**Tensorflow**

**1.tf.placeholder(dtype,shape=None, name=None)**

**此函数可以理解为形参，用于定义过程，在执行的时候再赋具体的值**

2**.tf.stack（**）这是一个矩阵拼接的函数，tf.unstack（）则是一个矩阵分解的函数

3.**tf.get\_variable**(name,shape,initializer): name就是变量的名称，shape是变量的维度，initializer是变量初始化的方式。

4**.tf.Variable**（initializer， name）：initializer是初始化参数，可以有5.tf.random\_normal，tf.constant，tf.constant等，name就是变量的名字，用法如下：

6. **tf.ones\_like**(tensor, dtype=None, name=None)

该方法实际上为一个拷贝函数：默认情况下，它会拷贝参数tensor的类型，维度等数据，并将其中的值设置为1．当参数dtype设置后，那么拷贝后的tensor对象

7. **tf.squeeze**(input, squeeze\_dims=None, name=None)

Removes dimensions of size 1 from the shape of a tensor.   
从tensor中删除所有大小是1的维度

8. **tf.nn.l2\_loss**(t, name=None)

解释：这个函数的作用是利用 L2 范数来计算张量的误差值，但是没有开方并且只取 L2 范数的值的一半，具体如下：

output = sum(t \*\* 2) / 2

9. 函数training()通过梯度下降法为最小化损失函数增加了相关的优化操作，在训练过程中，先实例化一个优化函数，比如 tf.train.GradientDescentOptimizer，并基于一定的学习率进行梯度优化训练：

optimizer = tf.train.GradientDescentOptimizer(learning\_rate)

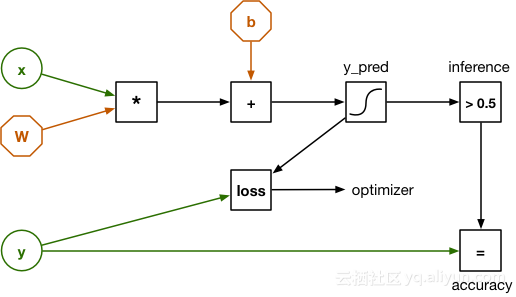
然后，可以设置 一个用于记录全局训练步骤的单值。以及使用minimize()操作，该操作不仅可以优化更新训练的模型参数，也可以为全局步骤(global step)计数。与其他tensorflow操作类似，这些训练操作都需要在[tf.session](http://blog.csdn.net/lenbow/article/details/52181159" \t "_blank)会话中进行

global\_step = tf.Variable(0, name='global\_step', trainable=False)

train\_op = optimizer.minimize(loss, global\_step=global\_step)

12．tensorflow使用举例

TensorFlow上使用该分类器，首先需要创建一个计算图（computational graph）。计算图由表示进行运算的节点（nodes）以及节点之间流动的数据（data）组成。

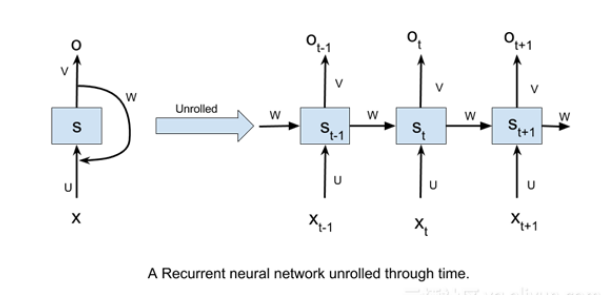


1.标量数字就是0阶张量；2.向量是一阶张量；

3.矩阵是二阶张量；4.三维数组是三阶张量

14.LSTM简介

展开RNN



1.Xt是指时间步长t的输入。

2.St是指时间步长t处的隐藏状态。它可以被可视化为网络的“内存”。

3.Ot指的是输出在时间步长t。

4.U，V和W是**所有时间步长共享的参数**。该参数共享的意义在于，我们的模型在不同输入的时间步长可以执行相同的任务。

1. input\_size 这个参数不能使用，使用的是num\_units

2. state\_is\_tuple 官方建议设置为True。此时，输入和输出的states为c(cell状态)和h（输出）的二元组

3. 输入、输出、cell的维度相同，都是 batch\_size \* num\_units，

cell = tf.contrib.rnn.BasicLSTMCell(num\_units, forget\_bias=0.0, state\_is\_tuple=True)　　#指定num\_units

