

何时机器可以学习

笔记本: machine learning

创建时间: 2019/9/20 15:25

更新时间: 2019/9/22 15:56

作者: wsb

何时机器可以学习

一、机器学习概述

(一)、什么是机器学习

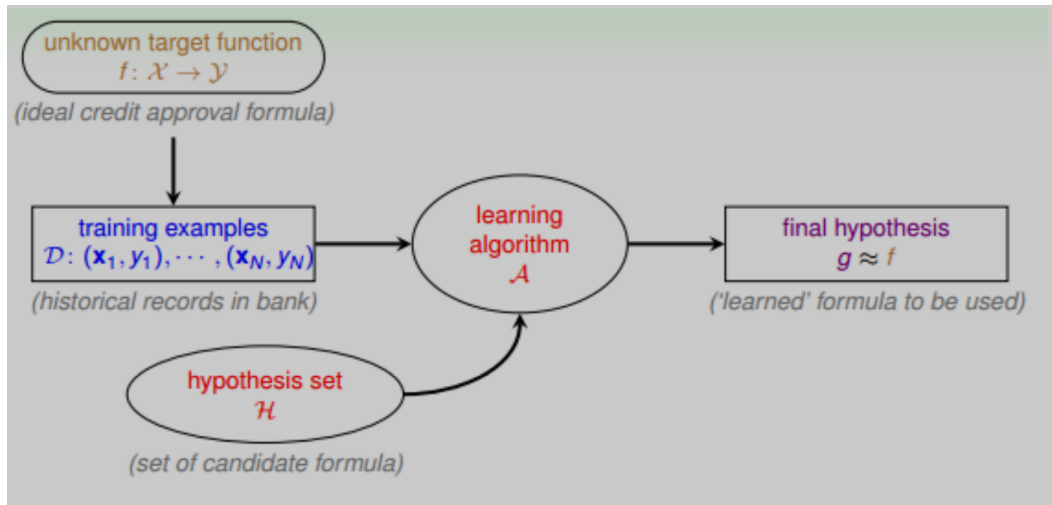
1. 机器学习定义: 机器从数据中总结经验, 从数据中找出某种规律或者模型, 用来解决实际问题。
2. 何时使用机器学习: 三个条件
 - (1) 事物本身有潜在规律
 - (2) 某些问题难以用普通变成解决
 - (3) 有大量的数据可以使用

(二)、机器学习应用——无处不在

(三)、机器学习组成

1. 基本术语:
 - (1) 输入 x
 - (2) 输出 y
 - (3) 目标函数 f 即最接近实际样本分布的规律
 - (4) 训练样本 data
 - (5) 假设 hypothesis: 一个机器学习模型对应的hypothesis会有很多, 通过算法A, 我们将选择一个最合适的hypothesis对应的函数称为 h_g , g 能最好地表示事物的内在规律, 也是我们最终想要得到的模型表达式。

2. 实际过程中，机器学习的流程：



首先对于理想的目标函数 f ，我们未知，但是因为它是理想的，我们手中的训练集 D 一定是它生成的，我们假设他是监督学习，即有 x ，也有 y 。机器学习的过程，就是根据先验知识选择模型 H （一组hypothesis） H 中包含了许多不同的hypothesis，然后通过算法 A ，我们对 D 进行训练，筛选出最好的hypothesis，它对应的函数表达式 g 就是我们最终要求的。一般情况下， g 最能接近目标函数 f ，这样，机器学习的流程就完成了。

（四）、机器学习和其他领域的关系

1. 数据挖掘
2. 人工智能
3. 统计学

事实上，机器学习和这三个领域都是相通的，基本类似，但是也不完全一样。机器学习是这三个领域的**有力工具**，这三个领域也是机器学习可以**广泛应用**的领域，总的来说，他们之间没有十分明确的界限

