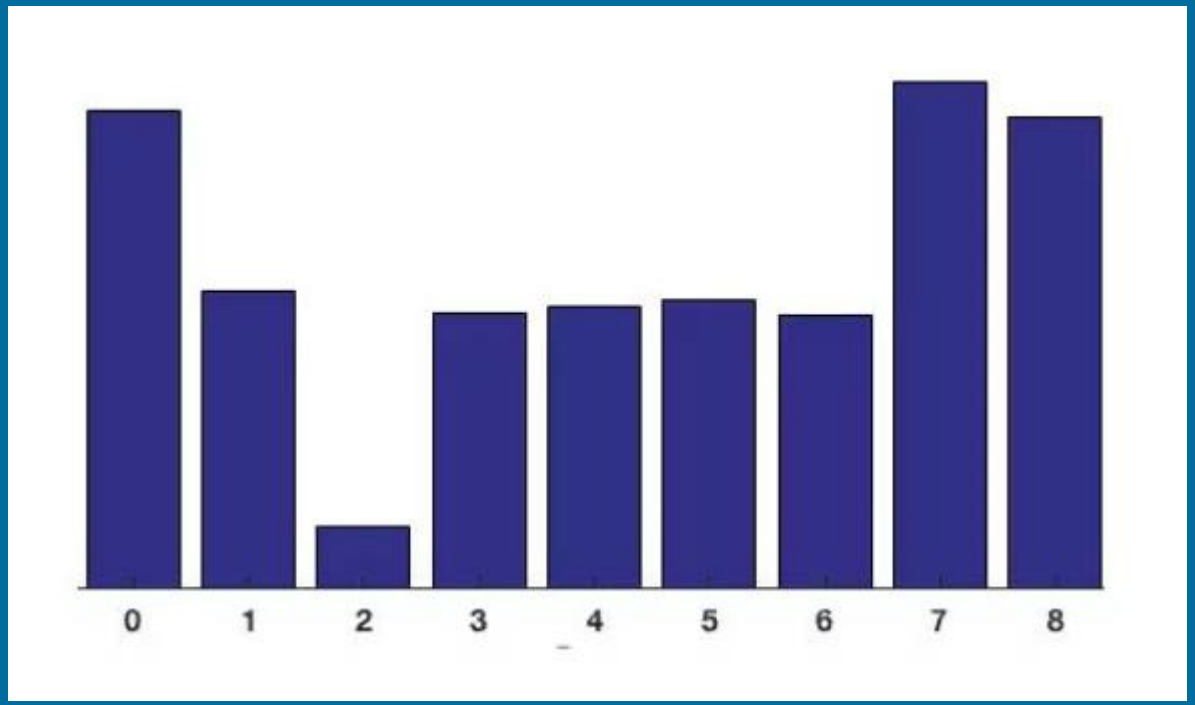
Gradient Direction

Center : The RGB patch and gradients represented using arrows. Right : The gradients in the same patch represented as numbers

