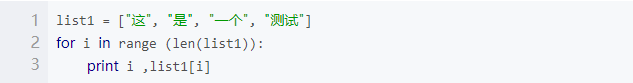
2019.9.24

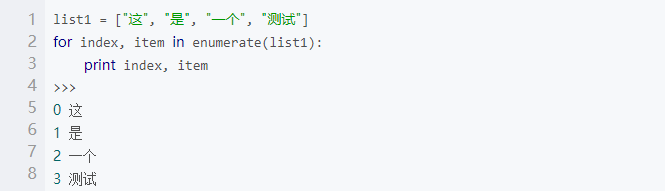
复现KinomeX

1. **enumerate()**

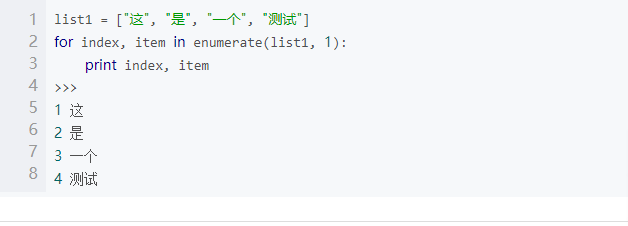
如果是一个列表，既要遍历索引又要遍历元素时，首先可以这样写：



上述方法有些累赘，利用enumerate()会更加直接和优美：



enumerate还可以接收第二个参数，用于指定索引起始值，如：



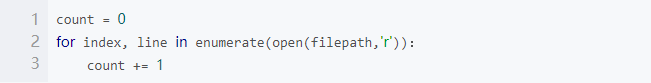
**补充**

如果要统计文件的行数，可以这样写：



这种方法简单，但是可能比较慢，当文件比较大时甚至不能工作。

可以利用enumerate()：

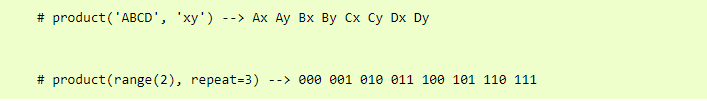


1. **itertools** (python内置的模块)

<https://blog.csdn.net/neweastsun/article/details/51965226>

1. product()

product(iter1,iter2, ... iterN, [repeat=1]);创建一个迭代器，生成表示item1，item2等中的项目的笛卡尔积的元组，repeat是一个关键字参数，指定重复生成序列的次数。



for i in product([1, 2, 3], [4, 5], [6, 7]):

    print i

(1, 4, 6)

(1, 4, 7)

(1, 5, 6)

(1, 5, 7)

(2, 4, 6)

(2, 4, 7)

(2, 5, 6)

(2, 5, 7)

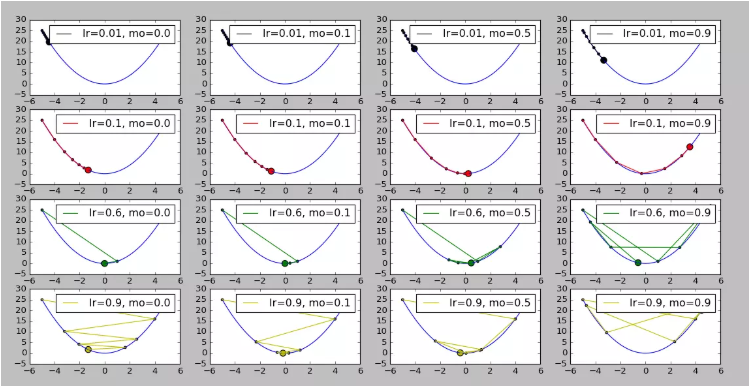
(3, 4, 6)

(3, 4, 7)

(3, 5, 6)

(3, 5, 7)

1. **momentum**冲量（梯度下降法中使用）
2. 冲量对学习率的影响，一行的图的学习率lr一样，每一列的momentum一样，最左列为不使用momentum时的收敛情况



