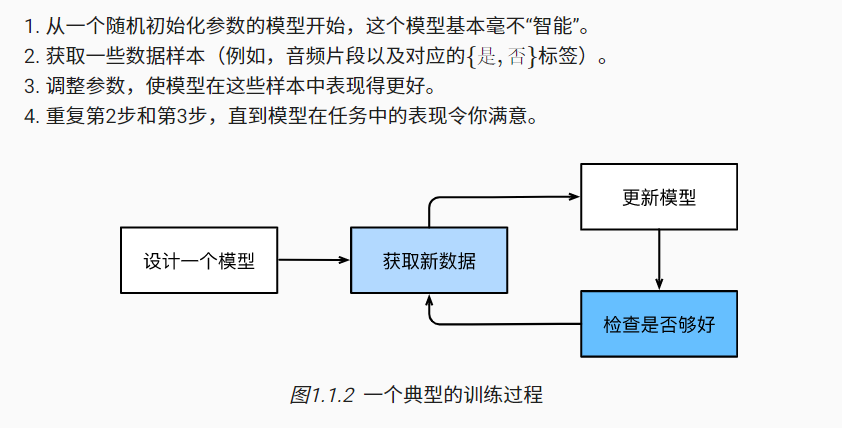
通常，即使我们不知道如何明确地告诉计算机如何从输入映射到输出，即使你不知道如何编写计算机程序来识别“Alexa”这个词，你的大脑自己也能够识别它。

那么到底什么是参数呢？ 你可以把参数看作是旋钮，我们可以转动旋钮来调整程序的行为。 任一调整参数后的程序，我们称为模型（model）。 通过操作参数而生成的所有不同程序（输入-输出映射）的集合称为“模型族”。 使用数据集来选择参数的元程序被称为学习算法（learning algorithm）。

正如你可能猜到的，如果我们只是随机设置所有按钮（模型参数），我们的模型不太可能识别出“Alexa”、“Hey Siri”或任何其他单词。 在机器学习中，学习（learning）是一个训练模型的过程。

总而言之，我们没有编写唤醒词识别器，而是编写了一个“学习”程序

仅仅拥有海量的数据是不够的，我们还需要正确的数据。 如果数据中充满了错误，或者如果数据的特征不能预测任务目标，那么模型很可能无效。 有一句古语很好地反映了这个现象：“输入的是垃圾，输出的也是垃圾。”（“Garbage in, garbage out.”） 此外，糟糕的预测性能甚至会加倍放大事态的严重性。

当任务在试图预测数值时，最常见的损失函数是平方误差（squared error），

当一个模型在训练集上表现良好，但不能推广到测试集时，我们说这个模型是“过拟合”（overfitting）的。 就像在现实生活中，尽管模拟考试考得很好，真正的考试不一定百发百中。

### 优化算法大多流行的优化算法通常基于一种基本方法–梯度下降（gradient descent）

#### 回归任何有关“多少”的问题很可能就是回归问题。比如：

这个手术需要多少小时？

在未来六小时，这个镇会有多少降雨量？

 假设你在市场上寻找新房子，你可能需要估计一栋房子的公平市场价值。 销售价格，即标签，是一个数值。 当标签取任意数值时，我们称之为回归问题。 我们的目标是生成一个模型，它的预测非常接近实际标签值。