计算梯度图像

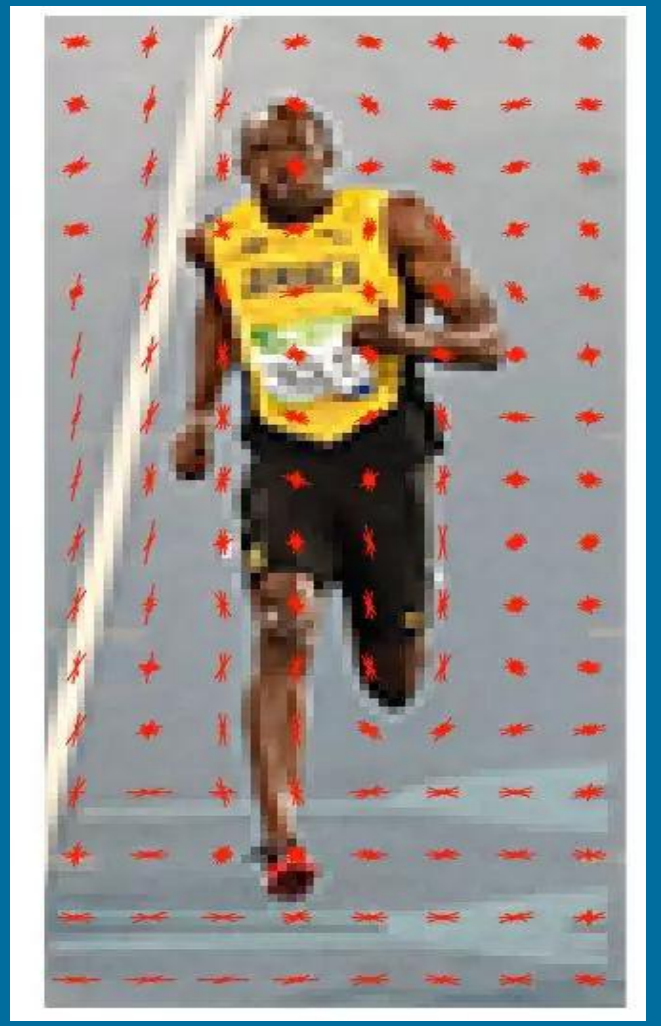

80 36 5 10 0 64 90 73
37 9 9 179 78 27 169 166
87 136 173 39 102 163 152 176
76 13 1 168 159 22 125 143
120 70 14 150 145 144 145 143
58 86 119 98 100 101 133 113
30 65 157 75 78 165 145 124
11 170 91 4 110 17 133 110
Gradient Direction
2 3 4 4 3 4 2 2
5 11 17 13 7 9 3 4
11 21 23 27 22 17 4 6
23 99 165 135 85 32 26 2
91 155 133 136 144 152 57 28
98 196 76 38 26 60 170 51
165 60 60 27 77 85 43 136
71 13 34 23 108 27 48 110
Gradient Magnitude
2
2
2
0
20
40
60
80
100
120
140
160
Histogram of Gradients

计算梯度图像-HOG



计算HOG特征向量

- 通常HOG特征描述子是画出8*8网格中9*1归一化的直方图，见图。你可以发现直方图的主要方向捕捉了这个人的外形，特别是躯干和腿。



HOG代码实现

库函数介绍——dlib

- 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

或者先下载再安装：
https://pypi.org/project/dlib/19.4.0/
然后打开cmd，cd到dlib的whl文件所在文件夹，pip install dlib-19.4.0-cp35-cp35m-win_amd64.whl

库函数介绍——skimage（scikit-image ）

- 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

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

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 × 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: $x_1, x_2, \ldots, x_N$ 中的任何一个.

矩阵 W 的尺寸为 K × 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).

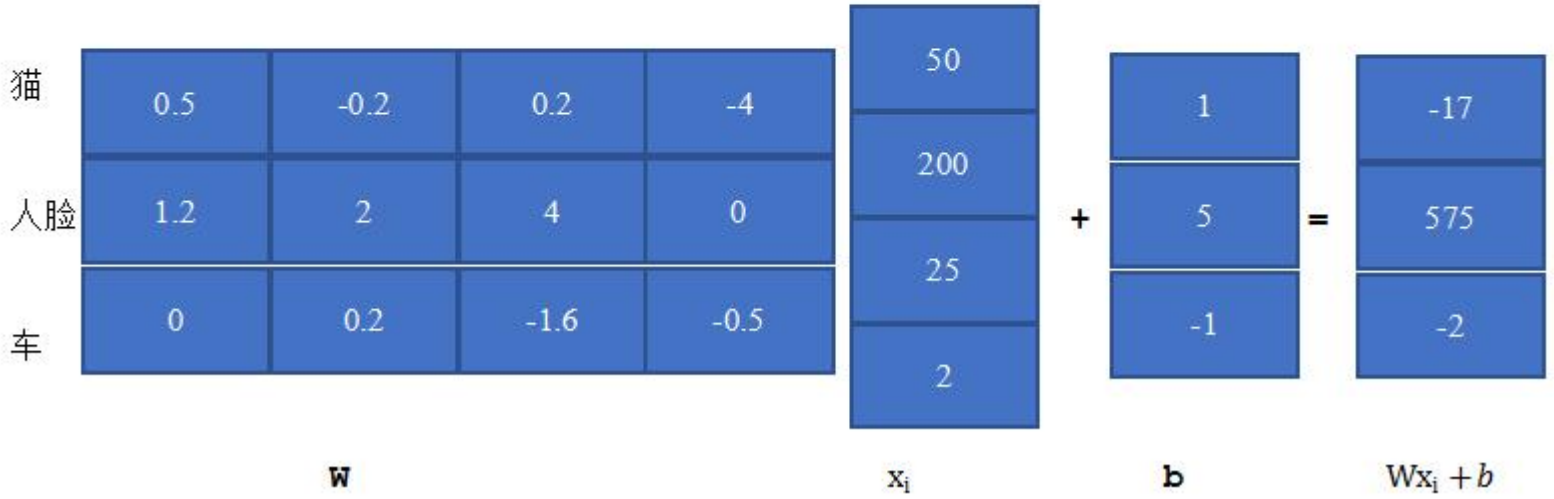
注意: 训练数据用来学习参数 W 和 b, 学习完成后, 我们可以舍弃训练集.

