深度学习

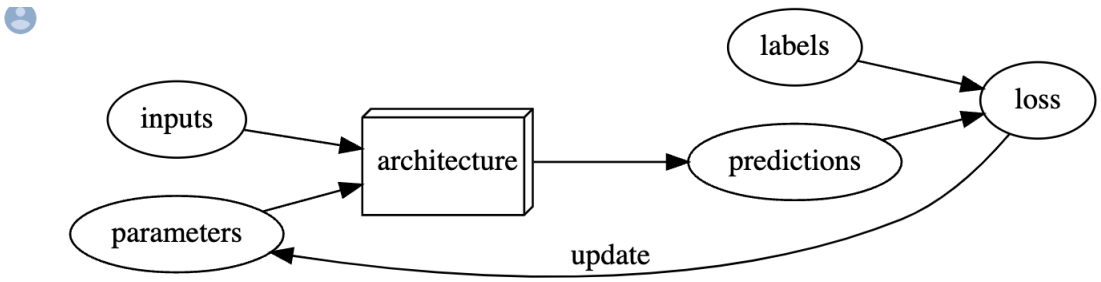
Created by	ⓨ yuan xiao
Created time	November 3, 2023 12:01 PM
Tags	Basic

学习来源

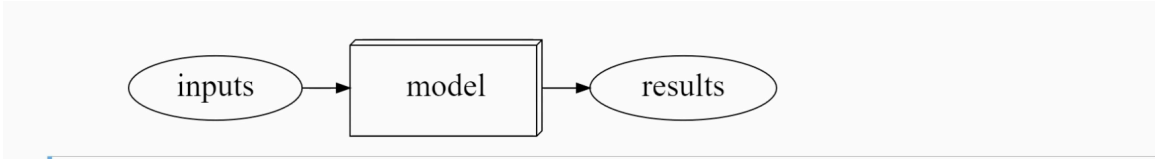
https://course.fast.ai/

▼ 深度学习概念

▼ 流程概述



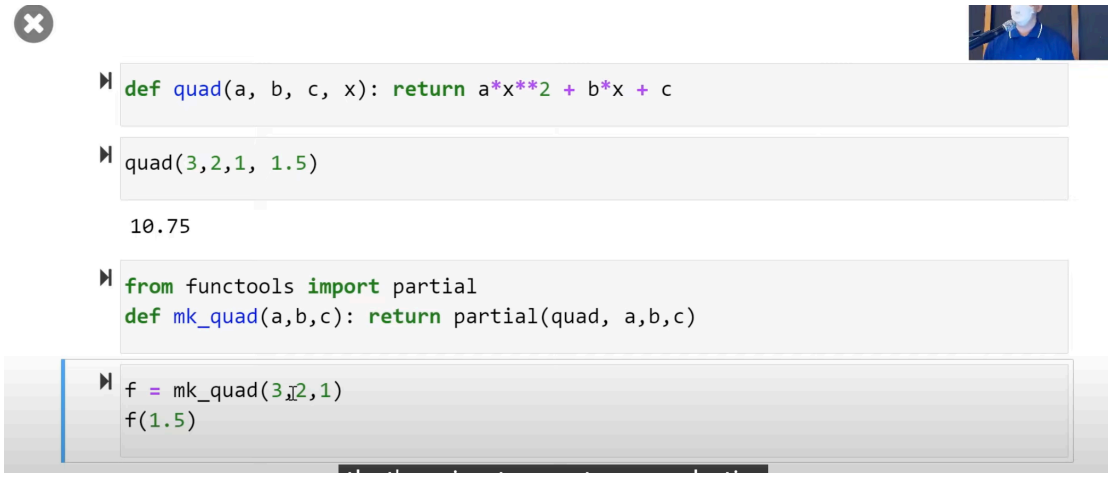
将input 的数据与parameters(数学方程式的系数)给到 architecture(数学方程式),得出一个 predictions,再去根据labels与predictions得出loss值(即学习结果如何),如不满意则调整parameters再来一遍.一遍流程则为一次epoch



