

# 情報生命科学演習

## Ada-Boostの実装

齊藤 太郎

[leo@cb.k.u-tokyo.ac.jp](mailto:leo@cb.k.u-tokyo.ac.jp)

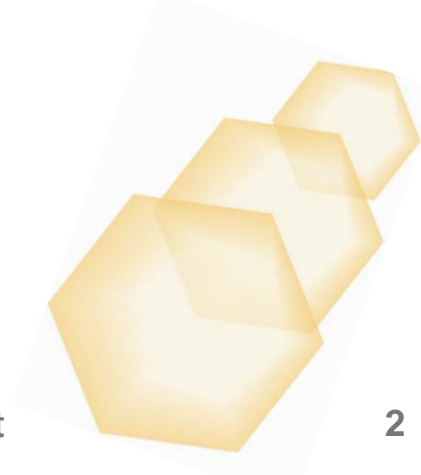
July 10<sup>th</sup>, 2007



# Ada-Boost アルゴリズム

情報生命科学演習: AdaBoostの実装

- 講義資料とサンプルプログラム
  - <http://www.xerial.org/lecture/AdaBoost>
- Ada-Boost アルゴリズム
  - プログラミングの難易度としてはやさしめ
  - 実装のポイント
    - Tab区切りデータ（あるいはその他のデータ形式）の読み込み
    - クラス分類器の理解
      - 入力（多次元データ）に対して、目標属性の値を予測するもの
    - ランダムに答える（50%）より正答率の良い、クラス分類器をプログラムから自動生成
      - WeakLearner
    - クラス分類器を組み合わせる
      - 各分類器にサンプル毎に異なる重みを与える



# プログラミングの考え方

情報生命科学演習： AdaBoostの実装

- アルゴリズムのどの部分をクラスにするか？
  - 考え方の基準
    - データのまとめり
      - テーブルデータ？
    - 機能のまとめり
      - クラス分類器？
    - アルゴリズムのまとめり
      - Ada-Boostの一連の流れ
  - クラス内のデータは、クラス自身に処理させるのが原則
    - クラスの状態がいつ変更されるかわかりやすくなる
    - 例： SampleWeightクラスは、ただのdoubleの配列だが、その配列上でできる操作を限定している
  - サンプルプログラムとは違うクラス設計も十分考えられます

# プログラムの品質

情報生命科学演習: AdaBoostの実装

- ソフトウェア工学の観点から
  - 1年後にソースコードを読んで内容が理解できることが理想
  - コメントを適宜入れる
  - プログラムの変数名、メソッド名が自然言語に近くなるように配慮
    - Eclipseなどのエディタの進化で、長い変数名を入力するのが数年前よりかなり楽になっています
      - Eclipseでは、Ctrl + SPACEで、入力を補完
- プログラムが正しく動くことを確認
  - 各クラスごとに、テストコードを作成
  - 大規模なクラス設計の変更時にも、デバッグが容易に

# ファイルの読み込み

情報生命科学演習: AdaBoostの実装

```
BufferedReader reader =  
    new BufferedReader(new FileReader(fileName));  
String line;  
while((line = reader.readLine()) != null ) {  
    String[] column = line.split("¥t");  
}
```

- **FileReaderは、ディスクI/Oを行うクラス**
  - 現在のハードディスクはレコードとほぼ同じ構造
    - 一回転でデータを沢山読み込んでおきたい
  - そのため、BufferedReaderを仲介に挟み、メモリにデータを貯めておく (バッファリング)
- **while( (line = reader.readLine()) != null) の意味**
  - readLine()は、データがないときにnull を返す
  - (line = reader.readLine()) という式は、左辺値、すなわちlineの値を返す
- **split("¥t") : データを"¥t"を目印に分割 (正規表現)**

# WeakLearner

情報生命科学演習： AdaBoostの実装

- Ada-Boost アルゴリズムでの要件
  - サンプルデータでの正答率が50%以上の分類器 (classifier) を作ることができればよい
- 取り組み方の例
  - ためしに、ある列 (column)のデータが 1 (true)なら、 trueと予測するclassifierを作ってみる
  - そのようなclassifierを各列、 予測のtrue/falseの組み合わせで、brute-forceで見つける
  - 発展：
    - 2列以上のデータの組み合わせから、目標属性の値を予測できるようにプログラムを拡張する
  - サンプルデータにそのclassifierを適用し、 error率を計算

# レポートの内容

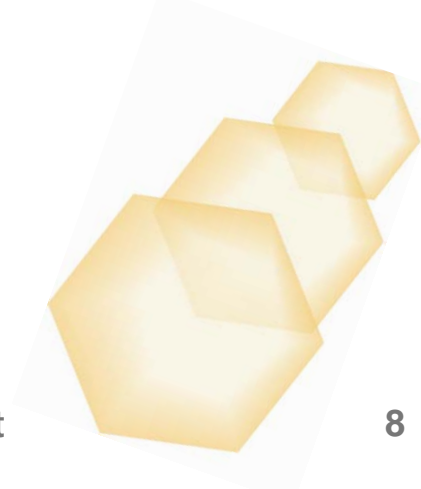
情報生命科学演習: AdaBoostの実装

1. プログラムのソースコード
  - サンプルプログラムを与える
  - 自ら考えて実装した場合は可以上の評価
  - サンプルプログラムをそのまま使用した場合は可以下の評価
2. データ構造と手続きの流れの説明
3. サンプルデータ（2値）から計算された分類器を示すこと
4. 重み、エラー率の変動についての解析
5. データ数を100, 1000 と増加させた例を作って計算時間とエラー率を測定する
6. WeakLearn が新しいクラス分類器を出力できない例
7. WeakLearn が( $T1=1$ ) かつ( $T2=0$ ) のように2つの属性を使ったクラス分類器を出力できるように拡張する

