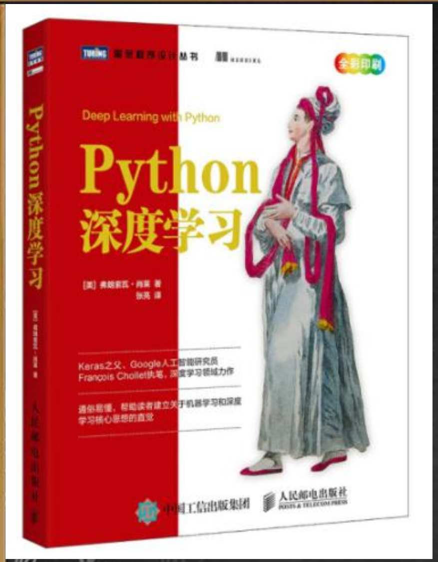


“神经网络的引擎：基于梯度的优化”

祥水村东头@AI识堂

《Python 深度学习》读书笔记系列课程(对应2.4节 pp36-41)

本次胶片内容、及涉及相关代码均可移步至Github进行下载

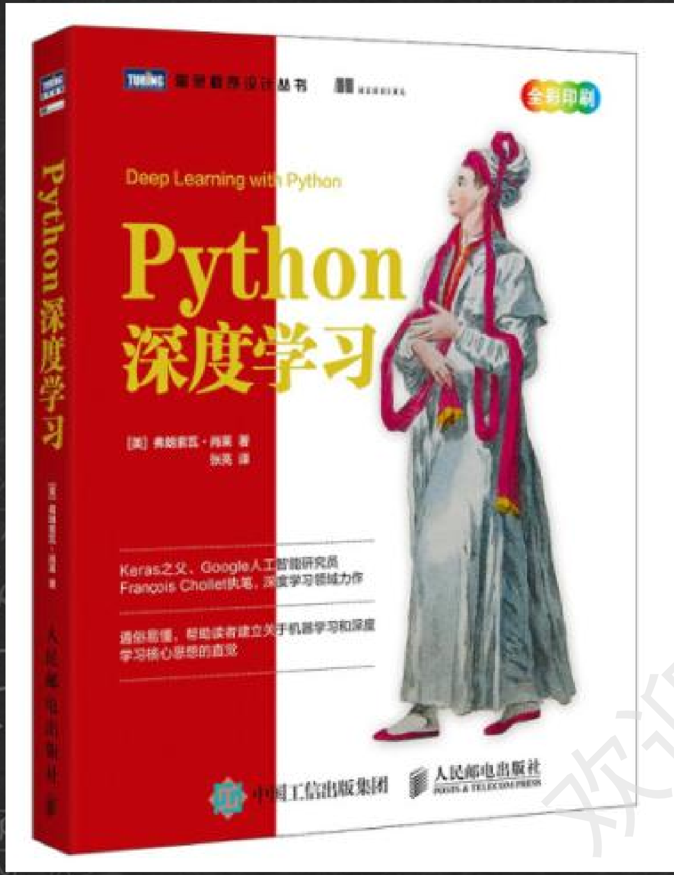
我的代码 Github 地址：
https://github.com/david-cal/Reading-Note-for-Chollet-of-Deep-Learning-with-Python

欢迎关注B站：祥水村东头

目录

欢迎关注B站：梓水村东头

1 模型训练4部曲，何谓“学习”



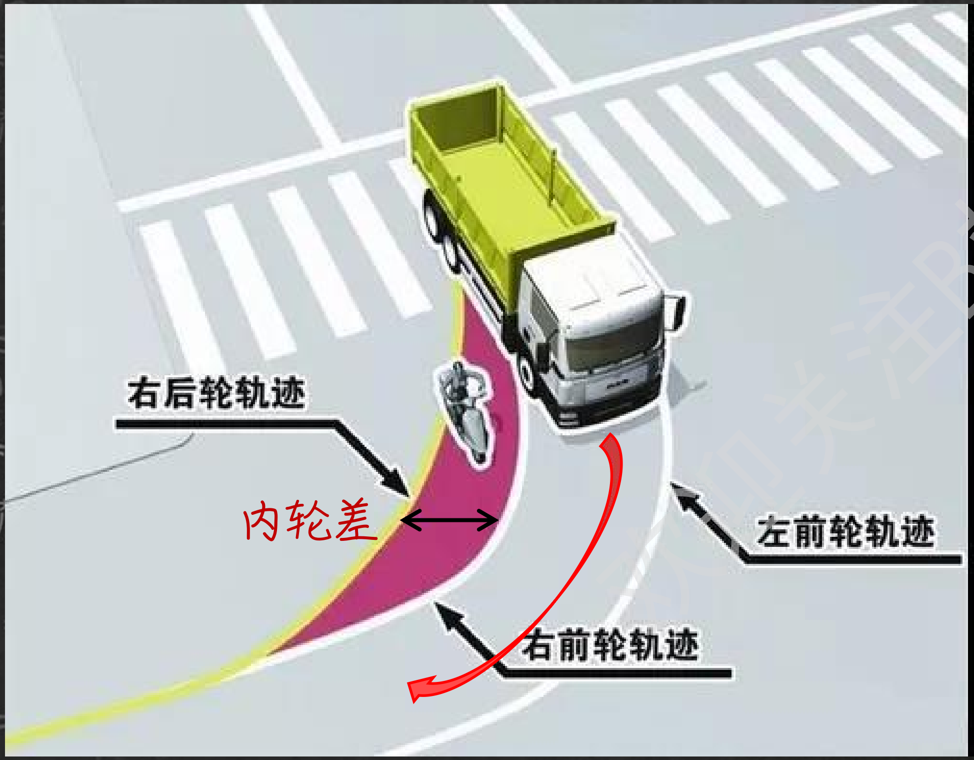
- 所谓“深度学习”，究竟是如何学习的？
- 本质上就是不断调整网络模型中的参数使其输出最佳结果
- 先来观察下现实世界中的“学习”！

