多数候補からの統計モデル選択に伴う
誤分類率の推定と検定

倉　橋　一　成

要旨

近年コンピュータの計算能力の発展に伴い、機械学習や統計的学習の分野で予測を行うための様々なアルゴリズムやモデルが提案されている。作成したモデルの予測性能は、結果変数をカテゴリ変数とした上で予測結果の誤分類率を計算し、評価することが一般的である。統計モデルは簡単にいくつも試すことができるが、試したモデルのうち最も良いモデルを選択すると、選択バイアスが混入する。そのため本論文ではこのバイアスを加味した誤分類率の検定方法を提案する。また統計モデルの性能を視覚的に確認できる K-sample Plot（K's Plot）の提案を行う。これは Bootstrap 法や Cross Validation 法と違い、サンプル数が増えたときの誤分類の減少と検出力の増加を視覚的に評価できる新しい手法である。

誤分類率の検定の性能をシミュレーションで確認し、K's Plot の性質はマイクロアレイや健康診断といった実データで評価した。シミュレーションの設定は、結果変数は 2 値、説明変数の数が 100 と 1000、サンプルの数が 20 と 200（片群の数は半分）、結果変数と説明変数の関連が無い場合と線形の関連がある場合、説明変数のグループと結果変数に関連がある状況で行った。それぞれ 2,000 回行っている。検定を行った際の α エラーはそれぞれの設定で 1.2%から 2.0%（この検定は片側検定なので有意水準は 2.5%）、検出力はサンプル数が 20 のとき 5.0%

から 37.0%、200 のとき 100.0%であった。また実データでは、マイクロアレイはサンプルが数個から数十個しかないのに対して説明変数が数千から数十万あるデータの典型として、健康診断データはサンプルが数千から数十万あるのに対して説明変数が数十しかないデータの典型として取り上げている。マイクロアレイによる白血病のサブタイプ予測・癌種予測や、健康診断データによるメタボリックシンドロームリスクの予測を行う場合の統計モデルで K's Plot を描き、統計モデルの性質を視覚的に評価できた。

序文

近年コンピュータの計算能力の発展に伴い、予測を行うための様々なアルゴリズムやモデルが提案されている。これらの予測アルゴリズム・統計モデルは機械学習・統計的学習と呼ばれ、Support Vector Machine（SVM）、Neural Network（NN）、Bayesian Networkなどが代表的な手法である。機械学習は音声認識・画像認識・生体認証・テキストマイニング・遺伝子など幅広い分野で利用されることが多く、様々な種類のデータを対象としており、機械学習を適用する根本的な目的はある現象を予測することである。古典的な統計学では統計モデルを作成するために同時に扱う説明変数は多くても数十程度であった[1]。しかしコンピュータの発展によって極めて大きなデータを簡単に扱うことが出来るようになり、今や多くの分野で説明変数の数が数千、数万、数十万となっている。そ

れに対して多くの研究では、解析対象データの観測数・サンプル数は非常に少なく、数十～数百程度である。そのため一般に“$n \ll m$問題”が起こる[2]。これは変数の数（m）がサンプル数（n）よりも非常に多いという意味であり、統計モデルの性能が著しく低下する等の問題が起こる。

作成したモデルの予測性能は、結果変数をカテゴリ変数とした上で予測結果の誤分類率を計算し、評価することが一般的である。誤分類率は 1 から正解率（検出力）を引いたものなので、正解率を用いて評価することと同値である。誤分類率の計算の際、作成した統計モデルをそのままデータに当てはめるとover-fitting（過適合）が起こるため、真の誤分類率（Err^{true}、真のエラー）を過小評価すると言われている。Efron は、統計モデルをそのままデータに当てはめた誤分類率を apparent error（Err^{app}、見せかけのエラー）と呼び、Err^{app} が Err^{true} を過小評価することを示した[3,4]。さらに Bootstrap や Cross-Validation（CV）によって推定される誤分類率（Err^{boot}、Err^{CV}）にはバイアスが無いことをシミュレーションによって示した[3,5]。Bootstrap と CV は手持ちのデータを学習データと検証データの 2 つに分け、学習データで統計モデルを作成し、検証データで当てはめて誤分類率を評価するという方法である。