**简单分析一下运行结果：**

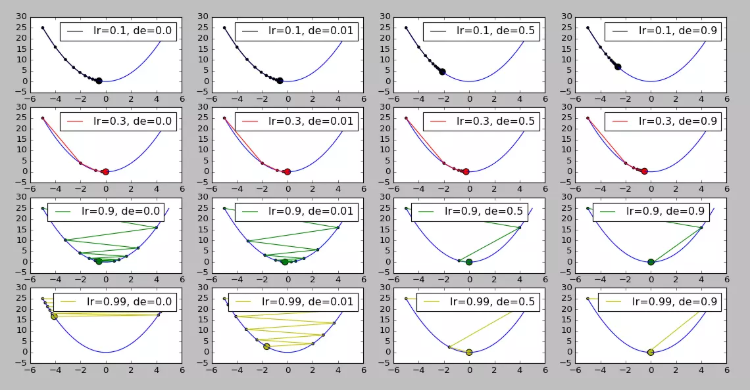
从第一行可看出：在学习率较小的时候，适当的momentum能够起到一个加速收敛速度的作用。

从第四行可看出：在学习率较大的时候，适当的momentum能够起到一个减小收敛时震荡幅度的作用。

然而在第二行与第三行的最后一列图片中也发现了一个问题，当momentum较大时，原本能够正确收敛的时候却因为刹不住车跑过头了。那么怎么继续解决这个新出现的问题呢？

1. **decay** 学习率衰减因子
   * decay越小，学习率衰减地越慢，当decay = 0时，学习率保持不变。
   * decay越大，学习率衰减地越快，当decay = 1时，学习率衰减最快。

运行结果如下图所示，其中每行图片的学习率一样、decay依次增加，每列图片decay一样，学习率依次增加：

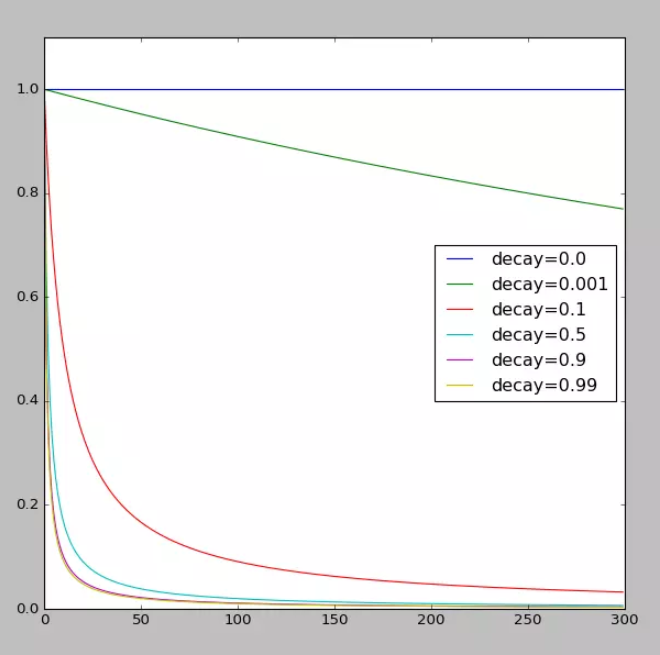


**简单分析一下结果：**

* 在所有行中均可以看出，decay越大，学习率衰减地越快。
* 在第三行与第四行可看到，decay确实能够对震荡起到减缓的作用。

**运行结果如下图所示。**

可以看到，当decay为0.1时，50次迭代后学习率已从1.0急剧降低到了0.2。如果decay设置得太大，则可能会收敛到一个不是极值的地方呢。看来调参真是任重而道远：



1. **cross\_entropy\_with\_logits** : TensorFlow中四种不同交叉熵函数

<https://blog.csdn.net/qq_35203425/article/details/79773459>

* tf.nn.softmax\_cross\_entropy\_with\_logits()
* tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits()
* tf.nn.sigmoid\_cross\_entropy\_with\_logits()
* tf.nn.weighted\_cross\_entropy\_with\_logits()

1. **Batch Normalization** 批量标准化

<https://blog.csdn.net/marsggbo/article/details/77771497>

* Accelerating Deep Network Training by Reducing Internal Covariate Shift
* 在训练开始的时候，通过显式地迫使网络上的激活函数让整个网络上采用高斯分布来初始化神经网络。
* 在实际操作中，运用这项技术相当于在全连接层(或者卷积层，我们很快将会看到)后面嵌入BatchNorm层，并嵌在非线性(层)前。

