# Lecture 8　Noise and Error

整理者: LobbyBoy* 2020年2月21日

## 1. Noise and Probabilistic Target

### 1) noise的类型

本章我们将讨论训练资料存在“噪声”(杂讯，noise)的情况。在之前的推导中我们有一个非常“平凡”的假设，即存在某个目标函数$f$与输入空间的潜在分布$P$。而训练数据中的一个个$\mathbf{x}$是根据潜在分布$P$采样得到的，对应的一个个$y$是$f(\mathbf{x})$的结果，这是没有noise的情况。下面我们在数据中引入noise，分两种情况(以“银行信用卡审批”为例)：

noise in $y$：$y$中存在noise分两种情况。第一，弃真存伪型noise：例如一个客户应该被通过，即应该被标记为+1，但却被错误地拒绝，即被误标为-1；第二，重复不同型noise，比如两个客户的数据$\mathbf{x}$一模一样，但一个被接受(+1)，另一个被拒绝(-1)。

noise in $\mathbf{x}$：即输入(input)数据不精确，存在错误。比如客户自己填表的时候因为手误把自己的年龄写错了等等。

可见，在引入noise后，问题好像变得十分复杂，我们想要知道之前我们在无noise情况下推导出来的VC bound是否适用于存在noise的情况。

### 2) 罐子(bin)与弹珠(marble)

回到罐子(bin)与弹珠(marble)的比喻。

之前我们说，想要估计罐子中橙色弹珠的比例，我们可以通过从罐子中拿出一些弹珠的方式，用拿出弹珠中橙色弹珠的比例作为罐子中橙子弹珠比例的估计量，并且这种估计方式的准确度被Hoeffding不等式所保证。

引申到机器学习中，我们将输入空间$\mathcal{X}$中的一个个$\mathbf{x}$看作一颗颗弹珠，这些$\mathbf{x}$服从一个潜在的未知的分布$P$，就如罐子中的弹珠一样：有些弹珠被抽到的概率高，有些弹珠被抽到的概率低，对应于输入空间$\mathcal{X}$中的每个$\mathbf{x}$被采样到的概率各不相同。之后我们用目标函数$f$与某一个hypothesis $h$给弹珠们“染色”——$f(\mathbf{x}) \neq h(\mathbf{x})$的$\mathbf{x}$被“染”成橙色，其余的

*本笔记根据台湾大学林轩田教授于线上教育平台Coursera开设的“机器学习基石”课程整理而成(课程内容见: https://www.coursera.org/learn/ntumlone-mathematicalfoundations/home/welcome)。笔记内的大多数图片来自于林老师的课程slides。感谢林老师能够将如此精彩的课程通过线上平台同所有人分享，thanks!

被“染”成绿色。我们希望得到但却无法直接获得的泛化误差$E_{out}(h)$就是输入空间中“橙色”$\mathbf{x}$的比例。我们可以用样本中的橙色比例估计总体中的橙色比例，这是有Hoeffding保证的。而样本中的橙色比例，就是经验误差$E_{in}$。因此用经验误差估计泛化误差，也是有Hoeffding保证的。

现在我们将弹珠想象成为一种颜色可以发生变化的弹珠，即probabilistic marbles(or noisy marbles)。例如，某个弹珠在30%的时间里是橙色的，70%的时间里是绿色的；也就是该弹珠是橙色的概率为0.3，是绿色的概率是0.7；也就是该弹珠这一秒可能是橙色的，下一秒就可能变成了绿色，但总体而言处于绿色状态的时间更多。在这种设定下，抽出的样本中橙色弹珠的比例又包含着什么信息呢？它反映了在抽样的那一瞬间，罐子中橙色弹珠的比例。因此，Hoeffding依然能够保证样本中的橙色弹珠比例是抽样瞬间罐子中橙色弹珠比例的良好估计。

将上面的论述引申到含有noise的机器学习情景中。一颗颗弹珠还是对应输入空间中的一个个$\mathbf{x}$，但我们不再假设存在某个目标函数$f$，而是假设存在一个条件分布$P(y|\mathbf{x})$，例如：某个输入$\mathbf{x}=\mathbf{x_0}$，其对应的条件分布为：

$$P(y_0=+1|\mathbf{x}=\mathbf{x_0})=0.7,\quad P(y_0=-1|\mathbf{x}=\mathbf{x_0})=0.3$$