当数据模型训练完后,则可以直接拿来使用,如图

▼ 知识节点

▼ architecture



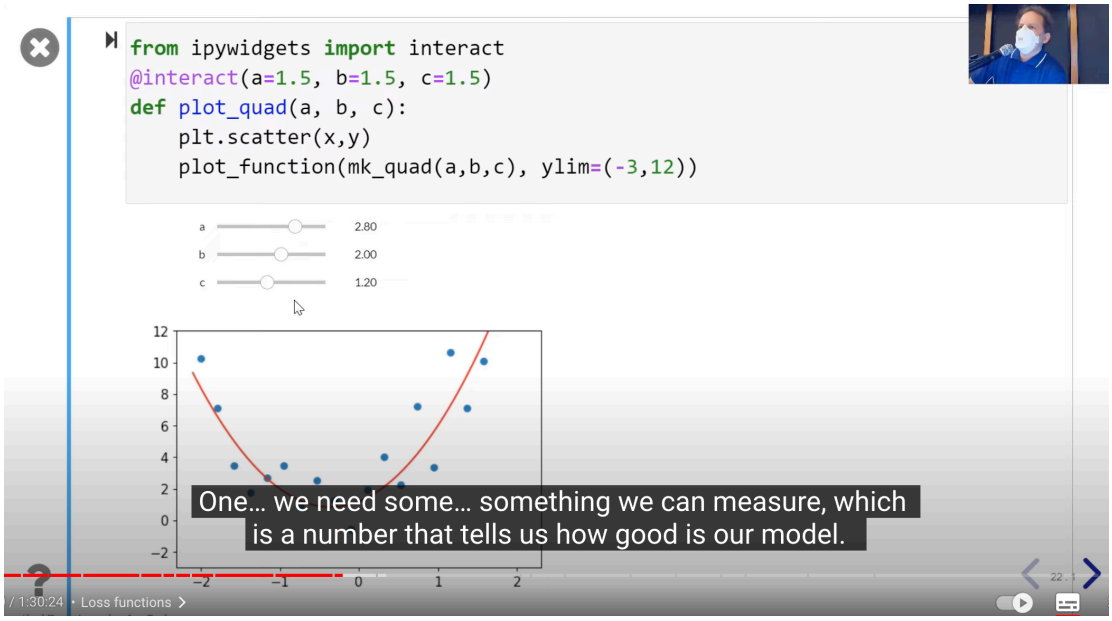
what: architecture即是数学方程式(譬如多重ReLU函数),如上图(图中是比较简单的样例)

why: 通过architecture,然后不断调整parameters来构建模型.

▼ parameters

what: parameters,即是architecture(数学方程式)的系数

why: 通过不断调整architecture的系数,使得得出最优解的结果与系数,而最优解的系数与architecture则成为最终的模型



如上图,是一个最简化版的方程式(architecture)调参示例

深度学习的过程,可理解为是一个不断调整parameters的过程,而调整parameters,即是调整这个方程式(architecture)的系数.

还需要注意的是调整parameters顺序,按顺序调整完每个系数后(如图则是a->b→c,即为一轮),再来一轮,调优效果会更好!!!

▼ loss

what: 用于评估architecture使用不同parameters计算出来结果的效果的指标方法

why:有了loss,则把每次调整parameters后的效果给指标化了,通过loss指标更易于判断每次parameters调整效果如何

In [11]:
```
def mae(preds, acts): return (torch.abs(preds-acts)).mean()
```

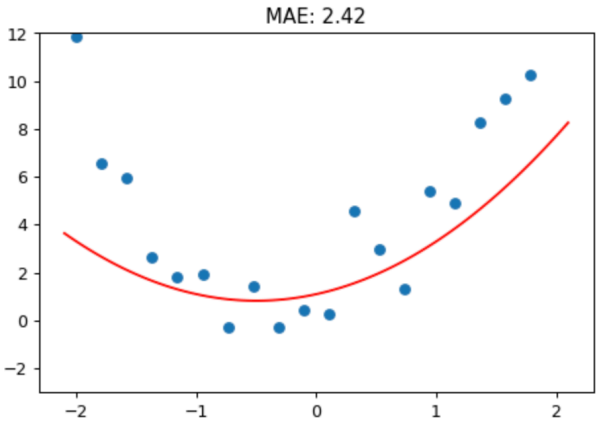
loss有很多种方案,如上图则是mae方案,通过计算preds(architecture与parameters计算出来的预测值)与实际值相减后的平均值,得出误差,mae值越小则误差越小,否则反之

In [12]:
```
@interact(a=1.1, b=1.1, c=1.1)
def plot_quad(a, b, c):
    f = mk_quad(a,b,c)
    plt.scatter(x,y)
    loss = mae(f(x), y)
    plot_function(f, ylim=(-3,12), title=f"MAE: {loss:.2f}")
```

a 1.10

b 1.10

c 1.10



有了loss后,只需要查看loss值,即可知道每次调整parameters后的效果如何

▼ gradient descent

what:梯度下降,用于优化parameter的一种方案

why: 用于优化parameter

how: 通过计算每个parameter在loss函数中的梯度值,即可知道每个parameter(系数)进一步调整后,loss的结果如何.例如,当parameter在loss函数中计算的梯度为负值,则表示进一步增加parameter则会使loss值降低

