

发件人: Yulin Wu yw4923@nyu.edu
 主题:
 日期: 2020年9月9日 下午5:34
 收件人:



Adaptive boosting

2020年9月6日 星期日
 上午2:50

Motivation:

给出10张苹果的图片, 10张不是苹果的图片, 希望能教会小学生们如何辨别苹果。

老师: 一开始, 这20张图片都同样重要, michael你来说说看如何辨别苹果呢?

michael说: 苹果是circular的。根据这个circular分类器, 有些图片被分对了, 有些被分错了。(如false positive, false negative)

★ 此时, 全班的小学生知道了苹果是circular的。

老师: 现在将被分错的图片都放大, 分对的图片缩小。那么现在, nina你说说看如何辨别苹果呢?



Tina: 苹果是red的。根据这个red的分类器, 有些图片被分对了, 有些被分错了。

★ 此时, 全班的小学生知道了苹果是circular的, 也是red的。

老师: 现在将被分错的图片都放大, 分对的图片缩小。那么现在, Joey你说说看如何辨别苹果呢?



(Class): Apples are somewhat circular and somewhat red.

Joey: 苹果可能是green的。

★ 此时, 全班的小学生知道了苹果是circular的, 也是red的, 也可能是green的。

Boosting:

上述的过程中, 一个小学生的判断就是一个分类器。老师的动作就是: 每训练一个分类器, 就将其犯错误的数据权重加大, 来训练下一个分类器。而全班同学学到的就是这些分类器的blending, 又因为这些分类器的权重相当于bootstrapping, 所以全班同学学到的是这些分类器的bagging。

boosting的性质:

1. 能保证分类器的diversity。
直观来讲, g_2 是在 g_1 做不好的数据集上做得好, 那么他们肯定会挺不一样的。
2. e_{in} 和 e_{out} 相差不多。
3. 效率高, 能在 $\log(N)$ 的时间内将 e_{in} 训练接近0。

性质1的严格证明:

$$g_t \leftarrow \underset{h \in H}{\operatorname{argmin}} \left(\sum_{n=1}^N u_n^{(t-1)} [y_n \neq h(\mathbf{x}_n)] \right)$$

$$g_{t+1} \leftarrow \underset{h \in H}{\operatorname{argmin}} \left(\sum_{n=1}^N u_n^{(t)} [y_n \neq h(\mathbf{x}_n)] \right)$$

if g_t 'not good' for $u^{(t+1)} \Rightarrow g_t$ -like hypotheses not returned as g_{t+1}

要保证 g_t 在 g_{t+1} 的权重数据集上表现得不好, 即我要令 $\frac{\sum_{n=1}^N u_n^{(t+1)} [y_n \neq g_t(\mathbf{x}_n)]}{\sum_{n=1}^N u_n^{(t+1)}} = \frac{1}{2}$

Adaptive Boosting
 Bootstrapping as Re-weighting Process

$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$

Iteration t :
 $\mathcal{D}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$

weight E_{in} on \mathcal{D} :
 $E_{in}^{(t)}(h) = \frac{1}{4} \sum_{n=1}^N u_n^{(t-1)} [y_n \neq h(\mathbf{x}_n)]$

E_{in} on \mathcal{D}_t :
 $E_{in}^{(t)}(h) = \frac{1}{4} \sum_{n=1}^N [y_n \neq h(\mathbf{x}_n)]$

$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)$
 $(\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)$
 (\mathbf{x}_5, y_5)

Adaptive Boosting
 Theoretical Guarantee of AdaBoost

From VC bound

$$E_{out}(G) \leq E_{in}(G) + O\left(\sqrt{\frac{O(d_{VC}(H)) \log(N)}{n}}\right)$$

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



'Optimal' Re-weighting

want: $\frac{\sum_{n=1}^N u_n^{(t+1)} [y_n \neq g_t(\mathbf{x}_n)]}{\sum_{n=1}^N u_n^{(t+1)}} = \frac{1}{2}$, where

$\bullet_{n=1}^N u_n^{(t+1)} [y_n \neq g_t(\mathbf{x}_n)] + \bullet_{n=1}^N u_n^{(t+1)} [y_n = g_t(\mathbf{x}_n)]$

need: (total $u_n^{(t+1)}$ of incorrect) = (total $u_n^{(t+1)}$ of correct)

one possibility by re-scaling (multiplying) weights, if

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

则我要如何设计 $u_n^{(t+1)}$, 使得橙色方块 == 绿色圆点?