欢迎关注B站：祥文村东头

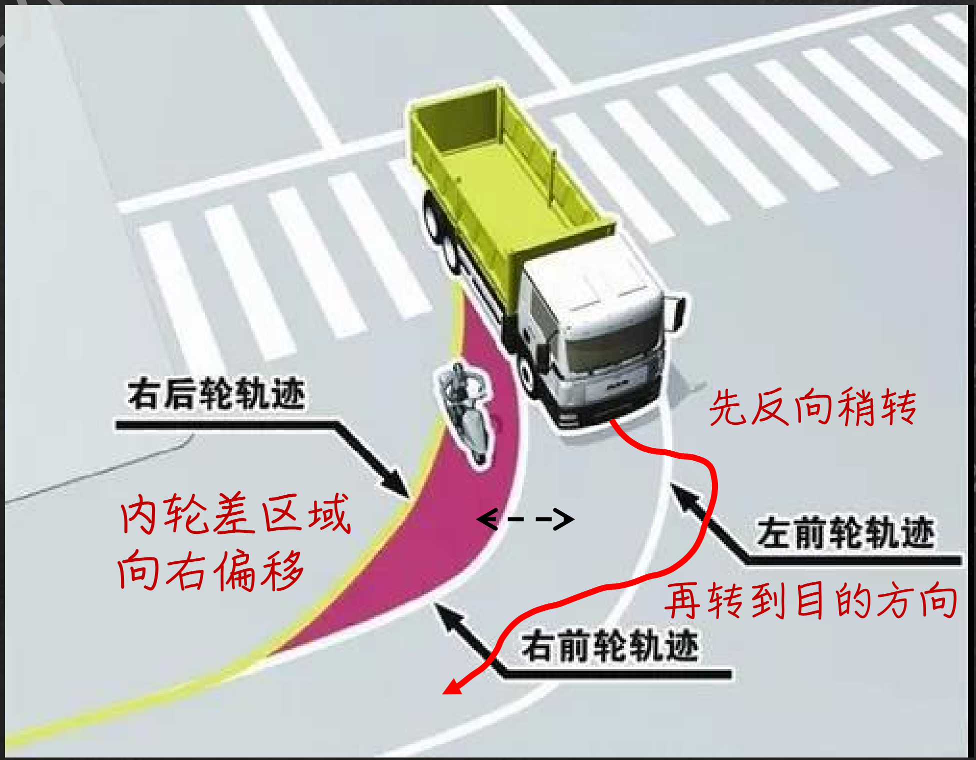
现实世界中的“学习”——驾驶经验的学习

- 大型货车在拐弯时，由于车身较长，右后轮与右前轮会形成内轮差，会对驾驶员造成视觉盲区，若恰有行人或小车在此区域，则会造成交通事故。

- 驾驶员在驾驶大型车辆拐弯时，提前观察好前方左右，若看到侧方有行人或车辆，先向反方向稍打轮，然后再向目的方向拐弯，为处于原本内轮差的物体预留出足够空间。（欲左先右、欲右先左）



学习后



1 模型训练4部曲，训练循环

- 现实世界中的“学习”

观察侧方情况
↓
按照当前方位、速度驾驶
↓
评估侧方物体与本车距离
↓
根据与周边物体距离，调整驾驶方位、速度

- 神经网络中的“学习”

批量抽取训练集中的样本(x)和标签(y)数据 —— 批导入
↓
x在网络上逐层传递，并得到y_pred —— 前传
↓
评估y_pred与y之间的误差 —— 定损
↓
根据误差调整网络中的各个参数 —— 后传（反传）*核心内容