In [19]:
```
abc.grad
```

Out[19]:
```
tensor([-1.3529, -0.0316, -0.5000])
```

In [21]:
```
for i in range(10):
    loss = quad_mae(abc)
    loss.backward()
    with torch.no_grad(): abc -= abc.grad*0.01
    print(f'step={i}; loss={loss:.2f}')
```

如图,

1.abc为三个parameter,通过调用quad_mae方法获取该组parameter在architecture的loss值

2.调用pytorch的backward计算梯度值

3.根据梯度值来调整parameter值,来优化loss值.如果梯度为负数,则意味着需要增加parameters(loss值越低意味着误差越低),否则反之.

注意这里图片实例中,因为abc梯度值是负数,所以abc需要使用减号,减去梯度与0.01相乘的值,来达到增加系数的效果.当梯度值为整数时,这里符号也不需要变,所以称为梯度下降

梯度计算注意:在使用pytorch计算梯度时,当对tensor(pytorch的一个张量对象)使用requires_grad_方法时,会标注为该tensor会被标记用于梯度计算,后续每对该tensor进行计算操作时,pytorch都会建立新节点用于记录新旧值与一些信息,当最后调用pytorch的损失函数时,会将之前的这些计算操作的节点构建成

计算图,一旦调用损失函数返回的loss对象的backward时,就会对该tensor对象的grad属性写入梯度计算值(通过之前该tensor构建好的计算图计算得来)

▼ ReLU/Rectifier Linear Unit(线性整流函数)

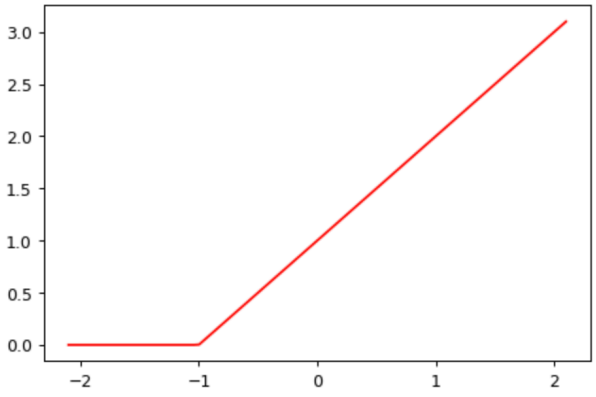
22]:
```
def rectified_linear(m,b,x):
    y = m*x+b
    return torch.clip(y, 0.)
```

what: 线性函数且固定y值最小值为0的函数,则是线ReLU.也可以理解为是architecture

why: 通过多重的ReLU函数的组合,可以组合出任意复杂的函数形式.然后通过不断调整parameters来构建出最合适的函数形式或称为模型.

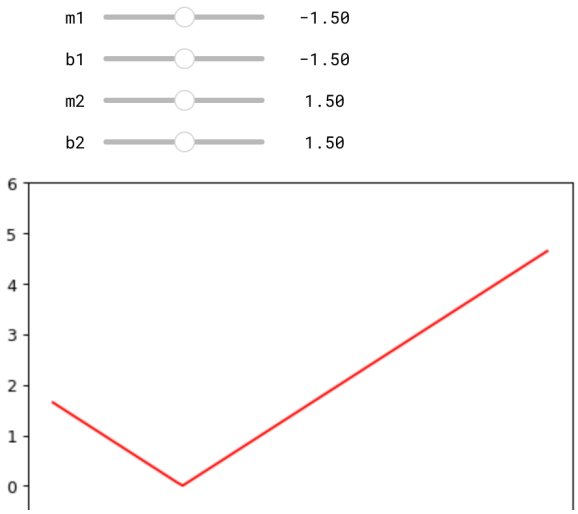
▼ 简单示例

[24]:
```
import torch.nn.functional as F
def rectified_linear2(m,b,x): return F.relu(m*x+b)
plot_function(partial(rectified_linear2, 1,1))
```



```
def double_relu(m1,b1,m2,b2,x):
    return rectified_linear(m1,b1,x) + rectified_linear(m2,b2,x)

@interact(m1=-1.5, b1=-1.5, m2=1.5, b2=1.5)
def plot_double_relu(m1, b1, m2, b2):
    plot_function(partial(double_relu, m1,b1,m2,b2), ylim=(-1,6))
```



