信息论

课程介绍

- 信息量
- 熵
- 相对熵/KL散度(重点)
- 交叉熵(重点)

一、信息量

1.1 信息量的引出

一个信息是否有信息量要看这个信息到底是你知道还是不知道，关键是给你带来多少的确定性。(也可以简单的理解为就是惊喜程度)

分析以下两个例子给我们带来的信息量。

例1:比如说我是一个男生，没有任何信息。因为这是确定的。

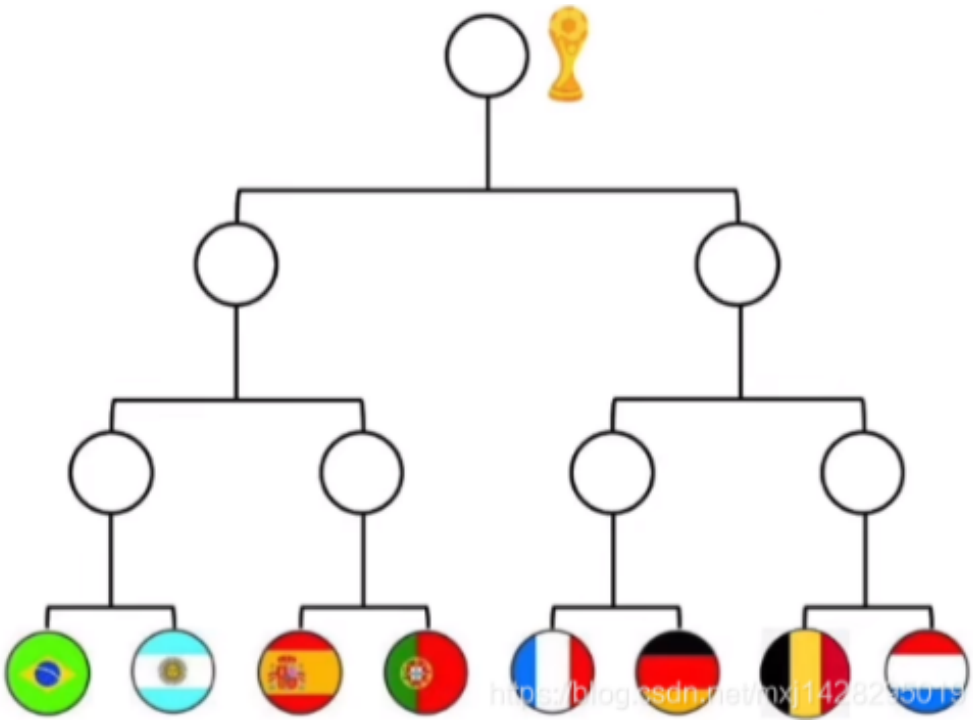
例2: 如果我们班上一个同学，买彩票中大奖啦。

> 对于班上其他同学而言，这个消息可能是一个惊喜，也可能是一个令人震惊的消息。因为中大奖的概率较低，大多数人可能没有预期到这样的事件会发生。

再比如: 以下是八个球队获得冠军的概率都是1/8，阿根廷进入决赛和阿根廷获得冠军的所得到的信息量是不一样的，一个是从1/8到1/2，一个是1/8到1，很显然阿根廷获得冠军这个带来的信息量更大

- 如果你什么消息都没有听说，有人问你阿根廷夺冠没有啊，你回答说不知道。
- 随后，你看到一个消息，说阿根廷已经进决赛了，这个时候再问你阿根廷夺冠没有啊，你还是说不知道

这两个不知道带来的信息一样吗?



也就是说，信息量它应该是，某个消息对“某个事件的确定程度改变了多少”进行的衡量。而确定性改变了多少，其实也就是前面说的那个概率的改变，阿根廷夺冠从原来的1/8变成了1/2。

那如何定量的描述信息量呢？

1.2 信息大小的衡量

信息量

信息论的基本想法是一个不太可能的事件居然发生了，要比一个非常可能的事件发生，能提供更多的信息。消息说：“今天早上太阳升起” 信息量是如此之少以至于没有必要发送，但一条消息说：“今天早上有日食” 信息量就很丰富。

