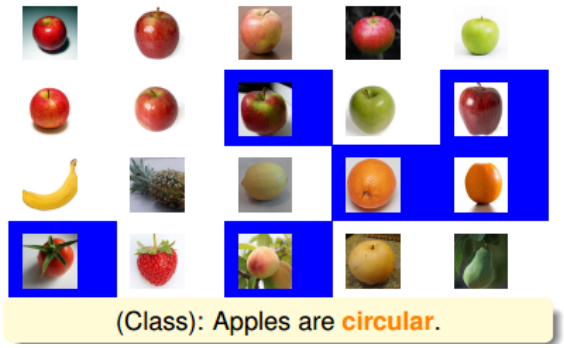
Adaptive Boosting

1.Motivation of Boosting

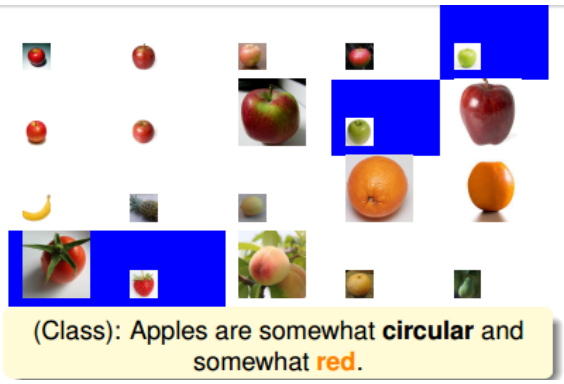
20张图片包括它的标签都是已知的。
根据苹果是圆形的这个判断，大部分苹果能被识别，但是也存在错误。



蓝色区域代表分类错误。

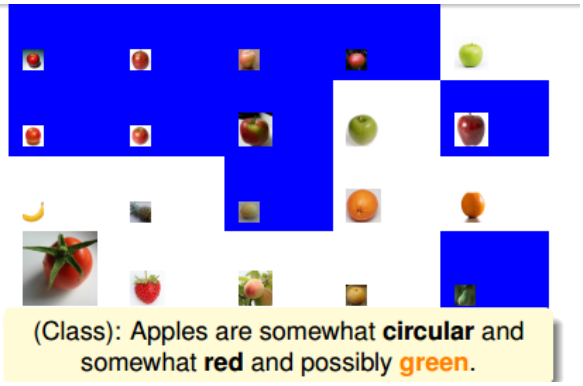
把蓝色区域（分类错误的图片）放大，分类正确的图片缩小，这样在接下来的分类中就会更加注重这些错误样本。

根据苹果是红色这个信息，得到的结果（蓝色区域代表分类错误）：



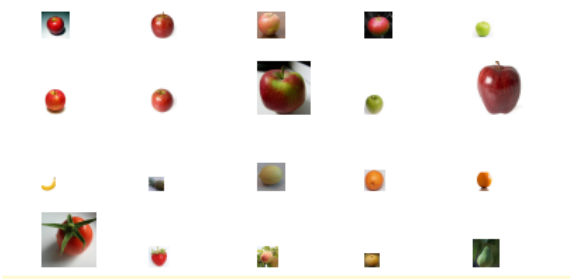
然后将分类错误的样本放大化，其它正确的样本缩小化。

苹果也可能是绿色的

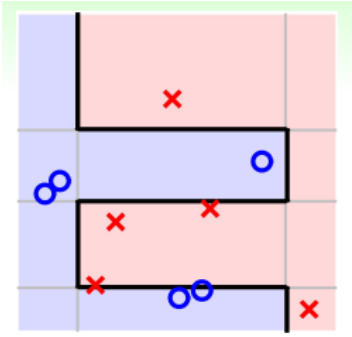


蓝色区域的图片代表分类错误
把这些分类错误的样本放大化，其它正确的样本缩小化，在下一轮判断继续对其修正。

上面有梗的才是苹果



苹果被定义为：圆的，红色的，也可能是绿色的，上面有梗。从一个一个的推导过程中，我们似乎得到一个较为准确的苹果的定义。虽然可能不是非常准确，但是要比单一的条件要好得多。
简单的hypotheses g_t，将所有g_t融合，得到很好的预测模型G。