训练循环

欢迎关注B站：祥水村东头

再回顾

批量抽取训练集中的样本(x)和标签(y)数据

批导入

x在网络上逐层传递，并得到y_pred

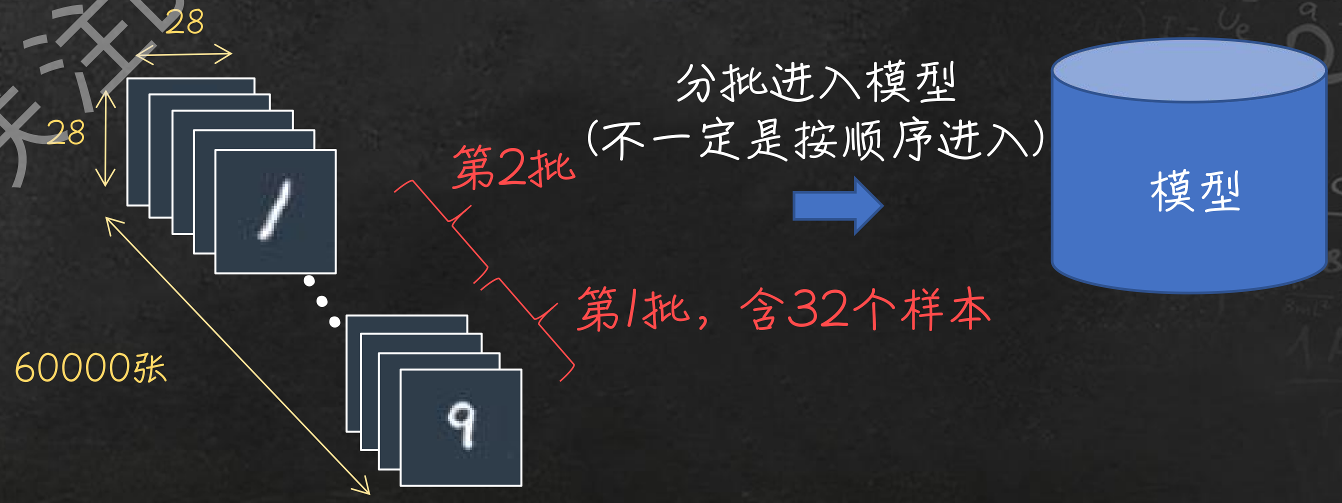
评估y_pred与y之间的误差

根据误差调整网络中的各个参数

训练循环

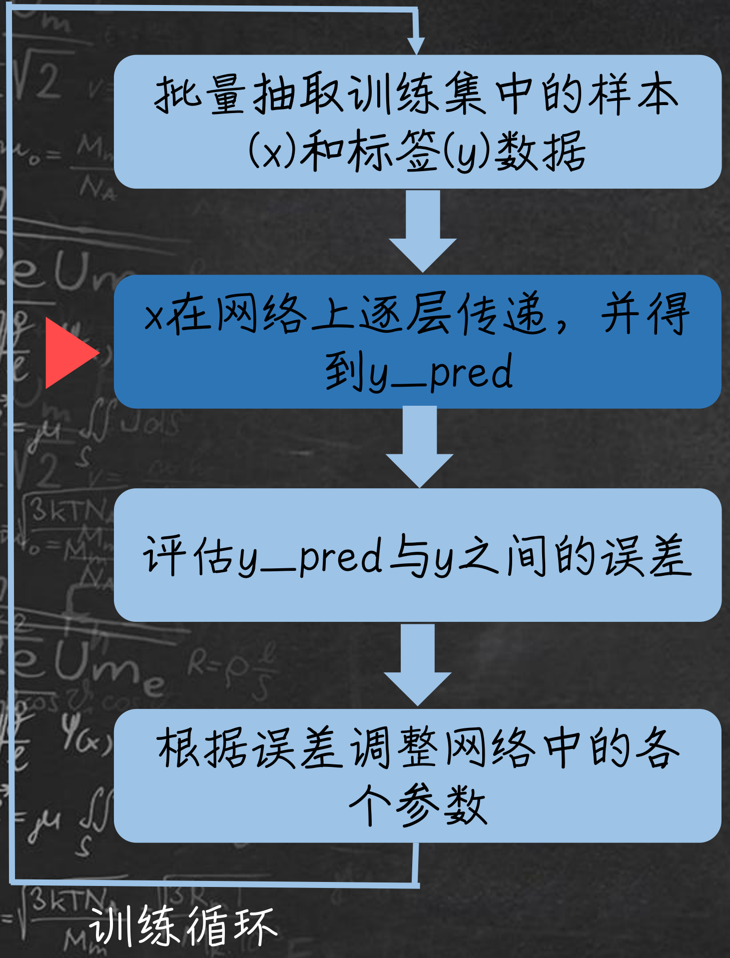
```
#训练模型
network.fit( processed_train_images,
processed_train_labels, batch_size = 32,
epochs = 10)
```

