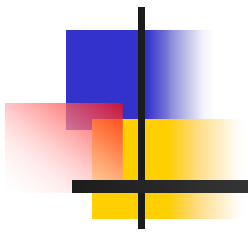
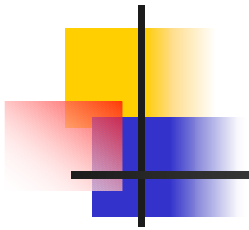


SplusによるAdaBoostの実装

筑波大学　社会工学研究科

佐野　夏樹



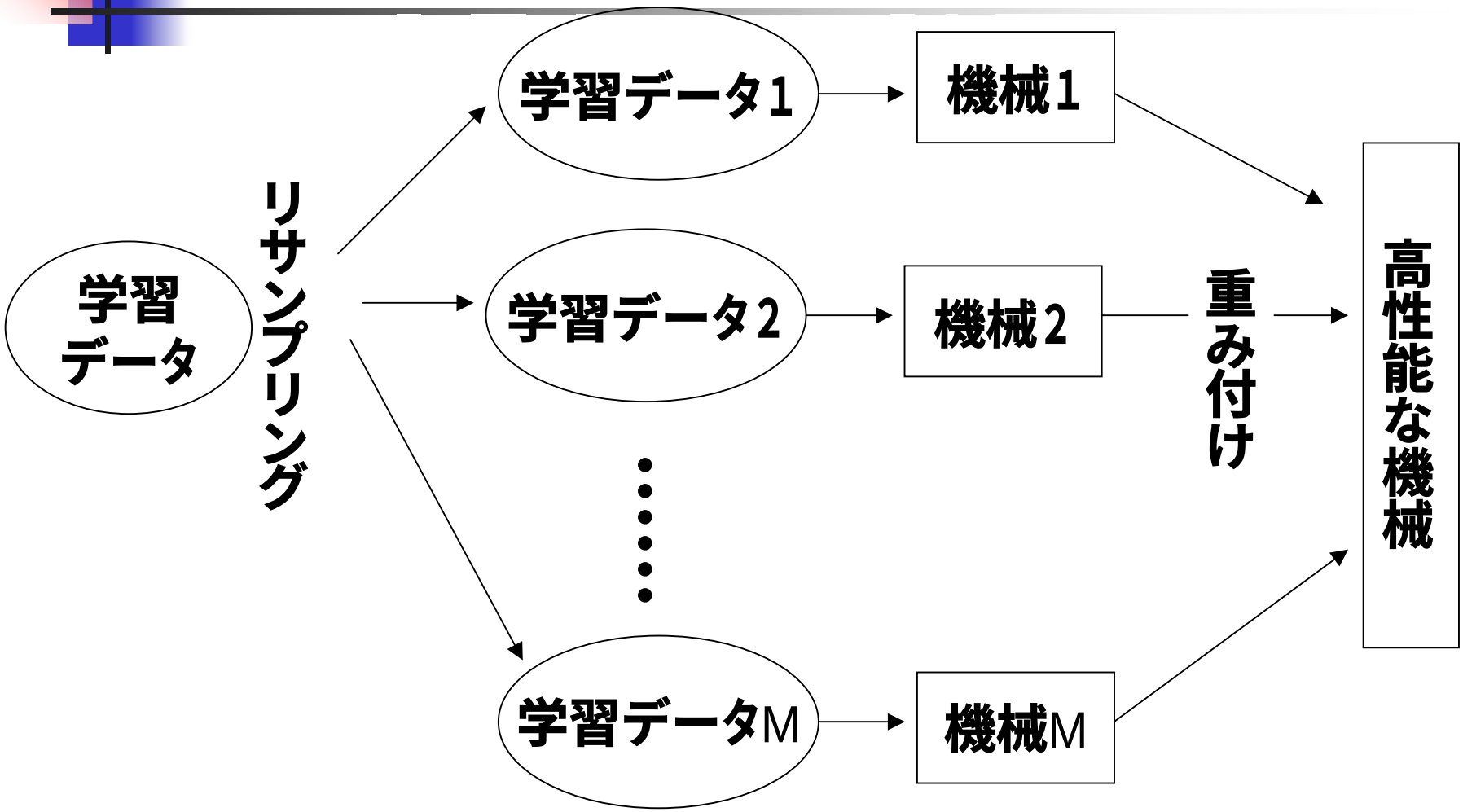
Boosting

- それほど能力の高くない学習機械をたくさん集めて，個々の機械より本質的に高い能力を持つ学習機械を作りだすこと

主なブースティング手法