- students: simple hypotheses g_t (like vertical/horizontal lines)
- (Class): sophisticated hypothesis G (like black curve)
- Teacher: a tactic learning algorithm that directs the students to focus on key examples

不同的判断代表不同的hypotheses g_t
最终得到的结果定义就代表hypothesis G
演算法将注意力集中到错误样本，从而得到更好的定义。

2.Diversity by Re-weighting

Bagging的核心是bootstrapping，通过对原始数据集D不断进行bootstrap的抽样动作，得到与D类似的数据集$\hat{D}_t$，每组$\hat{D}_t$都能得到相应的g_t，从而进行aggregation的操作。
假如包含四个样本的D经过bootstrap，得到新的$\hat{D}_t$如下：

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)\}$$

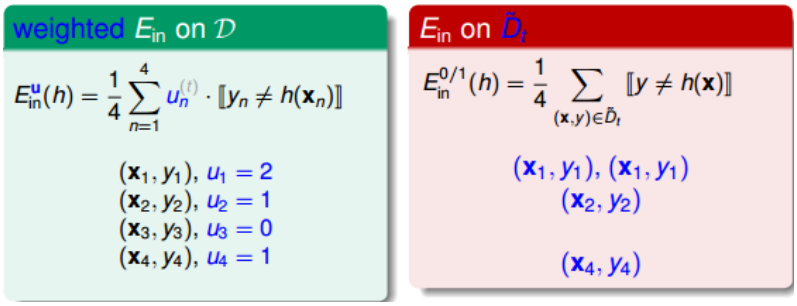
$$\xrightarrow{\text{bootstrap}} \tilde{\mathcal{D}}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_4, y_4)\}$$

那么，对于新的$\hat{D}t$，把它交给base algorithm，找出Ein最小时对应的g_t，如下图右边所示。

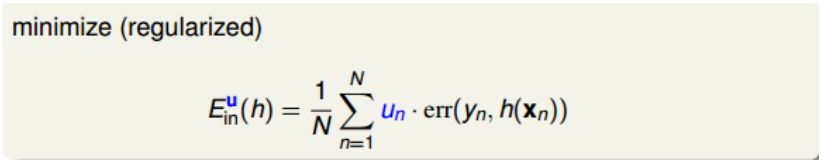
$$E_{in}^{0/1}(h) = \frac{1}{4}\sum_{n=1}^{4}[y \neq h(x)]$$

由于$\hat{D}_t$完全是D经过bootstrap得到的，其中样本(x_1, y_1)出现2次，(x_2, y_2)出现1次，(x_3, y_3)出现0次，(x_4, y_4)出现1次。引入一个参数u_i来表示原D中第i个样本在$\hat{D}_t$中出现的次数，如下图左边所示。

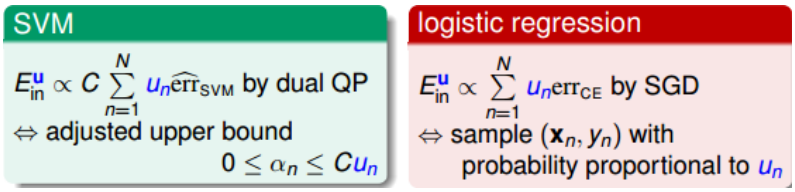
$$E_{in}^{u}(h) = \frac{1}{4}\sum_{n=1}^{4} u_n^{(t)} \cdot [y_n \neq h(x)]$$



