为何机器可以学习

一、训练 vs 测试

（一）、重述和预览

之前的note1中我们主要探讨了如下的两个机器学习核心问题：

1. $E_{in}(g) \approx E_{out}(g)$ —— （1）
2. $E_{in}(g)$足够小 ——— （2）

- M很小的时候，由霍夫丁不等式，（1）式容易满足，但是由于hypothesis选择较少（2）不容易满足
- M很大的时候，（2）容易满足，但是$E_{in}(g)$和$E_{out}(g)$可能差的比较大

所以，对于M的选择直接影响到机器学习的两个核心问题了。因此M不能太大也不能太小。倘若M**无限大**，那么是否就不可以机器学习了？但是PLA再无限条直线的时候还是跑的好好的。我们试图尽可能把M控制在一个有限的m_H内，这样就可以解决了。

（二）、有效线的数量

霍夫丁不等式：$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$ ，M是hypothesis的个数。
我们考虑每个hypothesis下BAD events B_m级联的形式满足下列不等式：

$$P[B_1 or B_2 or B_3 or ... or B_M] \leq P[B_1] + P[B_2] + ... + P[B_M]$$

当$M = \infty$的时候，上述不等式右侧很大，因为我们扩大了上界，union bound 过大。和实际情况不吻合。因此我们线看看实例来得到启发。

我们看看Perceptron中在给定的N个点中最多有多少个情况可以用一条直线划分这些点。

lines in	2D
N	$effective(N)$
1	2
2	4
3	8
4	14 < 16

我们霍夫丁不等式：$P[|E_{in}(g)-E_{out}(g)|>\epsilon]\leq 2effective(N)e^{-2\epsilon^2N}$
我们发现，$effective(N)$的数量总是小于2^N，因此我们如果能保证这个关系持续下去，右侧的不等式接近于零，那么即使M无限大，支线的种类也是有限的，机器学习也是可行的！

（三）、有效hypothesis的数量

1. 二分类(dichotomy)
   dichotomy就是将空间中的点(例如二维平面)用一条直线分成正类、负类。令H是将平面上的点用直线分开的所有hypothesis集合那么dichotomy H和hypothesis的关系是：hypothesis H 是平面上所有直线的集合，个数可能是无限个，dichotomy是能将点完全用直线分开的直线种类，上界是2^N个，我们要做的就是用dichotomy代替M。
2. 成长函数(grow function)，即 $m_H(H)$，上界是2^N，也就是之前说的effective *lines* 的最大值。
   成长函数的定义是：对于由N个点组成的不同集合中，某集合对应的dichotomy最大，那么这个dichotomy值就是$m_H(H)$,上界是2^N

$$m_H(N)=\max_{x_1,x_2,x_3,...,x_N\in X}|H(x_1,x_2,...,x_N|$$

   成长函数其实就是effective(N)的最大值，根据成长函数的定义，二维平面上，$m_H(H)$和N的关系如上所示。我们列举几种常见模型的**growth function**:

(1) Positive Rays: $m_H(H) = N + 1$，取一点和最后一点构成线段（可以是它自己），还可以都不取

(2) Positive Intervals: $m_H(H) = \binom{N}{2} + n + 1$，

- 从N个点中选两个作为线段来分隔
- N个点每个自己构成线段来分隔
- 全部是反类

(3) 凸多边形封闭曲线，我们在把所有的正类连起来作为划分区域即可。这样对于每一种情况我们都能做到有效（*此处的line是曲线*）我们把这种能完全分类的情况叫做shattered，对应growth function 是2^N个

（四）、Break Point

model	$m_H(N)$
positive rays	$m_H(N) = N + 1$
positive intervals	$m_H(N) = \binom{2}{N+1} + 1$
convex sets	$m_H(N) = 2^N$
2D perceptron	???

根据之前的分析，我们知道2D perceptron在 N=4的时候就没办法做出所有的16个点的dichotomy了，于是我们把4称作2D perceptron的**break point**

model	$m_H(N)$	**break point**
positive rays	$m_H(N) = N + 1 = O(N)$	2
positive intervals	$m_H(N) = \binom{2}{N+1} + 1 = O(N^2)$	3
convex sets	$m_H(N) = 2^N$	no
2D perceptron	???	4

通过观察，我们猜测如下结论：

$$m_H(N) = O(N^{k-1})$$

二、一体化理论

下面详细讨论上述猜想

（一）、Break Point k 的限制

shatter: 对N个点，能够分解为2^N种dichotomies

我们就看k=2的时候成长函数 $m_H(N)$是多少。

- N=1,$m_H(H) = 2$
- N=2,由K知任意两点都不能被shattered，$m_H(2) \leq 3$
  我们也可以得到$m_H(3) = 4$
- N>k时，break point 限制了$m_H(N)$的大小，影响成长函数$m_H(N)$的因素主要由两个：
  - 抽样数据集N
  - break point k（假设类型）

如果给定N,k,能够证明$m_H(N)$的最大值上界是多项式的，根据霍夫丁不等式，就可以用$m_H(N)$代替M，下证$m_H(N)$上界是poly(N)