我们通过这种基本想法来量化信息。

• 非常可能发生的事件信息量要比较少，并且极端情况下，确保能够发生的事件应该没有信息量。

• 较不可能发生的事件具有更高的信息量。

• 独立事件应具有增量的信息。例如，投掷的硬币两次正面朝上传递的信息量，应该是投掷一次硬币正面朝上的信息量的两倍。

为了满足上述三个性质，我们定义一个事件 x 信息量（self-information）为

$$I(x) = -\log P(x).$$

可以简单理解为: 一个事件由原来的不确定变得确定，它的难度越大，信息量也就越大。(与概率成反比)

二、熵

2.1 熵的概念

在科普内容里面，都是说**熵是对一个系统的混乱程度的度量。**

系统的混乱程度到底是什么意思？为什么用熵可以去描述系统的混乱程度？

以下两张图分别表示两个系统:

系统1: 表示A、B 两个摔跤选手都是70公斤选手，各自获胜的概率均是1/2

系统2: 表示C、D两个选手分别是80公斤 和 30公斤选手，获取概率分别是0.99 和 0.01

从这两个系统上我们其实可以看出，系统1、系统2进行比较，当前使用不确定程度描述两场比赛，这个不确定程度就是我们通常说的混乱

程度，这个混乱程度主要是结果不确定，所以很混乱。

系统2不出意外的话，肯定是男方获胜，也就是最后的结果确定性更高。系统1谁赢都有可能，所以最后结果是什么就很不确定。

这里我是用不确定的程度来描述两场比赛，这个不确定的程度也就是我们日常说的混乱程度，系统1因为结果特别不确定，所以很混乱。

通过上面的案例既然和概率以及不确定性有关，前面讲的信息量可以排上用场。

系统1: p(A) = p(B) = 1/2

信息量: 均为 I(A) = I(B) = -log(1/2) = 0.69

系统2: p(C) = 0.99 p(D) = 0.01

信息量分别为: I(C) = -log(0.99) = 0.01 I(D) = - log(0.01) = 4.6

从直观上看出，系统1 所包含的信息少于系统2，因为系统1的信息量加起来才是0.69 * 2，没有系统2的女方获胜的信息量(4.6)大，

如果用信息量来表示熵，是不是就会有问题。明明系统1更不确定，但是计算出来却是系统1的信息量更少。

那怎么办? 一个系统整体的信息量，还需要看事件对整个系统的贡献多少信息量。

熵就是所有事件对应的信息量的加权和。就是整个系统里面信息量的期望值。

2.3 熵的定义

信息是处理单个的输出。如果我们需要量化对整个系统中概率分布中的不确定性总量进行量化。

熵是一个系统里信息量的期望值

$$H(X) = -\sum_i p_i log p_i$$

一个随机事件发生的概率越小，则发生后的信息量就越大；

一个随机事件的发生概率越大，则发生后的信息量也越小。

X是随机变量，而这个随机变量有各种可能的取值，比如x1, x2...等等，pi是当X = x1时候的概率。注意前面的负号，因为概率是小

于1的值，所以求log之后就成了负数，为了保证信息熵为正，在前面加上了负号，负负得正。

备注: 熵其实并不神秘，和长度、体积、温度等一样都是用来做衡量使用。

2.4 案例

需求：计算信息熵（x1,x2,x3 已知概率分布）

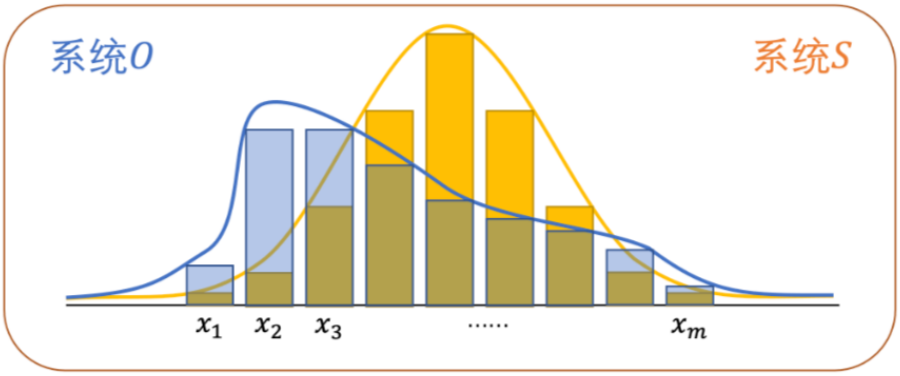
```
import numpy as np
# 计算信息熵，已知概率分布
def get_ent(x):
    ent = 0.0
    for i in x:
        # log为自然对数(底数为e：2.7)
        ent -= i * np.log(i)
    return ent
# 第一个信号的概率0.4 ....
x1 = np.array([0.4, 0.4, 0.2])                  # 概率分布
x2 = np.array([0.2, 0.2, 0.2, 0.2, 0.2])        # 概率分布
x3 = np.array([1])                              # 概率分布
print(get_ent(x1)), print(get_ent(x2)), print(get_ent(x3))
```