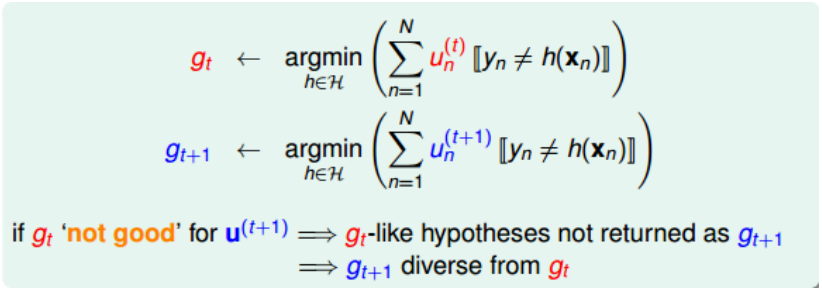
参数u相当于是权重因子，当$\hat{D}_t$中第i个样本出现的次数越多的时候，那么对应的u_i越大，表示在error function中对该样本的惩罚越多。



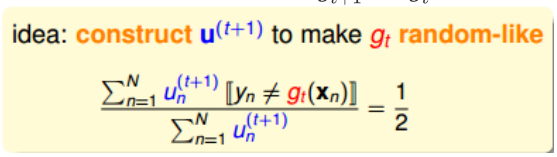
在logistic regression中，同样可以对每个犯错误的样本乘以相应的u_n，作为惩罚因子。u_n表示该错误点出现的次数，u_n越大，则对应的惩罚因子越大，则在最小化error时就应该更加重视这些点。



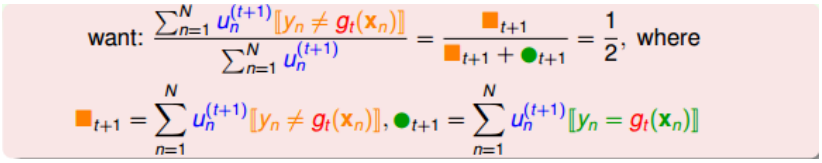
g_t越不一样，其aggregation的效果越好



利用g_t在使用$u_n^{(t+1)}$的时候表现很差的条件，越差越好。
这样的做法就能最大限度地保证g_{t+1}会与g_t有较大的差异性。



分式中分子可以看成g_t作用下犯错误的点，而分母可以看成犯错的点和没有犯错误的点的集合，即所有样本点。
其中犯错误的点和没有犯错误的点分别用橘色方块和绿色圆圈表示：



在g_t作用下，让犯错的$u_n^{(t+1)}$数量和没有犯错的$u_n^{(t+1)}$数量一致（包含权重u_n^{t+1}）就可以使分式结果为）0.5。
一种简单的方法就是利用放大和缩小的思想（本节课开始引入识别苹果的例子中提到的放大图片和缩小图片就是这个目的），将犯错误的u_n^t和没有犯错误的u_n^t做相应的乘积操作，使得二者值变成相等。

或者利用犯错的比例来做。
一般求解方式是令犯错率为ϵ_t，在计算$u_n^{(t+1)}$的时候，u_n^t分别乘以$(1-\epsilon_t)$和ϵ_t。

- need: $\underbrace{(\text{total } u_n^{(t+1)} \text{ of incorrect})}_{\blacksquare_{t+1}} = \underbrace{(\text{total } u_n^{(t+1)} \text{ of correct})}_{\bullet_{t+1}}$
- one possibility by **re-scaling (multiplying) weights**, if

(total $u_n^{(t)}$ of incorrect) = 1126 ; (weighted incorrect rate) = $\frac{1126}{7337}$	(total $u_n^{(t)}$ of correct) = 6211 ; (weighted correct rate) = $\frac{6211}{7337}$
incorrect: $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 6211$	correct: $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 1126$

'optimal' re-weighting under weighted incorrect rate ϵ_t:
multiply incorrect $\propto (1-\epsilon_t)$; multiply correct $\propto \epsilon_t$

3.Adaptive Boosting Algorithm

新的尺度因子：

$$\diamond t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$$

对错的u_n^t，将它乘以$\diamond t$；对于正确的u_n^t，将它除以$\diamond t$。
这种操作跟之前介绍的分别乘以$(1-\epsilon_t)$和ϵ_t的效果是一样的。