（二）、Bounding Function: Basic Cases

引入新的函数bounding function,**B(N,k)**

$$B(N,k) = \max_k m_H(N) \leq_{???} poly(N)$$

我们已知的结论有：

- B(2,2) = 3 (max < 4)
- B(3,2) = 4

1. k=1, $B(N,k) \equiv 1$
2. k>N, 由于N比k小，我们有2^N条dichotomy，$B(N,k) = 2^N$
3. k=N，去掉一个就满足了$B(N,k) = 2^N - 1$
4. k<N, 情况比较复杂，给出推导过程：
   以B(4,3)为例，首先想着能否构建B(4,3)与B(3,x)之间的关系。
   首先，B(4,3)的情况共有11组，也就是说再加一种dichotomy，任意三点都会被shatter

number	x_1	x_2	x_3	x_4
01	o	o	o	o
02	x	o	o	o
03	o	x	o	o
04	o	o	x	o
05	o	o	o	x
06	x	x	o	x
07	x	o	x	o
08	x	o	o	x
09	o	x	x	o
10	o	x	o	x
11	o	o	x	x

对于这11种dichotomy分组，目前分成两组，分别是粗体和非粗体部分，粗体就是x_1, x_2, x_3一致，x_4不同但是成对，非粗体是single的

number	x_1	x_2	x_3	x_4
01	o	o	o	o
05	o	o	o	x
02	x	o	o	o
08	x	o	o	x
03	o	x	o	o
10	o	x	o	x
04	o	o	x	o
11	o	o	x	x
06	x	x	o	x
07	x	o	x	o
09	o	x	x	o

粗体去重后得到的4个不同的vector成为α,相应的非粗体的部分是β

$$B(4,3) = 2\alpha + \beta$$

又由α, β构成的所有三点组合也不能shatter $\alpha + \beta \leq B(3,3)$
另一方面，由于α和x_4是成对出现的，且α不能被任意三点shatter所以推导出α中x_1 x_2 x_3是不能被任意两点shatter，否则如果能被任意两点shatter，加上x_4之后就能被三点shatter,所以我们有

- $B(4,3) = 2\alpha + \beta$

- $\alpha+\beta \leq B(3,3)$
- $\alpha \leq B(3,2)$

因此我们有， $B(4,3) \leq B(3,3)+B(3,2)$
一般地，我们泛化得到如下公式：

- $B(N,k)=2\alpha+\beta$
- $\alpha+\beta \leq B(N-1,k)$
- $\alpha \leq B(N-1,k-1)$

$B(N,k) \leq B(N,k-1)+B(N-1,k-1)$
继而有，

$$B(N,k) \leq \sum_{i=0}^{k-1}\binom{N}{i} = O(N^{k-1})$$

右侧是一个多项式，我们完成了要求。进一步推广，$B(N,k) \leq N^{k-1}$

（三）、简易证明

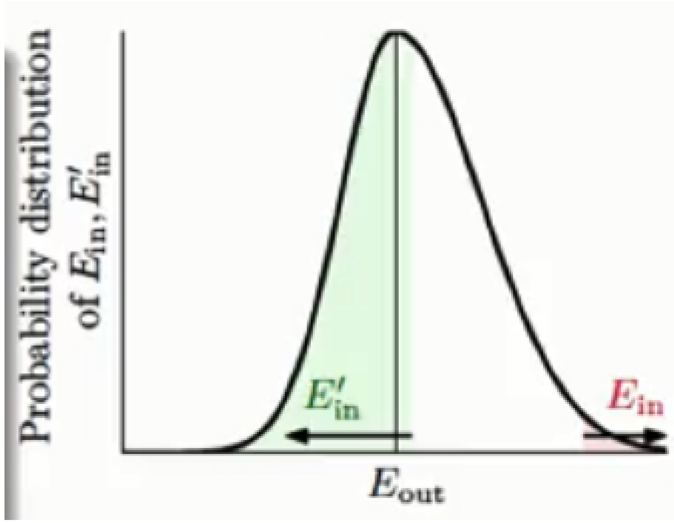
我们看看能不能把 $m_H(N)$替换M（其实没那么简单）
我们有

$$P[\exists h \in H\ s.t\ |E_{in}(h)-E_{out}(h)>\epsilon] \leq 2m_H(N)e^{-2\epsilon^2 N}$$

实际上是：

$$P[\exists h \in H\ s.t\ |E_{in}(h)-E_{out}(h)>\epsilon] \leq 2\cdot 2m_H(2N)e^{-2\frac{1}{16}\epsilon^2 N}$$

证明分为三步来进行

1. 用$E'_{in}(N)$代替$E_{out}(N)$,这是因为$E_{out}(N)$衡量的是h在全体输入上的错误，全体输入包含了位置的无限多个输入，所以我们希望找到一个有限的数来代替$E_{out}(N)$。我们假设有一个测试集D'，测试集的数据量仅有N个，那么原来的$E_{out}(N)$其实就相当于D'的E_{in}，记为E'_{in}，D'被我们称作 **Ghost Data**，如果E_{in}离E_{out}很远，那么E_{in}离E'_{in}也会很远，因此我们可以用E'_{in}来代替E_{out}了，但是要加入一些常数来修正一些东西:
   