一个方法是: 将犯错的数据和正确的数据的比例拉到1:1

将 g_t 预测错误的数据集复制 * g_t 预测正确的数据集的比例 (这是一个数字) 倍
 将 g_t 预测正确的数据集复制 * g_t 预测错误的数据集的比例 (这是一个数字) 倍

incorrect: $u_n^{(t+1)} \leftarrow u_n^{(t)} - 6211$ correct: $u_n^{(t+1)} \leftarrow u_n^{(t)} + 1120$

如果让 ϵ_t 表示犯错的比例那么

'optimal' re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^N u_n^{(t)} [y_n \neq g(x_n)]}{\sum_{n=1}^N u_n^{(t)}}$

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

根据这个思想, 我们定义菱形 Φ 正比例于 $1 - \epsilon_t$, 反比例于 ϵ_t .

Scaling Factor

'optimal' re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^N u_n^{(t)} [y_n \neq g(x_n)]}{\sum_{n=1}^N u_n^{(t)}}$

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

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

incorrect \leftarrow incorrect $\cdot \Phi_t$
correct \leftarrow correct $/ \Phi_t$

• equivalent to optimal re-weighting

• $\Phi_t \geq 1$ iff $\epsilon_t \leq \frac{1}{2}$

—physical meaning: **scale up incorrect; scale down correct**

—like what Teacher does

看到了吗! 发现能够保证 g_t 和 g_{t+1} 很不同的条件就是: **scale up incorrect, scale down correct**.

就是刚才老师做的事情, 证明了boosting的性质1.

boosting algorithm 流程:

A Preliminary Algorithm

$u^{(1)} = ?$

for $t = 1, 2, \dots, T$

① obtain g_t by $\mathcal{A}(\mathcal{D}, u^{(t)})$,
where \mathcal{A} tries to minimize $u^{(t)}$ -weighted 0/1 error

② update $u^{(t)}$ to $u^{(t+1)}$ by $\Phi_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$,
where ϵ_t = weighted error (incorrect) rate of g_t .

return $G(x) = ?$

• want g_t 'best' for E_{in} : $u_n^{(1)} = \frac{1}{N}$

• $G(x)$:

• uniform? but g_t very bad for E_{in} (why? :-))

• linear, non-linear? *as you wish*

next: a special algorithm to aggregate
linearly on the fly with theoretical guarantee

Adaptive Boosting (AdaBoost) Algorithm

$u^{(1)} = [\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}]$ ✓

for $t = 1, 2, \dots, T$

① obtain g_t by $\mathcal{A}(\mathcal{D}, u^{(t)})$,
where \mathcal{A} tries to minimize $u^{(t)}$ -weighted 0/1 error

② update $u^{(t)}$ to $u^{(t+1)}$ by

$[y_n \neq g_t(x_n)]$ (incorrect examples): $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot \Phi_t$

$[y_n = g_t(x_n)]$ (correct examples): $u_n^{(t+1)} \leftarrow u_n^{(t)} / \Phi_t$

where $\Phi_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ and $\epsilon_t = \frac{\sum_{n=1}^N u_n^{(t)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^N u_n^{(t)}}$

③ compute $\alpha_t = \ln(\Phi_t)$

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

训练 g_t

scale up incorrect, scale down correct

训练 g_{t+1}

scale up incorrect, scale down correct

最后将 g blending

blending 的 weight 怎么解决呢?

如果你表现好, 我才将你放入我最终的blending.

菱形越大表示 g 这个分类器表现得越好.

• wish: large α_t for 'good' $g_t \Leftarrow \alpha_t = \text{monotonic}(\Phi_t)$

• will take $\alpha_t = \ln(\Phi_t)$

• $\epsilon_t = \frac{1}{2} \Rightarrow \Phi_t = 1 \Rightarrow \alpha_t = 0$ (bad g_t , zero weight)

• $\epsilon_t = 0 \Rightarrow \Phi_t = \infty \Rightarrow \alpha_t = \infty$ (super g_t , superior weight)

Adaptive Boosting = weak base learning algorithm \mathcal{A} (Student)

+ optimal re-weighting factor Φ_t (Teacher)

+ 'magic' linear aggregation α_t (Class)