\_initial\_state = cell.zero\_state(batch\_size, tf.float32)　　　　　　　　　　　　　　　　　　　#指定batch\_size,将c和h全部初始化为0，shape全是batch\_size \* num\_units，

15.pandas.read\_csv参数整理

读取CSV（逗号分割）文件到DataFrame也支持文件的部分导入和选择迭代

16.python计算分位数使用numpy包：

import numpy as np

a=np.array(([1,2,3,4]))

np.median(a)#中位数

np.percentile(a,95)#95%分位数

17用法：

for image in images[:2]:

print("shape: {0}, min: {1}, max: {2}".format(

image.shape, image.min(), image.max()))

Output:

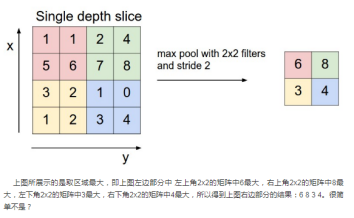
shape: (141, 142, 3), min: 0, max: 255

shape: (120, 123, 3), min: 0, max: 255

18、ReLU激励层http://img.blog.csdn.net/20160703124215945

池化pool层

    说了，池化，简言之，即取区域平均或最大，如下图所示



<http://blog.csdn.net/u014365862/article/details/54865609>

19下采样（池化）层

下采样，即池化，目的是减小特征图，池化规模一般为2×2。常用的池化方法有：

最大池化（Max Pooling）。如图5所示。

均值池化（Mean Pooling）。如图6所示。

高斯池化。借鉴高斯模糊的方法。

可训练池化。训练函数 ff ，接受4个点为输入，输出1个点。

20．sess = tf.Session()

result = sess.run(product)

print result

sess.close()

自动完成关闭session操作

**with** tf.Session() **as** sess:

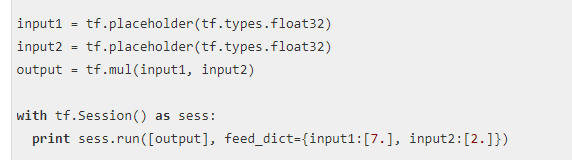
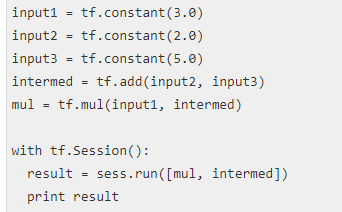
result = sess.run([product])

**print** result

21 Feed机制

该机制可以临时替代图中的任意操作中的 tensor 可以对图中任何操作提交补丁, 直接插入一个 tensor.

feed 使用一个 tensor 值临时替换一个操作的输出结果. 你可以提供 feed 数据作为 run() 调用的参数. feed 只在调用它的方法内有效, 方法结束, feed 就会消失. 最常见的用例是将某些特殊的操作指定为 "feed" 操作, 标记的方法是使用 tf.placeholder() 为这些操作创建占位符.



22CNN原理

http://www.hackcv.com/index.php/archives/104/?hmsr=toutiao.io&utm\_medium=toutiao.io&utm\_source=toutiao.io

1图像卷积，通俗讲就是矩阵对应元素相乘，池化就是采样，有最大值池化，就是局部取最大值；均值池化，就是取平均值等。

一张图，卷积变小了，池化又变小了，最卷积变小了，在池化，又变小了。。。。。最后取出想要的东西。

2图像是像素值的矩阵。本质上来说，每张图像都可以表示为像素值的矩阵：

3在 CNN 的术语中，3x3 的矩阵叫做“滤波器（filter）”或者“核（kernel）”或者“特征检测器（feature detector）”**，通过在图像上滑动滤波器并计算点乘得到矩阵**叫做“卷积特征（Convolved Feature）”或者“激活图（Activation Map）”或者“**特征图**（Feature Map）”。**记住滤波器在原始输入图像上的作用是特征检测器。**