线性分类器

- 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的矩阵乘积. 每个分类就是这个

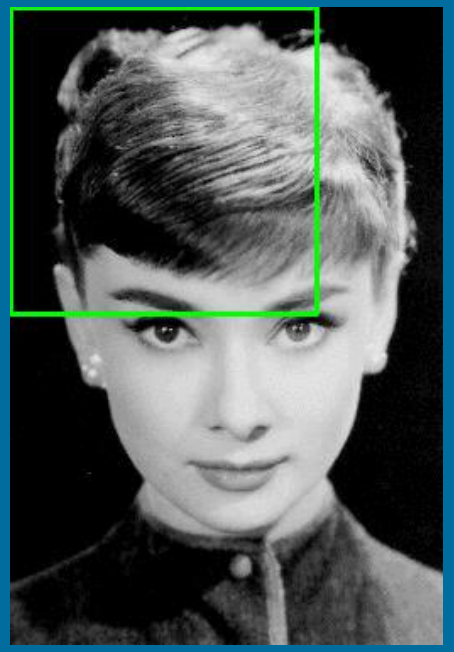
例. 矩阵乘法举例: $Wx_i + b$

```
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
```



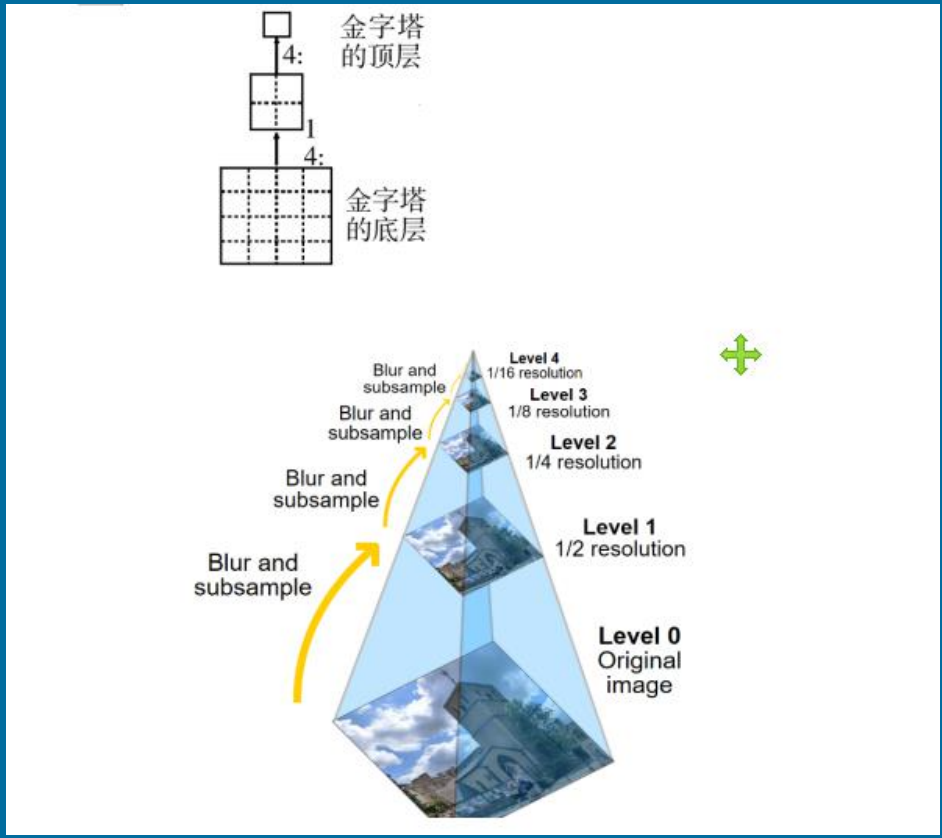
滑动窗口检测

- 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.