三、相对熵(KL散度)

3.1 相对熵的概念

用于度量两个概率分布间的差异性信息(掌握)

下图表示的是两个系统的概率分布，其中系统S代表的是真实的规律，系统O代表的是机器学习模型里面猜测的那个规律。



这两个不同的概率分布怎么去描述他们的差异。前面我们学习了熵，如果两个系统的概率分布相同，那系统的熵一定是相等的， 如果这

两个系统越接近，熵应该是越接近。

但是反过来，如果两个系统的熵相同。两个系统的概率分布一定相同吗？并不一定，比如说 我花了1万块钱学习，1万块旅游。虽然价格

一样，但是差别很大。

系统A:

```
P(X1) = 0.3
P(X2) = 0.7
```

系统B:

```
P(X1) = 0.7
P(X2) = 0.3
```

```
print(-(0.4 * np.log(0.4) + 0.3 * np.log(0.3)))
print(-(0.3 * np.log(0.3) + 0.4 * np.log(0.4)))
```

熵相同但是概率分布不同

不能直接比较系统的熵。怎么办? 使用KL散度, 每一个事件xi对应的信息量 ， 都拿来做了比较。如果每一个信息量相同，那么两个概率分布相同。

事件x_i在S系统中发生的概率

$$D_{KL}(S \parallel O) := \sum_{i=1}^{m} P_S(x_i) \cdot \left(I(P_O(x_i)) - I(P_S(x_i)) \right)$$

事件x_i在两个系统中信息量的差值

求和的是两个系统中同一个事件的信息量的差值，权重Ps(xi)是其中一个系统里这个事件的概率值

KL散度会在两个系统中挑选一个**作为基准(这里使用的是S系统作为基准)**， 另一个系统与这个基准进行比较

$$D_{KL}(S \parallel O) := \sum_{i=1}^{m} P_S(x_i) \cdot \left(I(P_O(x_i)) - I(P_S(x_i))\right)$$

$$= \sum_{i=1}^{m} P_S(x_i) \cdot ((-\log_2 P_O(x_i)) - (-\log_2 P_S(x_i)))$$

$$= \boxed{\sum_{i=1}^{m} P_S(x_i) \cdot (-\log_2 P_O(x_i))} - \sum_{i=1}^{m} P_S(x_i) \cdot (-\log_2 P_S(x_i))$$

这就是交叉熵　　系统S的熵

通过变形之后我们能够发现，KL散度可以被分成两部分，其中后面那个部分计算出来的就是系统S的熵，这部分与系统O无关。

系统S的熵已经确定的。真正决定KL散度其实是前面那部分，系统O的大小决定着KL散度的大小。

这部分内容被称为交叉熵

3.2 交叉熵

通过前面的描述 我们在比较两个系统概率分布的距离。不需要去计算KL散度，只需要计算交叉熵即可。故模型预测出来的概率分布O, 与

真实的概率分布S接近。如果使用KL散度描述的话，KL散度要接近数值0。

使用交叉熵描述怎么描述?

