Adaboost的簡介與相關理論的推導

Cheng-Yu Lin 林鉦育, (aha)
ntuaha@gmail.com

December 26, 2012

目錄

Abstract

18 Candidates for the Top 10 Algorithms in Data Mining[]裡面在Bagging and Boosting的候選演算法 AdaBoost，"A decision-theoretic generalization of on-line learning and an application to boosting[]" (1997)。

Chapter 1

Introduction

作者用了非常巧妙的Weighted Majority Algorithm[]去更新它對於各個弱模型的權重，所以能夠有效地控制模型在組合的時候誤差能夠控制在某個定量之下，也因此這個方法可以結合使用到多種一般模型之下去運作，比方說賭博, 多結果預測, 重複實驗, 以及預測多維空間中的點這類問題的衆多弱模型，都可以利用這個演算法將其組合成一個强模型．但前提是你要有多個比隨機亂猜還要好的模型．否則都只是來亂的．這個方法還有以下兩個好處：

- 不需要任何跟要組合的弱模型相關的資訊
- 不只能解決離散的答案的猜測問題（如猜下一次骰子會出幾點），還能解決預測數線上某一個想要的點的問題．

但是，前面所說明的是一個利用多模型組合的方式，在本文稱之為避險演算法 *(Hedge Algorithm)*，而真正的*Adaboost*則是不斷地利用訓練資料(training data)用一個特殊的手法把比隨機還要好的預測模型(誤差率小於1/2)的誤差更進一步地降低．

因此在本文中不論是避險演算法還是Adaboost演算法的設計理念都非常值得去花時間去體會與學習．

Chapter 2

Hedge Algorithm

2.1 Scenario for Hedge Algorithm

讓我們先來看看以下適用於避險演算法幾個問題:

2.1.1 Gambler (賭徒問題)

假設你是一個賭徒，你已經有５個賭海明（冥）燈,你已經想好每一次下注都是固定金額，只是你對於這五個賭海明燈會根據他們先前的表現來決定下注的比率，假設說你已經有一個鐵定的明（冥）燈，那當然你就100%跟著下注（或者下反注），但事實上這是不可能的．你唯一能做的就是希望能夠讓你的表現大部份都跟著績效最好的那一對明燈差不多就好．也因此你會需要一個能夠根據先前表現狀況動態調整下注比率的演算法，而這個就是避險演算法會告訴你的事情．

2.1.2 Finite classification problem (有限類別分類問題)

假設 有K個類別的的預設模型組B和已經設定標簽好的訓練集S，那麼我們可以設定 B裡面的任一一個預測模型在預測錯誤S的時候給他損失比例為 1,正確的時候損失比例為0, 利用這樣的損失比例，可以利用避險演算法來找出B裡面適合的模型．

2.1.3 Rock, paper, and Scissors (剪刀石頭布)

可以將這個懲罰值用當預測模型猜贏實際對方出的時候給於損失比例為0, 平手給0.5, 輸的時候給1．

$You/Computer$	S	R	P
S	0.5	1	0
R	0	0.5	1
P	1	0	0.5

//TODO 利用Game Theory 重新來看這個問題(沒看懂...[Example3][2] QQ)

2.1.4 找尋R^n空間中的特定點

考慮在一個R^n 空間中的單位球找點，那麼其損失比例等於預測值到實際值得歐幾里德測距函數

2.2 Concept

假定有N個投資策略，目前也知道在這個時刻點t，每個投資策略i的損失比例是$l_i^t \quad 1 \leq i \leq N, \quad l^t \in [0,1]^N$ 同時也有每個時刻的投資分配比例p^t ,在這個比例下，經過時間T所造成的投資損失為

$$L_{Hedge} = \sum_{t=1}^{T} \sum_{i=1}^{N} p_i^t l_i^t \qquad \forall i \quad p^t \in [0,1]^N$$