可以简单理解为，该input对应的output有70%的概率为+1，有另外30%的概率为-1。之前我们用目标函数$f$与某个hypothesis $h$给所有$\mathbf{x}$染色，现在我们用条件分布$P(y|\mathbf{x})$与某个hypothesis $h$给所有$\mathbf{x}$染色：还是上面那个input $\mathbf{x_0}$，假设有$h(\mathbf{x_0})=+1$；那么$h(\mathbf{x_0})=y_0$的概率为0.7，不等的概率为0.3，也就是——$\mathbf{x_0}$被染成橙色的概率为0.3，被染成绿色的概率为0.7。这样，输入空间中的一个个$\mathbf{x}$，也变成了一颗颗可以变色的弹珠。由于样本中橙色弹珠的比例估计了抽样瞬间罐子中橙色弹珠的比例，因此样本中$y\neq h(\mathbf{x})$的数据比例估计了抽样瞬间总体中$y\neq h(\mathbf{x})$的数据比例，也就是$y\neq h(\mathbf{x})$的概率，只不过这个概率并不是deterministic的，而是probabilistic时的一个realization。

综上，我们依然可以在noise的场景中使用Hoeffding不等式，那么VC bound的结论当然也可以运用到有noise的场景，但需要做出一点变化：原来我们是假设输入空间上有一个潜在分布$P$，现在我们不仅要假设这个$P$的存在，还要假设一个条件分布$P(y|\mathbf{x})$——两者综合一下，也就是我们需要有一个关于$\mathbf{x}$和$y$的联合分布：

$$(\mathbf{x},y)\overset{i.i.d.}{\sim}P(\mathbf{x},y)$$

### 3) 目标分布：Target Distribution

在无noise的情况下，我们假设了目标函数$f$的存在；在有noise的情况下，我们则用一个条件分布$P(y|\mathbf{x})$来刻画noise带来的不确定性。这里的条件分布被称为“目标分布”，对

应于无noise情况下的“目标函数”。还是用之前的例子：某个输入$\mathbf{x} = \mathbf{x_0}$，其对应的条件分布为：

$$P(y_0 = +1|\mathbf{x} = \mathbf{x_0}) = 0.7, \quad P(y_0 = -1|\mathbf{x} = \mathbf{x_0}) = 0.3$$

我们将其看作是“ideal mini-target”与“noise”的叠加：因为+1的概率更大，因此我们将+1看作是ideal mini-target，而将-1看作是noise。而无noise的情形也可以看作是这种方式的一个特例：

$$P(y|\mathbf{x}) = 1 \quad for \quad y = f(\mathbf{x})$$

$$P(y|\mathbf{x}) = 0 \quad for \quad y \neq f(\mathbf{x})$$

如果我们将目标分布下output的不确定性分解成为ideal mini-target与noise来看待的话，我们的“学习目标”就变成了：对于经常出现的input较为较准确地预测其ideal mini-target——predict ideal mini-target(w.r.t $P(y|\mathbf{x})$) on often-seen inputs (w.r.t $P(\mathbf{x})$)。

而这个目标的确能够被实现：刚才我们已经论述过，样本中$y \neq h(\mathbf{x})$的数据比例估计了抽样瞬间总体中$y \neq h(\mathbf{x})$的数据比例，而我们将目标分布中出现的概率较大的结果看作是ideal mini-target，因此我们相信在抽样瞬间输入空间中大多数$\mathbf{x}$所对应的$y$都处于其ideal mini-target上——所以，样本中$y \neq h(\mathbf{x})$的数据比例，实际上也就估计了ideal mini-target value($\mathbf{x}$)$\neq h(\mathbf{x})$的概率。

## 2. Error Measure

回忆在无noise的情形下，对于二元分类问题，我们如何衡量某一个hypothesis $h$与目标函数$f$的“接近程度”？用$E_{out}$：

$$E_{out}(h) = E_{\mathbf{x}\sim P}\Big[I(h(\mathbf{x}) \neq f(\mathbf{x}))\Big]$$