#### 分类 在分类问题中，我们希望模型能够预测样本属于哪个类别（category，正式称为类

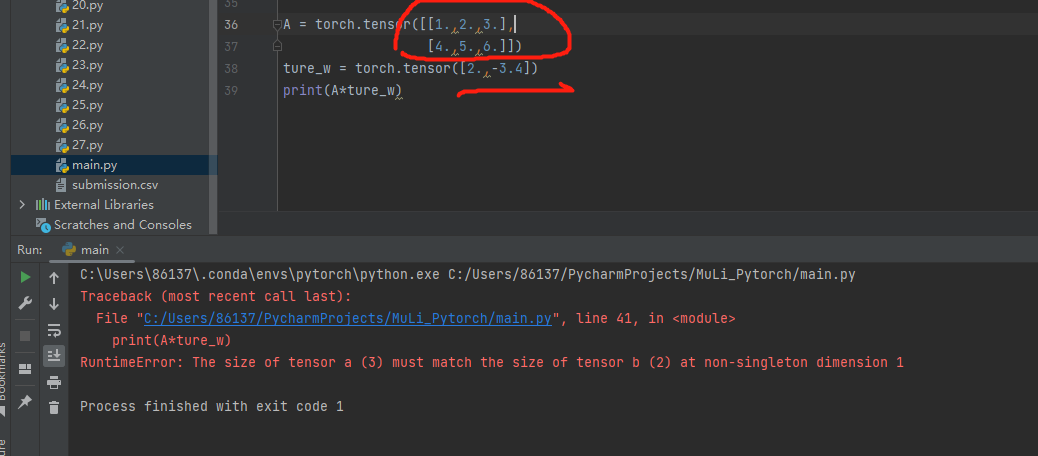
例如，对于手写数字，我们可能有10类，分别数字0到9。 最简单的分类问题是只有两类，我们称之为“二元分类”。 例如，数据集可能由动物图像组成，标签可能是猫狗{猫,狗}两类。 在回归中，我们训练一个回归函数来输出一个数值； 而在分类中，我们训练一个分类器，它的输出即为预测的类别。