- Bagging: Bootstrapによってリサンプリングを行いデータをつくりなおし，得られた学習機械を一様に足し合わせる．
- AdaBoost：学習データに対するリサンプリングの際に重みを逐次的に更新していき，できた機械を最後に学習機械に対する重みをつけて足し合わせて統合学習機械をつくる．

リサンプリングによる方法



問題設定

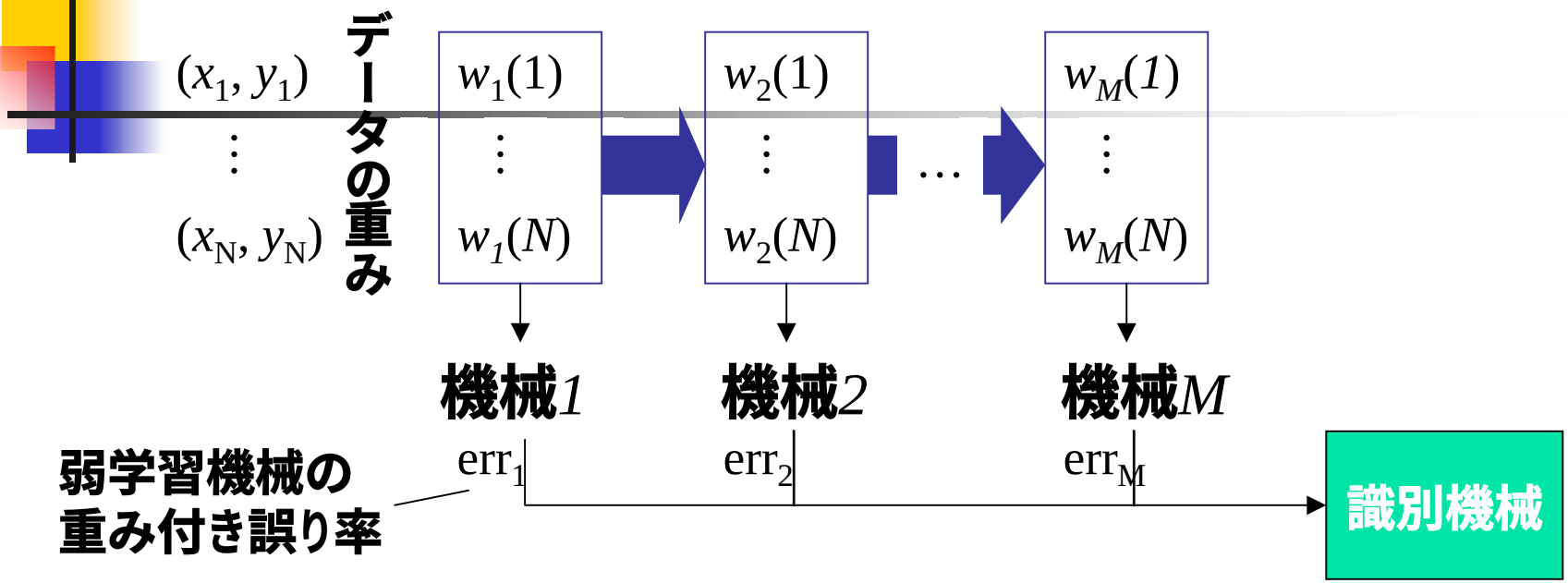
- 2クラス識別問題

  例えば 商品を購入・非購入など

  クラスは{-1,1}に変換する.