1. **权重初始化方法**（对Keras层设置初始化权重）

<https://keras-cn.readthedocs.io/en/latest/other/initializations/>

1. 初始化的关键字：kernel\_initializer 和 bias\_initializer
2. 所有初始化方法的父类，不能直接使用，如果想要定义自己的初始化方法，要继承此类。
3. 预定义初始化方法：
4. Zeros: 全零初始化

keras.initializers.Zeros()

1. Ones: 全1初始化

keras.initializers.Ones()

1. Constant: 初始化为固定值value

keras.initializers.Constant(value=0)

1. RandomNormal: 正态分布初始化

keras.initializers.RandomNormal(mean=0.0, stddev=0.05, seed=None))

* mean：均值
* stddev：标准差
* seed：随机数种子

1. RandomUniform: 均匀分布初始化

keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)

* minval：均匀分布下边界
* maxval：均匀分布上边界
* seed：随机数种子

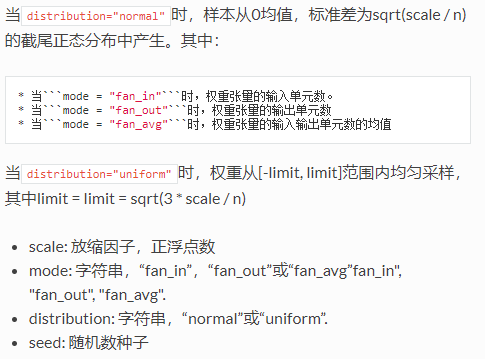
1. TruncatedNormal 截尾高斯分布初始化

keras.initializers.TruncatedNormal(mean=0.0, stddev=0.05, seed=None)

该初始化方法与RandomNormal类似，但位于均值两个标准差以外的数据将会被丢弃并重新生成，形成截尾分布。该分布是神经网络权重和滤波器的推荐初始化方法。

1. VarianceScaling 该初始化方法能够自适应目标张量的shape。

keras.initializers.VarianceScaling(scale=1.0, mode='fan\_in', distribution='normal', seed=None)



1. Orthogonal 随机正交矩阵初始化

keras.initializers.Orthogonal(gain=1.0, seed=None)

* + - * gain: 正交矩阵的乘性系数
      * seed：随机数种子

1. Identiy 单位矩阵初始化，仅适用于2D方阵

keras.initializers.Identity(gain=1.0)

* gain：单位矩阵的乘性系数

1. lecun\_uniform: LeCun均匀分布初始化方法

lecun\_uniform(seed=None)

参数由[-limit, limit]的区间中均匀采样获得，其中limit=sqrt(3 / fan\_in), fin\_in是权重向量的输入单元数（扇入）

1. lecun\_normal LeCun正态分布初始化方法

lecun\_normal(seed=None)

参数由0均值，标准差为stddev = sqrt(1 / fan\_in)的正态分布产生，其中fan\_in和fan\_out是权重张量的扇入扇出（即输入和输出单元数目）

1. glorot\_normal: Glorot正态分布初始化方法，也称作Xavier正态分布初始化

glorot\_normal(seed=None)

参数由0均值，标准差为sqrt(2 / (fan\_in + fan\_out))的正态分布产生，其中fan\_in和fan\_out是权重张量的扇入扇出（即输入和输出单元数目）

1. glorot\_uniform Glorot均匀分布初始化方法，又成Xavier均匀初始化

glorot\_uniform(seed=None)

参数从[-limit, limit]的均匀分布产生，其中limit为sqrt(6 / (fan\_in + fan\_out))。fan\_in为权值张量的输入单元数，fan\_out是权重张量的输出单元数。

1. he\_normal: He正态分布初始化方法

he\_normal(seed=None)

参数由0均值，标准差为sqrt(2 / fan\_in) 的正态分布产生，其中fan\_in权重张量的扇入

1. he\_uniform: LeCun均匀分布初始化方法

he\_normal(seed=None)

参数由[-limit, limit]的区间中均匀采样获得，其中limit=sqrt(6 / fan\_in), fin\_in是权重向量的输入单元数（扇入）

1. **dropout**

dropout是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。注意是暂时，对于随机梯度下降来说，由于是随机丢弃，故而每一个mini-batch都在训练不同的网络。