*每批次包含32个样本（缺省值）
*训练一次全量样本需要1875个批次
*全量样本要反复进入模型10轮

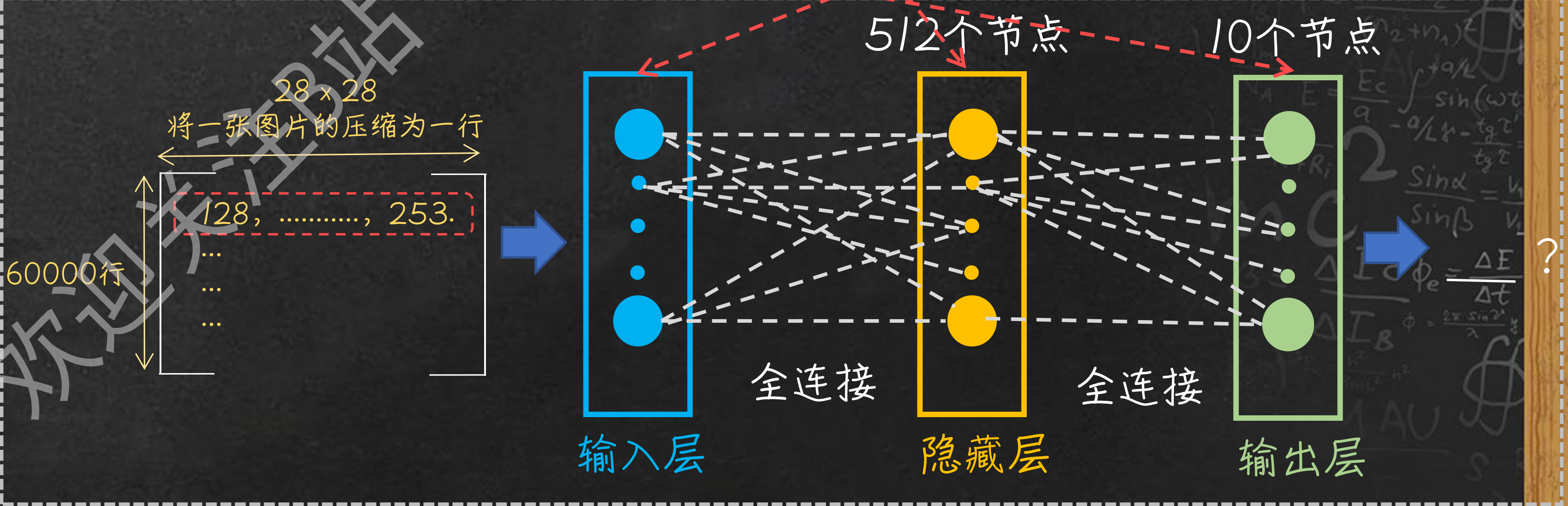


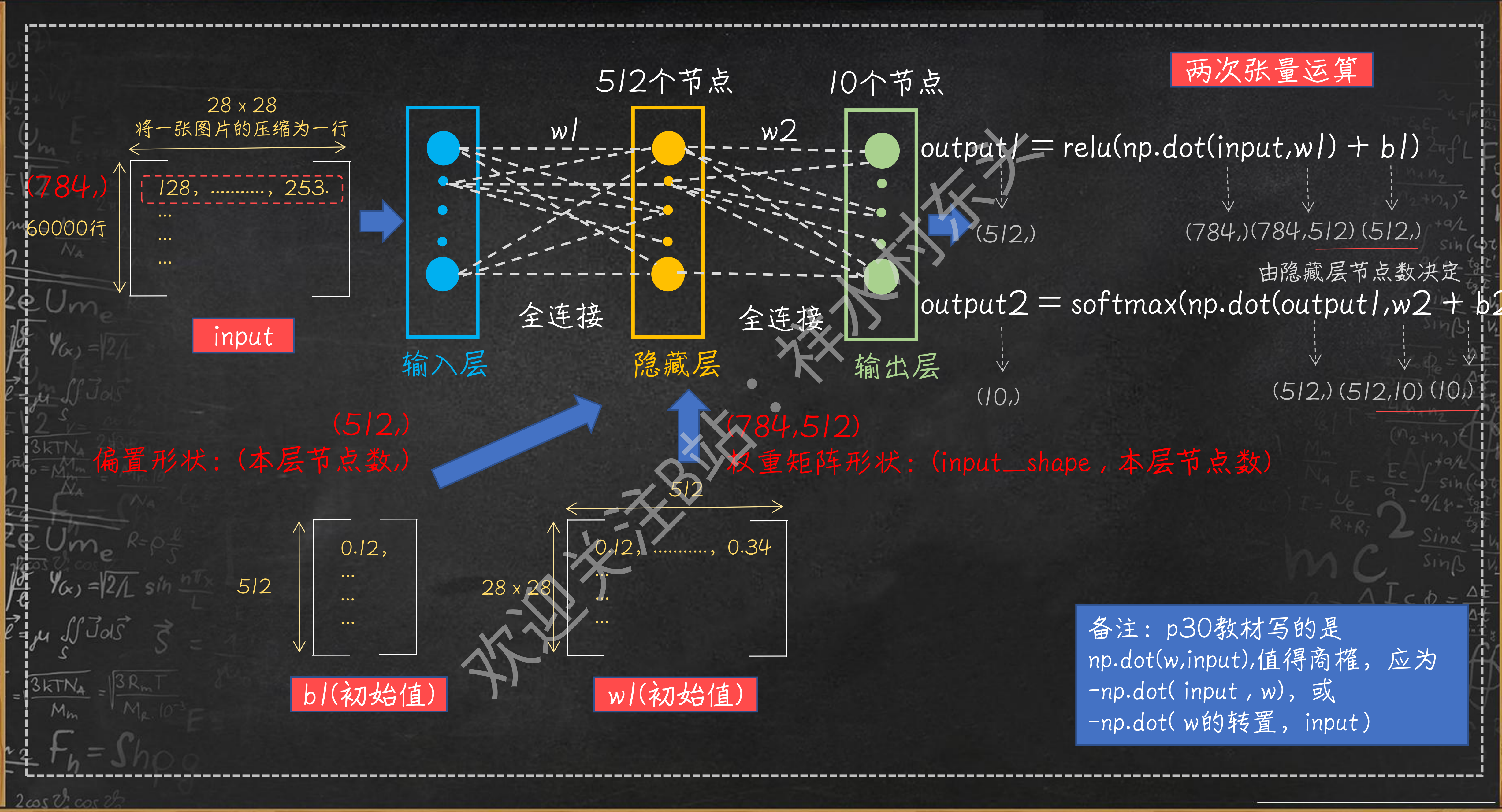
欢迎关注B站：樟树村东头

2 前传，输入张量从前至后逐层进行张量运算(forward pass)

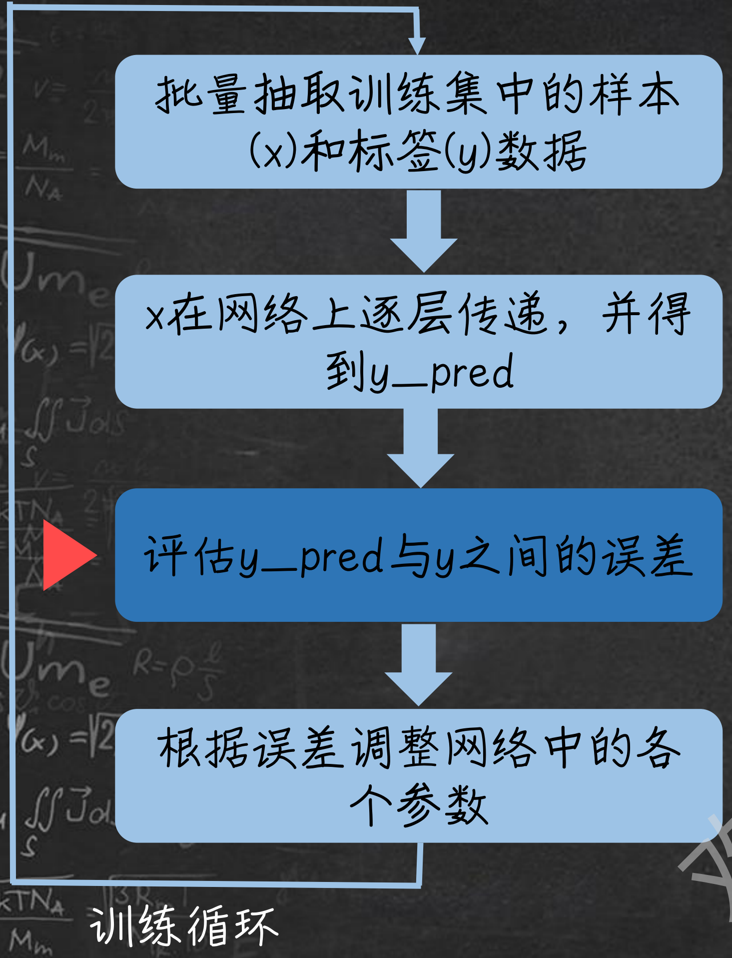



```
#初始化神经网络模型
network = models.Sequential()

#叠加首层——全连接层,首层还需要指定input_shape
network.add(layers.Dense(512,activation = 'relu', input_shape = (28 * 28 ,)))

#叠加第二层——全连接层，注意是躲分类问题，所以激活函数应使用softmax
network.add(layers.Dense(10, activation = 'softmax'))
```