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).



总结——人脸标识的步骤——代码实现

- 1 获取图片文件名; 并加载图片 io.imread(file_name)
- 2 用dlib内置类创建HOG人脸探测器 dlib.get_frontal_face_detector()
- 3 对加载的图片运行HOG人脸探测器,得到探测出的人脸 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()
- 4. 利用dlib自带的"人脸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
- 6.显示出带图片的窗口
- 7.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)

第三章　训练人脸识别分类器

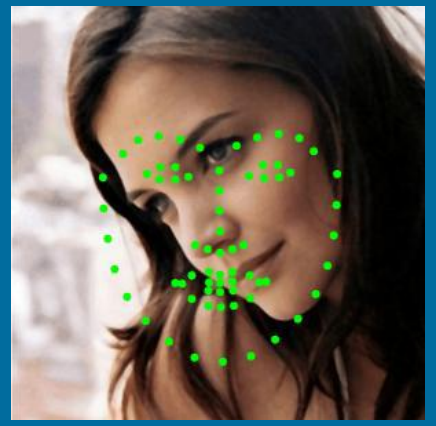
- 1.了解人脸校正和编码的基本过程.
- 2.K-近邻算法.
- 3.学习使用mglearn, skimage, matplotlib, numpy等Python库.

人脸识别步骤

- 分四个步骤来处理人脸识别.
- 第一,人脸探测;
- 第二,人脸校正和变换;
- 第三,人脸编码;
- 第四,区分出人脸.

人脸校正和变换

- 基本思想：我们将选出每张人脸上的68个特殊点 (landmarks) ---- 如眼睛的外边界, 眉毛的内边界, 等. 然后我们将运用一个机器学习算法在任意一张人脸上找出这些特殊点
- 当我们知道人脸的双眼和嘴的位置，我们将简单地旋转, 伸缩和剪切(shear)这张图片, 使得双眼和嘴尽可能地在每张图中都在相同的位置. 我们运用基本的图形变换，如保持平行线仍然平行的旋转变换和伸缩变换 (仿射变换 (affine transformations)) . 这一步操作为下一步提供了很大的方便.



达到的目标: 无论人脸如何转动, 我们都能将眼睛和嘴基本上置于照片中心固定的位置. 这将使我们下一步更精确

估算人脸的位置——Boosting算法 (了解)

- Boosting方法最初来源于分类问题，但也可以拓展到回归问题． Boosting方法的动机: 是将许多弱分类器组合起来构成一个强大的分类器集体.
- 最流行的boosting算法(Freund and Schapire，1997)： “AdaBoost.M1”

举例，对二分类问题, 分类器的**错误率**定义为：

$$err = \frac{1}{N}\sum_{\{i=1\}}^{N}$$

其中，$y_i \in \{-1, 1\}$，分类器用 $G(x_i)$表示.

弱分类器(weak classifier)：错误率仅仅比随机猜

boosting 方法的目标：相继应用弱分类器算法以不

$m = 1, 2, \ldots, M.$

最终的预测器：

$$G(x) = sign\left[\sum_{1}^{M} \alpha_m G_{m(x)}\right]$$

此处$\alpha_1, \alpha_2, \ldots, \alpha_M$. 由 boosting 算法计算而得，并为各个分类器$G_m(x)$提供权重.

该算法的目标：

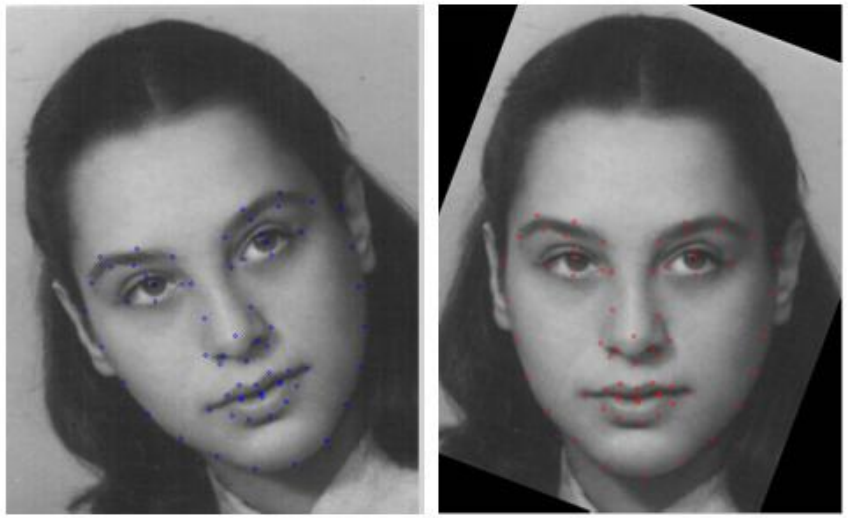
以一种有效的计算方法来精确地估算人脸(facial landmarks)的位置.