- 基本学習機械（ニューラルネットワーク，決定木など）を用意し，特徴データからクラスを予測する.

AdaBoostの概念図



- 学習データに対する重み$w_m(i)$
  - ひとつ前の機械が間違えたデータ → 重みを高く
  - ひとつ前の機械が正解したデータ → 重みを低く
- 学習機械に対する重みβ

  重み付き誤り率の大きい機械・・・ β小

  重み付き誤り率の小さい機械・・・ β大, M＝ラウンド数

AdaBoost M1 (Freund & Schapire 1996)

1.**重みの初期値を** $w_0(i) = 1/N,\ i = 1,2,\ldots,N$ **とする.**

2. For $m = 1$ to M: do

(a)**重み** w_{m-1} **を用いて学習データに機械** $f_m(x)$**をあてはめる.**

(b)**重み付き誤り率** $\mathrm{err}_m = \sum_{i=1}^{N} w_{m-1}(i) I(y_i \neq f_m(x_i))$

(c) **信頼度** $\beta_m = \log((1-\mathrm{err}_m)/\mathrm{err}_m)$**を求める.**

(d) $w_m(i) \leftarrow w_{m-1}(i) \cdot \exp[\beta_m \cdot I(y_i \neq f_m(x_i))],\ i = 1,2,\cdots,N$

重みを更新し $\sum_i w_m(i) = 1$ **となるように正規化する.**

3.**識別機械を** $\mathrm{sign}(\mathrm{F_m}(\mathrm{x})) = \mathrm{sign}[\sum_{\mathrm{m}=1}^{M} \beta_m f_m(x)]$ **とする.**

リサンプリングしない方法

リサンプリングせずに学習データに対する重み$w_m(i)$を直接学習機械に取り入れることができる（サンプリングエラーがない）.

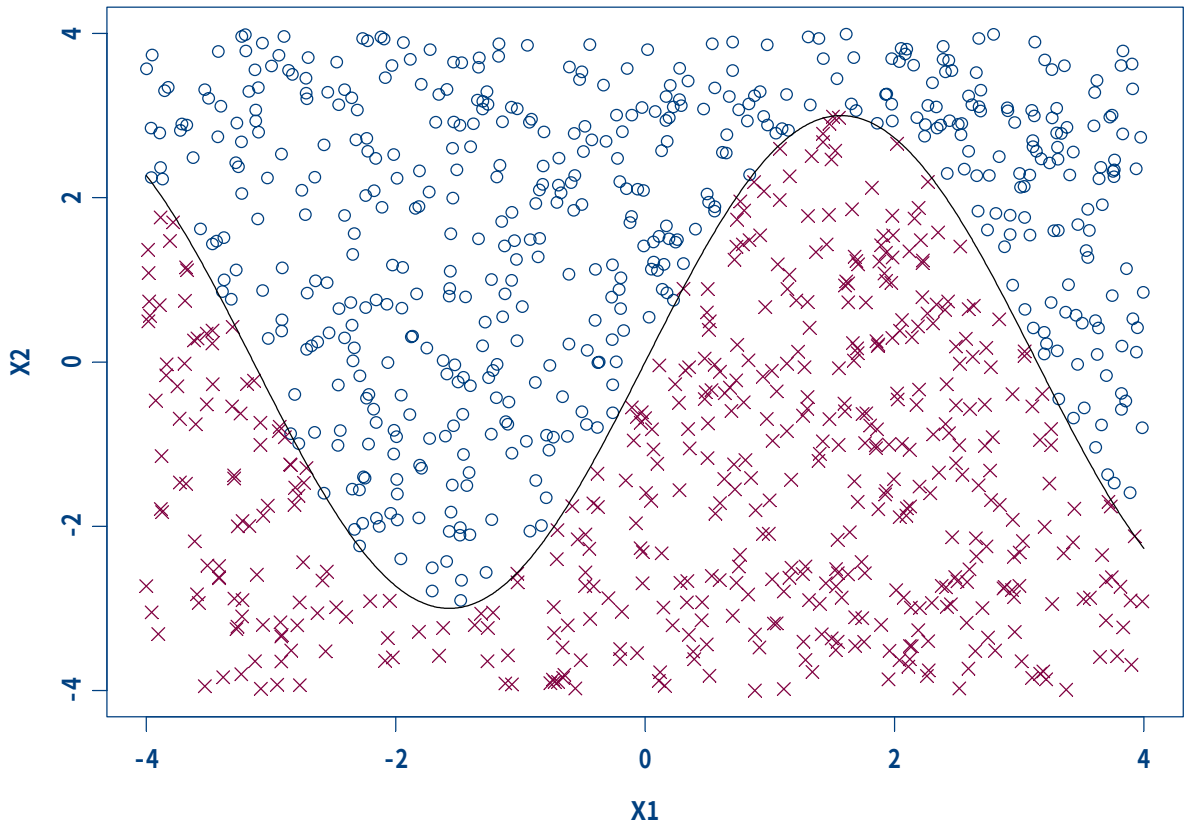
- **決定木の場合**

```
tree.model <- rpart(Y ~ ., weights = w, method =
    "class", data = Train)
```