4**特征图**的大小（卷积特征）**由下面三个参数控制**，我们需要在卷积前确定它们：

5**深度**（Depth）：深度对应的是卷积操作所需的滤波器个数。在下图的网络中，我们使用三个不同的滤波器对原始图像进行卷积操作，这样就可以生成三个不同的特征图。你可以把这三个特征图看作是堆叠的 2d 矩阵，那么，特征图的“深度”就是3。

6**步长**（Stride）：步长是我们在输入矩阵上滑动滤波矩阵的像素数。当步长为 1 时，我们每次移动滤波器一个像素的位置。当步长为 2 时，我们每次移动滤波器会跳过 2 个像素。步长越大，将会得到更小的特征图。

7**零填充**（Zero-padding）：有时，在输入矩阵的边缘使用零值进行填充，这样我们就可以对输入图像矩阵的边缘进行滤波。零填充的一大好处是可以让我们控制特征图的大小。使用零填充的也叫做泛卷积，不适用零填充的叫做严格卷积。这个概念在下面的参考文献 14 中介绍的非常详细。

8在上面图中，在每次的卷积操作后都使用了一个叫做 ReLU 的操作。是一个非线性操作。

9**.ReLU 操作可以**从下面的图中**理解**。它展示的 ReLU 操作是应用到上面图 6 得到的特征图之一。这里的输出特征图也可以看作是**“修正”过的特征图**。

ReLU 是一个元素级别的操作（应用到各个像素），并将特征图中的所有小于 0 的像素值设置为零。ReLU 的目的是在 ConvNet 中引入非线性，因为在大部分的我们希望 ConvNet 学习的实际数据是非线性的（卷积是一个线性操作——元素级别的矩阵相乘和相加，所以我们需要通过使用非线性函数 ReLU 来引入非线性。

10. 空间**池化**（Spatial Pooling）（也叫做亚采用或者下采样）。最大化、平均化、加和。

对于最大池化（Max Pooling），我们定义一个空间邻域（比如，2x2 的窗口），并从窗口内的修正特征图中取出最大的元素。除了取最大元素，我们也可以取平均（Average Pooling）或者对窗口内的元素求和。在实际中，最大池化被证明效果更好一些。我们以 2 个元素（也叫做“步长”）滑动我们 2x2 的窗口，并在每个区域内取最大值。如上图所示，这样操作可以**降低我们特征图的维度**。

11池化函数可以逐渐降低输入表示的空间尺度。特别地，池化：

1使输入表示（特征维度）变得更小，并且网络中的参数和计算的数量更加可控的减小，因此，可以控制过拟合

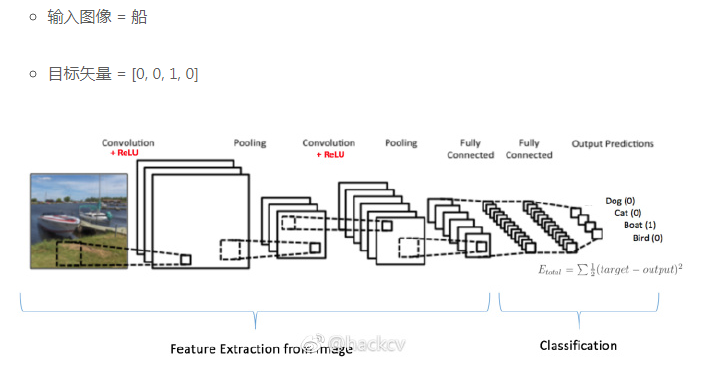
2使网络对于输入图像中更小的变化、冗余和变换变得不变性（输入的微小冗余将不会改变池化的输出——因为我们在局部邻域中使用了最大化/平均值的操作。

3帮助我们获取图像最大程度上的尺度不变性（准确的词是“不变性”）。它非常的强大，因为我们可以检测图像中的物体，无论它们位置在哪里。

这些层一起就可以从图像中提取有用的特征，并在网络中引入非线性，减少特征维度，同时保持这些特征具有某种程度上的尺度变化不变性。

第二组池化层的输出作为全连接层的输入，我们会在下一部分介绍全连接层。

卷积和池化层的输出表示了输入图像的高级特征。全连接层的目的是为了使用这些特征把输入图像基于训练数据集进行分类。比如，在下面图中我们进行的图像分类有四个可能的输出结果。