然而模型怎么判断得出这种“是”或“不是”的硬分类预测呢？ 我们可以试着用概率语言来理解模型。

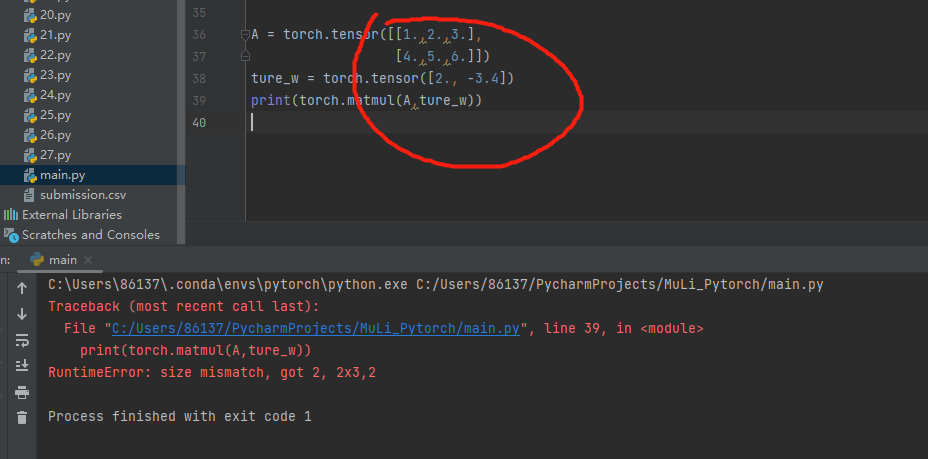
分类问题的常见损失函数被称为交叉熵（cross-entropy）

y = torch.matmul(X, w) + b # 生成X对应的预测值y，.matmul中禁止矩阵与标量的乘法。

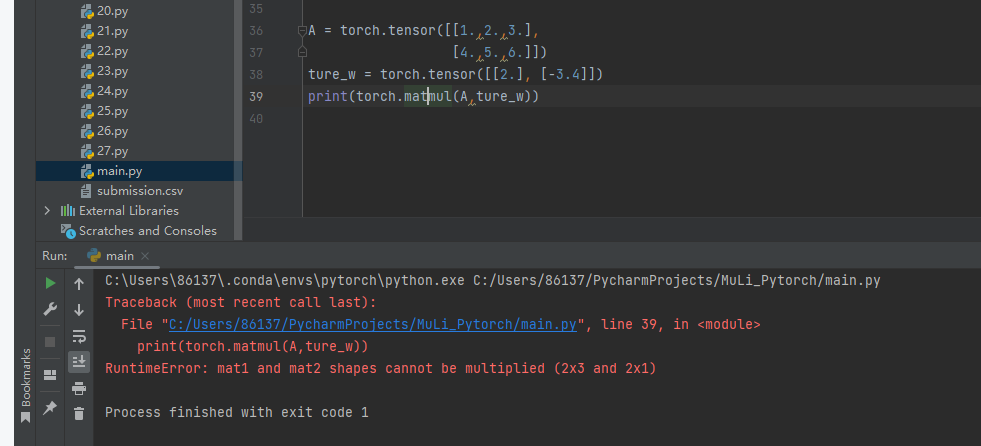
行数必须要相同，否则无法相乘



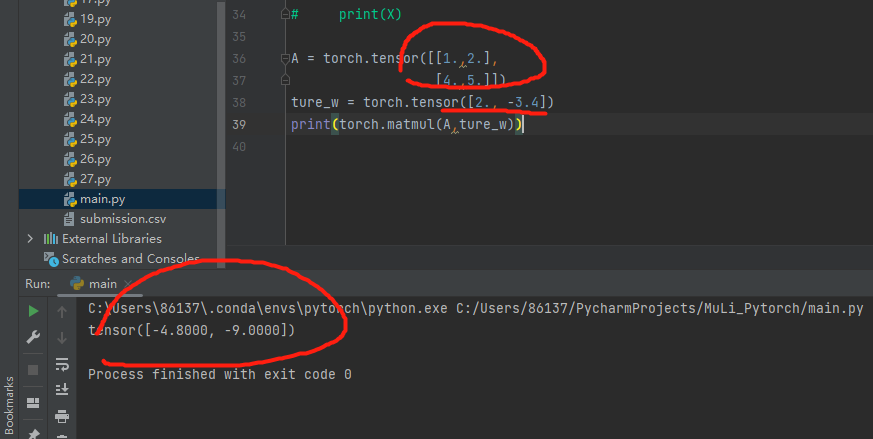
列数不同无法相乘，但是行数相同列数不同也无法相乘



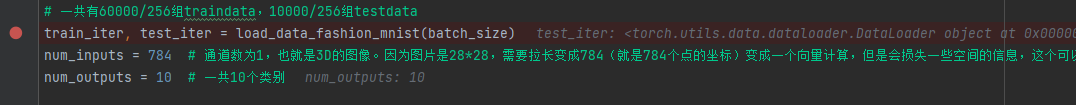
行数相同列数不同也无法相乘



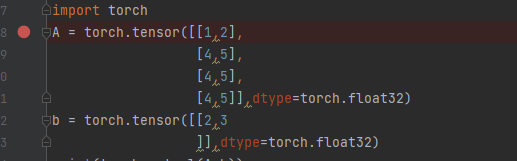
只有列数相同的时候才能matmul（更新：前面说法也不对，主要是要按照乘法规则和广播机制来算），# matmul是每一行元素一一对应相乘后相加，#单纯的乘法是一一对应相乘即可



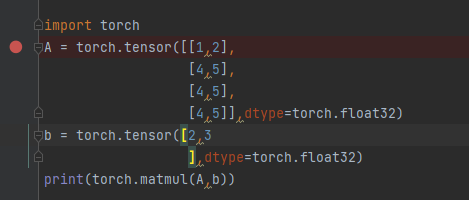
为什么要输入的图片28\*28展平呢？因为728行10列。这样就是每一列图片的特征是728个，x的形状是n\*d（其中特征维度（输入数量）为d，批量大小为n。）这样一次性可以处理256张特征数为728的图片