这里我们选择了indicator function $I\Big(h(\mathbf{x}) \neq f(\mathbf{x})\Big)$作为我们的“错误衡量”(error measure)，这也是十分自然的：判断错了就记1个点的错误，否则不记任何错误。该错误衡量方式又被称为“0-1错误”(0/1 error)，经常用于分类问题。

当然还有其他很多种错误度量方式，但它们大多都是pointwise型，称为pointwise error measure。也就是说，使用这样的错误度量能够告诉我们hypothesis $h$在每一个$\mathbf{x}$上犯的错误是多少：

$$err(h(\mathbf{x}), f(\mathbf{x}))$$

而hypothesis $h$总体而言与目标函数$f$相差多少就可以表示为“平均错误度量”：

$$E_{out}(h) = E_{\mathbf{x}\sim P}\Big[err(h(\mathbf{x}), f(\mathbf{x}))\Big]$$

hypothesis $h$经验误差$E_{in}(h)$则为：

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N} err\Big(h(\mathbf{x_n}), f(\mathbf{x_n})\Big)$$

两种最重要的pointwise型错误衡量，一个是0/1误差，另一个是“平方误差”(squared error)。这里我们将预测值记作$\tilde{y}$，真值记作$y$：

$$err(\tilde{y}, y) = (\tilde{y} - y)^2$$

平方误差经常被用于回归(regression)问题中。那么在同一个问题中使用不同的错误衡量会不会导致不同的结果呢？答案是肯定的，下面我们将重新回到有noise的情形中，看看不同的错误衡量会给学习过程带来怎样的影响。

我们先po出一个claim：目标分布$P(y|\mathbf{x})$与错误衡量$err$共同决定了ideal mini-target $f(\mathbf{x})$，which is 我们在noise环境下希望学到“等价的目标函数”。举例如下：

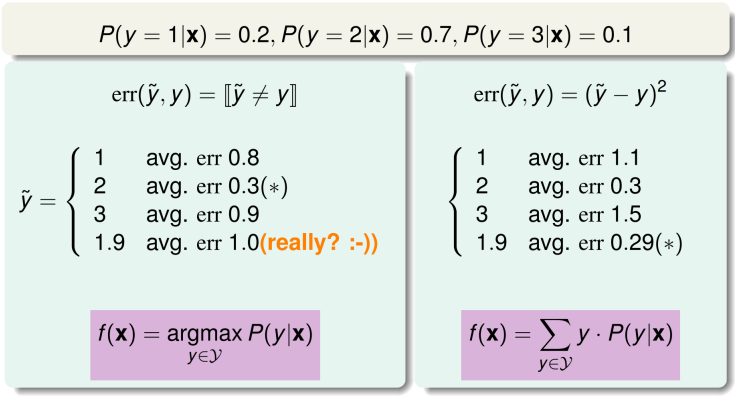


图 1: 某$\mathbf{x}$的ideal mini-target

下面我们来分别推导0/1误差与平方误差下的ideal mini-target的形式。首先是0/1误差，对于某$\mathbf{x}$，其ideal mini-target为：

$$f(\mathbf{x}) = \underset{\tilde{y}}{argmax}\, E_{y\sim P(y|\mathbf{x})}\Big[I(y \neq \tilde{y})\Big]$$

我们可以将RHS中的最优化目标函数写成：

$$E_{y\sim P(y|\mathbf{x})}\Big[I(y \neq \tilde{y})\Big] = \sum_{y_i\in\mathcal{Y}} P(y_i|\mathbf{x}) \cdot I(y_i \neq \tilde{y})$$

要想上式最大，那必有：

$$f(\mathbf{x}) = \underset{y \in \mathcal{Y}}{argmax}\, P(y|\mathbf{x})$$

而在平方误差的情形中，对于某$\mathbf{x}$，其ideal mini-target为：

$$f(\mathbf{x}) = \underset{\tilde{y}}{argmax}\, E_{y \sim P(y|\mathbf{x})}\left[(y - \tilde{y})^2\right]$$