像滤波器数量、滤波器大小、网络结构等这样的参数，在第一步前都是固定的，在训练过程中保持不变——仅仅是滤波器矩阵的值和连接权重在更新

接着，是第二个**卷积层**，它有5个Filter。每个Fitler都把前面**下采样**之后的**3**个Feature Map卷积在一起，得到一个新的Feature Map。

在 CNN 的术语中，3x3 的矩阵叫做“滤波器（filter）”或者“核（kernel）”或者“特征检测器（feature detector）”，通过在图像上滑动滤波器并计算点乘得到矩阵叫做“卷积特征（Convolved Feature）”或者“激活图（Activation Map）”或者“特征图（Feature Map）”。记住滤波器在原始输入图像上的作用是特征检测器。

**12 tf.nn.conv2d(input, filter, strides, padding, use\_cudnn\_on\_gpu=None, name=None)**

除去name参数用以指定该操作的name，与方法有关的一共五个参数：

第一个参数input：指需要做卷积的输入图像，它要求是一个Tensor，具有**[batch, in\_height, in\_width, in\_channels]**这样的shape，具体含义是**[训练时一个batch的图片数量, 图片高度, 图片宽度, 图像通道数]**，注意这是一个4维的Tensor，要求类型为float32和float64其中之一

第二个参数filter：相当于CNN中的卷积核，它要求是一个Tensor，具有[filter\_height, filter\_width, in\_channels, out\_channels]这样的shape，具体含义是**[卷积核的高度，卷积核的宽度，图像通道数，卷积核个数]**，要求类型与参数input相同，有一个地方需要注意，第三维in\_channels，就是参数input的第四维

第三个参数strides：卷积时在图像每一维的步长，这是一个一维的向量，长度4

第四个参数padding：string类型的量，只能是"SAME","VALID"其中之一，这个值决定了不同的卷积方式（后面会介绍）

第五个参数：use\_cudnn\_on\_gpu:bool类型，是否使用cudnn加速，默认true

**13.tf.nn.max\_pool(value, ksize, strides, padding, name=None)**

参数是四个，和卷积很类似：

第一个参数value：需要池化的输入，一般池化层接在卷积层后面，所以输入通常是feature map，依然是**[batch, height, width, channels]**这样的shape

第二个参数ksize：池化窗口的大小，取一个四维向量，一般是**[1, height, width, 1]**，因为我们不想在batch和channels上做池化，所以这两个维度设为了1

**理解这句话，就是对pool的ksize对上面的conv2d一一对应，就是**[1, height, width, 1]与**[训练时一个batch的图片数量, 图片高度, 图片宽度, 图像通道数]，一般【1，2，2,1】就是对图片进行2x2池化**

第三个参数strides：和卷积类似，窗口在每一个维度上滑动的步长，一般也是[1, stride,stride, 1]

第四个参数padding：和卷积类似，可以取'VALID' 或者'SAME'

**14．tf.nn.relu(features, name = None)**

解释：这个函数的作用是计算激活函数relu，即max(features, 0)。即将矩阵中每行的非最大值置0。

15.\_image = tf.reshape(x, [-1,28, 28, 1])

# -1表示任意数量的样本数,大小为28x28深度为一的张量

# 可以忽略(其实是用深度为28的,28x1的张量,来表示28x28深度为1的张量)

16.不推荐使用SummaryWriter （来自tensorflow.[Python](http://lib.csdn.net/base/python" \t "_blank" \o "Python知识库).training.summary\_io），将在2016-11-30之后删除。 更新说明：

请切换到tf.summary.FileWriter接口和行为是相同的; 这只是一个重命名

17.官方文档中给出的用法是：numpy.random.randint(low,high=None,size=None,dtype)   
生成在半开半闭区间**[low,high)**上离散均匀分布的整数值;若**high=None**，则取值区间变为**[0,low)**

**18.**  我们经常在训练完一个模型之后希望保存训练的结果，这些结果指的是模型的参数，以便下次迭代的训练或者用作测试。Tensorflow针对这一需求提供了[Saver](https://www.tensorflow.org/versions/r0.9/api_docs/python/state_ops.html" \l "Saver" \t "_blank)类。