如上两图,通过不断叠加ReLU函数以及调整parameters,可以生成出不同的函数图形或是函数式子,这意味着该函数形式可以匹出任意情况的输入与输出值

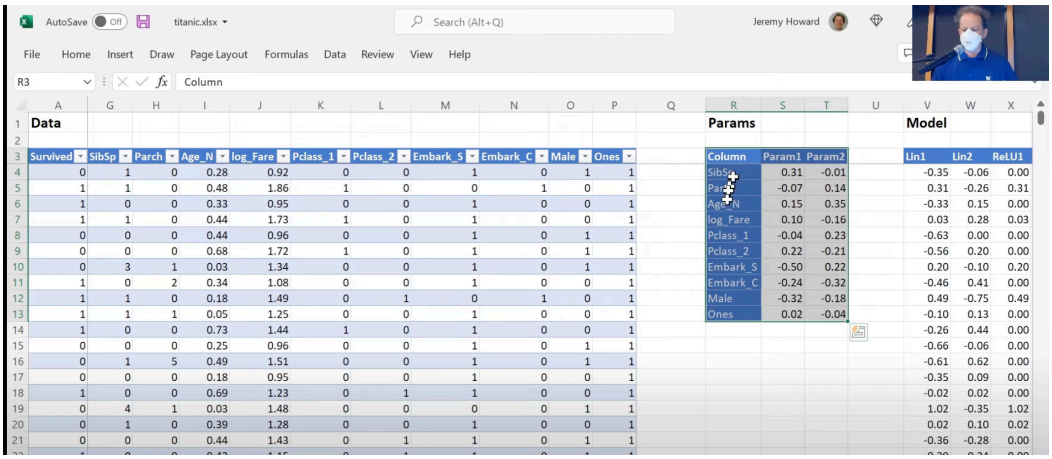
譬如,训练图片识别模型,可将要训练图片像素坐标值以及rgb值,作为x变量代入ReLU函数(多个x变量意味着多个ReLU函数)中,计算出该位置预测的rgb值,将该值与训练的图片对应像素坐标值的rgb值进行loss运算,根据loss值,再使用梯度下降进行parameters调整,即可不断优化这个模型.(自己的猜测与理解)

▼ 关于模型如何识别图像每个像素连续性的问题(自己的猜测与理解)

Q:训练与对比的是每个像素对应的rgb值,不会导致模型后续在识别时,当图片中物体位置不同时而识别不出来吗?

A:不会,在训练图片中的物体时,可以通过添加物体在不同位置的图片,来增强学习效果.这样在训练中,调整parameters时,由于两个图片中物体位置不同,会计算出较高的loss值,根据loss值过高,我们可以增加训练次数来达到训练的准确度

▼ 使用矩阵计算ReLU



如图

多重ReLU例子: (ax1+c)+(bx2+c)....

data侧的每一行为多重ReLU函数的x变量值,对应例子中的x1,x2等等

params侧的每一行为系数值,也就是parameters.对应上面例子的a,b.

data最后一列的Ones与params最后一行的const则对应例子中c的总和,算是一种简化技巧

data每一行用于执行一次多重ReLU组合函数,从而得出predic值

如此通过data的每一行乘以paramters的一列,便可使用矩阵乘法计算ReLU

当有多层parameters时,每多一列paramters,即为多一层神经网络的layer层,将每一层layer计算出的ReLU结果的和作为predict

▼ metrics

what: 度量模型好坏的指标,指标有多种,且每种指标的意义不同,譬如Pearson correlation coefficient指标是用于比较两个变量的关联程度(比如人民收入与房价中位数两个变量),越关联则值越高.

why: 用于测量模型好坏

补充:

loss与metrics是有区别的,loss由于能取得较平滑的函数图像(因为loss函数可以使用平方),便于梯度下降计算;而metrics的每个不同指标体现在函数图像上,可能是非常曲折的,不利于梯度计算.譬如当使用accurate用于梯度计算时,每轮训练的accurate值可能会上下波动,所以不利于梯度下降计算.

当某个指标不错时,并不意味着其他指标都会取得很好的效果,且当指标值不错时,并不意味着该模型落地后一定不错!!!