防止过拟合的方法：

* 提前终止（当验证集上的效果变差的时候）
* L1和L2正则化加权
* soft weight sharing

1. **isinstance()**

判断一个对象是否是一个已知的类型，类似type()

isinstance(object, classinfo)

isinstance() 与 type() 区别：

* + type() 不会认为子类是一种父类类型，不考虑继承关系。
  + isinstance() 会认为子类是一种父类类型，考虑继承关系。

如果要判断两个类型是否相同推荐使用 isinstance()。



1. 分类中解决类别不平衡问题

<https://zhuanlan.zhihu.com/p/36381828>

1. transformation

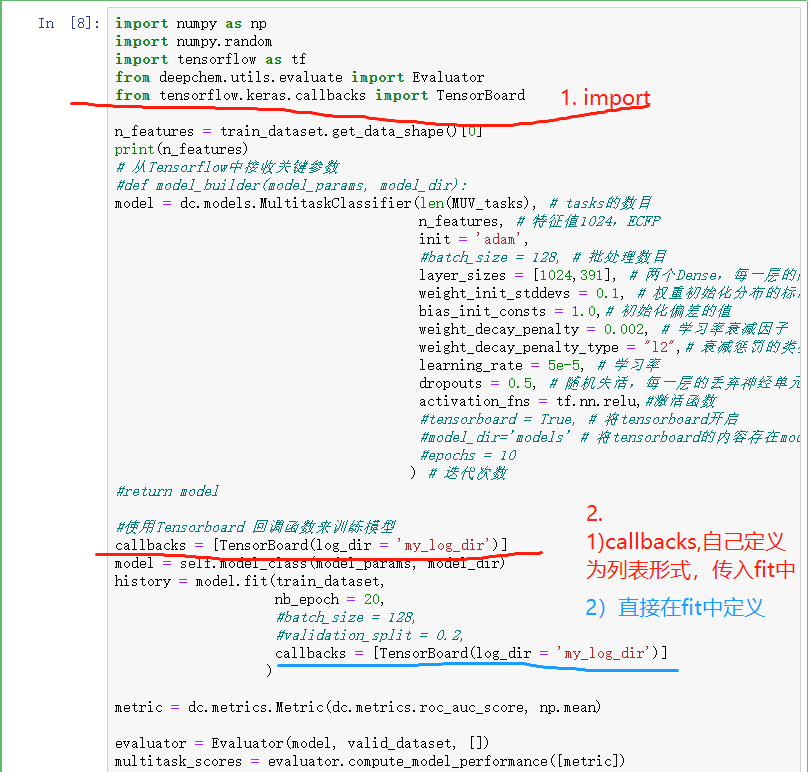
<https://zhuanlan.zhihu.com/p/30751039>

1. 开启tensorboard
   * + 1. 在deepchem中

在构建模型时，将tensorboard = True, model\_dir='models'传入



* + - 1. 在tensorflow中



* + - 1. 在tensorboard文件存储位置上一级加载deepchem虚拟环境，并输入：

tensorboard --logdir models/ --port 6006

打开[http://localhost:6006](http://localhost:6006/" \t "_blank) 网站

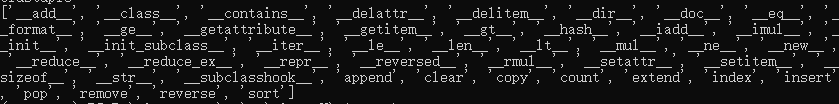
1. print(\*)

输出为文件句柄，可通过dir访问

例如：print(clustuple)



print(dir(clustuple))



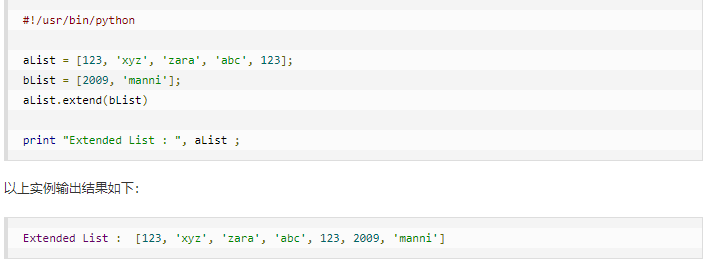
print(clustuple.\_\_add\_\_)



1. extend()

在列表末尾一次性追加另一序列中的多个值（用新列表扩展原来的列表）

list.extend(seq)



1. jupyter notebook

当模块的代码发生改变时，jupyter的环境还是原来的环境，需要重新添加kernel <https://blog.csdn.net/u012151283/article/details/54565467>