两次张量运算
28 x 28
将一张图片的压缩为一行
(784,)
60000行
128，…………，253.
input
512个节点
10个节点
w1
w2
输入层
隐藏层
输出层
全连接
全连接
output1 = relu(np.dot(input,w1) + b1)
(512,)
(784,)(784,512) (512,)
由隐藏层节点数决定
output2 = softmax(np.dot(output1,w2 + b2))
(10,)
(512,) (512,10) (10,)
(512,)
偏置形状：(本层节点数,)
(784,512)
权重矩阵形状：(input_shape，本层节点数)
512
0.12，
512
28 x 28
0.12，…………，0.34
b1(初始值)
w1(初始值)
备注：p30教材写的是
np.dot(w,input),值得商榷，应为
-np.dot(input，w)，或
-np.dot(w的转置，input)

3 定损，输入数据在网络上所产生的损失（loss function）




```
#编译模型，预设优化器、损失函数、监控指标
network.compile(optimizer = 'rmsprop',
loss = 'categorical_crossentropy',
metrics = ['accuracy'])
```

多分类问题
（场景）

交叉熵
（损失算法，loss function或error function）

何为熵(entropy)？

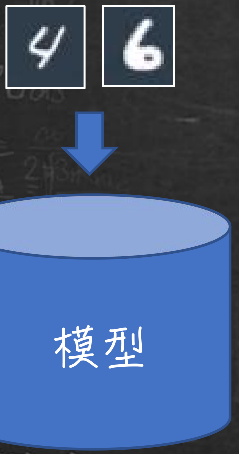
-1865年由德国物理学家克劳修斯提出，用于描述能量退化；“熵增”则表示事物本体由有序到无序的过程；这一概念也从物理领域延伸到生化、信息学、管理学、社会学等诸多领域。

-在本文中，熵主要用于描述模型预测结果与真实结果之间的差异程度

3 定损，输入数据在网络上所产生的损失

如何计算 categorical_crossentropy ?

- 假设有两个样本——“4”、“6”，依次进入模型
- 可分别得到模型在两个样本上的输出概率，每个样本对应10个类别上的概率值



类别(class)	样本“4”预测输出概率	样本“4”真实输出概率	样本“6”预测输出概率	样本“6”真实输出概率
0	0	0	0	0
1	0	0	0	0
2	0.7	0	0	0
3	0	0	0	0
4	0.3	1	0	0
5	0	0	0	0
6	0	0	0.4	1
7	0	0	0	0
8	0	0	0	0
9	0	0	0.6	0

以cross_entropy算法作为
loss function计算公式

$$loss = -\sum_{i=1}^{n} \hat{y}_{i1} log y_{i1} + \hat{y}_{i2} log y_{i2} + \cdots + \hat{y}_{im} log y_{im}$$

样本数　以e为底数的运算　真实概率　预测概率

网络在样本“4”上的损失值

预测概率

-(0xlog(0) + ...+ 0xlog(0.7) +.... + 1xlog(0.3) +...) = 1.2

真实概率

网络在样本“6”上的损失值

真实概率

-(0xlog(0) + ...+ 1xlog(0.4) +.... + 0xlog(0.6) +...) = 0.9

预测概率

求和

2.1

网络在2个样本上的总损失

欢迎关注B站：祥水村东头

- 在现实世界中，熵值越大、代表物质越不稳定；反之，熵越小、越稳定
- 同理，在网络模型中，以cross_entropy作为衡量模型损失的度量(loss function)，其值越小，表示模型预测准确程度越高
- 当cross_entropy = 0时，表示预测与真实值完全一致
- 因此，我们的目标是需要找到使loss function值达到极小值的模型参数组合（各种w和b）