如果$\epsilon_t \leq \frac{1}{2}$，得到$\diamond t \geq 1$，那么接下来错误的$u_n^t$与$\diamond t$的乘积就相当于把错误点放大了，而正确的$u_n^t$与$\diamond t$的相除就相当于把正确点缩小了。
也就是能够将注意力更多地放在犯错误的点上。
通过这种scaling-up incorrect的操作，能够保证得到不同于g_t的g_{t+1}。

define scaling factor $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

$$\begin{array}{rcl} \text{incorrect} & \leftarrow & \text{incorrect} \cdot \blacklozenge_t \\ \text{correct} & \leftarrow & \text{correct} / \blacklozenge_t \end{array}$$

- **equivalent** to optimal re-weighting
- $\blacklozenge_t \geq 1$ iff $\epsilon_t \leq \frac{1}{2}$
  —physical meaning: **scale up incorrect**; **scale down correct**
  —like what Teacher does

得到一个初步的演算法。其核心步骤是每次迭代时，利用$\diamond t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$把$u_t$更新为$u_{t+1}$。
具体迭代步骤如下：

$\mathbf{u}^{(1)} = ?$
for $t = 1, 2, \ldots, T$
1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$,
   where $\mathcal{A}$ tries to minimize $\mathbf{u}^{(t)}$-weighted 0/1 error
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$,
   where ϵ_t = weighted error (incorrect) rate of g_t

return $G(\mathbf{x}) = ?$

一般来说，为了保证第一次E_{in}最小的话，设$u^{(1)} = \frac{1}{N}$。
对所有的g(t)进行linear或者non-linear组合来得到G(t)。

- want g_1 'best' for E_{in}: $u_n^{(1)} = \frac{1}{N}$
- $G(\mathbf{x})$:
  - uniform? but g_2 very bad for E_{in} (why? :-))
  - linear, non-linear? as you wish

将所有的g(t)进行linear组合。
方法是计算$g(t)$的同时，就能计算得到其线性组合系数α_t，即aggregate linearly on the fly。
这种算法使最终求得g_{t+1}的时候，所有g_t的线性组合系数α也求得了，不用再重新计算α了。

这种Linear Aggregation on the Fly算法流程为：

$\mathbf{u}^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$
for $t = 1, 2, \ldots, T$
1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$, where ...
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, where ...
3. compute $\alpha_t = \ln(\blacklozenge_t)$

return $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$

α_t与ϵ_t是相关的：ϵ_t越小，对应的α_t应该越大，ϵ_t越大，对应的α_t应该越小。
又因为$\diamond t$与ϵ_t是正相关的，所以，α_t应该是$\diamond t$的单调函数。
我们构造α_t为：

$$\alpha_t = ln(\diamond t)$$

α_t这样取值是有物理意义的
例如当$\epsilon_t = \frac{1}{2}$时，error很大，跟随机过程没什么两样，此时对应的$\diamond t = 1$，$\alpha_t = 0$，即此g_t对G没有什么贡献，权重应该设为零。
而当$\epsilon_t = 0$时，没有error，表示该g_t预测非常准，此时对应的$\diamond t = \infty$，$\alpha_t = \infty$，即此g_t对G贡献非常大，权重应该设为无穷大。

- wish: large α_t for 'good' $g_t \Longleftarrow \alpha_t = \text{monotonic}(\blacklozenge_t)$
- will take $\alpha_t = \ln(\blacklozenge_t)$
  - $\epsilon_t = \frac{1}{2} \Longrightarrow \blacklozenge_t = 1 \Longrightarrow \alpha_t = 0$ (bad g_t zero weight)
  - $\epsilon_t = 0 \Longrightarrow \blacklozenge_t = \infty \Longrightarrow \alpha_t = \infty$ (super g_t superior weight)

这种算法被称为Adaptive Boosting。它由三部分构成：base learning algorithm A，re-weighting factor $\diamond t$和linear aggregation α_t。这三部分分别对应于我们在本节课开始介绍的例子中的Student，Teacher和Class。