しかし、ただ Bootstrap や CV を行えばバイアスが入らないわけではない。Simon はマイクロアレイデータの状況を想定してシミュレーションを行い、CV を説明

変数の選択後に行うと誤分類率を過小評価することを示している[6]。彼が行ったシミュレーションは、1,000 遺伝子（説明変数）を使って、遺伝子と全く関連のない群の変数（結果変数）を予測するものである。説明変数と結果変数に関連が無いので、どんな優れた統計モデルでも誤分類率は 0.5 となるはずである。Simon は説明変数の中から結果変数と相関の高いいくつかの変数を選択し（変数選択）、その変数を用いて予測する（予測）という 2 段階のプロセスで統計モデルを構築した。その際、変数選択を含めて CV を行うと誤分類率は 0.5 付近であったのに対し、変数選択をした後に予測の部分だけを CV すると誤分類率は約 0.1 となり、大きなバイアスが入ることが確認できた。これらの議論によって誤分類率をバイアス無く推定する為には、変数選択を含めた統計モデルの作成工程を全て通して CV を行わなくてはならないことが分かる。

近年、統計モデルは機械学習や生物情報学の分野で多数提案されており、多くの無料解析パッケージも利用することができる。例えばクラスタリング手法だけでも 40 以上の方法があるという報告もある[7]。統計学全般で広く利用されている統計解析ソフト R は、完全無料で利用でき信頼性も高い[8]。マイクロアレイの分野では R での解析パッケージが特に充実しており[9]、また BRB-Array tools という解析ソフトも無料で利用可能である[10]。これらのソフトを利用すればスーパーコンピュータを使わなくても、大規模データに対して複雑なモデルを簡単

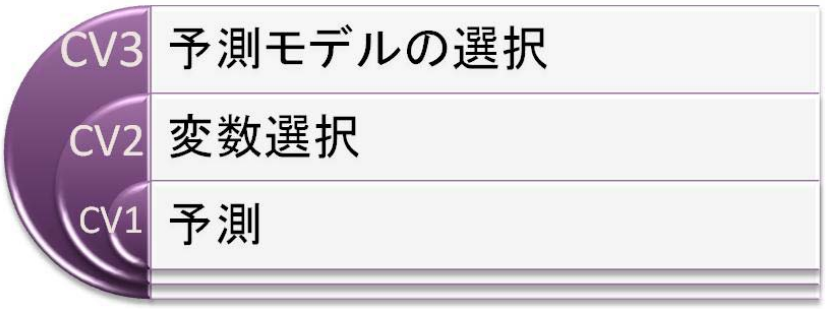
に当てはめることができる。例えば R のパッケージに caret というものがあり、このパッケージでは Bootstrap や CV を行いながら多数の統計モデルを構築することができる[11]。付録の表に示すように実行できるモデルの数は実に多数あり、現存する統計モデルのほとんどは適用することが可能である。

高次元データ（$n \ll m$ のデータ）の場合の予測アルゴリズムは、Simon が行ったシミュレーションのように、変数選択又は変数縮小（以降まとめて変数縮小と表現する）と予測の 2 ステップで構成されることが多い。例えば Golub らはマイクロアレイで急性骨髄性白血病（AML）と急性リンパ性白血病（ALL）を予測するためにシグナル・ノイズ比（標準化統計量）の大きい遺伝子を選択し、選択された遺伝子で判別分析を行った[12]。また乳がんの予後予測を行う Oncotype DX という遺伝子測定キットの統計モデル作成の際は、再発と相関の高い遺伝子を選択後に主成分分析（Principal Component Analysis: PCA）で縮小し、線形モデルで予測を行っている[13]。Tothill らや Kurahashi らは原発不明がんの予測を行うために原発がんのマイクロアレイデータを利用して統計モデルを作成しているが、やはりがん種と相関の高い遺伝子の選択・縮小後、SVM や判別分析で予測を行うという流れになっている[14,15]。