   结合图片解释我们有，$\frac{1}{2}\mathbb{P}[\exists h \in H\ s.t\ |E_{in}(h) - E_{out}(h) > \epsilon] \leq \mathbb{P}[\exists h \in H\ s.t\ |E_{in}(h) - E_{out}(h) > \frac{\epsilon}{2}|]$
2. 通过种类分解H
$$BAD \leq 2\mathbb{P}[\exists h \in H\ s.t.|E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}] \leq 2m_H(2N)P[fixed\ h\ s.t.|\ E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}]$$
3. 直接用霍夫丁不等式：
   考虑有2N个例子的罐子，选N个作为E_{in},其他的作$E'_{in}(E_{out})$
   有$|E_{in} - E'_{in}| > \frac{\epsilon}{2} \iff |E_{in} - \frac{E_{in}+E'_{in}}{2}| > \frac{\epsilon}{4}$

$$
\begin{aligned}
BAD &\le 2m_H(2N)P[fixed\ h\ s.t. |\ E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}] \\
&\le 2m_H(2N) \cdot 2e^{-2(\frac{\epsilon}{4})^2 N}
\end{aligned}
$$

最后我们得到了新的不等式，称之为Vapnik-Chervonenkis(VC) bound:

$$
\mathbb{P}[\exists h \in H s.t. |E_{in}(h) - E_{out}(h)| > \epsilon] \le 4m_H(2N)e^{-\frac{1}{8}\epsilon^2 N}
$$

对于 2D perceptron, break point 是4，$m_H(N) = O(N^3)$。因此，我们可以说2D perceptrons是可以济宁机器学习的，只要找到hypothesis能让$E_{in} \approx 0$就能保证$E_{in} \approx E_{out}$

三、 VC Dimension

(一）、回顾

1. 机器学习能够进行的条件:

- 假设空间H的size M是有限的，即当N足够大，对于假设空间中任意一个假设g，$E_{in} \approx E_{out}$
- 利用算法A从假设空间H中，挑选一个g，使得$E_{in}(g) \approx 0$，则$E_{out}(g) \approx 0$

2. train & test

- train: 让损失期望$E_{in}(g) \approx 0$
- test: 让算法用到新样本的时候的损失期望尽可能小，$E_{out} \approx 0$

(二）、VC Dimension 的定义

我们知道若一个假设空间H 有break point k 那么它的成长函数是有界的，上界称之为Bound Function.根据数学归纳法，Bound Function也是有界的，而且上界是N^{k-1}我们通过观察可以发现N^{k-1}比B(N,k)松弛很多。

$$\begin{aligned}
&\mathbb{P}_{\mathbb{D}}[|E_{in}(g)-E_{out}(g)|>\epsilon] \\
\leq &\mathbb{P}_{\mathbb{D}}[\exists h \in H s.t. |E_{in}(h)-E_{out}(h)|>\epsilon] \\
\leq &4m_H(2N)e^{-\frac{1}{8}\epsilon^2 N} \\
\overset{if\ k\ exists}{\leq} &4(2N)^{k-1}e^{-\frac{1}{8}\epsilon^2 N}
\end{aligned}$$

这样不等式就只和k，N有关了，一般情况下，N足够大，所以我们只考虑k值。我们有如下结论：

- 若假设空间H有break point k 而且N足够大，那么根据VC bound theory,算法有良好的泛化能力。
- 在假设空间中选择一个矩g，让$E_{in} \approx 0$则其在全集数据中的错误率会较低。

接下来介绍VC Dimension:某假设集H能够shatter的最多inputs的个数，也就是最大完全正确的分类能力。 （只要存在一种分布的inputs能够正确分类就满足）易知 $d_{vc}=breakpoint-1$。因此，如果假设集H的d_{vc}确定了，我们就满足了第一个条件，那么与**算法**、**样本数据分布**无关。

（三）、Perceptrons 的 VC Dimension

2D的时候，k=4，$d_{vc}=3$，我们根据VC Bound理论，当N足够大的时候，满足第一个条件。如果还能找到矩g满足第二个条件，我们就能证明PLA是可以学习的。然后我们试图考虑多为的情况。我们猜测$d_{vc}=d+1$，其中d是维数。我们用夹逼原则证明。

1. 在d维中，我们只要找到某一类的d+1个inputs可以被shatter的话，那么必然得到$d_{vc} \geq d+1$构造X是d维度的，有d+1个inputs，每个inputs加上第零个维度的常数项1，得到X的矩阵：
$$\begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \\ ... \\ x_{d+1}^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & ... & 0 \\ 1 & 1 & 0 & ... & 0 \\ 1 & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... \\ 1 & 0 & 0 & ... & 1 \end{bmatrix}$$
矩阵中，每一行代表一个input，每个input都是d+1维的，一共有d+1个inputs。X很显然是可逆的。
shatter的本质就是假设空间H对于X的所有情况的判断都是正确的，也就是说总能找到权重W，满足$XW=y, W=X^{-1}y$,因此所有的inputs都可以被shatter，因此$d_{vc} \geq d+1$