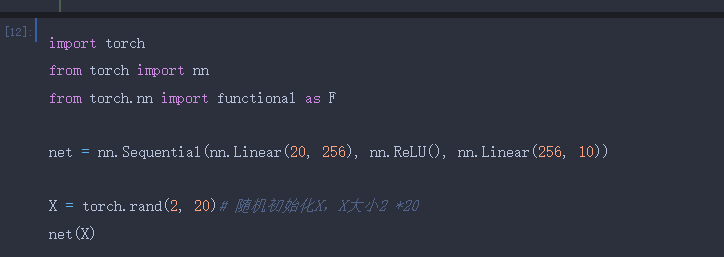
这个b是一行两列（因为是二维的）



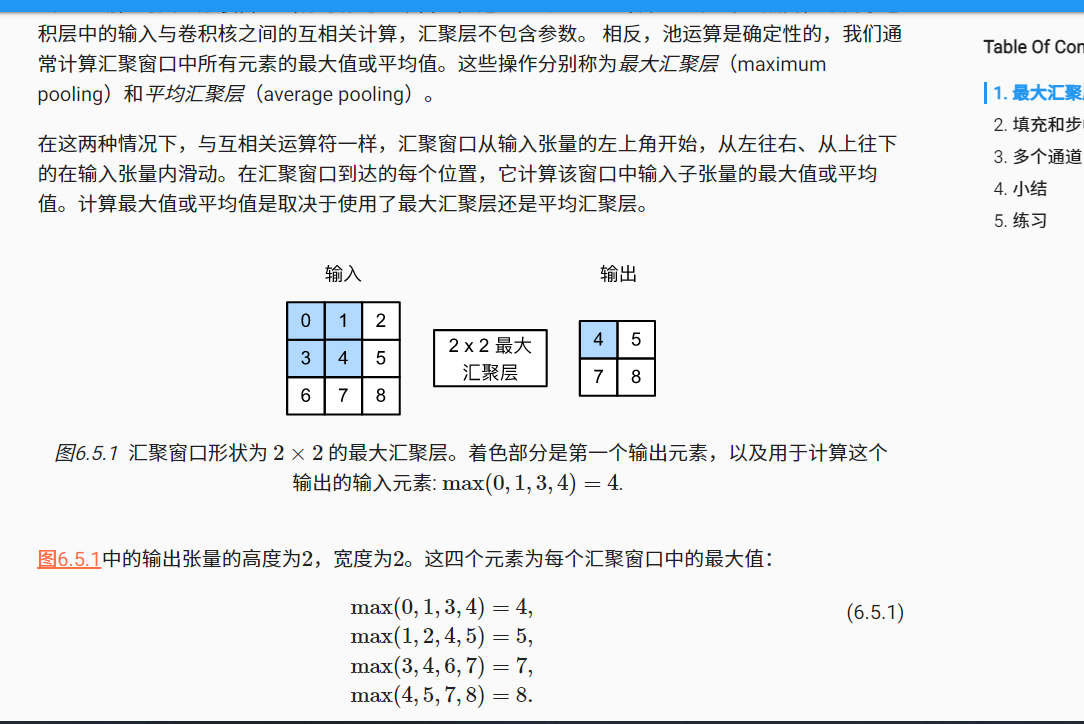
而这个b是2，就单纯的2【因为是一维的】（有两个数字）



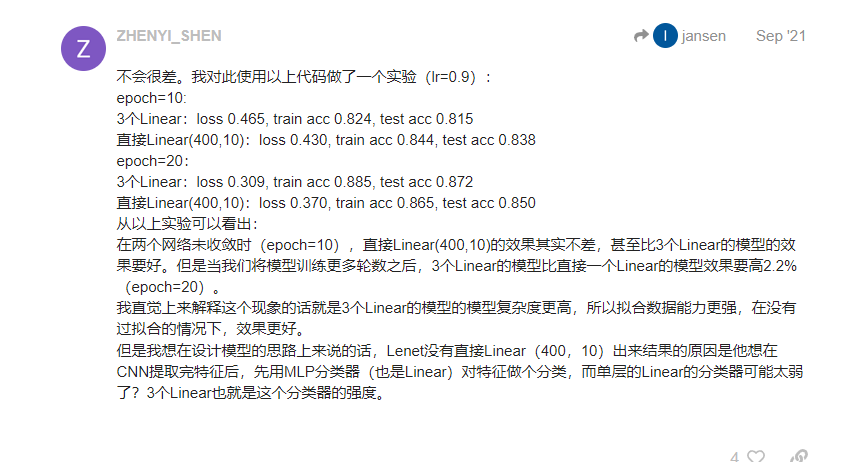
我们一直在通过net(X)调用我们的模型来获得模型的输出。 这实际上是net.\_\_call\_\_(X)的简写。 这个前向传播函数非常简单： 它将列表中的每个块连接在一起，将每个块的输出作为下一个块的输入。