仿射变换（旋转和变换等）

仿射变换将原坐标(x, y)变换为新坐标(x', y')：

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} A_{00}x + A_{01}y + A_{02} \\ A_{10}x + A_{11}y + A_{12} \\ 1 \end{pmatrix}$$

通过这公式，可计算出原图像经过变换后的新图像.
根据眼睛坐标进行人脸对齐, 计算变换后对应坐标. 图片如下：



人脸图像变换前后及 68 个面部关键点

人脸的编码1——测量人脸，提取特征

- 我们需要提取人脸的特征值! 我们的方法是: 提取每一张人脸的几个基本测量值．然后我们可以以同样的方式测量新的人脸照片，并找出与之有最接近的测量值的已知人脸.　例如，我们可能测量每一只耳朵的尺寸, 两眼间距, 鼻子长度等.
- 我们应该从每一张脸上收集哪些测量值来建立我们的带有标签的＂熟人＂数据库呢？　耳朵大小？眉毛长度?　眼睛大小？ 鼻子宽度？ 最精确的方法是让计算机自己去决定该收集哪些测量值．对于人脸的哪些测量值最重要这一问题，深度学习算法是一种很好的方法! 具体的解决办法是训练一个深度卷积神经网络, 并训练它对每张人脸产生128个测量值.

人脸的编码2——为人脸编码

- 即使有大型计算机, 训练神经网络也非常耗时．但是一旦神经网络训练完成，它便可以对任意一张从未见过的人脸产生出测量值．所以，训练只需要运行一次！ OpenFace的研究人员已经训练除了一些神经网络，我们可以直接使用 (参考Brandon Amos and team)! 我们需要亲自做的就是：使我们的人脸照片输入至他们已经训练好的神经网络中以得到那128个测量值.
- 问: 这128个数值分别测量的是什么呢?
- 我们关注的是这个网络在看同一个人的两张不同的照片时，产出几乎一样的数.（安装：pip install face_recognition ）

对图片中的人脸编码, 函数 face_recognition.face_encodings()
模块 face_recognition.api 中的函数 face_encodings .

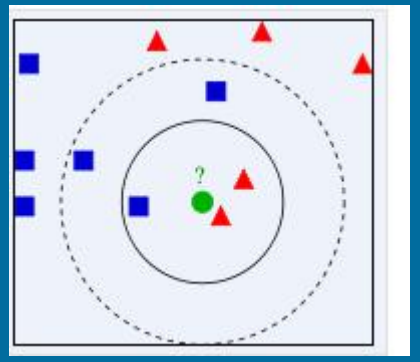
```python
def face_encodings(face_image, known_face_locations=None, num_jitters=1)
    """
    对给定的图片中的每一张人脸, 返回 128-dimension 人脸编码.
```

技能学习——Python画图（mglearn、matplotlib）

- 利用mglearn处理数据集（安装mglearn: pip install mglearn ）
- 利用matplotlib画图

KNN算法介绍（k-NearestNeighbor）

- K最近邻，就是k个最近的邻居的意思，说的是每个样本都可以用它最接近的k个邻居来代表。只计算“最近的”邻居样本。
- K最近邻分类算法是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。KNN算法中，所选择的邻居都是已经正确分类的对象。
- 对于距离的度量，我们有很多的距离度量方式，但是最常用的是欧式距离，即对于两个n维向量x和y，两者的欧式距离定义为：



$$D(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2}=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

第四章　人脸识别系统的测试

- 人脸识别系统实现完整步骤
- 1　准备数据集
- 2　训练模型
- 3　使用模型
- 4　显示结果