2. 在d维中，如果对于任何的d+2个inputs一定不能被shatter的话那么不等式就成立了。我们继续构造一个X，包含d+2个inputs，有d+1列，d+2行。由于某一列一定可以被另外d+1个线性表示，我们有

$$X_{d+2} = a_1X_1 + a_2X_2 + ... + a_{d+1}X_{d+1}$$

$a_1 > 0, a_2, a_3, ..., a_{d+1} < 0$
如果X1是正类，X2,X3,...,Xd均为负类，则存在W，得到如下表达式：

$$W^TX_{d+2} = a_1W^TX_1 + a_2W^TX_2 + ... + a_{d+1}W^TX_{d+1} > 0$$

那么X_{d+2}一定是正类，无法得到负类的情况了。所以d+2个inputs一定无法被shatter

综合1、2，我们有$d_{vc} = d + 1$

（四）、VC Dimension 的物理直观

上述公式中W就是features，即自由度，可以自由调节。VC Dimension代表了假设空间的分类能力，即反应了H的自由度，产生dichotomy的数量，也就是features的个数（近似）我们发现M和$_{vc}$是成正比的

（五）、解释VC Dimension

下面我们将深入探讨。首先，把VC Bound重述一下：

$$\mathbb{P}[BAD \iff |E_{in}(g) - E_{out}(g)| > \epsilon] \leq 4(2N)^{dvc}e^{-\frac{1}{8}\epsilon^2N} = \sigma$$

根据之前的泛化不等式，如果$|E_{in} - E_{out}| > \epsilon$,也就是说BAD的概率不大于$\sigma$,出现**good**的概率就不小于1-$\sigma$,那么我们对上述不等式进行重新推导：

$$with\ probability \geq 1-\sigma, GOOD: |E_{in}(g) - E_{out}(g)| \leq \epsilon$$

$$set\ \sigma = 4(2N)^{dvc} e^{-\frac{1}{8}\epsilon^2 N}$$

$$\frac{\sigma}{4(2N)^{dvc}} = e^{-\frac{1}{8}\epsilon^2 N}$$

$$ln(\frac{4(2N)^{dvc}}{\sigma}) = \frac{1}{8}\epsilon^2 N$$

$$\sqrt{\frac{8}{N} ln(\frac{4(2N)^{dvc}}{\sigma})} = \epsilon$$

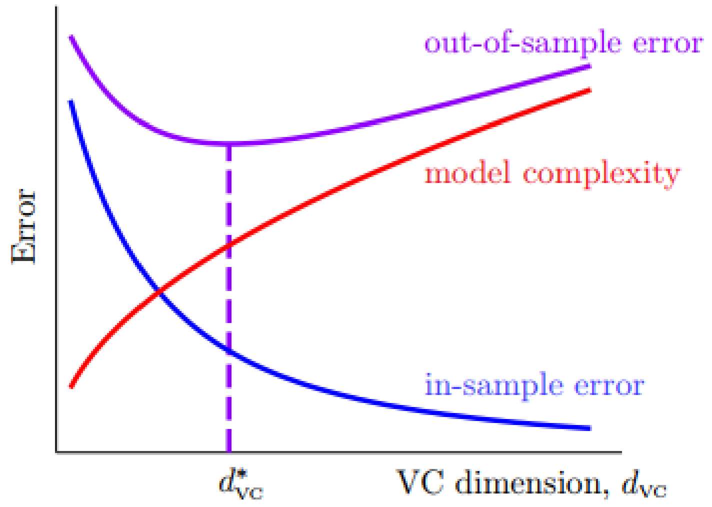
ϵ体现了假设空间H的泛化能力，ϵ越小，泛化能力越大

我们于是就有，$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} ln(\frac{4(2N)^{dvc}}{\sigma})}$

其中$\sqrt{\frac{8}{N} ln(\frac{4(2N)^{dvc}}{\sigma})} = \Omega(N, H, \sigma)$是模型复杂度。至此我们已经推导出泛化误差$E_{out}$的上界

with a high probability,

$$E_{\text{out}}(g) \le E_{\text{in}}(g) + \underbrace{\sqrt{\frac{8}{N}\ln\left(\frac{4(2N)^{d_{\text{VC}}}}{\delta}\right)}}_{\Omega(N,\mathcal{H},\delta)}$$



- $d_{\text{VC}} \uparrow$: $E_{\text{in}} \downarrow$ but $\Omega \uparrow$
- $d_{\text{VC}} \downarrow$: $\Omega \downarrow$ but $E_{\text{in}} \uparrow$
- best d^*_{VC} in the middle

powerful $\mathcal{H}$ not always good!

如图我们可以得到如下结论:

- d_{vc}越大，E_{in}越小，Ω越复杂
- d_{vc}越小，E_{in}越大，Ω越简单
- 随着d_{vc}增大，E_{out}会先减小再增大

我们应该选择合适的d_{vc}，选择的features个数要合适，不能太大也不能太小。
注：VC Bound是比较宽松的，但是如何把它收紧却并非易事。好在VC Bound对于所有的模型的宽松程度几乎都是一样的。所以不同模型之间还是可以横向比较。从而VC Bound宽松对于机器学习的可行性还是没有太大影响。