1 Saver类提供了向checkpoints文件保存和从checkpoints文件中恢复变量的相关方法。Checkpoints文件是一个二进制文件，它把变量名映射到对应的tensor值 。

2只要提供一个计数器，当计数器触发时，Saver类可以自动的生成checkpoint文件。这让我们可以在训练过程中保存多个中间结果。例如，我们可以保存每一步训练的结果。

3为了避免填满整个磁盘，Saver可以自动的管理Checkpoints文件。例如，我们可以指定保存最近的N个Checkpoints文件。

比如两个卷积核就可以将生成两幅图像，这两幅图像可以看做是一张图像的不同的通道。

5、IOError、AttributeError、ImportError、IndentationError、IndexError、KeyError、SyntaxError、NameError分别代表什么异常

IOError：输入输出异常

AttributeError：试图访问一个对象没有的属性

ImportError：无法引入模块或包，基本是路径问题

IndentationError：语法错误，代码没有正确的对齐

IndexError：下标索引超出序列边界

KeyError:试图访问你字典里不存在的键

SyntaxError:Python代码逻辑语法出错，不能执行

NameError:使用一个还未赋予对象的变量

7、列出几种魔法方法并简要介绍用途

\_\_init\_\_:对象初始化方法

\_\_new\_\_:创建对象时候执行的方法，单列模式会用到

\_\_str\_\_:当使用print输出对象的时候，只要自己定义了\_\_str\_\_(self)方法，那么就会打印从在这个方法中return的数据

\_\_del\_\_:删除对象执行的方法

Pandas库的理解

两个数据类型：Series,  DataFrame

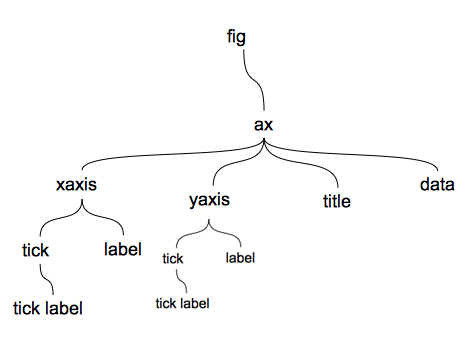
基于上述数据类型的各类操作：基本操作、运算操作、特征类操作、关联类操作

|  |  |
| --- | --- |
| NumPy | Pandas |
| 基础数据类型 | 扩展数据类型 |
| 关注数据的结构表达 | 关注数据的应用表达 |
| 维度：数据间关系 | 数据与索引间关系 |

dropout 是指 随机用一定概率 把一些 节点失效，进行参与训练 放置数据整理上陷入overfitting 局部最优解。

**fig, ax = plt.subplots(figsize=(12, 9))**

fig代表绘图窗口（Figure），ax代表这个绘图窗口上的坐标系（axes）。后面的ax.xxx则是表示对ax坐标系进行xxx操作



|  |
| --- |
| scatter/scatter3做散点的能力更强，因为他可以对散点进行单独设置 所以消耗也比plot/plot3大 所以如果每个散点都是一致的时候，还是用plot/plot3好以下 如果要做一些plot没法完成的事情那就只能用scatter了  pandas |

loc——通过行标签索引行数据   
iloc——通过行号索引行数据   
ix——通过行标签或者行号索引行数据（基于loc和iloc 的混合）   
同理，索引列数据也是如此！plt.xticks(rotation=90) #设置双坐标轴,右侧Y轴

所以问题当中df.drop（‘列名’, axis=1）代表将‘列名’对应的列标签（们）沿着水平的方向依次删掉**,即删掉了一列**。**区别:**如果我们调用df.mean(axis=1),我们将得到按行计算的均值

**轴用来为超过一维的数组定义的属性，二维数据拥有两个轴：第0轴沿着行的垂直往下，第1轴沿着列的方向水平延伸.**

**8**损失函数L(Y, P(Y|X))表达的是样本X在分类Y的情况下，使概率P(Y|X)达到最大值（换言之，**就是利用已知的样本分布，找到最有可能（即最大概率）导致这种分布的参数值；或者说什么样的参数才能使我们观测到目前这组数据的概率最大**）。因为log函数是单调递增的，所以logP(Y|X)也会达到最大值，因此在前面加上负号之后，最大化P(Y|X)就等价于最小化L了。