Adaptive Boosting = weak base learning algorithm $\mathcal{A}$ (Student)
+ optimal re-weighting factor $\blacklozenge_t$ (Teacher)
+ 'magic' linear aggregation α_t (Class)

综上所述，完整的adaptive boosting（AdaBoost）Algorithm流程如下：

$\mathbf{u}^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$
for $t = 1, 2, \ldots, T$
1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$,
   where $\mathcal{A}$ tries to minimize $\mathbf{u}^{(t)}$-weighted 0/1 error
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by
   $[\![y_n \neq g_t(\mathbf{x}_n)]\!]$ (incorrect examples): $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot \blacklozenge_t$
   $[\![y_n = g_t(\mathbf{x}_n)]\!]$ (correct examples): $u_n^{(t+1)} \leftarrow u_n^{(t)} / \blacklozenge_t$
   where $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ and $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)}}$
3. compute $\alpha_t = \ln(\blacklozenge_t)$

return $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$

从VC bound角度来看，AdaBoost算法理论上满足：

- From VC bound

$$E_{out}(G) \leq E_{in}(G) + O\left(\sqrt{\underbrace{O(d_{VC}(\mathcal{H}) \cdot T \log T)}_{d_{VC} \text{ of all possible } G} \cdot \frac{\log N}{N}}\right)$$

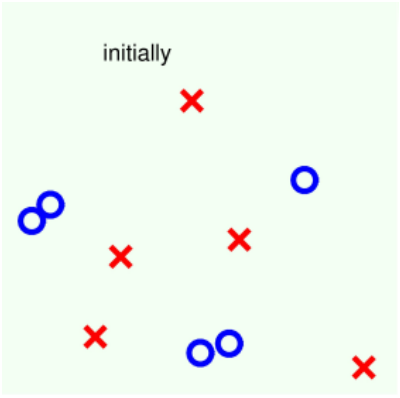
- **first term can be small**:
  $E_{in}(G) = 0$ after $T = O(\log N)$ iterations if $\epsilon_t \leq \epsilon < \frac{1}{2}$ always
- **second term can be small**:
  overall d_{VC} grows "slowly" with T

只要每次的$\epsilon_t \leq \epsilon < \frac{1}{2}$，即所选择的矩g比乱猜的表现好一点点，那么经过每次迭代之后，矩g的表现都会比原来更好一些，逐渐变强，最终得到$E_{in} = 0$且E_{out}很小。

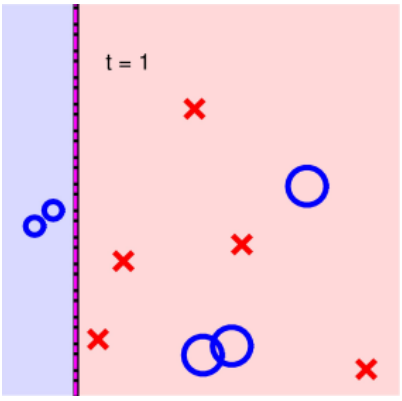
boosting view of AdaBoost:
if $\mathcal{A}$ is weak but always **slightly better than random** ($\epsilon_t \leq \epsilon < \frac{1}{2}$),
then (AdaBoost+$\mathcal{A}$) can be strong ($E_{\text{in}} = 0$ and E_{out} small)

4.Adaptive Boosting in Action

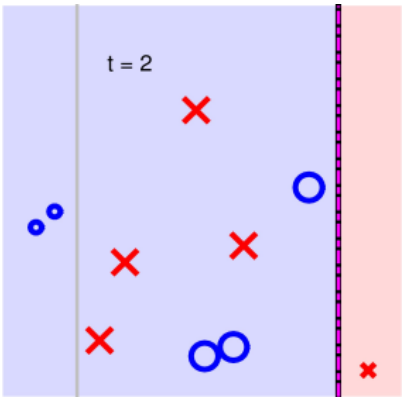
eg:AdaBoost使用decision stump解决实际问题:
二维平面上分布一些正负样本点，利用decision stump来做切割。