同時，如果只考慮單一投資策略i ，經過時間T後，投資損失為

$$L_i = \sum_{t=1}^{T} l_i^t \tag{2.2.1}$$

因此，我們會希望所設計的避險演算法能夠找出，在每個時刻t中最適合的p^t 使得L_{Hedge} 與擁有最小的損失策略的差距最小．也就是追求:

$$\min_{\beta}(L_{Hedge}(\beta) - \min_{i} L_i)$$

有趣的事情是，隨著時間的變化，可能在$T = 5$的最小損失策略是二號，但到了$T = 10$時卻變成是六號，但是避險演算法到了$T = 10$時一樣可以保證你會靠近這個六號策略的誤差．針對這個問題，作者利用這樣的一個直覺得需求設計了避險演算法．

2.3 Algorithm

這裡我們將先介紹這個避險演算法(Hedge Algorithm)

Algorithm 1 避險演算法Hedge Algorithm with parameter β

Input: β:
一個自由選擇的調整加權參數變化幅度的參數

Input: w^t: $\sum_{i=1}^{N} w_i^1 = 1$
加權向量，要使用的策略數目為其向量維度，此外一開始向量和為1，可以直接當作投資分配比例p^t，但之後並不一定，因此使用的時候均需歸一化

Output:

1: **for** Do for t = 1,2, ...,T **do**

2: 計算投資分配

$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$

3: 計算當時刻T每個策略的損失比例 （假設你已經要開始比較這個時間點t 避險演算法和各策略i的優劣）

$$L^t = l^t \in [0,1]^N$$

4: 計算在該時刻t避險演算法的損失比例

$$L_{Hedge}(\beta)^t = p^t l^t$$

5: 計算下一個時間點$t+1$的加權向量

$$w_i^{t+1} = w_i^t \beta^{l_i^t} \tag{2.3.1}$$

6: **end for**

2.4 Analysis of Hedge Algorithm

考慮一個避險演算法，其參數為β，考慮現在有一個N維的加權向量

$$w^t = (w_1^t, ..., w_n^t) \qquad \forall i, w_i^t > 0 \quad \sum_i w_i^1 = 1$$

但由於這個加權向量我們在第一個時刻點的時候不知道該如何設定比較好，所以就一視同仁的全部設定為

$$w_i^1 = \frac{1}{N}$$

也因此在時間t的時候,可以定義我們的分配比例向量p為

$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$

然後當我們已知每一個決策演算法在這個時刻點t的失敗率為l^t ，那麼利用*multiplicaiton rule* 我們可以得到下一個時刻點用來猜測的加權向量為

$$w_i^{t+1} = w_i^t \dot{b}^{l_i^t}$$

如果要寫得更一般化的話，可以將上式的更新公式改成是與自己設定的β的函式$U_b(l_i^t)$

$$w_i^{t+1} = w_i^t \dot{U}_b(l_i^t)$$

同時這個$U_b : [0,1] -> [0,1] \quad b : [0,1]$ 還滿足(利用Weighted Majority Algorithm)

$$b^r \leq U_b(r) \leq 1 - (1-b)r \quad \forall r \in [0,1]$$

最後一步就是這個避險演算法最精華的一步!,因為這一步，所以我們可以繼續往下利用這件事情

$$\begin{aligned}
\sum_{t=1}^{N} w_i^{t+1} &= \sum_{t=1}^{N} w_i^t \beta^{l_i^t} \\
&\leq \sum_{i=1}^{N} w_i^t (1 - (1-\beta) l_i^t) \\
&= (\sum_{i=1}^{N} w_i^t)(1 - (1-\beta) p^t l^t) \\
&\leq (\sum_{i=1}^{N} w_i^1) \prod_{t=1}^{T} (1 - (1-\beta) p^t l^t) \\
&\leq \exp(-(1-\beta) \sum_{t=1}^{T} b^t l^t) \\
&= \exp(-(1-\beta) L_{Hedge}(\beta))
\end{aligned}$$

- 第二行用了 Weighted Majority Algorithm 才有辦法從第一行得到
- 第三行則是提出加權向量的總和，並根據策略分配比率的定義得到
- 第四行則是不斷地將t時刻的分配向量猜解到時刻1的總和（這時候總和等於1)
- 第五行利用 $1+x$ 小於等於 e^x
- 最後利用 L_{Hedge} 定義去置換