二、感知机模型

（一）、Perceptron Hypothesis Set

1. 引例：某银行要根据用户信息来判断该用户是否可以发信用卡。这是一个典型的机器学习问题，因为首先发信用卡存在某种规则，其次一般的编程不好解决，再次我们有银行里面很多用户的数据，因此ML可以派上用场。我们要根据 D ，通过 A ，在 H 中选择最好的 h ，得到 g 来接近目标函数 f ，也就是根据先验知识建立是否发信用卡的模型。我们如下定义：发信用卡（+1），不发信用卡（-1）。
2. 模型选择：这个是最重要的一步，也就是考虑Hypothesis Set，它很大程度上影响了机器学习的表现。这里我们选择的是感知机模型
3. 一些变量的说明：

| | | |
|------------|-------------|--------------|
| 个人信息：特征向量x | 特征权重：w | 阈值：threshold |
| 总共有d个特征 | 每个特征对应不同的权重 | 设定的值 |

我们的目的就是计算出所有特征加权和以及阈值threshold，然后比较。
大于threshold的时候 $h(x)=1$,否则 $h(x)=-1$

- For $\mathbf{x} = (x_1, x_2, \dots, x_d)$ 'features of customer', compute a weighted 'score' and

approve credit if

$$\sum_{i=1}^d w_i x_i > \text{threshold}$$

deny credit if

$$\sum_{i=1}^d w_i x_i < \text{threshold}$$
- $\mathcal{Y}: \{+1(\text{good}), -1(\text{bad})\}$, 0 ignored—linear formula $h \in \mathcal{H}$ are

$$h(\mathbf{x}) = \text{sign} \left(\left(\sum_{i=1}^d w_i x_i \right) - \text{threshold} \right)$$

我们为了方便起见:

$$w_0 = -\text{threshold}$$

因此，我们有：

$$h(x) = \text{sign}(w^T x)$$

为了更加清晰说明，我们先考虑二位的感知机模型

$$h(x) = \text{sign}(w^T x) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$$

其中 $w_0 + w_1 x_1 + w_2 x_2 = 0$ 是平面上一条分类直线，直线一侧是正类，直线另一侧是负类。权重w不同。对应于平面上不同的直线。

因此，大体上来说，所谓的感知机，在这个模型上就是一条直线，我们成为linear classifiers，但是感知器线性分类不限定在二维空间中，这是一个泛指在3D中，线性分类就用平面表示，在更高维度中，线性分类用超平面表示。

linear(binary) classifiers: 形如 $w^T x$ 的限定模型

(二)、Perceptron Learning Algorithm(PLA)

有了模型是之后，我们考虑算法A。也就是找到一个A，来选择一个最好的直线能把平面上所有的正类负类完全分开。

1. 思想：逐点修正，先随便取一条直线，然后对第一个错误进行修正，然后再依次修改剩下的错误。

For $t = 0, 1, \dots$

- ① find a **mistake** of \mathbf{w}_t called $(\mathbf{x}_{n(t)}, y_{n(t)})$

$$\text{sign}(\mathbf{w}_t^T \mathbf{x}_{n(t)}) \neq y_{n(t)}$$

- ② (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

... until **no more mistakes**
return **last \mathbf{w} (called \mathbf{w}_{PLA}) as g**

2. 操作流程：如果第一个错误发生在正类上，也就是说它被分成了负类那我们有

$$\mathbf{w}^T \mathbf{x}_{n(t)} < 0$$

也就是说， \mathbf{w} 和 \mathbf{x} 夹角是钝角， \mathbf{w} 是直线的法向量。

那么修正的方法就是让它们之间的夹角变成锐角，同理对于 \mathbf{w} 和 \mathbf{x} 夹角是锐角的情况我们的做法就是让他们之间的夹角变成钝角。因此一般的做法就是

$$\mathbf{w} := \mathbf{w} + y\mathbf{x}, \quad y = \pm 1$$

一句话总结这个算法就是：“**A fault confessed is half redressed 知错能改。**”

注：PLA算法可收敛的条件是D是线性可分的，否则可能没法收敛

3. 实际操作中，我们可以一个逐个点遍历，发现错误就进行修正，直到所有点全部分类正确 —— **Cyclic PLA**

Cyclic PLA

For $t = 0, 1, \dots$

- ① find **the next** mistake of \mathbf{w}_t called $(\mathbf{x}_{n(t)}, y_{n(t)})$

$$\text{sign}(\mathbf{w}_t^T \mathbf{x}_{n(t)}) \neq y_{n(t)}$$

- ② correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

... until **a full cycle of not encountering mistakes**

(三)、PLA算法的可行性

PLA什么时候会停下来？直到全部分类正确。否则不会停止。

两个问题：

- PLA算法在线性不可分的情况下咋办
- PLA停下来的时候是否就有 $f \approx g$, 倘若停不下来是否有 $f \approx g$?