9. Hinge 损失函数的标准形式:

L(y)=max(0,1−y~~y~~),y=±1

可以看出，当|y|>=1时，L(y)=0。

10而**正则化项一般是模型复杂度的单调递增函数，模型越复杂，正则化值就越大**，所以可以把正则化项加入到损失函数里面作为惩罚项，这样就保证了模型训练的精度和复杂度之间的平衡。

常见的模型集成方法有：

**Bagging**：将多个模型（基学习器）的预测结果简单地加权平均或者投票；

**Boosting**: 其思想有点像“知错能改”，每训练一个基学习器，是为了弥补上一个基学习器所犯的错误。其中著名的算法有AdaBoost，Gradient Boost。Gradient Boost Decision Tree就用到了这种思想；

**Stacking**： 用新的模型（次学习器）去学习怎么组合那些基学习器，如果把Bagging看作是多个基分类器的线性组合，那么Stacking就是多个基分类器的非线性组合；

**Blending**： Blending和Stacking有些类似，具体的区别应该在于数据的划分。

但是，神经网络的具体实现存在以下问题：

梯度弥散(Gradient Vanishment)

连接权重W初始化

随机梯度(SGD)学习速率的调整

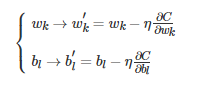
过拟合（这是所有机器学习算法所共同面对的问题）

深度学习基础之-梯度弥散和梯度爆炸及解决办法

<https://blog.csdn.net/u013403054/article/details/78424095>

为什么梯度的负方向是局部下降最快的方向？共线方向

**而 df(x)正是代表函数值在x处的**梯度**。前面又说明了v的方向和df(x)方向相同的时候，点积值（变化值）最大，所以说明了梯度方向是函数局部上升最快的方向。也就证明了梯度的负方向是局部下降最快的方向！**f(x+v)-f(x)=d f(x)v ,v指的是自变量的变化量

AB=abcos(thea) 

**过拟合**是机器学习中一个常见的问题，我们希望训练出来的模型具有很好的泛化能力。在神经网络中，因为参数众多，经常出现参数比数据还多的情况，这就非常容易出现只是记忆了训练集特征的情况。为了解决这个问题，Hinton老爷子提出了一个思路简单但是非常有效的方法，**Dropout**。它的大致思路是，在训练时将神经网络某一层的输出节点数据随机丢弃一部分，就是这么任性！这种做法实质上等于创造出了很多新的随机样本，通过增大样本量、减少特征数量来防止过拟合。

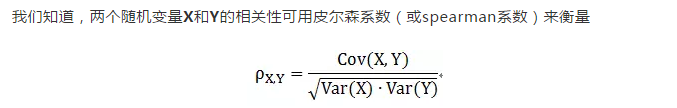
刻画数据相对分散性的度量是**变异系数**，计算公式为变异系数是一个无量纲的量。

CV=100\*标准差/均值（%）

a, b = 1, 2

切片的语法：[起始:结束:步长]

注意：选取的区间属于左闭右开型，即从"起始"位开始，到"结束"位的前一位结束（不包含结束位本身)。

Python切片的语法：[起始:结束:步长]

**注意：选取的区间属于左闭右开型，即从"起始"位开始，到"结束"位的前一位结束（不包含结束位本身)。**

mystr.split(str=" ", 2) 字符串转化成链表

以字符串str中空格为间隔转换成链表

append通过append可以向列表添加元素

extend通过extend可以将另一个集合中的元素逐一添加到列表中

index和find一样就是找

del：根据下标进行删除 del movieName[2]

pop：删除最后一个元素 movieName.pop()

remove：根据元素的值进行删除movieName.remove('指环王')

sort方法是将list按特定顺序重新排列，默认为由小到大，参数reverse=True可改为倒序，由大到小（先由小到大，再改我由大到小）。

reverse方法是将list逆置。

demo:clear清空整个字典 info.clear()