我们可以将RHS中的最优化目标函数写成：

$$E_{y \sim P(y|\mathbf{x})}\left[(y - \tilde{y})^2\right] = \left(E_{y \sim P(y|\mathbf{x})}(y - \tilde{y})\right)^2 + Var(y|\mathbf{x})$$

要想上式最大，那必有：

$$E_{y \sim P(y|\mathbf{x})}(y - \tilde{y}) = 0$$

即：

$$f(\mathbf{x}) = E_{y \sim P(y|\mathbf{x})}(y) = \sum_{y \in \mathcal{Y}} y \cdot P(y|\mathbf{x})$$

最后，我们将更新learning flow，引入错误衡量：

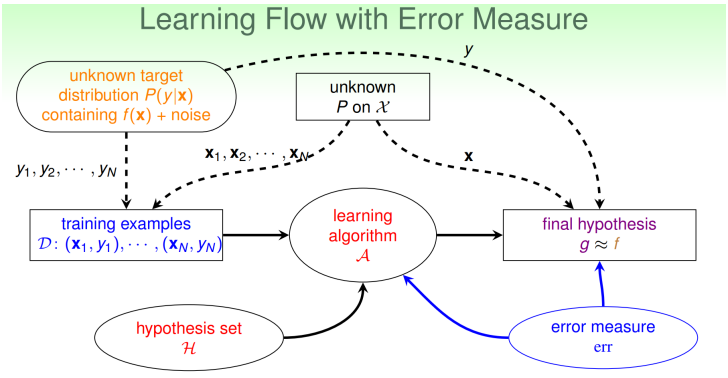


图 2: learning flow with error measure

## 3. Algorithm Error Measure

回忆此前我们使用0/1误差度量分类错误：只要预测值与实际值不符，就记一个错误点。然而，在实际中错误可以被分为两种类型：false accept，即实际值为-1但预测值为+1；false reject，即实际值为+1但预测值为-1。我们之前的做法是，无论是false accept还是false

reject，都记同样的一个错误点。但有些时候我们可能更不希望发生其中的一类错误，而把另一类错误看得没有那么可怕。

例如，某个超市使用指纹识别系统判断一位顾客是不是会员，以决定是否给其提供折扣，那么该超市一定更害怕false reject这种错误：false accept也就是损失一些钱的问题，但false reject可能会惹恼一个常客，导致其流失。因此该超市在训练指纹识别系统的时候，面临的是这样一个“成本矩阵”(cost matrix)：

| | | $g$ | |
|---|---|---|---|
| | | +1 | -1 |
| $f$ | +1 | 0 | 10 |
| | -1 | 1 | 0 |

图 3: cost matrix for supermarket

再例如，CIA的机密档案室使用指纹识别系统作为门钥匙，那么CIA一定更害怕false accept这种错误：false reject可能只是使自己的员工发发牢骚，但是false accept可能会将危险人物带入档案室，导致机密泄露。因此CIA面临的成本矩阵为：

| | | $g$ | |
|---|---|---|---|
| | | +1 | -1 |
| $f$ | +1 | 0 | 1 |
| | -1 | 1000 | 0 |

图 4: cost matrix for CIA

可见，不同的场景中即便都是在用0/1错误衡量，但在具体的操作上仍然不尽相同。也就是说，错误衡量在很大程度上是问题导向的(application/user-dependent)：你面临怎样的问题，就选择相应的合适的错误衡量。然而在现实中，我们往往很难去把握那个所谓的真正合适的错误衡量方式$err$，而是选择某种替代的$\hat{err}$：

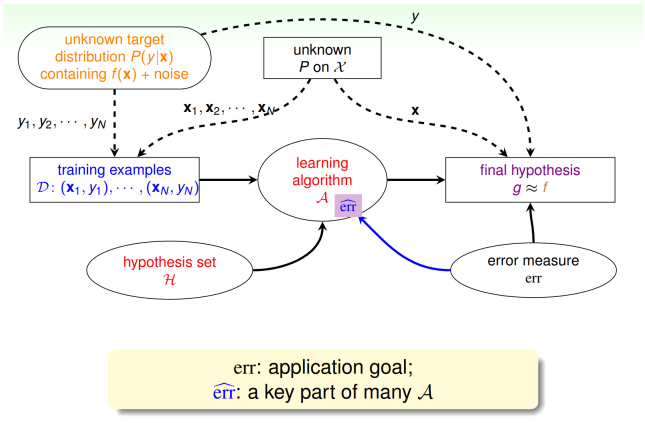


图 5: learning flow with algorithmic error measure

## 4. Weighted Classification

假设我们现在面临的是CIA的成本矩阵，即false accept的惩罚力度是false reject的1000倍。那么我们的$E_{in}(h)$与$E_{out}(h)$可以写成：