线性可分的情况：如果有一直线可以完全二分类，我们假设此时权重是 w_f ，那么我们必然对每个点：

$$y_{n(t)} w_f^T x_{n(t)} \geq \min_n y_n w_f^T x_n > 0$$

我们希望证明每一次操作后 w_t 和 w_f 越来越接近：

- 模长没有太大改变
- 夹角变小

$$\begin{aligned} w_f^T w_{t+1} &= w_f^T (w_t + y_{n(t)} x_{n(t)}) \\ &\geq w_f^T w_t + \min_n y_n w_f^T x_n \\ &> w_f^T w_t + 0. \end{aligned}$$

w_t changed only when mistake

$$\Leftrightarrow \text{sign}(w_t^T x_{n(t)}) \neq y_{n(t)} \Leftrightarrow y_{n(t)} w_t^T x_{n(t)} \leq 0$$

- mistake 'limits' $\|w_t\|^2$ growth, even when updating with 'longest' x_n

$$\begin{aligned} \|w_{t+1}\|^2 &= \|w_t + y_{n(t)} x_{n(t)}\|^2 \\ &= \|w_t\|^2 + 2y_{n(t)} w_t^T x_{n(t)} + \|y_{n(t)} x_{n(t)}\|^2 \\ &\leq \|w_t\|^2 + 0 + \|y_{n(t)} x_{n(t)}\|^2 \\ &\leq \|w_t\|^2 + \max_n \|y_n x_n\|^2 \end{aligned}$$

第一个式子说明 w_t 和 w_f 之间的内积变小，第二个式子说明了 w_{t+1} 和 w_t 之间的差距被控制了，因此模长不会差距太大
然后我们证明有限次会停下来，也就是T是有限的

Given 3 conditions

1 w_f perfectly fitting those data means:

$$y_{n(t)} w_f^T x_{n(t)} \geq \min_n y_n w_f^T x_n > 0 \quad (1)$$

2 w_t changed only when making mistakes means:

$$\text{sign}(w_t^T x_{n(t)}) \neq y_{n(t)} \Leftrightarrow y_{n(t)} w_t^T x_{n(t)} \leq 0 \quad (2)$$

3 we make corrections by:

$$w_t = w_{t-1} + y_{n(t)} x_{n(t)} \quad (3)$$

We want to prove after t mistake corrections starting from 0, we will get:

$$\frac{w_f^T}{\|w_f\|} \frac{w_t}{\|w_t\|} \geq \sqrt{T} \cdot \text{constant}$$

it means 2 things:

1 Because the above equation is less equal than 1, we also prove the T will have upper boundary, thus the algorithm will stop at some time;

2 the left part the above equation means the angle of w_f and w_t , and their angle is decreasing, in other words w_t is approaching w_f , thus we are on the right way to get the solution

Here is the prove:

the primary principle is to replace all the variables which in above equations are w_t and $\|w_t\|$

for w_t we have:

$$\begin{aligned} w_f^T w_t &= w_f^T (w_{t-1} + y_{n(t-1)} x_{n(t-1)}) && \text{using (3)} \\ &\geq w_f^T w_{t-1} + \min_n y_n w_f^T x_n && \text{using (1)} \\ &\geq w_0 + T \cdot \min_n y_n w_f^T x_n && \text{applying T times} \\ &\geq T \cdot \min_n y_n w_f^T x_n \end{aligned}$$

for $\|w_t\|$ we have:

$$\begin{aligned} \|w_t\|^2 &= \|w_{t-1} + y_{n(t-1)} x_{n(t-1)}\|^2 && \text{using (3)} \\ &= \|w_{t-1}\|^2 + 2y_{n(t-1)} w_{t-1}^T x_{n(t-1)} + \|y_{n(t-1)} x_{n(t-1)}\|^2 \\ &\leq \|w_{t-1}\|^2 + 0 + \|y_{n(t-1)} x_{n(t-1)}\|^2 && \text{using (2)} \\ &\leq \|w_{t-1}\|^2 + \max_n \|x_n\|^2 \\ &\leq \|w_0\| + T \cdot \max_n \|x_n\|^2 = T \cdot \max_n \|x_n\|^2 \end{aligned}$$