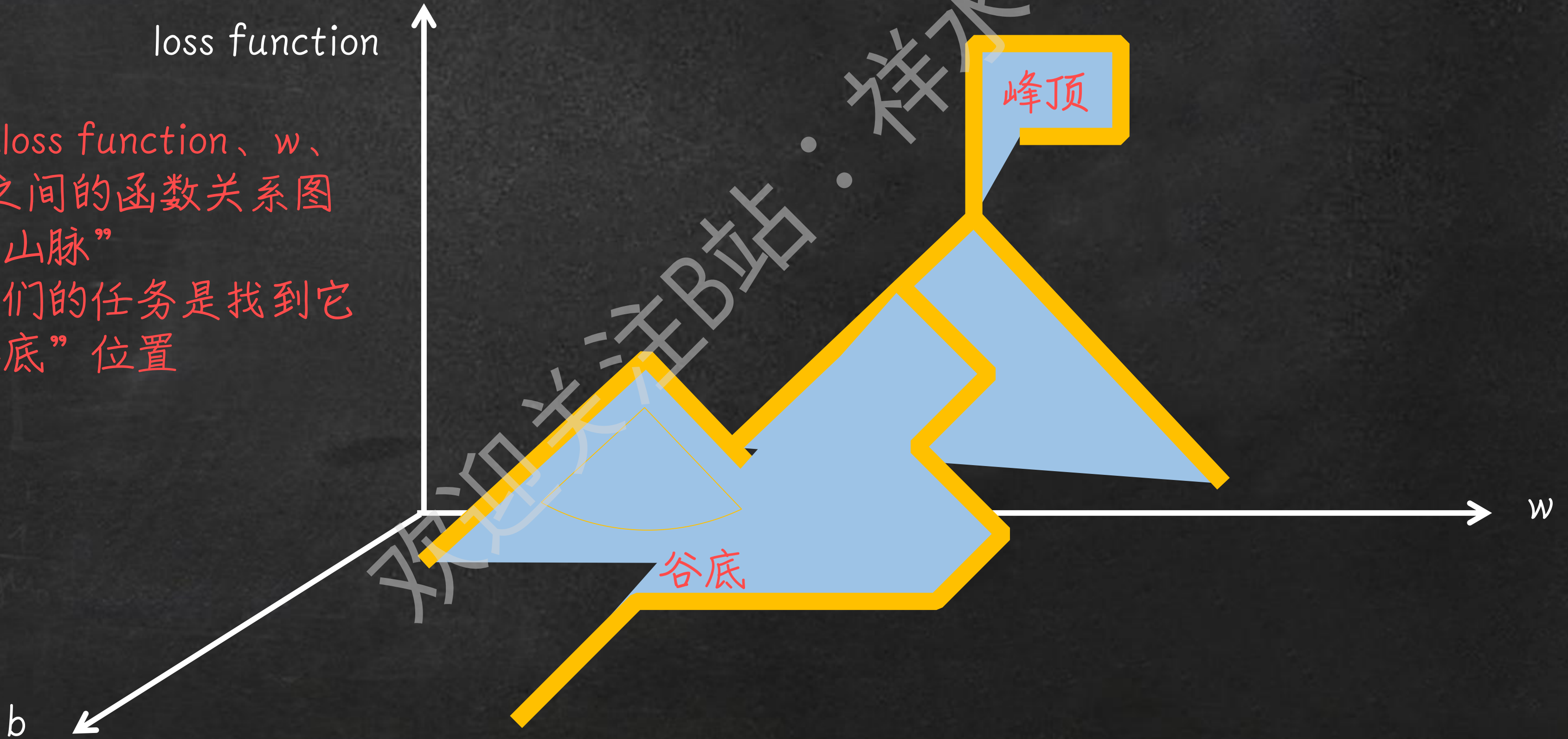
欢迎关注B站：祥水村东头

4 回传，向损失函数变小的方向更新模型参数（back propagation）

- 假设参与模型张量运算的参数只有2个——w和b，即y′ = np.dot(w ,input) + b
- loss function是衡量真实值y与预测值y′之间的差异程度
- 因此，我们可以绘制出loss function、w、b三者之间的函数关系图

✓ 如果把loss function、w、b三者之间的函数关系图看成“山脉”