▼ learn rate

In [21]:
```
for i in range(10):
    loss = quad_mae(abc)
    loss.backward()
    with torch.no_grad(): abc -= abc.grad*0.01
    print(f'step={i}; loss={loss:.2f}')
```

what: 计算梯度下降时,每次与梯度相乘的比率,即上图中的0.01

why:合理的learn rate有助于加速模型训练;以及达到最佳效果,即当梯度为0时,loss值最低

▼ 名词解释

▼ 概览

```
[[dljargon]]
.Deep learning vocabulary
[options="header"]
|=====
| Term | Meaning
|Label | The data that we're trying to predict, such as "dog" or "cat"
|Architecture | The _template_ of the model that we're trying to fit; the actual mathematical function that we're passing the input d
|Model | The combination of the architecture with a particular set of parameters
|Parameters | The values in the model that change what task it can do, and are updated through model training
|Fit | Update the parameters of the model such that the predictions of the model using the input data match the target labels
|Train | A synonym for _fit_
|Pretrained model | A model that has already been trained, generally using a large dataset, and will be fine-tuned
|Fine-tune | Update a pretrained model for a different task
|Epoch | One complete pass through the input data
|Loss | A measure of how good the model is, chosen to drive training via SGD
|Metric | A measurement of how good the model is, using the validation set, chosen for human consumption
|Validation set | A set of data held out from training, used only for measuring how good the model is
|Training set | The data used for fitting the model; does not include any data from the validation set
|Overfitting | Training a model in such a way that it _remembers_ specific features of the input data, rather than generalizing well
|CNN | Convolutional neural network; a type of neural network that works particularly well for computer vision tasks
|=====
```

▼ overfit (过度拟合)

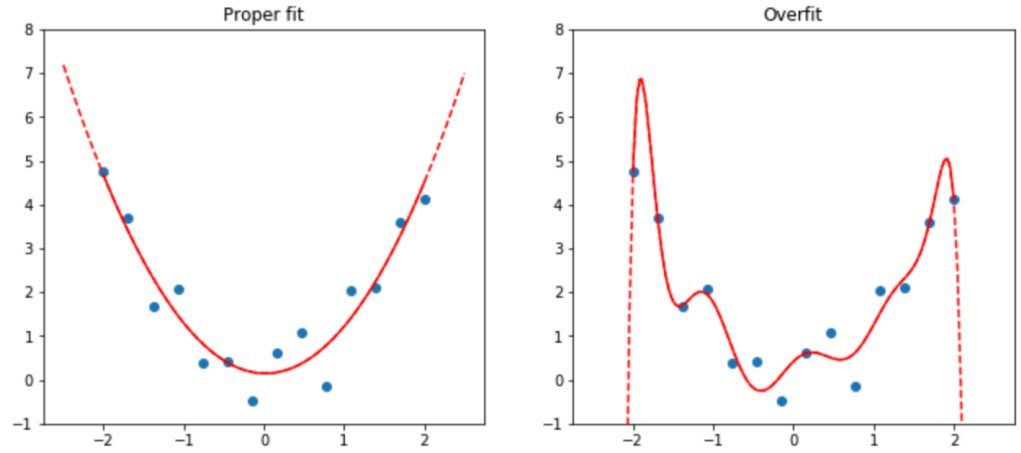
what: 模型在训练时记住了过多的特征值

why: 记住过多训练数据中的特征值,会使得训练后的模型只针对于这部分训练数据有效,不太具有展望性.

fastai will *always* show you your model's accuracy using *only* the validation set, *never* the training set. This is absolutely critical, because if you train a large enough model for a long enough time, it will eventually memorize the label of every item in your dataset! The result will not actually be a useful model, because what we care about is how well our model works on *previously unseen images*. That is always our goal when creating a model: for it to be useful on data that the model only sees in the future, after it has been trained.

Even when your model has not fully memorized all your data, earlier on in training it may have memorized certain parts of it. As a result, the longer you train for, the better your accuracy will get on the training set; the validation set accuracy will also improve for a while, but eventually it will start getting worse as the model starts to memorize the training set, rather than finding generalizable underlying patterns in the data. When this happens, we say that the model is *overfitting*.

<> shows what happens when you overfit, using a simplified example where we have just one parameter, and some randomly generated data based on the function `x**2`. As you can see, although the predictions in the overfit model are accurate for data near the observed data points, they are way off when outside of that range.



因为训练模型的最终目的是要作用在未来的图片(即不在当前训练集的图片),所以当训练时间过久,会出现模型逐渐记住过多训练数据中的特征值(也就是过度拟合),使得训练后的模型只针对于这部分训练数据有效,不太具有展望性.