四、噪声和错误

（一）、噪声和概率目标

本节讨论如果D中存在Noise的时候VC Dimension的推导是否依然成立呢
Data Set中的Noise一般有三种情况：

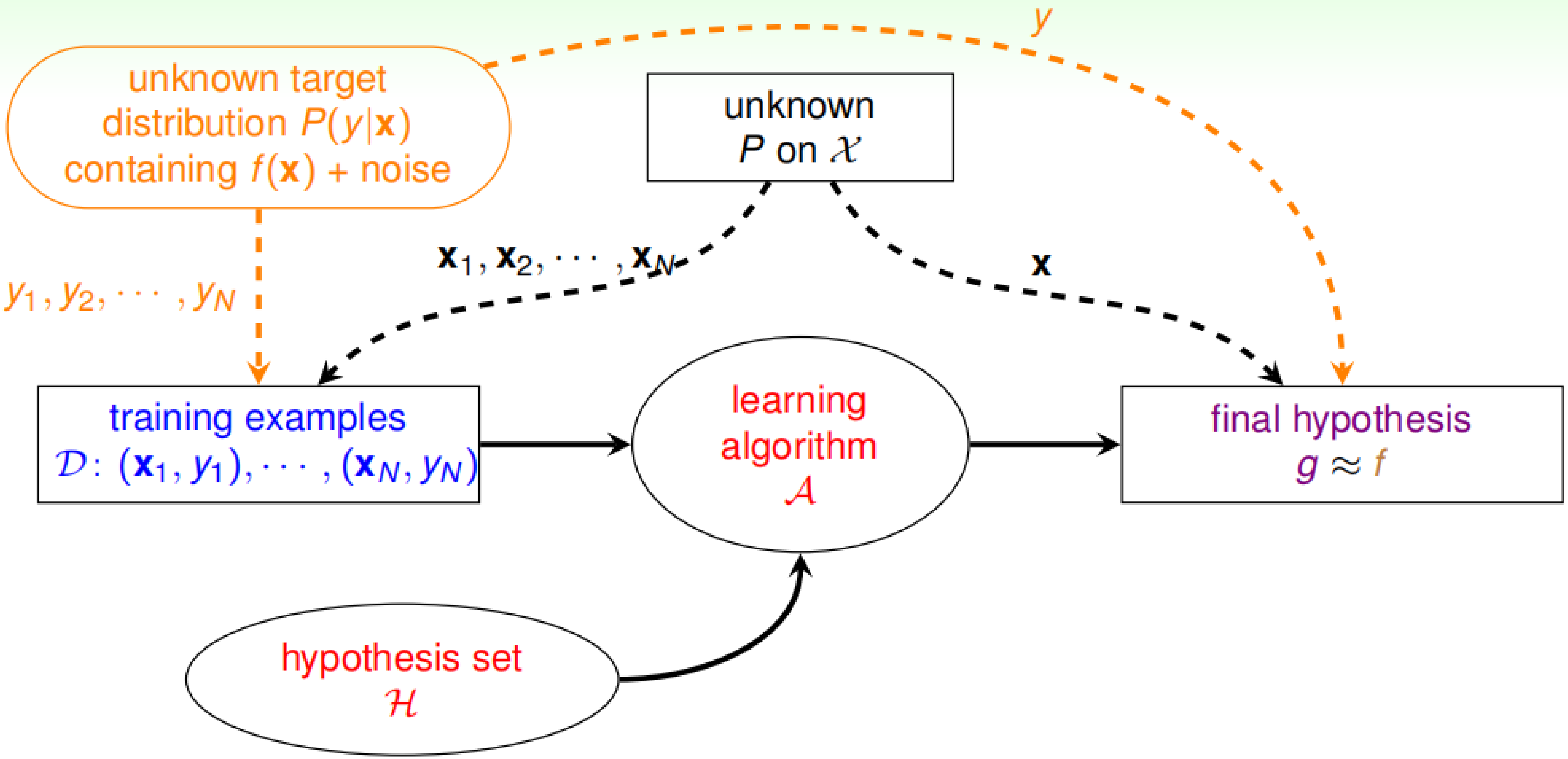
- 人为因素导致的误分类，假正类，假反类
- 同样特征的样本被分成不同的类
- 样本的特征被错误记录使用

类比我们之前Pocket Algorithm中出现noise的情况，我们发现这个时候还是可以解决。
之前数据集是确定的没有Noise的时候，我们成为**Deterministic**。现在有Noise了，也就是说在某点处不再是确定分布，而应该是概率分布了，也就是说对于每一个(x,y)而言出现的概率是$P(y|x)$。我们可以证明，如果D是按照$P(y|x)$概率分布而且是i.i.d的，那么之前证明机器可以学习的方法依然奏效。VC Dimension有限仍然可以推断$E_{in} \approx E_{out}$

$$VC\ holds\ for\ \underbrace{x \overset{i.i.d}{\sim} P(x),\ y \overset{i.i.d}{\sim} P(y|x)}_{(x,y) \overset{i.i.d}{\sim} P(x,y)}$$

$P(y|x)$称之为目标分布 **Target Distribution**，实际上对于**Deterministic Target**我们仍然可以看作是特殊情况。
在引入noise的情况下，我们就有了新的学习流程图：