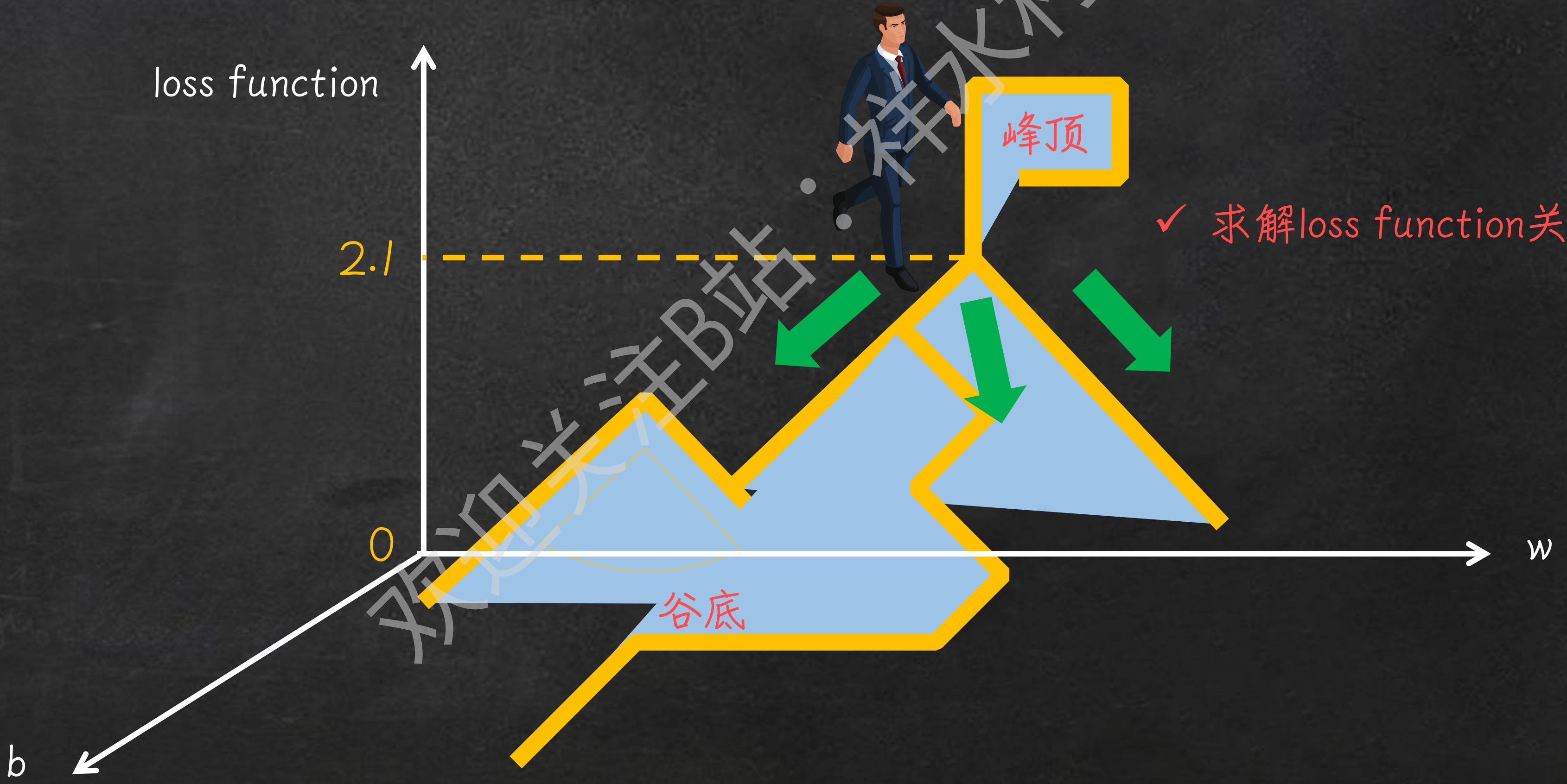
✓ 那么我们的任务是找到它的“谷底”位置



欢迎关注B站：祥水村东头

4 回传，向损失函数变小的方向更新模型参数（back propagation）

- 假设网络模型参数初始值让我们在峰顶的高度（如cross entropy = 2.1）
- 此时，我们脚下有很多条路可以下山，那么究竟应该走哪条路可以让我们达到谷底的位置？



✓ 求解loss function关于w和b的导数

欢迎关注B站：祥水村务子

回顾：中学阶段我们所学的导数

- 假设一个连续的光滑函数 $y = f(x)$，即x的微小变化只能带来y的微小变化，则有

$$y + \Delta y = f(x + \Delta x)$$

- 在某点附近，当 Δx 足够小时，可以将f近似为斜率为a的线性函数，则

$$\Delta y = a * \Delta x$$

- 斜率a称为f在某点的导数（derivative）

✓ $a > 0$，Δx将导致f(x)变大
✓ $a < 0$，Δx将导致f(x)变小
✓ $a = 0$，即为函数的极值点
✓ a的绝对值表示f(x)增大/小的快慢

- 因此，如果希望f(x)变小，x需要向a的相反方向移动

张量的导数

- 同理，若要找到使loss function最小的w和b的参数组合，需要求解loss function关于w和b的导数

$$L + \Delta L = Loss\_Function((w + \Delta w), (b + \Delta b))$$

- 梯度（gradient），张量的“导数”
- 求解梯度为0时所对应的张量，即找到极值点，此方法成为解析法
- 但是，在神经网络中的张量运算极其复杂，w和b的个数动辄成千上万，我们无法通过解析法找到极值点
- 梯度下降法：基于当前损失值对w和b一点点调节，即沿着梯度值的反方向不断更新w和b，这样loss function会逐渐变小，直到我们找到极小值

梯度 gradient $= \left(\partial(L)/\partial(w), \partial(L)/\partial(b) \right)$

更新 $w' = w - learning\_rate * \partial(L)/\partial(w)$

$b' = b - learning\_rate * \partial(L)/\partial(b)$

下降 $L' = Loss\_Function(w', b')$

- 梯度， gradient = （ $\partial(L)/\partial(w)$, $\partial(L)/\partial(b)$ ）
- 更新 ， $w' = w -$ learning_rate * $\partial(L)/\partial(w)$ ， $b' = b -$ learning_rate * $\partial(L)/\partial(b)$
- 下降 ， L' = Loss_Function (w' , b')