- 当交叉熵的值大于系统S的熵时，我们的目标是寻找交叉熵最小的值
- 当交叉熵的值小于系统S的熵时，我们的目标是寻找交叉熵最大的值

经过牛人的证明, 交叉熵的值一定是会大于等于系统S的熵的, (这里的证明省略) 因此只需要求出交叉熵最小值即可。

$$H(p,q) = \sum_{x} p(x) \cdot log\left(\frac{1}{q(x)}\right)$$

```python
import numpy as np
def cross_entropy(x, y):
    x = np.float_(x)
    y = np.float_(y)
    return np.sum(x * np.log(1 / y))

x1 = [(1, 1, 0), (0.8, 0.9, 0.1)]
x2 = [(0, 1, 1), (0.9, 0.2, 0.2)]
print(cross_entropy(x1[0], x1[1]))
print(cross_entropy(x2[0], x2[1]))
```

四、损失函数

4.1、均方差损失函数（万能损失函数，可以做回归和分类）

```
# numpy 实现
loss = np.mean((np.array([1, 2]) - np.array([3, 4])) ** 2)
print(loss.item())

# pytorch 实现
loss = torch.mean((tensor([1., 2.]) - tensor([3., 4.])) ** 2)
print(loss.item())

# pytorch实现
loss = torch.nn.MSELoss()(tensor([1., 2.]), tensor([3., 4.]))
print(loss.item())
```

4.2、交叉熵损失函数（更适合做分类问题）

```
# pytorch 实现
entroy = torch.nn.CrossEntropyLoss()
loss = entroy(tensor([1., 2.]), tensor([3., 4.]))
print(loss)
```

五、案例

5.1 案例1:

例: 假设某个字符发射器随机发出0和1两种字符，且其真实发出概率分布为A, 现在有两人的观察概率分布B与C, 各个分布如下:

$$A(0) = 1/2,\ A(1) = 1/2$$

$$B(0) = 1/4,\ B(1) = 3/4$$

$$C(0) = 1/8,\ C(1) = 7/8$$

B和C哪个更接近实际分布A

引入相对熵

$$\mathrm{KL}(P\|Q) = -\sum_{x\in X} P(x)\log\frac{1}{P(x)} + \sum_{x\in X} P(x)\log\frac{1}{Q(x)} = \sum_{x\in X} P(x)\log\frac{P(x)}{Q(x)}$$

计算得:

$$\mathrm{KL}\left(A\|B\right) = 1/2\log(\frac{1/2}{1/4}) + 1/2\log(\frac{1/2}{3/4}) = 1/2\log(4/3)$$

$$\mathrm{KL}(A\|C) = 1/2\log(\frac{1/2}{1/8}) + 1/2\log(\frac{1/2}{7/8}) = 1/2\log(16/7)$$

实际上B比C更接近实际分布A(因为与A分布的相对熵的较小)

5.2 案例2:

用于度量两个概率分布间的差异性信息，语言模型的性能通常用交叉熵来衡量。是神经网络(机器学习)中的损失函数。

说人话就是用于度量对同一个随机变量的非真实分布(预测分布)与真实分布之间的差距。

假设现在有一个样本集, 两个概率分布p、q，其中p为真实分布，q为非真实分布。

H(p)是信息熵

$$H(p)=\sum_i p(i)\cdot log\left(\frac{1}{p(i)}\right)$$

此时就将H(p,q)称之为交叉熵，交叉熵的计算方式如下：

$$H(p,q)=\sum_x p(x)\cdot log\left(\frac{1}{q(x)}\right)$$

例:

姓名	预测选中的概率
球球	0.8
冷檬	0.1
小玲	0.1

姓名	真实选中的概率
球球	1
冷檬	0
小玲	0

为什么我们不使用前面的KL散度来衡量这种差异性勒?

分析:
带入KL散度公式

H(P) = 1 * log(1) +0*log(0)* + *0*log(0) = 0 (注意这里实际会加上一个很小的值不是是0)

KL散度 = H(p, q) - H(p) = H(p, q) - 0 = 交叉熵

预测越准确，交叉熵越小

交叉熵只跟真实标签的预测值有关系。

交叉熵取值范围?