The New Learning Flow



VC still works, **pocket algorithm explained :-)**

（二）、 误差测量

机器学习考虑的问题是g和f到底有多相近，我们一直使用E_{out}进行误差的估计。现在我们考虑一般的错误量度：

Error Measure

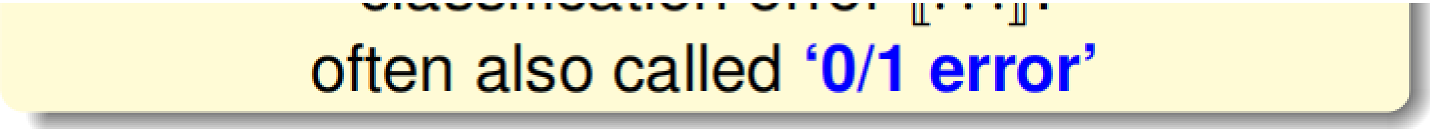
final hypothesis
$g \approx f$

- how well? previously, considered out-of-sample measure

$$E_{\text{out}}(g) = \underset{\mathbf{x} \sim P}{\mathcal{E}} [\![g(\mathbf{x}) \neq f(\mathbf{x})]\!]$$

- more generally, **error measure** $E(g, f)$
- naturally considered
  - out-of-sample: averaged over unknown $\mathbf{x}$
  - pointwise: evaluated on one $\mathbf{x}$
  - classification: $[\![$prediction $\neq$ target$]\!]$

classification error $[\![\ldots]\!]$:



分为三种：

- out-of-sample: 样本外的位置数据
- pointwise：对每个数据点进行测试

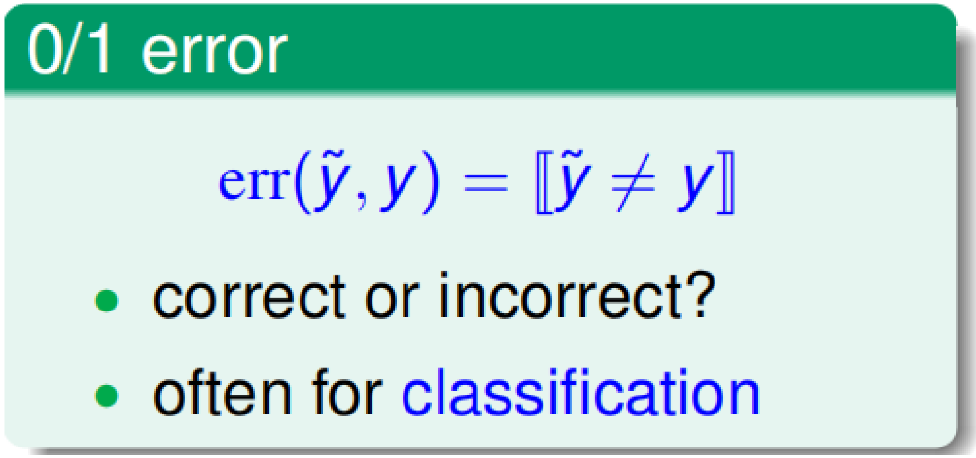
  PointWise Error:

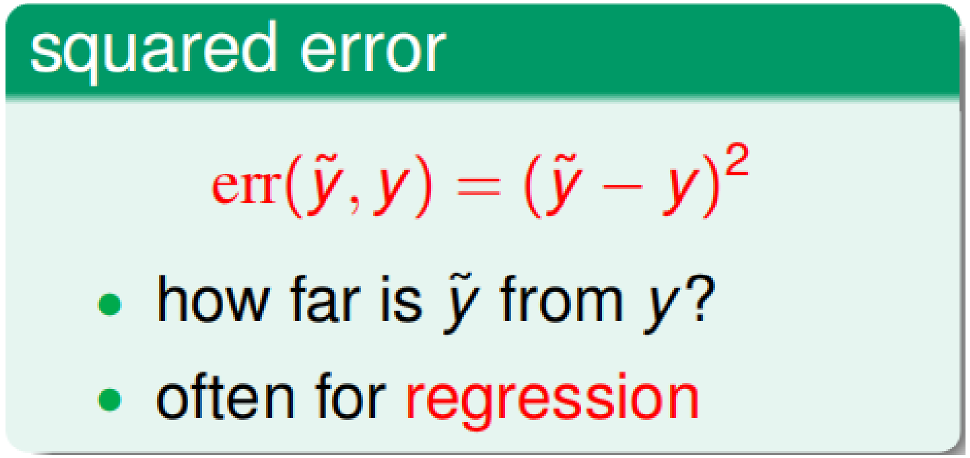
  $E_{in}(g) = \frac{1}{N}\sum_{n=1}^{N} err(g_{(}x_n), f(x_n))$

  $E_{out}(g) = \underset{x\sim P}{\epsilon} err(g(x), f(x))$

  这是机器学习中最简单最常用的错误衡量方式。一般分成两类

  1. 0/1 error——classification
  2. squared error——regression





- classification：看prediction和target是否一致，classification error通常称为0/1 error

理想的Mini-Target由P(y|x)和err共同决定，通过下面例子中我们可以看出：

Ideal Mini-Target

interplay between noise and error:

$P(y|\mathbf{x})$ and err define ideal mini-target $f(\mathbf{x})$

$$P(y=1|\mathbf{x}) = 0.2, P(y=2|\mathbf{x}) = 0.7, P(y=3|\mathbf{x}) = 0.1$$

$$\mathrm{err}(\tilde{y}, y) = [\![\tilde{y} \neq y]\!]$$

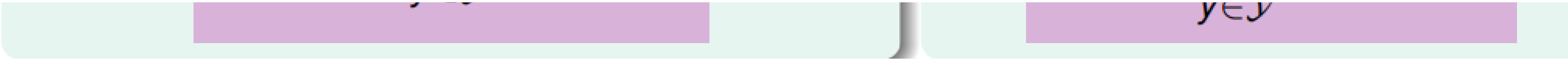
$$\tilde{y} = \begin{cases} 1 & \text{avg. err } 0.8 \\ 2 & \text{avg. err } 0.3 (*) \\ 3 & \text{avg. err } 0.9 \\ 1.9 & \text{avg. err } 1.0 \text{ (really? :-))} \end{cases}$$