このように統計モデルは変数縮小と予測の無数にある組み合わせから選びながら作成することになる。このときの統計モデルの選択基準は、数理的な仮定

がデータに合っているかや誤分類率が低い（正解率が高い）かなどである。しかし統計モデルが複雑になれば数理的な仮定を確認することが難しくなるため、実際には単純に誤分類率が最も小さい統計モデルを選択している状況が多いであろう。この際、Simon の結果によれば統計モデルの選択全体を通して CV を行わなくてはならないが、ほとんどの研究ではそれぞれの統計モデルの中で CV は行ったとしても、選択まで含めての CV は行なっていないと思われる。つまり、いくつも統計モデルを試して誤分類率を推定して最も低かったモデルを採用すると、例え変数選択を含めた CV を行っていてもバイアスが入ってしまう危険がある。CV を行った結果のうち最も小さい誤分類率を CV selection error（Err^{CVS}）とすると、Err^{CVS} は Err^{true} を過小評価していることを本研究で示す。【図 1】のように、予測のみで行った CV を CV1、変数選択も含めて行った CV を CV2、統計モデルの選択も含めて行った CV を CV3 とする。Simon の研究では CV1 ではバイアスが入るため、CV2 を行うべきだという結論が示された。しかし統計モデルをいくつも試して CV2 を計算し、その中から最も良い誤分類率を選択する（Err^{CVS}）とバイアスになる。そのため CV3 まで行えばバイアスが無いのだが、CV3 を行うためには高度なプログラミングと計算時間が必要になる。そのため CV3 を計算しなくてもバイアスの無い推論が行えるように、本研究では Err^{CVS} が帰無仮説に等しいかどうかの検定方法を提案する。この検定は統計モ

デルの選択バイアスがある状況での誤分類率の検定である。さらに、Err^{CVS} を検定する方法以外で、統計モデルの性能を視覚的に確認する K-sample Plot（K's Plot）という方法も提案する。この方法は Bootstrap や CV 法と違い、サンプル数が増えたときの誤分類の減少と検出力の増加を視覚的に評価できる新しい手法である。



【図 1】Cross Validation（CV）を行う階層

方法

1. データ

本研究では数十から数千の説明変数と 2 値の結果変数から構成されているデータを用いて、統計モデルを構成する状況を想定する。このようなデータは、マイクロアレイやSNPデータによる疾病予測、健康診断や医療データによるリスクスコアの開発などを行うときの構造である。特に今回実データとして用いたものは、マイクロアレイによる白血病のサブタイプの予測、同じくマイクロアレイによる癌種の予測、健康診断データによるメタボリックシンドロームリスクの予測である。前 2 つのマイクロアレイデータはサンプルが数個から数十個しかないのに対して説明変数が数千から数十万あるデータの典型として取り上げ、最後のデータはサンプルが数千から数十万あるのに対して説明変数が数十しかないデータの典型として取り上げている。また本論文での解析は全て統計解析ソフトR 2.11.1 (http://cran.r-project.org/)で行った。

2. 統計モデルの説明

観測値の数が n である 2 値結果変数ベクトル Y を、m 個の連続説明変数行列 X を使って予測する状況を考える（各群のサンプル数は $n/2$ ）。説明変数を利用して変数縮小と予測の 2 ステップで結果変数を予測し、この一連の流れによって作られたモデルを 1 つの統計モデルという。変数縮小で o 種類、予測で p 種類の

手法が統計モデルの候補になっているということは、統計モデルは最大で $N_{alg} = o \times p$ 種類試すことになる。以下に本研究で利用する変数縮小と予測の方法を説明する。本研究ではそれぞれ 4 種類ずつを組み合わせ、16 種の統計モデルを試すこととした。変数縮小の 4 種を A～D、予測方法の 4 種を E～H で示す。

2.1. 変数縮小

A. t 検定

2 群間の t 検定を行い P 値が 0.05 未満の変数を選択する。$P < 0.05$ となる変数が無い場合は最も P 値が小さい変数を 1 つだけ選択する。また $P < 0.05$ となる変数が 10 個以上ある場合は、P 値が小さい変数から 10 変数を選択する。