补充:

- validation set与test set的数据,最好用训练时未使用过的数据.譬如validation set使用不同于training set的数据,test set的数据要不同于training set与validation set的数据.这样能尽可能使模型更具有展望性

▼ data argumentation(数据增强)

what: 通过将一张图像拆分成,多张由改变原始图片的灯光亮度,旋转方向等反差方式的图片来进行训练

why: 解决input的图片需要处理成相同大小给gput的问题,以及训练量的问题.

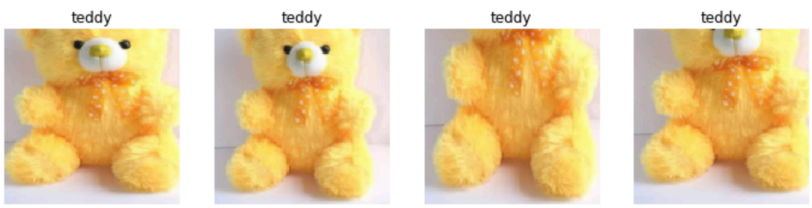
模型在学习的时候,一次epoch是需要将多组图像给到gpu,且必须要是相同大小形式,这就会带来input数据图像大小不同需要处理的问题.

如果只是将图片进行裁剪或挤压.则会导致一些特征值学习不到或是不正确.

如果只是将图片周边用黑边填充来保证图像宽高比正常,则会浪费大多计算在无用的黑边上.

因此引出了data arumentaion,由于未训练的neural network并不知道什么是图片,即便它旋转了或是裁剪等等,所以我们可以通过将一张图片中物体的不同位置,拆成成多张图片来进行训练也是一样的效果.

```
bears = bears.new(item_tfms=RandomResizedCrop(128, min_scale=0.3))
dls = bears.dataloaders(path)
dls.train.show_batch(max_n=4, nrows=1, unique=True)
```



▼ 新旧术语

- The functional form of the *model* is called its *architecture* (but be careful—sometimes people use *model* as a synonym of *architecture*, so this can get confusing).
- The *weights* are called *parameters*.
- The *predictions* are calculated from the *independent variable*, which is the *data* not including the *labels*.
- The *results* of the model are called *predictions*.
- The measure of *performance* is called the *loss*.
- The loss depends not only on the predictions, but also the correct *labels* (also known as *targets* or the *dependent variable*); e.g., "dog" or "cat."

一些深度学习术语由旧往新的转换,如图,如model→architecture;weights->parameters

▼ NLP

▼ Embedding

what: 动词角度,可理解为映射的意思.名词角度可理解为映射后的向量

why:用于将单词或字符向量化，便于后期文本分析等。譬如将每个input id映射到向量空间中，距离相近的代表两个字符意思相近，否则反之。

注意事项

- 通常使用pretrained来减少训练时间,我们只需喂一些我们的数据给它即可.它是通过修改pretrained model最后一层layer来适应我们想要的结果,这个过程

叫fine-tune(微调)

- 注意电脑内存的使用量,当内存使用过高时会进行内存交换,导致一部分程序数据会存在硬盘中,是的gpu无法读取数据

▼ Tokenization与Numericalization

what: 将文本拆分成词和词组,再将词和词组数字化

why:因为深度学习的矩阵计算时,只能数字计算,所以需要将文本转化为数字(input id)

注意:在使用nlp的预训练模型时,对应模型会有对应的词库,用于针对该模型输入的字符进行数字化.

▼ sequence length

what: 描述一个raw text经过tokenizer后,token加上padding等的数量(可以简单理解tokenizer后token的数量)

why: 这个数值代表一个模型,能处理的最大token数量,超过将会被截断

▼ 补充

▼ Transformer解释

Transformer是神经网络的一种结构或是架构,还有其他架构譬如CNN(卷积神经网络)等,Transformer架构目前用于nlp领域很厉害,但不意味着不能用于其他领域(计算机视觉等)(Transformers则是hugging face提供了创建和使用共享模型的功能的库,但只限于Transformer架构的模型)

▼ 快速部署

可使用gradio(一个快速生成交互的python库)部署到hugging-face的space上或是自己搭建的python后台服务上

▼ YOLO

YOLO是一个框架,基于特定的**基础网络(backbone)**,如 ResNet (一种神经网络架构或模型的一部分),其预先基于大型图像分类数据集进行预训练,得到了**获取图像特征能力**.YOLO提供能力,能够**加载预训练模型**和**新的数据集**来进行**训练和微调**,生成最终的任务模型