applying above 2 conclusions to the left part of the equation we finally get:

$$\begin{aligned}
 \frac{w_f^T}{\|w_f\|} \frac{w_T}{\|w_T\|} &= \frac{T \cdot \min_n y_n w_f^T x_n}{\|w_f^T\| \cdot \|w_t\|} \\
 &\geq \frac{T \cdot \min_n y_n w_f^T x_n}{\|w_f^T\| \cdot \sqrt{T} \cdot \max_n \|x_n\|} \\
 &\geq \frac{\sqrt{T} \cdot \min_n y_n w_f^T x_n}{\|w_f^T\| \cdot \max_n \|x_n\|} = \sqrt{T} \cdot \text{constant}
 \end{aligned}$$

Because the $\frac{w_f^T}{\|w_f\|} \frac{w_T}{\|w_T\|} \leq 1$, we finally have:

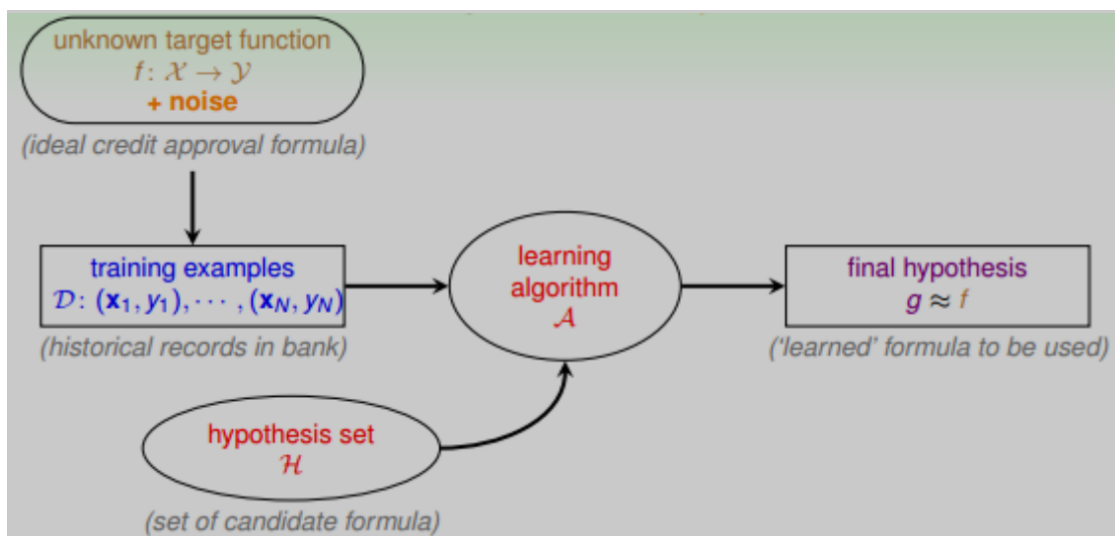
$$\begin{aligned}
 \frac{\sqrt{T} \cdot \min_n y_n w_f^T x_n}{\|w_f^T\| \cdot \max_n \|x_n\|} &\leq 1 \\
 \Leftrightarrow T &\leq \frac{\max_n \|x_n\|^2 \cdot \|w_f^T\|^2}{\min_n y_n w_f^T x_n} = \frac{R^2}{\rho^2}
 \end{aligned}$$