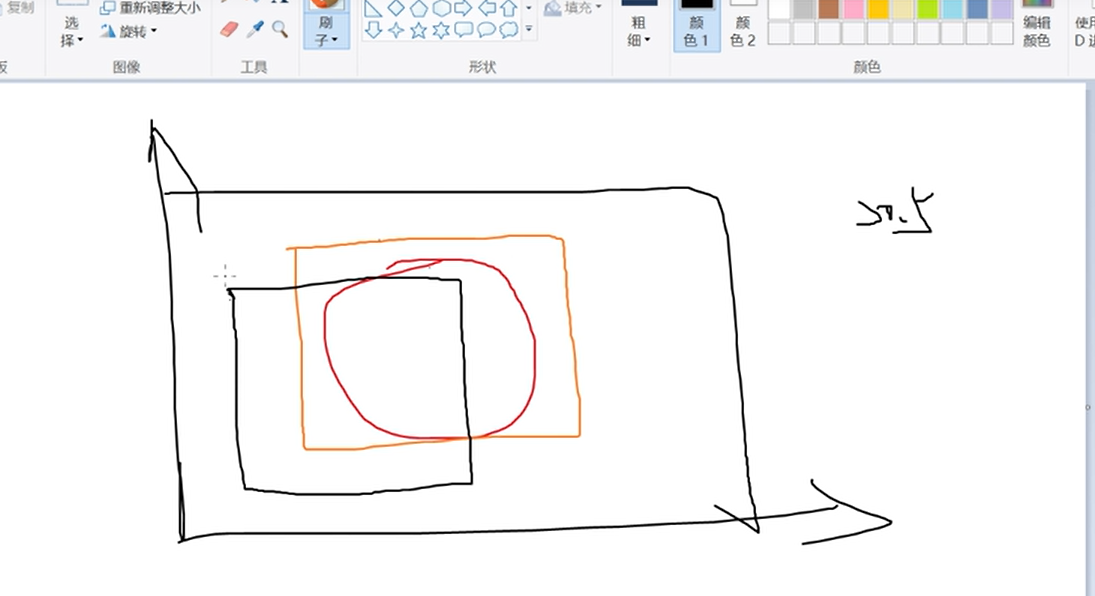
## 最大汇聚层和平均汇聚层[¶](https://zh-v2.d2l.ai/chapter_convolutional-neural-networks/pooling.html" \l "id2" \o "Permalink to this headline)