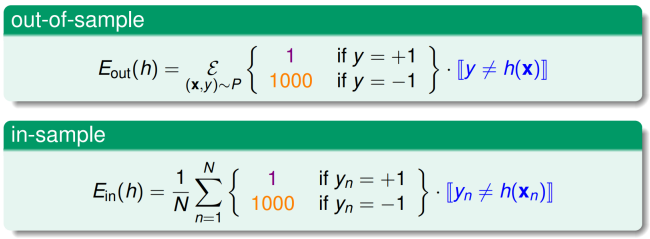


图 6: New $E_{in}(h)$ and $E_{out}(h)$

现在我们要解决的这个分类问题就叫做“weighted classification”，我们将这里的经验误差记作$E_{in}^{w}(h)$，以和普通的、等权重的分类问题中的$E_{in}^{0/1}(h)$区别开来：

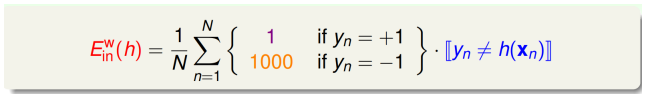


图 7: $E_{in}^{w}(h)$

要解决weighted classification，我们依然需要最小化经验误差，即最小化$E_{in}^{0/1}(h)$。如果资料是线性可分的，我们用PLA算法没有问题。但如果资料不是线性可分的，我们必须使用pocket算法，则需要做出一些调整。一个容易想到的变化是：原来我们是将$\mathbf{w}_{t+1}$与$\hat{\mathbf{w}}$的$E_{in}^{0/1}$进行比较，若$\mathbf{w}_{t+1}$更小则将其赋值给$\hat{\mathbf{w}}$，而现在则是比较$E_{in}^{w}$。

Pocket能够保证我们在迭代的过程中$E_{in}^{0/1}$逐渐变小，但没有直接保证$E_{in}^{w}$能够逐渐变小，我们还需做出其他调整。想象原问题的一个等价形式：

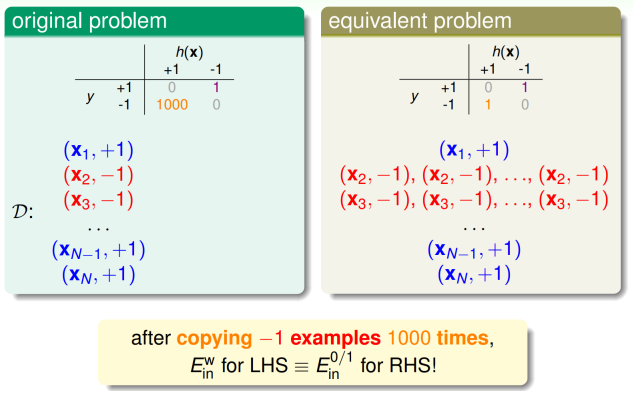


图 8: Virtual copying

也就是将“反例”($y = -1$的例子)拷贝1000份，然后把成本矩阵换成原来的等权重形式。在等价问题中，hypothesis每在反例上犯一个错误，$E_{in}^{0/1}$中都要加1000，这就相当于$E_{in}^{w}$中

反例犯一个错误。因此$E_{in}^{0/1} = E_{in}^{w}(h)$，我们在等价问题上做Pocket即可。

最后，其实我们不用真的将每个反例拷贝1000份。在Pocket算法中，一个样本被拷贝1000份，实质上是它的“经验分布”改变了，更容易被“抽”到了，因此只需在执行Pocket算法的check mistake时，将反例被挑中的概率提高即可(具体提高多少需要具体计算)。