# レポートの提出方法

情報生命科学演習：AdaBoostの実装

- 締切日：7月31日（火） 23:59
- 宛先：[leo@cb.k.u-tokyo.ac.jp](mailto:leo@cb.k.u-tokyo.ac.jp)
- メールのsubjectを、**[report]**（半角文字で表記）とすること
- プログラムのソースコード、レポート文章をひとつのフォルダにまとめ、zip, tarなどの形式に圧縮
  - Windowsでは、Lhacaなどのフリーウェアで圧縮できます
- 氏名・学籍番号を忘れずに



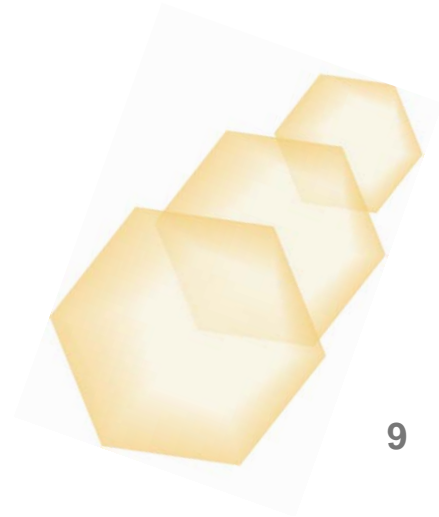
# 情報生命科学演習

## Ada-Boostの実装 (補足)

齊藤 太郎

[leo@cb.k.u-tokyo.ac.jp](mailto:leo@cb.k.u-tokyo.ac.jp)

July 17<sup>th</sup>, 2007



# Loggerの役割

情報生命科学演習：AdaBoostの実装

- プログラムの途中の状態を知りたい
  - Eclipseのデバッグ機能を使う
    - ブレークポイント
    - ループ回数が多くなると状態の変化を追うのが大変
  - System.out.println でデバッグ用のメッセージを表示
    - 不要になったときに、println文を消し忘れる
    - 再度必要になったときに、また書き直すのが手間
- 解決法：Loggerを使う
  - 必要なときにログ表示をON、不要なときにはOFF
  - 既存のライブラリ(log4jなど)もあるが、自作しても数百行程度のコード
    - サンプルプログラム：org.xerial.util.log.Logger

# Logger の使い方

情報生命科学演習: AdaBoostの実装

```
Logger _logger = Logger.getLogger("AdaBoost.main");

long t = System.currentTimeMillis();
_logger.info("entering the loop");
for(int i=0; i<100; i++)
{
    _logger.debug("loop count: " + i);
    // do something
    _logger.trace("more detailed debug message");
}
t = System.currentTimeMillis() - t; // ループの実行時間を計測
_logger.info("left the loop: time=" + t + " milli sec.");
```

# LogのOn/Off

情報生命科学演習： AdaBoostの実装

- すべてのログをオフにする
  - `Logger.getRootLogger().setLogLevel(Logger.OFF);`
- 特定のログのみをオフにする
  - `Logger.getLogger("AdaBoost.main").setLogLevel(Logger.OFF);`
    - AdaBoost.mainをprefixにもつLoggerの出力をOFFにする
- ログレベルの切り替え
  - TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF
- 使用例：
  - `Logger.getLogger("AdaBoost").setLogLevel(Logger.INFO)`
    - AdaBoostをprefixに持つLoggerでは、INFOより下（つまり、TRACE, DEBUG)のメッセージは表示しない
  - `Logger.getLogger("AdaBoost.WeakLearn").setLogLevel(Logger.WARN)`
    - AdaBoost.WeakLearnをprefixに持つLoggerでは、WARN以上のメッセージのみを出力
- Log4jなどのライブラリでも使い方は同様

# 設定ファイルでLoggerの出力を制御

情報生命科学演習: AdaBoostの実装

- サンプル中のLoggerプログラム

- Java Virtual Machine (JVM)の引数として、log.configという変数に設定ファイル名を指定しておく、最初に設定ファイルを読んで、ログレベルを設定する

- 使用例 :

- > java **-Dlog.config=log\_config.txt** -jar AdaBoost.jar ...

- log\_config.txtの内容

```
AdaBoost.main = debug
AdaBoost.WeakLearn = trace
```

- 補足

- 設定ファイルを使い、プログラムの動作を変えたい

- java.util.Properties クラスが便利

- 詳細はJDK のドキュメントを参照してください

- あるいは、Loggerのソースコードを参照