B. 主成分分析（principal component analysis: PCA）

PCA は変数全体がもつばらつきをなるべく保ったまま、次元を縮小するために利用される。数理的には分散共分散行列の特異値分解を行っている。説明変数行列 X （$n \times m$ 行列）の分散共分散行列（$m \times m$ 行列）を V とすると、以下のように特異値分解を行う。

$$V = B^T AB$$

A は対角成分以外が 0 で対角成分は非負である $m \times m$ 行列である。このとき A の

対角成分a_iを特異値、Bの列ベクトルb_i特異ベクトルという（$i=1,...,rank(B)$）。このb_iを使って計算できるXb_iが主成分スコアとなり、変換前の変数Xのばらつきのうち$a_i/\Sigma_i a_i$の割合（特異値割合）だけのばらつきを表現している。主成分スコアの最大数はVの階級に等しい。実用上は全ての主成分スコアを利用するのではなく、例えば累積特異値割合が 0.8 以上となるように選択されたり、上位いくつかの主成分スコアを選択したりする。本研究では累積特異値割合に応じて決定するのではなく、常に第 5 主成分スコアまでを選択することにする。つまり特異ベクトルによって以下のように次元縮小する。

$$PCA_i = Xb_i,\ i=1,...,5$$

C. Partial Least Squares（PLS）

PLS 法は PCA のように変数の次元を縮小するために利用される方法である。主成分分析と大きく異なる点は結果変数の情報を利用することであり、結果変数と相関が高くなるように説明変数を縮小する。そのため PCA は教師無し法（unsupervised method）、PLS は教師有り法（supervised method）と呼ばれることもある。まず$t_1 = Xw_1$となる PLS スコアt_1を用いて次のようにX、Yの推定値を表現する。

$$\hat{X} = t_1 p_1^T \qquad where \quad p_1^T = \left(t_1 t_1^T\right)^{-1} t_1^T X,$$

$$\hat{Y} = t_1 c_1^T \qquad where \quad c_1^T = \left(t_1 t_1^T\right)^{-1} t_1^T Y$$

ここでt_1は$u_1 = Yq_1$との共分散$t_1^T u_1$が最大となるように最適化される。ここでw_1とq_1はそれぞれ$X^T YY^T X$と$Y^T XX^T Y$の第一固有ベクトルとなっている。次に$X - \hat{X}$と$Y - \hat{Y}$を用いて同様にt_2を推定する。以下同様にしてPLSスコアであるt_iを作っていく。PLSスコアも主成分スコアと同様にVの階級に等しい数だけ推定できるが、全てを利用されることは無く、CVなどで推定された予測誤差が最も小さくなる数だけ利用することが多い。主成分スコアと同じように、本研究ではこのような基準で決定するのではなく、常に第5PLSスコアまでを選択することにする。

$$PLS_i = Xw_i, \ i = 1, ..., 5$$

D. クラスタリングによる次元縮小

クラスタリングは一般的に変数間の距離を定義して計算し、距離が小さいもの同士をまとめる手法である。代表的なクラスタリング手法にWard法があり、この方法での距離は以下の式で定義される。

$$D_{KL} = \frac{\left\| \bar{X}_K - \bar{X}_L \right\|^2}{\left(1/N_K + 1/N_L\right)}$$

ここでK、Lはあるクラスターを表している。これを総当たりで計算していき距

離の小さいものから順に、階層的にまとめていく。Ward 法の距離は分散分析（ANOVA）や線形モデルでの平方和に等しいという性質がある。階層的クラスタリングを利用して変数縮小する場合は、樹形図を長方形に区切るようにして変数をまとめる。本研究ではクラスターが 5 つできるように区切り、同じクラスター内の変数の平均値をそのクラスターのスコアとする。

$$Clus_i = \sum_{k \in C_i} \frac{x_k}{m_i},\ i = 1, ..., 5$$

ここでC_iはあるi番目のクラスター、m_iはクラスターiに含まれる変数の数である。

2.2. 予測

変数縮小 A~D のいずれかの方法で説明変数行列XをX_{red}へと縮小し、縮小行列X_{red}を使ってYの予測を行う。このセクションでは表記を簡便にするため、X_{red}をXと表現する。