- Neural Network **の場合**

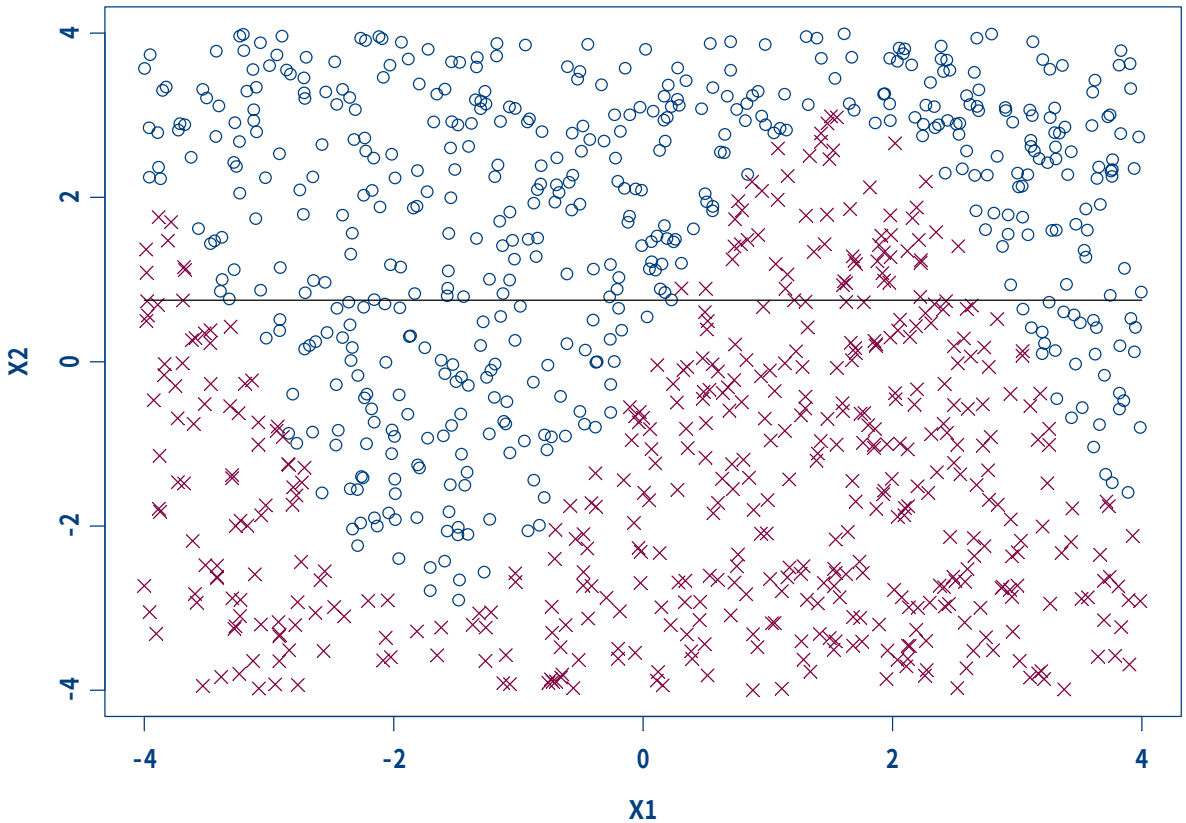
```
neural.model <- nnet(as.factor(Y) ~ ., data = Train,
    weights = w, size = 3, maxit = 1000)
```