梯度下降过程中需关注的两个问题

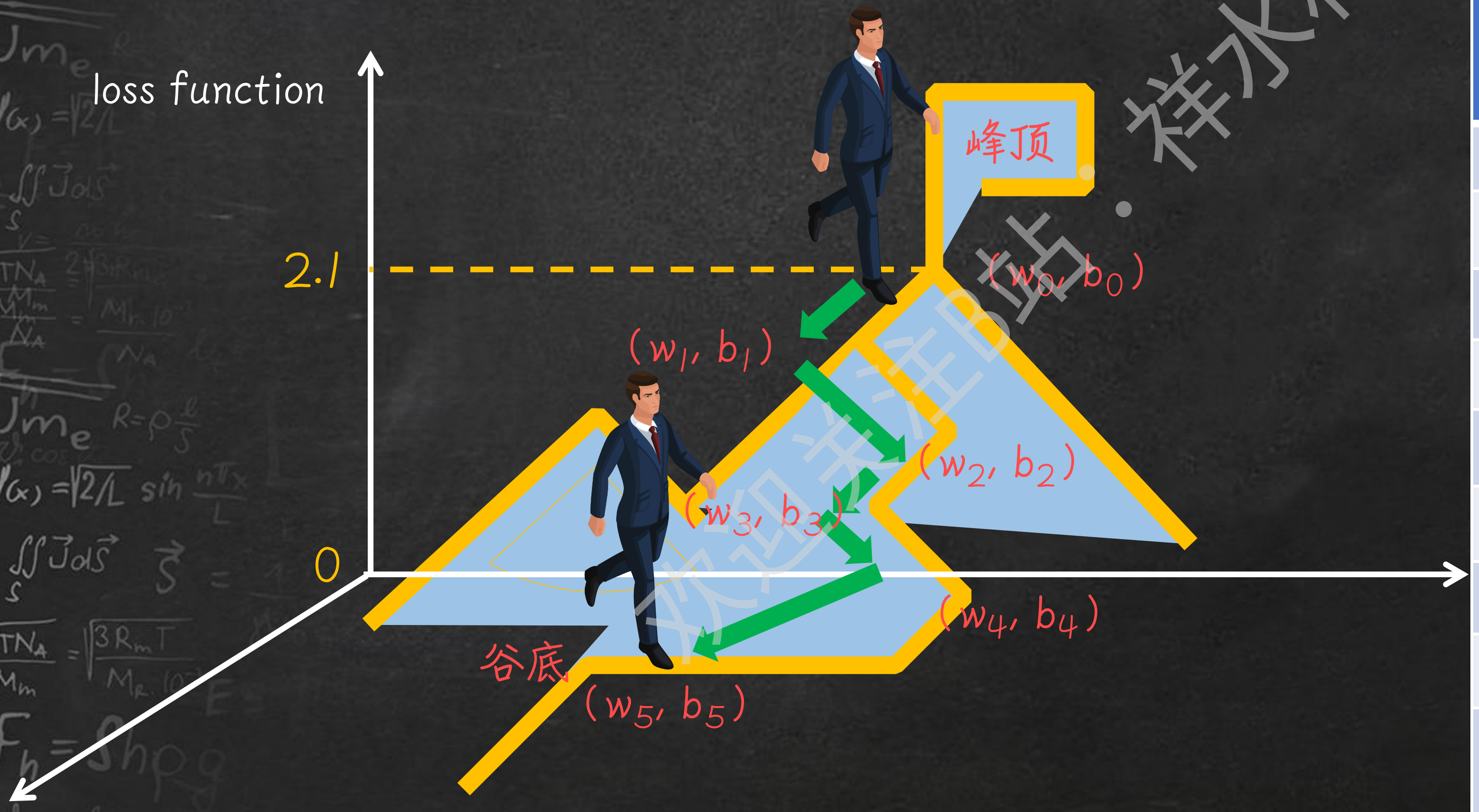
- 局部最优解 vs 全局最优解，梯度下降算法要充分考虑避免陷入局部最优解的情况
- 收敛速度，学习率（learning rate）是个重要参数，不宜太大、否则更新后的参数容易越过最优解；也不宜太小，否则参数更新很慢，使找到最优解的计算成本和时间成本增加

（示意图）

欢迎关注B站：淮水村东头

随机梯度下降过程（stochastic gradient descent,简称SGD）

- 以下图为例，每一次从训练集中抽取1个样本，然后计算模型损失值
- 求损失值关于参数的梯度，然后再回传并更新参数
- 再从训练集中抽取第2个样本，如此循环往复，直到找到最优解



模型参数更新轮次	模型参数	样本真实值预测值	loss function
初始	(w_0, b_0)	$x_0\ y_0\ y'_0$	2.1
1	(w_1, b_1)	$x_1\ y_1\ y'_1$	1.9
2	(w_2, b_2)	$x_2\ y_2\ y'_2$	1.7
3	(w_3, b_3)	$x_3\ y_3\ y'_3$	0.8
4	(w_4, b_4)	$x_4\ y_4\ y'_4$	0.5
5	(w_5, b_5)	$x_5\ y_5\ y'_5$	0.3
6	(w_6, b_6)	$x_6\ y_6\ y'_6$	1.4
7	(w_7, b_7)	$x_7\ y_7\ y'_7$	2.5
8	(w_8, b_8)	$x_8\ y_8\ y'_8$	3.3

欢迎关注B站：祥水村东头

真SGD、小批量SGD、批量SGD（p39）

- 真SGD：如上页所示，每一次求解梯度、更新参数的过程中，只抽取1个样本
- 小批量SGD：每一次求解梯度、更新参数的过程中，抽取部分样本
- 批量SGD：每一次求解梯度、更新参数的过程中，抽取全量样本

关于optimizer（p40）

- SGD还有很多种变体，称之为优化器（optimizer）
- 它们不仅计算本次参数更新，同时还会考虑上一次参数更新的情况
- 如 rmsprop（2.1节使用）、Adagrad等变体

```
network.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

关于反向传播 backward propagation（p41）

- 当模型层数不断叠加、每层节点数不断增加时，张量的运算会变得越来越复杂
- 假设模型中的每层就代表1种运算，那么包含3层的模型其运算公式为

$$f(w_1, w_2, w_3) = f_{layer\_1}(w_1, f_{layer\_2}(w_2, f_{layer\_3}(w_3)))$$

- 如果对上式求导，则需要用到“链式法则”：$f' = f'_{layer\_1} * f'_{layer\_2} * f'_{layer\_3}$
- 将“链式法则”应用于求解梯度、更新参数的过程中，则称为“反向传播”或“反式微分”

欢迎关注B站：祥水村东头

“人生之旅历途甚长，所争决不在一年半月，万不可因此着急失望，招精神上之萎靡”

——梁启超致梁思成家书

献给正在逐梦路上努力奔跑的你我他

感谢聆听

THANK YOU

欢迎关注B站：诺水村东头

本次胶片内容、及涉及相关代码均可移步至Github进行下载
感谢您的投币三连！

我的代码 Github 地址：
https://github.com/david-cal/Reading-Note-for-Chollet-of-Deep-Learning-with-Python

欢迎关注B站：祥水雨寿头