六、交叉熵与均方误差总结

交叉熵损失的计算公式如下：

$$H(y,\hat{y})=-\sum_{i=1}^{K} y_i\cdot\log(\hat{y}_i)$$

其中，y 是真实的概率分布，$\hat{y}$ 是模型的预测概率分布。

均方误差的计算公式如下：

$$MSE(y,\hat{y})=\frac{1}{K}\sum_{i=1}^{K}(y_i-\hat{y}_i)^2$$

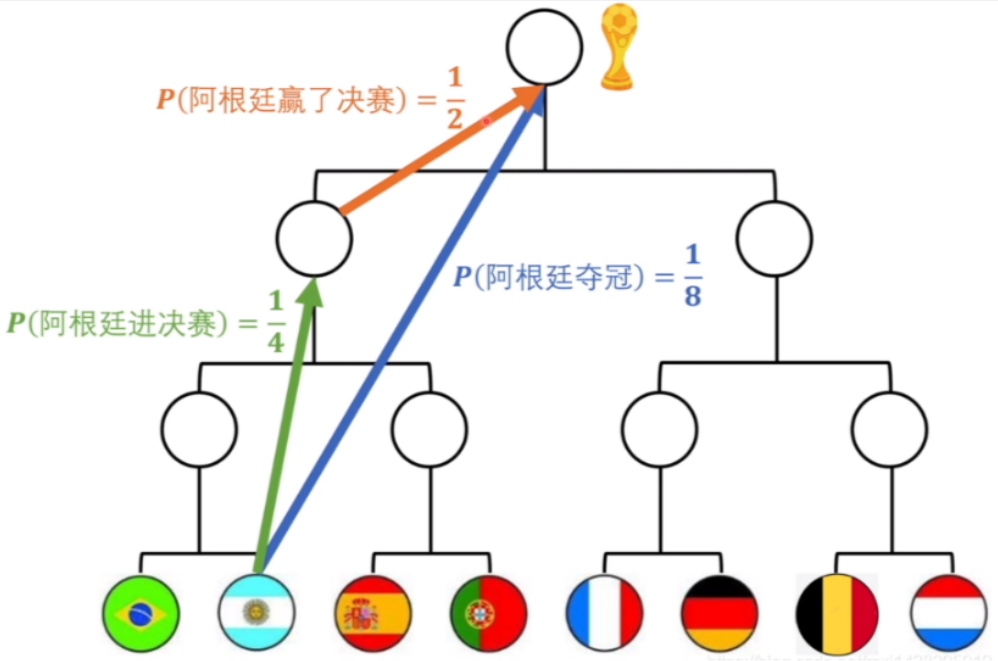
其中，y 是真实的概率分布，$\hat{y}$ 是模型的预测概率分布。

交叉熵损失函数在分类任务中更为常见，因为它对分类概率的误差敏感，特别是在多类别分类任务中。

均方误差在分类任务中**使用相对(注意是相对)**较少，因为它通常(**注意是通常**)用于回归任务，对于分类任务来说，它对单一样本的误差不够敏感。

> 敏感性：
> 交叉熵对于模型输出的概率与真实标签的概率之间的差异非常敏感。当模型对正确类别的预测概率较高时，交叉熵损失接近于零；而当模型对正确类别的预测概率较低时，交叉熵损失会迅速增大。

备注:



作业:

1. 案例: 给定字符串求信息熵

```
["a", "b", "c", "d", "e"]
["a", "b", "c", "a", "b"]
["a", "a", "a", "a", "a"]
```

```python
import numpy as np
def get_ent(x):  # 计算信息熵，未知概率分布
    length = len(x)
    dataDict = {}
    for letter in x:
        if dataDict.get(letter) is None:
            dataDict[letter] = 1
        else:
            dataDict[letter] += 1
```

```
    entropy = [-d / length * np.log(d / length) for d in
list(dataDict.values())]
    return sum(entropy)

print(get_ent(["a", "b", "c", "d", "e"]))
print(get_ent(["a", "b", "c", "a", "b"]))
print(get_ent(["a", "a", "a", "a", "a"]))
```

1. `print(get_ent(["a", "b", "c", "d", "e"]))`：这是一个包含5个不同元素的列表。由于每个元素都不同，熵较高，因为数据集的混乱度较大。你会得到一个较高的熵。
2. `print(get_ent(["a", "b", "c", "a", "b"]))`：这是一个包含5个元素的列表，其中有一些元素重复。相比于第一个例子，这个数据集的混乱度较小，因为有一些元素重复。你会得到一个介于第一个例子和第三个例子之间的熵值。
3. `print(get_ent(["a", "a", "a", "a", "a"]))`：这是一个包含5个相同元素的列表。由于所有元素都相同，数据集的混乱度最小。你会得到一个最低的熵值。

总体而言，信息熵越高，数据集的不确定性越大，反之越低则越有序

2、阶段性总结