Example Data



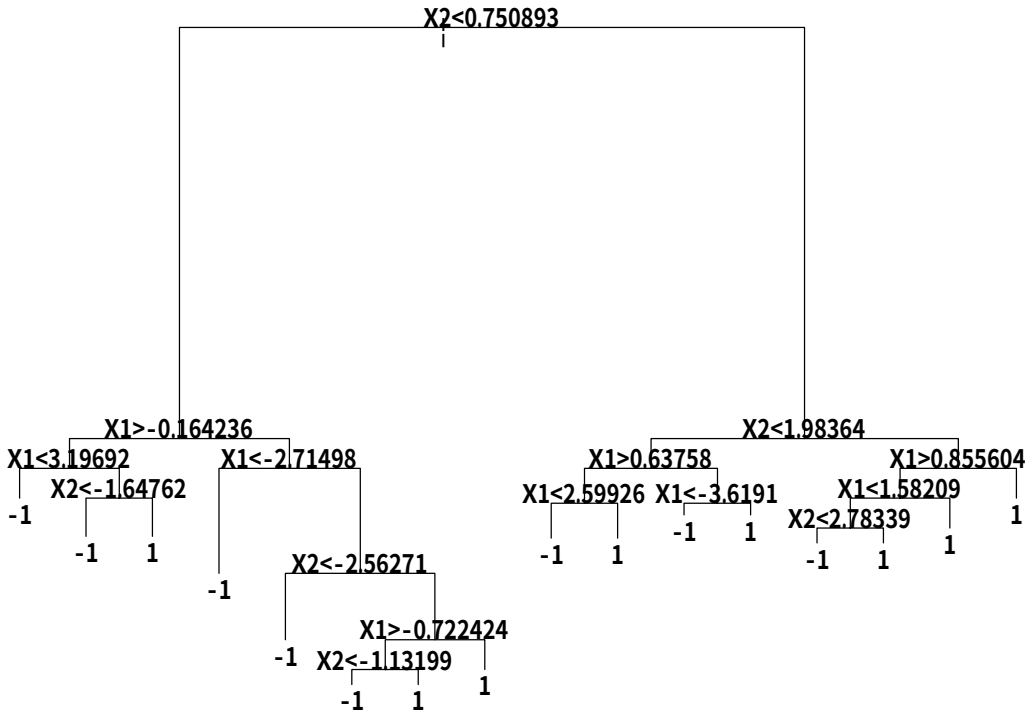
学習データ数１０００　テストデータ数１０００

2ノードのTree (Stump)