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

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

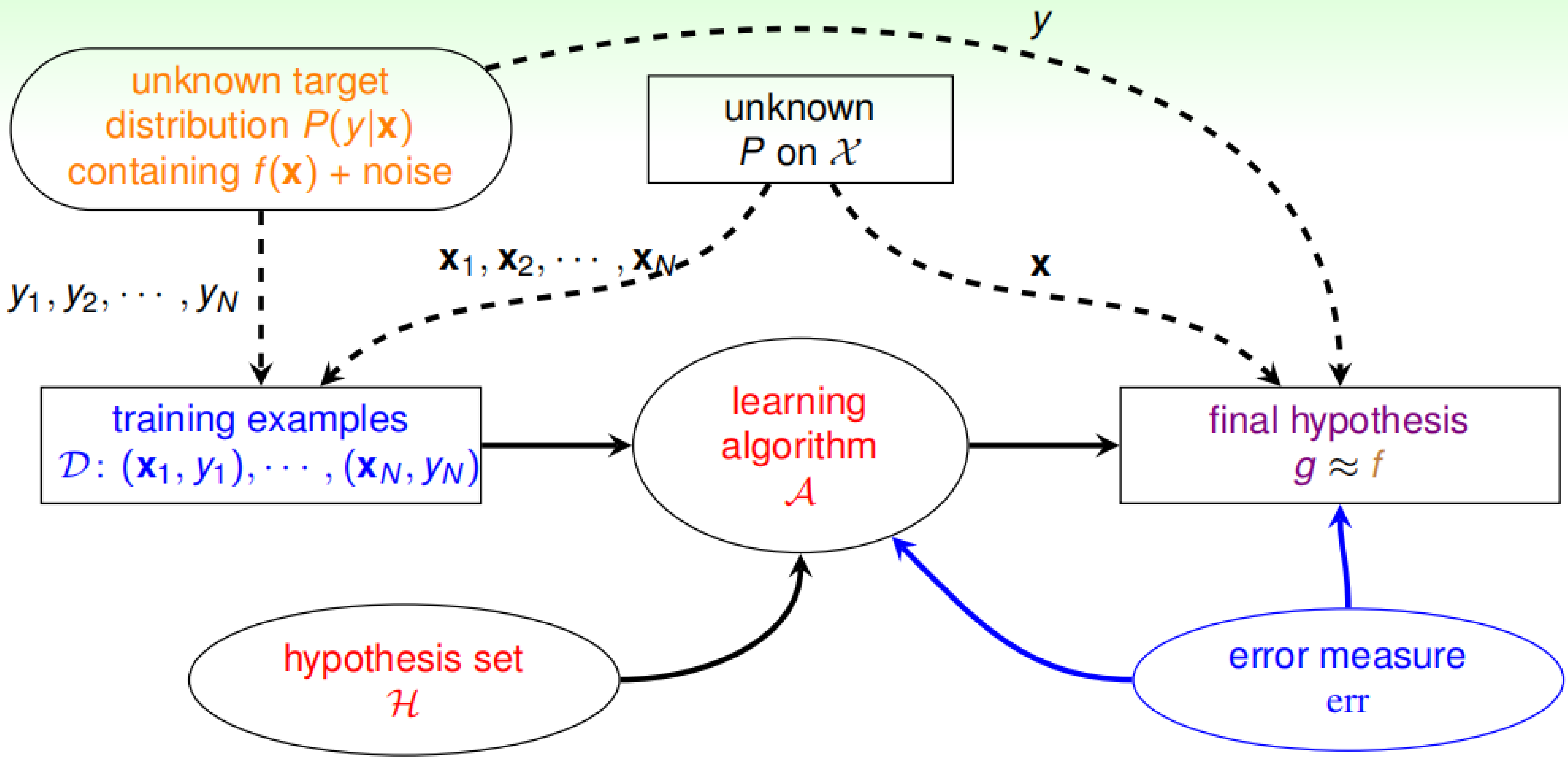
$$\begin{cases} 1 & \text{avg. err } 1.1 \\ 2 & \text{avg. err } 0.3 \\ 3 & \text{avg. err } 1.5 \\ 1.9 & \text{avg. err } 0.29 (*) \end{cases}$$

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



ideal mini-target的计算方法不一样。
有了错误的量度我们就会知道当前的矩g是好还是不好。并会让演算法不断修正，得到更好的g，让g和f靠得更近。因此，引入error measure之后，学习流程图如下所示：

Learning Flow with Error Measure



extended VC theory/'philosophy'

works for most $\mathcal{H}$ and err

（三）、算法误差衡量

1. Error有两种：
   false accept: 负类当成正类
   false reject：正类当成负类
   根据不同的问题，这两个error的权重应该有所不同。例如超市优惠和CIA指纹识别就有相反的模型。
2. 机器学习算法A对应的cost function error的估计有很多种方法，但是真实的err往往难以计算，我们一般可以采用plausible或者friendly两种方法来计算。