因此我們再簡單移項就可以得到式子 2.4.1

$$L_{Hedge}(\beta) \leq \frac{-\log(\sum_{i=1}^{N} w_i^{T+1})}{1-\beta} \tag{2.4.1}$$

根據式子 2.3.1知道

$$w_i^{T+1} = w_i^1 \prod_{t=1}^{T} \beta^{l_i^t} = w_i^1 \beta^{L_i} \tag{2.4.2}$$

Theorem 1. 給定任意個時刻點的策略損失比例向量，以及一個$1, \ldots, N$的非空子集合S，我們可以得到

$$L_{Hedge}(\beta) \leq \frac{-\log(\sum_{i\in S} w_i^1) - (\log\beta)\max_{t\in S} L_i}{1-\beta} \tag{2.4.3}$$

Proof.

$$\begin{aligned} \sum_{i=1}^{N} w_i^{T+1} &\geq \sum i \in S w_i^{T+1} \\ &= \sum_{i\in S} w_i^1 \beta^{L_i} \\ &\geq \beta^{\max_{i\in S} L_i} \sum_{i\in S} w_i^1 \end{aligned}$$

□

假定我們令這個子集S只包含$\arg\min_i L_i$，以及令一開始時間1時的加權向量w為$\frac{1}{N}$，亦即一開始對於每一個策略的看重程度都一樣，因此將式子 2.4.1帶入式子 2.4.3就能得到經過一段時間T後，避險演算法的誤差上界

$$L_{Hedge}(\beta) \leq \frac{\min_i L_i \log(\frac{1}{\beta}) + \log N}{1-\beta} \tag{2.4.4}$$

所以可以改寫為

$$L_{Hedge}(\beta) \leq c \min_i L_i \log(\frac{1}{\beta}) + c \log N$$

其中$c = \frac{\log(\frac{1}{\beta})}{1-\beta}$與$a = \frac{1}{1-\beta}$

根據paper中的定理三會告訴我們這個a與c的最小值就是如上面所述，所以現在問題回到我們該怎麼選擇最適合的β，使得避險演算法的上界中的a與c可以達到這個最小值．

2.5 How to choose parameter β

正常情況，亂選β不見得可以達到一個理想的誤差上界，因為很有可能組合後的表現很難靠近最佳解，但所幸我們有引理 1可以提供我們一個簡單的參數β選擇方式，且當調整加權參數的次數，避險演算法的誤差上界會一直遞減．引理 1是這麼說的

Lemma 1. 假定 $0 <= L <= \hat{L}$ 且 $0 < R <= \hat{R}$ ，令 $\beta = g(\frac{\hat{L}}{\hat{R}})$ 其中 $g(x) = \frac{1}{\sqrt{2/x}}$時，我們會得到下列不等式：

$$\frac{-L \log \beta + R}{1-\beta} \leq L + \sqrt{2\hat{L}R} + R \tag{2.5.1}$$

Proof. 令β等於引理 1所述$-\log\beta \leq \frac{1-\beta^2}{2\beta}$就能馬上得證 □

利用lemma 1, $\hat{R} = \log N$還有 $\hat{L}$最大就是T (因為假設給每一個時刻點都賠光就是1，有T個時刻點就是$1 * T = T$) ，配合式子 2.5.1就能得到，避險演算法損失比率上界為

$$L_{Hedge}(\beta) \leq \min_i Li + \sqrt{2\hat{L}\log N}\log N$$

考慮每一個時間點的平均損失比率為

$$\frac{L_{Hedge}(\beta)}{T} \leq \frac{\min_i L_i}{T} + \frac{\sqrt{2\hat{L}\log N}}{T}\frac{\log N}{T}$$

因為第一項是$O(\frac{1}{T})$，第二項$\hat{L}$的部分可以當作T，所以第二項是$O(\frac{1}{\sqrt{T}})$，第三項為$O(\frac{1}{T})$，因此其平均損失是跟著$O(\frac{1}{\sqrt{T}})$去遞減的，也就是說在避險演算法其損失上界可以在精心設計的β下，擁有一個會遞減的上界．