为什么模型，在展开之后，用了三个Linear层，有什么说法吗，  
如果直接从 Linear(400,10)，效果会很差吗？

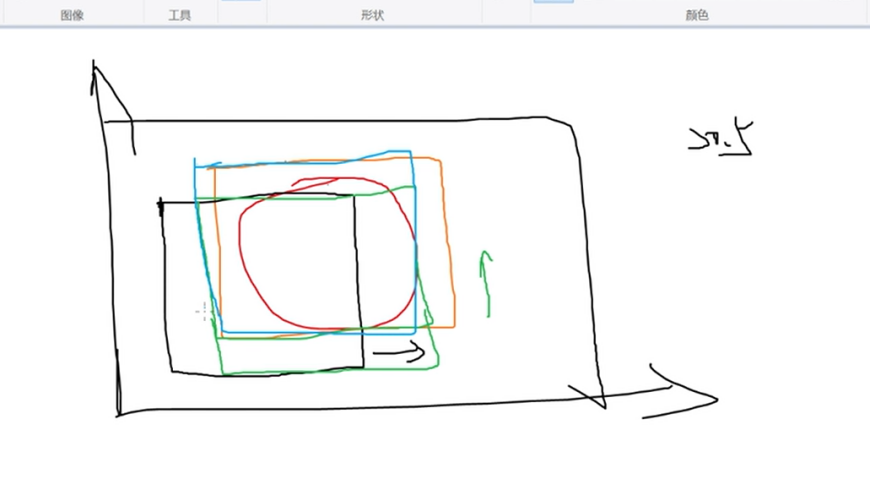


这个不叫一行24列，而是二维的，有24个元素，1只是补充了维度

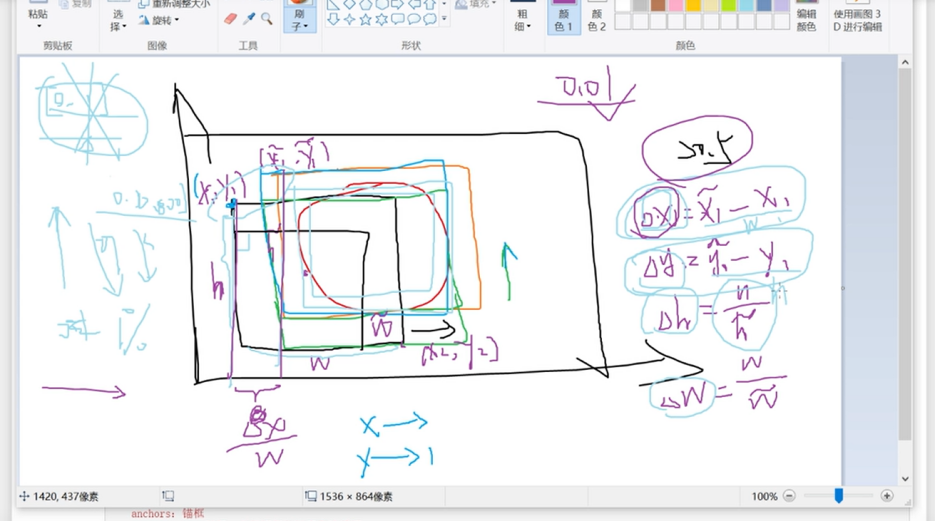


直观上锚框是大于threshold=0.5的，所以它把真实边界框分配给了锚框

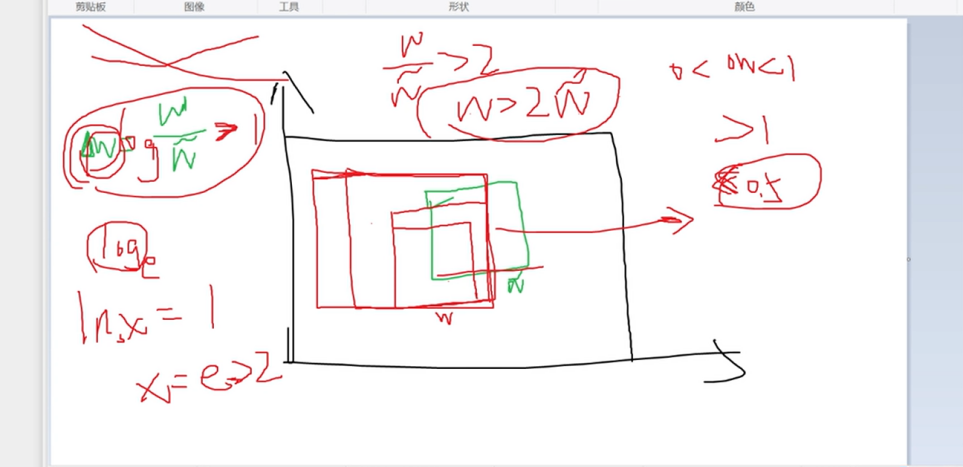
然后接下来就是做偏移了



这里老师话了5分钟讲为什么需要除以w。因为是梯度的问题，自己想想，太大太小都会怎样？？?还有不除以w的话真的会很大嘛，还是说在优化更好的策略呢？



还有w的占比为什么要log？ 也很讲究，加了log后占比很难达到1，又跟梯度有关

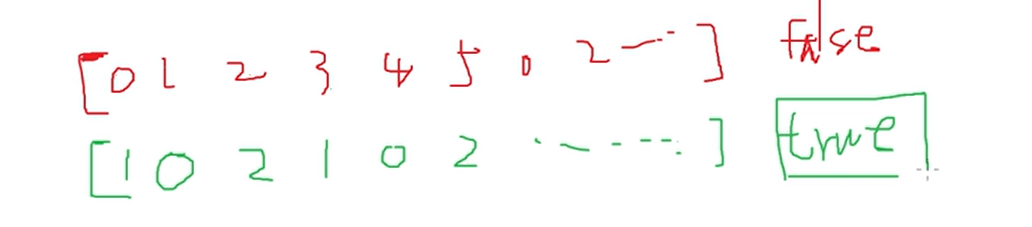


Eps的作用这么大》？/???



负类的图片就没不要反向传播了。减少运算

Cls\_anchors\_false和cls\_anchors\_true的区别，一个是真是边界框（含有类别的）一个是真实类别



布尔转换成0，1的方法