Take-home Message for Now

err is **application/user-dependent**

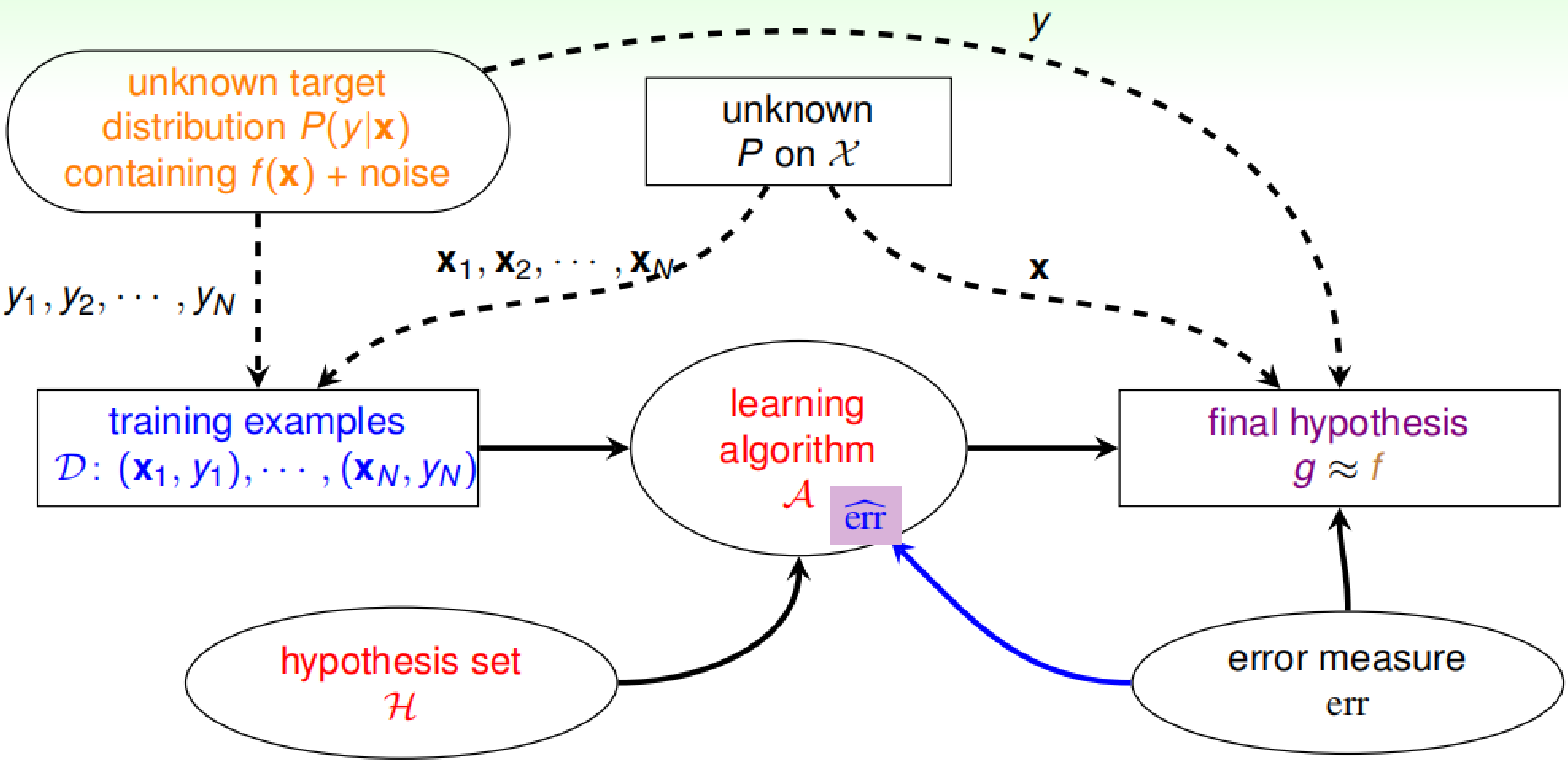
Algorithmic Error Measures $\widehat{\text{err}}$

- true: just err
- plausible:
  - 0/1: minimum 'flipping noise'—NP-hard to optimize, **remember? :-)**
  - squared: minimum Gaussian noise
- friendly: easy to optimize for $\mathcal{A}$
  - closed-form solution
  - convex objective function

$\widehat{\text{err}}$: more in next lectures

引入了algorithmic error measure之后，我们有新的学习流程图:

Learning Flow with Algorithmic Error Measure



err: application goal;

$\widehat{\text{err}}$: a key part of many $\mathcal{A}$

（四）、带权分类

机器学习中的cost function来自于这些error，也就是算法里面的迭代的目标函数，通过优化来使得Error(E_{in})不断变小，我们可以采取**virtual copying**的方法，就是假设出现的某一个(x,y)有weight个然后转化E_{in}^{w} 为$E_{in}^{0/1}$
那么，**weighted virtual copying**包含如下的东西：

- weighted PLA：
  随机找值为-1的错误点weight次
- weighted pocket replacement
  如果w_{t+1}表现得更好，就用它代替。