del info['name']

del info

返回一个包含所有（键，值）元祖的列表items

**cmp在比较字典数据时，先比较键，再比较值。**

**我们可以用id()来判断两个变量是否为同一个值的引用**

调用函数很简单的，通过 **函数名()**即可完成调用

修改全局变量 global 全局变量

可以根据\_\_name\_\_变量的结果能够判断出，是直接执行的python脚本还是被引入执行的，从而能够有选择性的执行测试代码

\_\_all\_\_影响导入哪些功能，\_\_iniSt\_\_影响包里面的py文件能不能让人用，前提还要在init文件中加all

深拷贝和浅拷贝

Python隔代回收。没相互引用，且计算不为0的放到第1代，其有相互引用，且不为0的还放在第0代

一.垃圾回收机制

Python中的垃圾回收是以引用计数为主，分代收集为辅。

LR比较适合解决线性可分的问题

使用dataframe[column\_name]，返回series格式数据。 series序列数据类似于list，你可以近似等同于list。 只不过返回数据中会多一列index索引。Series 是一个带有 **名称** 和索引的一维数组，既然是数组，肯定要说到的就是数组中的元素类型，在 Series 中包含的数据类型可以是整数、浮点、字符串、Python对象等。

# 存储了 4 个年龄：18/30/25/40

user\_age = pd.Series(data=[18, 30, 25, 40])

0 18

1 30

2 25

3 40

dtype: int64

Series 包含了 dict 的特点，也就意味着可以使用与 dict 类似的一些操作。我们可以将 index 中的元素看成是 dict 中的 key。Series 除了像 dict 外，也非常像 ndarray，这也就意味着可以采用切片操作。user\_age[:3]

数据集中，不同的列的数据可能在不同量级，如果直接进行分析。模型会认为数字大的影响力大，数字小的影响力小。 最终结果可能导致量级小的变量被剔除出模型。因此需要将数据归一化，变成同一量级的数据，这就是归一化操作。

DataFrame 是一个带有**索引**的二维数据结构，每列可以有自己的名字，并且可以有不同的数据类型。你可以把它想象成一个 excel 表格或者数据库中的一张表，DataFrame 是最常用的 Pandas 对象。

从表面上看解释器仅仅会解释这两句代码，因为函数在 没有被调用之前其内部代码不会被执行。@W1 def f1():等价于 f1=w1(f1)

当我们遇到python中——name——可以用来分辨程序是直接执行，还是被导入。

当我们是直接执行时。还不是在其他文件中导入时，——name——变量的值是main，就会执行下列if语句 if name==“”main“”，判断通过。执行之后的主程序代码。

10.**python中zip()** 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。

如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 \* 号操作符，可以将元组解压为列表。

>>>a = [1,2,3] >>> b = [4,5,6] >>> c = [4,5,6,7,8]

>>> zipped = zip(a,b) # 打包为元组的列表 [(1, 4), (2, 5), (3, 6)]

>>> zip(a,c) # 元素个数与最短的列表一致 [(1, 4), (2, 5), (3, 6)]

zip(\*zipped) # 与 zip 相反，可理解为解压，返回二维矩阵式 [(1, 2, 3), (4, 5, 6)]

11. numpy.random.uniform(low,high,size)

从一个均匀分布[low,high)中随机采样，注意定义域是左闭右开，即包含low，不包含high

# 查看某个指定环境的已安装包

conda list -n python34

# 查找package信息

conda search numpy

AdamOptimizer优化器

13.解决办法：

打开 Anaconda Prompt ，输入命令 activate tensorflow 以激活 tensorflow 环境

在 tensorflow 环境下安装 scikit-learn ，输入命令 conda install scikit-learn 即可

这样就在 tensorflow 环境安装了 scikit-learn ，可以成功 import 。

深度信念网络（DBN）

Signomd

1. 可以引入非线性（2）容易求导（3）可以把数据压缩，这样数据不容易发散。
2. Tanh [-1,1]
3. 。第一种很明显，可以通过使用别的激活函数；第二种可以使用层归一化；第三种是在权重的初始化上下功夫，第四种是构建新的网络结构~。