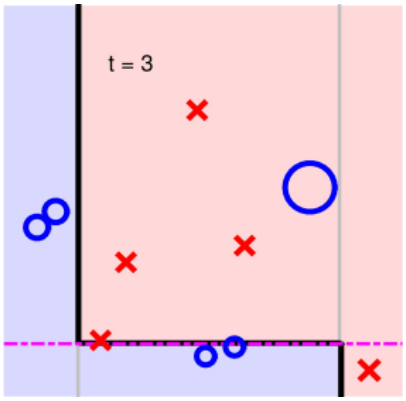
第一步:



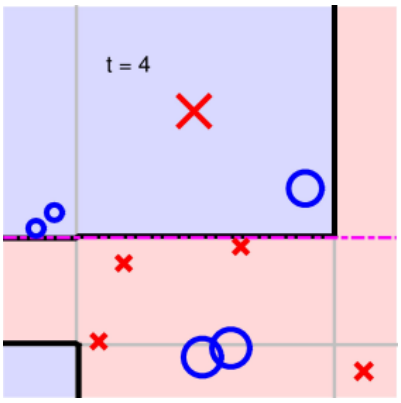
第二步:



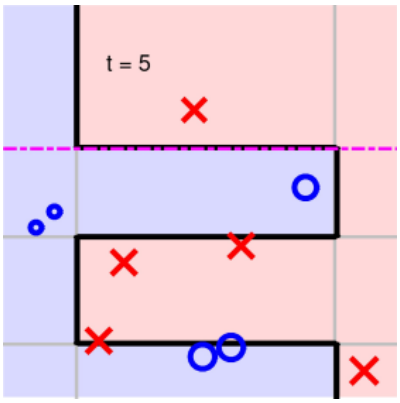
第三步:



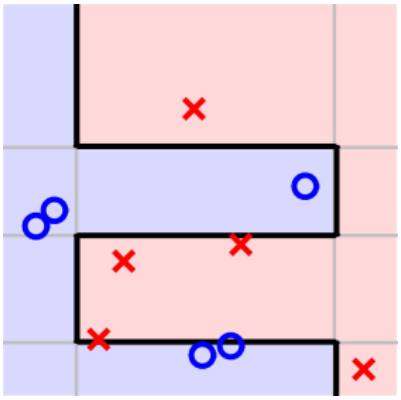
第四步：



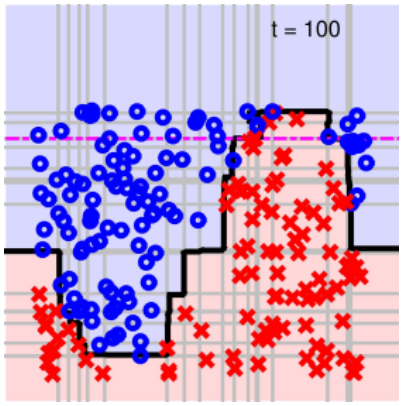
第五步：



经过5次迭代之后，所有的正负点已经被完全分开了，则最终得到的分类线为：



对于一个相对比较复杂的数据集，如下图所示。它的分界线从视觉上看应该是一个sin波的形式。如果我们再使用AdaBoost算法，通过decision stump来做切割。在迭代切割100次后，得到的分界线如下所示。



AdaBoost-Stump这种非线性模型得到的分界线对正负样本有较好的分离效果。

5.Summary

主要介绍了Adaptive Boosting。
Boosting的思想，即把许多“弱弱”的hypotheses合并起来，变成很强的预测模型。
算法如何实现，关键在于每次迭代时，给予样本不同的系数u，宗旨是放大错误样本，缩小正确样本，得到不同的小矩g。
并且在每次迭代时根据错误ϵ值的大小，给予不同g_t不同的权重。
最终由不同的g_t进行组合得到整体的预测模型G。
实际证明，Adaptive Boosting能够得到有效的预测模型。