不等式左边实际上就是余弦值，越来越接近1，说明正确率大致上逐渐提高，然后T也有上界，最终可以完全分类。

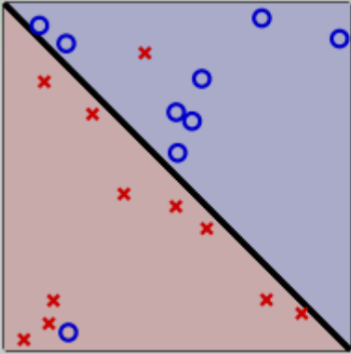
(三)、线性不可分的data

这种情况下 w_f 实际上并不存在，PLA不一定会停下来。

不过我们可以换一个思路，把非线性可分情况当成是数据集中参杂了一些noise



在非线性情况下，我们可以把条件放松，即不苛求每个点都分类正确，而是容忍有错误点，取错误点的个数最少时的权重 w ：



- assume 'little' noise: $y_n = f(\mathbf{x}_n)$ **usually**
- if so, $g \approx f$ on $\mathcal{D} \Leftrightarrow y_n = g(\mathbf{x}_n)$ **usually**
- how about

$$\mathbf{w}_g \leftarrow \operatorname{argmin}_{\mathbf{w}} \sum_{n=1}^N \mathbb{I}[y_n \neq \operatorname{sign}(\mathbf{w}^T \mathbf{x}_n)]$$

—NP-hard to solve, unfortunately

这个时候是**NP-hard**问题，难以求解。

因此我们试图修改一下PLA算法，称为**Packet Algorithm**

流程：

初始化权重 w_0 ，计算出在这条初始化的直线中，分类错误点的个数。然后对错误点进行修正，更新 w ，得到一条新的直线，在计算其对应的分类错误点的个数，如果更少，就更新直线。之后再经过 n 次迭代，不断得到错误点个数最少的直线

initialize pocket weights \hat{w}

For $t = 0, 1, \dots$

- 1 find a (random) mistake of \mathbf{w}_t called $(\mathbf{x}_{n(t)}, y_{n(t)})$
- 2 (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

- 3 if \mathbf{w}_{t+1} makes fewer mistakes than \hat{w} , replace \hat{w} by \mathbf{w}_{t+1}

...until **enough iterations**

return \hat{w} (called w_{POCKET}) as g

三、学习类型

(一)、在不同输出空间Y下的学习

除了二元分类，也有多元分类问题，多元分类的输出多余两个， $y=\{1,2,3,...,K\}, K>2$ 。一般多元分类的应用有**数字识别**、**图片内容识别**等等。

二元分类和多元分类都属于分类问题，他们的输出都是离散值。

如果输出是连续的，这类问题称为**回归问题**。

在NLP中还会遇到**结构化学习**，输出空间包含了某种结构在里面，一些解法通常是从多元分类问题中延伸出来的。

当然，如果按照输出空间概念划分的话，最基础最核心的两个类型就是**二元分类**和**回归**。

(二)、在不同数据标签 y_n 下学习

1. 如果是监督学习，我们既有 x 也有 y
2. 如果是无监督学习，我们是没有输出标签 y_n ，典型的无监督学习包括：
 - 聚类问题，例如对网页上的新闻自动分类
 - 密度估计，例如交通路况分析
 - 异常检测，比如用户网络流量监测
3. 如果是半监督学习，我们有一部分数据有输出标签 y_n ，另一部分数据没有输出标签 y_n 。例如药物检测因为成本和试验人群限制等问题只能有一部分输出标签
4. 如果是增强学习，我们通过模型输出反馈来学习，给定奖惩措施。例如通过用户点击选择而不断改进的广告系统

| supervised | unsupervised | semi-supervised | reinforcement | ... |
|------------|--------------|-----------------|----------------------------|-----|
| all y_n | no y_n | some y_n | implicit y_n by goodness | ... |

(三)、在不同的协议 $f(x,y)$ 下学习

1. Batch learning: **填鸭式**
每次获得的训练数据 D 是一批的，对其进行建模。应用最为广泛。
2. Online learning: **在线式**
在线学习模型，数据是实时更新的，根据数据一个个进来，被动同步更新我们的算法，例如PLA，每次对于一个错误的点来更新模型。
3. Active learning: **自学式**
让机器自己具备学习能力，例如手写数字识别，让它自己问问题。优势一般在获取样本label比较困难的时候，可以节约时间和成本，对一些重要的label提出需求。

(四)、在不同的输入空间X下学习

1. Concrete features: 具体特征

例如硬币分类中硬币的尺寸、重量，疾病诊断中病人信息
这个对于machine最容易理解使用的。

2. Raw features: 粗糙特征

例如手写数字识别中每个数字所在图片的 $m \times n$ 像素值，比如语音信号的频谱。
一般比较抽象，需要转换成concrete features,即Feature Transform

