テーマ３：計算論的学習理論とブースティング技法（その２）

本日の講義の目的

- ブースティング入門

復習：確率的近似学習（PAC 学習）の諸定義

学習の目標：確率的近似学習（PAC 学習）の目標を式で書くと次のようになる．

$$
\begin{array}{l}
\text{任意の学習パラメータ } n, \varepsilon, \delta \text{ に対し，} \\
\text{任意の } n \text{ 変数論理関数 } f_* \in \mathcal{C} \text{ に対し，そして} \\
\text{事例の全体 } X \text{ 上の任意の確率分布 } D_* \text{ に対し，} \\
\displaystyle\Pr_{S:(D_*)^m}\left[\begin{array}{l} \text{事例集合 } S \text{ 対し，学習アルゴリズムが} \\ S \text{ から求めた仮説 } h \text{ が以下を満たす} \\ (*) \quad \displaystyle\Pr_{\boldsymbol{a}:D_*}[\ f_*(\boldsymbol{a}) \neq h(\boldsymbol{a})\ ] \ \leq\ \varepsilon \end{array}\right] \ \geq\ 1-\delta.
\end{array}
$$

補足

- 学習パラメータ：ε は誤差上界，δ は失敗確率上界．
- 目標関数クラス $\mathcal{C}$：ルールの表し方全体．学習目標関数 f_* もその要素の一つ．
- 事例：入力例 $\boldsymbol{a}$ と判別値 $f_*(\boldsymbol{a})$ の組．

3.3. ブースティング技法：基礎編

目標関数を f_*，現在考えている確率分布を D とする．次の確率を仮説 h の優位度 (advantage) という．

$$\text{優位度} \ = \ \Pr_{\boldsymbol{a}:D}[\ f_*(\boldsymbol{a}) = h(\boldsymbol{a})\ ] - \frac{1}{2}.$$

この優位度が定数より大きな仮説，つまり，正解率が 1/2 より定数分以上大きな仮説 h を弱仮説という．

正解率が 1/2 というのは「(背後の) 法則がまったくわからない」ということに他ならない．それよりは少しはマシなのが弱仮説である．ブースティング技法では，どんな事例分布に対しても弱仮説ならば比較的容易に求めることができる，という仮定する．この仮定に基づいて，役に立つ (良い) 仮説を求める学習アルゴリズムを設計する手法がブースティング技法 (boosting technique) である．その概略を以下に示す．

Algorithm ブースティング・アルゴリズム概略
目標関数： n 変数の論理関数 $f_* \in \mathcal{C}$;（アルゴリズムには未知）
事例分布： D_0;（これは決まっていると考える．D_* と思ってもよい）
入力： 学習パラメータ n, ε, δ;
出力： n 変数の論理関数;
手順：

$t \leftarrow 1$; $D_1 \leftarrow D_0$;
repeat {
$h_t \leftarrow$ 分布 D_t のもと f_* に対する正解率の（最も）よい仮説;
（注：単純な仮説の集合の中で探す；その中に弱仮説は見つかる仮定）
$\gamma_t \leftarrow h_t$ の分布 D_t のもとでの優位度;
$\alpha_t \leftarrow h_t$ の重要度;
<u>統合仮説</u> f_t を次のように定義する：

$$f_t(\boldsymbol{x}) \;=\; \left(\sum_{1 \le i \le t} \alpha_i h_i(\boldsymbol{x}) \right) \text{の符号（}\pm 1\text{）}$$

if $\Pr_{\boldsymbol{a}:D_0}[\, f_*(\boldsymbol{a}) \neq f_t(\boldsymbol{a}) \,] \;<\; \varepsilon$ **then break**;
$D_{t+1} \leftarrow D_t$ と h_t から作られる新たな分布; $t \leftarrow t+1$;
} （ここまでが **repeat** の範囲）
得られた f_t を出力;
end.

上記はあくまでもアルゴリズムの概略である．実際には重要度 α_t と分布 D_t をどのように決めるかが重要である．この決め方を一つ定めるとアルゴリズムが一つ定まる，と考えてよいだろう．これを次のように定めたのが有名な **AdaBoost** である．

$$\begin{aligned} &\alpha_t \;=\; \ln \beta_t^{-1}, \quad w_t(\boldsymbol{x}) \;=\; D_0(\boldsymbol{x}) \cdot \prod_{1 \le i \le t-1} \beta_i^{h_i(\boldsymbol{x}) \cdot f_*(\boldsymbol{x})}, \quad D_t(\boldsymbol{x}) \;=\; \frac{w_t(\boldsymbol{x})}{W_t}. \\ &\text{ただし} \;\; \beta_t \;=\; \sqrt{\frac{1-2\gamma_t}{1+2\gamma_t}}, \quad W_t \;=\; \sum_{\boldsymbol{x} \in \{-1,+1\}^n} w_t(\boldsymbol{x}). \end{aligned} \tag{1}$$

この **AdaBoost** に対し次の性質が成り立つ．この性質を保証することができるアルゴリズムが，厳密な意味での <u>ブースティング・アルゴリズム</u> である．

定理 3.4. （**AdaBoost** のブースティング性）
目標関数を f_*，想定している事例分布を D_0，そしてアルゴリズムに与える誤判別確率の目標上界を $\varepsilon > 0$ とする．このもとで **AdaBoost** を実行した場合に，ある定数 γ_0 に対し，各 t 回目のブースティングステップで，優位度 $\gamma_t \ge \gamma_0$ となる弱仮設がつねに得られると仮定する．このとき，ブースティングステップ数（目標の精度までに必要なブースティングステップの回数）T は，次式のように抑えられる．

$$T \;\le\; \frac{1}{2\gamma_0^2} \ln \frac{1}{\varepsilon}.$$

3.4. オッカムの剃刀としての AdaBoost の利用

オッカムの定理（定理 1.3）が保証しているように，PAC 学習の目標のためには（十分大きな例数からなる）事例集合に無矛盾な仮説を作ればよかった．しかもなるべくコンパクトな仮説がよい．そのような仮説を作るアルゴリズムは「オッカムの剃刀」などと呼ばれている．ブースティング・アルゴリズムは，このオッカムの剃刀を設計するためにも使用できる．というか，そういう使用法が多い．ここでも **AdaBoost** を使ったオッカムの剃刀の設計法を紹介しよう．