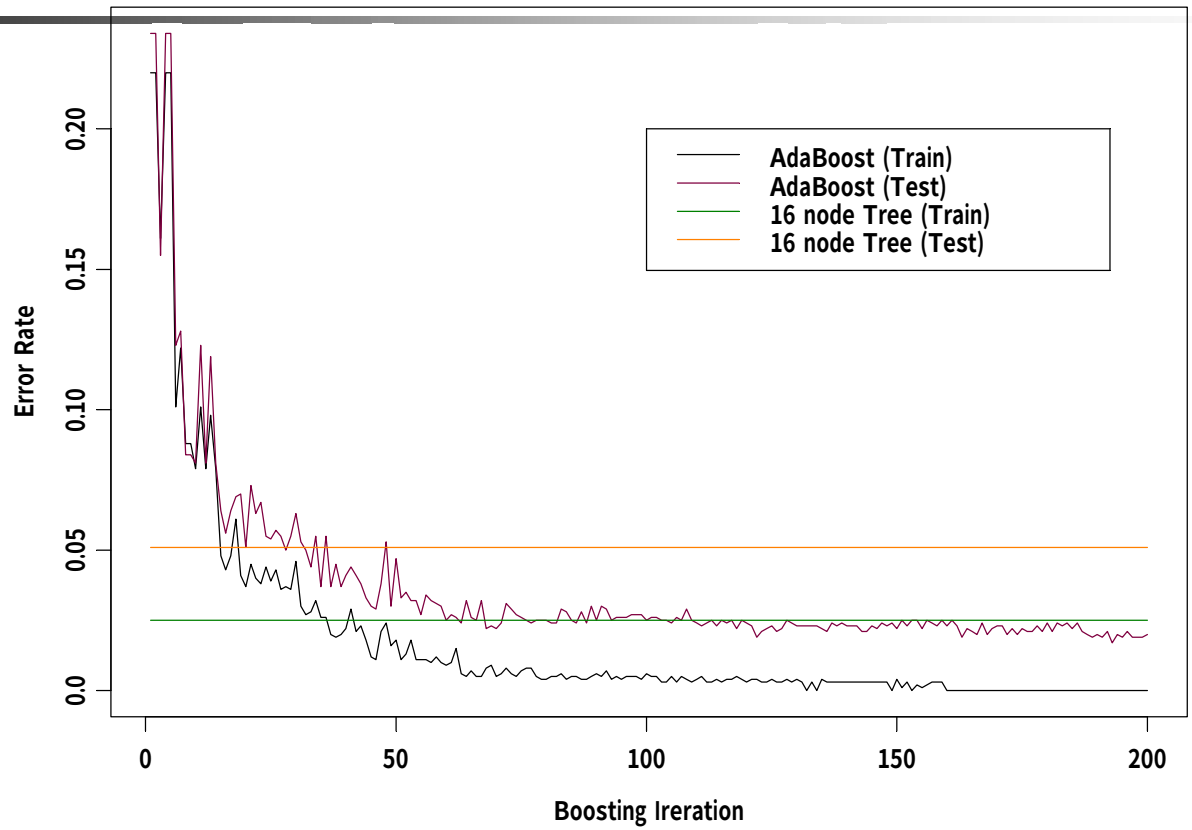
学習誤り率 (0．22) テスト誤り率 (0．23)