3. Abstract features: 抽象特征

完全抽象的特征，没有什么物理含义，对于machine处理很困难一般需要更多提取转换

四、机器学习的可行性

(一)、学习是一件不可能的事情

NFL定理：没有一个学习算法可以在任意领域总是产生最准确的学习器。算法的优越性只能在特定条件下论断。

(二)、概率模型

一个装有很多橙色球和绿色球的罐子中，我们能不能推断橙色球的比例 u ？统计学的方法就是随机取出 N 个球作为样本，计算出这些球中橙色球的比例 v ，那么就是估计出管子中橙色球比例为 v 。我们看一下 v, u 的接近程度——霍夫丁不等式：

$$P[|v-u| > \epsilon] \leq 2\exp(-2\epsilon^2 N)$$

说明当 N 很大的时候, v 和 u 会很接近，差值被限定在 ϵ 以内。

$v=u$ 被称为 **probably approximately correct(PAC)**

(三)、概率和学习联系起来

我们通过罐子模型来预测hypothesis和 f 一致的可能性。

罐子里的一颗弹珠类比为 x ,其中颜色为橙色表明 $h(x) \neq f$,绿色表明 $h(x) = f$ ，从管子中抽取的 N 个球类比为机器学习的训练样本 D ，且这两种抽样的样本和总体样本之间都是独立同分布的。因此只要 N 足够大且i.i.d就可以使用上述模型。

这里引入两个值 $E_{in}(h)$ 和 $E_{out}(h)$

- $E_{in}(h)$ 表示在**抽样样本**中, $h(x) \neq y_n$ 的概率
- $E_{out}(h)$ 表示在**实际样本**中, $h(x) \neq y_n$ 的概率

$$P[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2\exp(-2\epsilon^2 N)$$

一般如果N很大且h给定的时候，我们会有 $E_{in} \approx E_{out}$ ，但是这并不意味着 $g \approx f$ 。因为h是固定的，不能保证 $E_{in}(h)$ 足够小。

因此，我们一般会通过演算法A，先选择最好的h，使得 $E_{in}(h)$ 足够小，从而保证 $E_{out}(h)$ 很小。固定h之后使用新数据测试错误率。

(四)、和真实的学习联系

可能有的时候实验结果偏差很大，比如掷硬币五次都是正面朝上，抽到全部都是绿球也不一定说明罐子就是全部绿球。当罐子数目很多的时候可能会出现**Bad Example**，也就是 E_{in} 和 E_{out} 差别太大——选择过多会恶化不好的情形。

对于很多次抽样得到的不同数据集D，只要对于某一个hypothesis，D是**Bad Data**就说明它是bad。因此，根据霍夫丁不等式，上界可以表示为级联形式：

union bond:

$$\begin{aligned} P_D[\mathbf{BAD} \ D] &= P_D[\mathbf{BAD} \ D \text{ for } h_1 \text{ or } \mathbf{BAD} \ D \text{ for } h_2 \text{ or } \dots \mathbf{BAD} \ D \text{ for } h_M] \\ &\leq P_D[\mathbf{BAD} \text{ for } h_1] + P_D[\mathbf{BAD} \text{ for } h_2] + \dots + P_D[\mathbf{BAD} \text{ for } h_M] \quad (\text{union bond}) \\ &\leq 2\exp(-2\epsilon^2 N) + 2\exp(-2\epsilon^2 N) + \dots + 2\exp(-2\epsilon^2 N) \\ &= 2M\exp(-2\epsilon^2 N) \end{aligned}$$

其中，M是hypothesis的个数，N是样本D的容量， ϵ 是参数。

上述公式说明，M有限，N足够大的时候，**Bad Data**出现的概率就耕地了

因此我们就可以选择一个合适的算法，选择让 E_{in} 最小的 h_m 作为矩g（最好的h），一般能够保证 $g \approx f$ ，即有不错的泛化能力。

总的来说，

- 只要M有限，N足够大，通过A任选一个矩g，都有 $E_{in} \approx E_{out}$
- 若g能使 $E_{in} \approx 0$ ，**PAC**就可以让 $E_{out} \approx 0$

因此机器学习还是可行的！