といっても難しいことはない．先のブースティング・アルゴリズムの概略にあった分布 D_0 を，与えられた事例集合 S 上の等確率分布と考え，その分布のもとで $\varepsilon = 1/m$（ただし，m は S の要素数）とすればよい．この分布 D_0 のもとで，誤差判定が $1/m$ 未満ということは，誤判別確率が 0（つまりご判別無し）に等しい．具体的に示すと以下のようになる．

Algorithm AdaBoost を用いたオッカムの剃刀
目標関数： n 変数の論理関数 $f_* \in \mathcal{C}$;（アルゴリズムには未知）
入力： 事例集合 $S = \{(\boldsymbol{a}_1, f_*(\boldsymbol{a}_1)), ..., (\boldsymbol{a}_m, f_*(\boldsymbol{a}_m))\}$;
出力： S に無矛盾な仮説 f；
手順：
各 $\boldsymbol{a}_i$ に対し，$D_0(\boldsymbol{a}_i) = 1/m$ とする;
$t \leftarrow 1$; $D_1 \leftarrow D_0$;
repeat {
$h_t \leftarrow \mathrm{error}_t(h)$ が最低となる h;
ただし $\mathrm{error}_t(h) = \sum_{i \in Err(h)} D_t(\boldsymbol{a}_i)$ (where $Err(h) = \{i : h(\boldsymbol{a}_i) \neq f_*(\boldsymbol{a}_i)\}$)
$\gamma_t \leftarrow 0.5 - \mathrm{error}_t(h_t)$;
$\alpha_t \leftarrow$ 式 (1) に従って計算;
統合仮説 f_t を次のように定義する：

$$f_t(\boldsymbol{x}) = \left(\sum_{1 \le i \le t} \alpha_i h_i(\boldsymbol{x}) \right) \text{の符号（}\pm 1\text{）}$$

if $\mathrm{error}_0(f_t) = 0$ **then break**;
各 $\boldsymbol{a}_i$ に対し，$D_{t+1}(\boldsymbol{a}_i) \leftarrow$ 式 (1) に従って計算; —（※）
$t \leftarrow t + 1$;
}（ここまでが **repeat** の範囲）
得られた f_t を仮説 f として出力;
end.

ただし,（※）の計算だが，実際のプログラムでは，次のように重さ w_{t+1} を更新する計算の方が簡単だ（ただし $w_1(\boldsymbol{x}) = D_1(\boldsymbol{x})$ とする）．これが式 (1) の計算と同じであるこ

とは容易にチェックできるだろう．

$$w_{t+1}(\boldsymbol{x}) \;=\; \begin{cases} w_t(\boldsymbol{x}) \cdot \beta_t, & \text{if } h_t(\boldsymbol{x}) = f_*(\boldsymbol{x}), \text{ and} \\ w_t(\boldsymbol{x}) \cdot (1/\beta_t), & \text{if } h_t(\boldsymbol{x}) \neq f_*(\boldsymbol{x}). \end{cases}$$

先の **AdaBoost** のブースティング性を保証する定理を用いると，計算時間ブースティングステップ数と出力される仮説の大きさについて次のことが言える．

系 3.5. 事例数 m の事例集合 S に対する上記のアルゴリズムの実行を考える．毎回ブースティングステップで得られる弱仮説の優位度がつねに γ_0 以上とすると，ブースティングステップ数 T は $T \;\leq\; (\ln m)/(2\gamma_0^2)$ となる．

しかし，つねに優位度 γ_0 が保証されるとは限らないので，アルゴリズムにあるように $\mathrm{error}_0(f_t) = 0$ となるまで（別の言い方をすれば $\mathrm{error}_0(f_t) < 1/m$ となるまで）に固執すると停止しない可能性もある．実際には，適当なところで止めて使う場合も多い．

テーマ＃３での課題（〆切：以後，すべての〆切りを７月２２日（金）とします）

1. この例のように，全体のアルゴリズムの一部分の計算は何とか望み通りに実施できると仮定して（注），それから先はキッチリ解析し，アルゴリズムを設計している例を紹介せよ.（注：もちろん，無謀に仮定しても意味がない．妥当そうで，実際に実験してみるとうまく動くヒューリスティクスがあるのだが，それはとても複雑で解析ができない，というような場合である.）

2. もしかすると，このような例はアルゴリズムの世界だけではなく数学の世界にもあるかもしれない．つまり，ある部分だけは仮定して，もしもそれが成り立つとすると... ということで仮説の上に一つの理論が形成されているような数学の例があれば紹介して欲しい．

3. 実際に **AdaBoost** のプログラムを作成し，実験データ（たとえば `mushroom`）を用いて，その性能などを実験し，その結果・解析・考察を述べよ．訓練データ（つまり事例集合）として 1000 個くらいを使い，残りのデータを使って得られた仮説の良さを評価してみるとよい．ベストな仮説は何か？訓練データを多くするとどうなるか？高速化の工夫と効果は？等々，いろいろと調べられると思う．

参考：データならびにサンプルプログラムは

`http://www.is.titech.ac.jp/~watanabe/class/boost/` から得られる．