１６ノードのTree



学習誤り率（０．０３）テスト誤り率（０．０５）

AdaBoostの実験



学習誤り率（0）テスト誤り率（0．02）

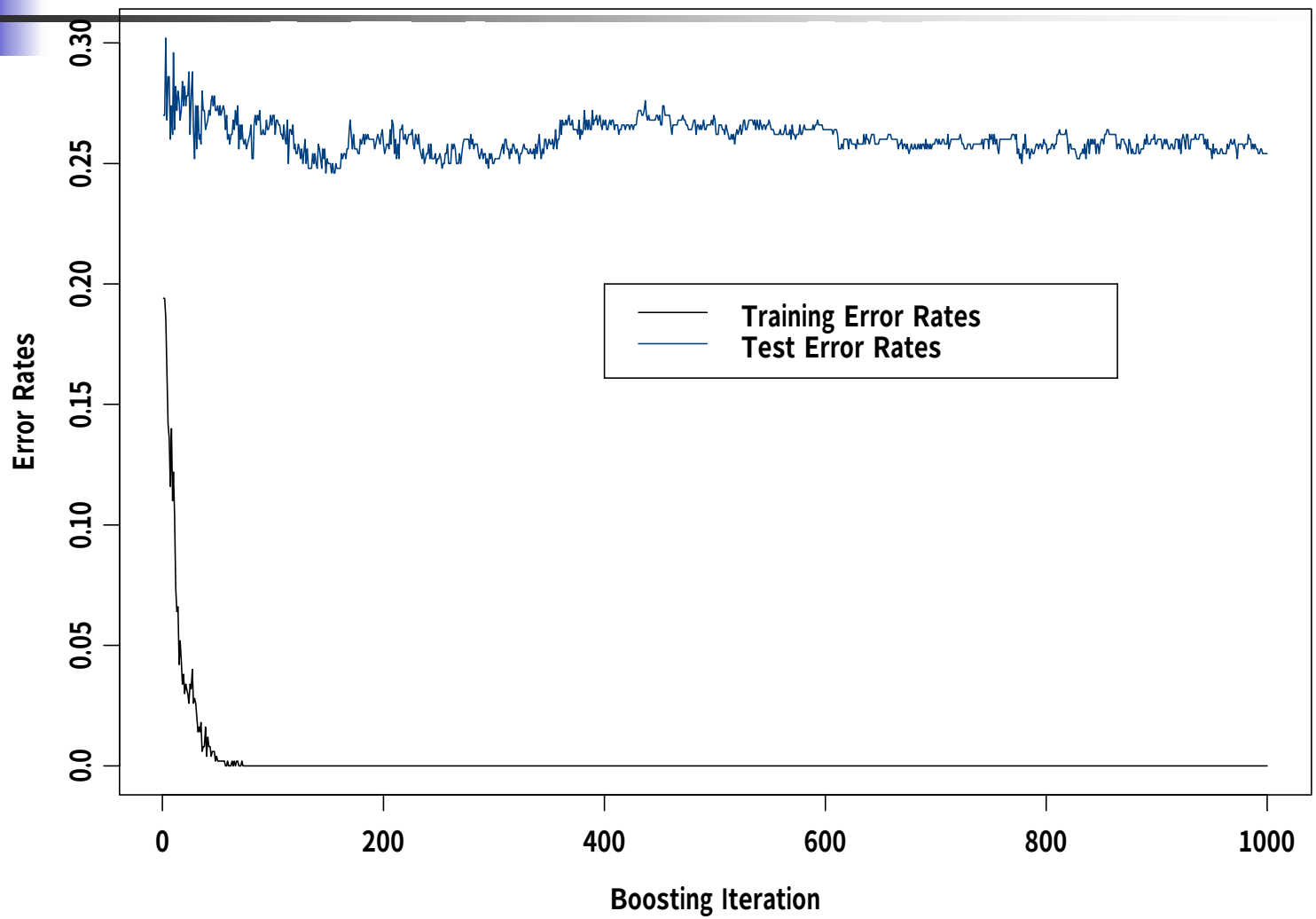
German Credit Data (Uci Repository)

- ドイツの銀行のクレジット返済に関するデータ
- サンプル数1000
- クレジットの履行，不履行を目的変数とする2値分類データ
- 説明変数は（当座預金口座額，継続月数，使用目的，利用金額，など24変数）である．

AdaBoostの適用

- **基本学習機械として隠れ層の素子数3の** Back Propagation Neural Netoworks **を使用した.**
- AdaBoost**の繰り返し回数は**1000**回**

AdaBoostの適用



結論

- シミュレーションで取り上げたようなノイズの混じっていないデータにおいてはAdaBoostの効果は顕著である.
- German Credit Data に対してAdaBoostを繰り返し数1000回まで適用した結果，テスト誤り率が0.254まで低下した.
- SplusによってAdaBoostが容易に実装･実行できる.

参考文献

- Y.Freund and R.E. Schapire.

  A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139, 1997.
- http://www.ics.uci.edu/mlearn/MLRepository.html

Appendix
ー SplusによるAdaBoostの実装 ー

Splusを用いてAdaBoostの実装，計算が容易に行えます．
行にサンプル，列に変数をとる学習データ，テストデータを用意する．
ただし1列目を教師データとする．教師データは{1,-1}と変換しておく．

```
> AdaBoost.Neural
function(Train, Test, M)
#  学習データ，テストデータ，最終的なBoositingのIteration 回数（ラウンド）の設定．
{
    options(object.size = 50000000)
    TrainY <- Train[, 1]       # 学習データの教師データの取得
    TestY <- Test[, 1]         # テストデータの教師データの取得
    TrainN <- length(TrainY)   # 学習データのサンプル数の取得
    TestN <- length(TestY)     # テストデータのサンプル数の取得
    Weak.learn <- matrix(rep(0, TrainN * M), ncol = M)
#  学習データに対する弱学習機械の予測結果を格納するマトリクスを用意
    Weak.test <- matrix(rep(0, TestN * M), ncol = M)
#  テストデータに対する弱学習機械の予測結果を格納するマトリクスを用意
```

```
error.table <- matrix(rep(0, 2 * M), ncol = 2)
# 各ラウンドにおける統合学習機械の誤り率を格納するマトリクスを用意
b <- rep(0, M)
# 弱学習機械に対する重みを格納するベクトルを用意
error.learn <- rep(0, M)
# 各ラウンドにおける弱学習機械の重み付き誤り率を格納するベクトルを用意
w <<- rep(1/TrainN, TrainN)
# 学習データに対する重みの初期値を設定
for(t in 1:M) { Mラウンドまで繰り返す
z.learn <- nnet(as.factor(Y) ~ ., data = Train, weights = w, size = 3, maxit = 100
# 弱学習機械として隠れ層の素子数3のBack-Propagation Neural Networkを使用
zp <- predict(z.learn, Train, type = "class")
# 学習データに対する弱学習機械の予測値を計算
zp <- as.integer(zp)
Weak.learn[, t] <- zp
if(sum(w[zp != TrainY]) == 0)
error.learn[t] <- 1e-025
else error.learn[t] <- sum(w[zp != TrainY])
# 重み付き誤り率の計算
b[t] <- 0.5 * log((1 - error.learn[t])/error.learn[t])
# 弱学習機械に対する重みを計算
```

```
        w[zp != TrainY] <<- w[zp != TrainY] * exp(b[t])
#  弱学習機械が誤った学習データに対する重みを更新
        w[zp == TrainY] <<- w[zp == TrainY] * exp( - b[t])
#  弱学習機械が正解した学習データに対する重みを更新
        pred <- predict.nnet(z.learn, Test[, -1], type = "class")
#  テストデータに対する弱学習機械の予測結果を計算
        Weak.test[, t] <- as.integer(pred)
      w <<- w/sum(w)
# 学習データに対する重みを正規化
      Fx <- Weak.learn %*% b
#  統合学習機械の学習データに対する出力を計算
      Gx <- Weak.test %*% b
#  統合学習機械のテストデータに対する出力を計算
    error.table[t, 1] <- length(Fx[sign(Fx) != TrainY])/TrainN
#  統合学習機械の学習誤り率の計算
    error.table[t, 2] <- length(Gx[sign(Gx) != TestY])/TestN
#  統合学習機械のテスト誤り率の計算
        print(c(error.table[t,  ], t))
      }
    return(error.table)
#  統合学習機械の誤り率のテーブルを返す
```

実行例

```
> AdaBoost.Neural(data.sin.2per.learn,data.sin.2per.test,10)
      [,1]  [,2]
 [1,] 0.034 0.033
 [2,] 0.034 0.033
 [3,] 0.034 0.033
 [4,] 0.034 0.033
 [5,] 0.034 0.033
 [6,] 0.034 0.033
 [7,] 0.036 0.037
 [8,] 0.033 0.034
 [9,] 0.033 0.034
[10,] 0.031 0.033
```