E. 判別分析

判別分析は説明変数Xの線形和もしくは非線形変換を用いてスコアを作り、結果変数Yのクラスを判別する方法である。XとYがそれぞれ以下の分布に従っていると仮定する。

$$p(y)=\pi_y,\ p(x|y)=\phi(x;\mu_y,V_y)$$

$\phi(x;\mu,V)$は平均ベクトルμ、分散共分散行列Vの多変量正規分布の確率密度関数である。また$p(y)=\pi_y$は事前確率とも言われる。この各パラメータの最尤推定量は次の対数尤度関数

$$\sum_{i=1}^{n}\log\phi(x_i;\mu_{yi},V_{yi})\pi_{yi}$$

を最大化し、以下のように推定される。

$$\hat{\pi}_y=\frac{n_y}{n},\ \hat{\mu}_y=\frac{1}{n_y}\sum_{i:y_i=y}x_i,\ \hat{\Sigma}_y=\frac{1}{n_y}\sum_{i:y_i=y}(x_i-\hat{\mu}_y)(x_i-\mu_y)^T$$

この推定値を使い、以下のようにYの事後確率を計算する。

$$p(y_j|x_i)=\frac{p(x_i|y_j)\pi_{y_j}}{\sum_{j:y_j=y}p(x_i|y_j)\pi_{y_j}},$$

$$p(x_i|y_j)\pi_{y_j}=\exp\left(-0.5D_{x_iy_j}-0.5\ln|V_{y_j}|-0.5M\ln 2\pi+\ln\pi_{y_j}\right),$$

$$D_{x_iy_j}=(x_i-\hat{\mu}_{y_j})^T V_{y_j}^{-1}(x_i-\mu_{y_j})$$

この事後確率の最も大きいクラスに分類するのが判別分析である。このモデルでの判別分析は 2 次判別分析と呼ばれ判別面は 2 次曲面となる。また$V_y=V$のように各クラスの共分散行列が等しいと仮定すると判別面は超平面となり、線形判別分析と呼ばれる。本研究ではこの線形判別分析を適用する。

F. Support Vector Machine（SVM）

SVM は 2 群間のマージンが最も大きくなるように判別面を構成する方法である。マージンとは各群から判別面までの距離である。まず以下の関数をカーネル関数と呼ぶ。

$$K_{ij} = k\left(x_i, x_j\right) = \phi\left(x_i\right)^T \phi\left(x_j\right)$$

SVM はカーネル関数で表現される式を次のように最小化することでパラメータを推定する。

$$\min_{\alpha \in R^p, b \in R} \frac{1}{2}\alpha^T K \alpha \quad \text{subject to} \quad y_i\left(\sum_{j=1}^{n} \alpha_j K_{ij} + b\right) \geq 1, \; i = 1, \ldots, n$$

この解を $\hat{\alpha}$ 、$\hat{b}$ とすると判別面は以下の関数となる。

$$\sum_{i=1}^{n} \hat{\alpha}_i k\left(x_i, x\right) + \hat{b} = 0$$

カーネル関数はいくつかの関数が提案されている。

線形カーネル： $k\left(x_i, x_j\right) = x_i^T x_j$

多項式カーネル： $k\left(x_i, x_j\right) = \left(\gamma x_i^T x_j + \delta\right)^d$

ガウシアンカーネル： $k\left(x_i, x_j\right) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right)$

シグモイドカーネル： $k\left(x_i, x_j\right) = \tanh\left(\gamma x_i^T x_j + \delta\right)$

本研究では多項式カーネルを使い、次数は 3 とする。

G. K 近傍法

K 近傍法はモデルの指定は行わず、パラメータ推定もしないノンパラメトリックな予測方法である。テストサンプル x_i と学習サンプル x_j のユークリッド距離を計算し、近い順に K 個学習サンプルを選ぶ。選ばれた学習サンプルのラベル Y の最も多いクラスにテストサンプルを分類する。最多のラベルが同数の場合はランダムに割りつける。 K の数は CV を行ったときの誤分類率が最も小さい値を利用することもあり、 $K=1$ の場合は one-nearest neighbor 法と呼ばれることもある。本研究ではこの $K=1$ とした K 近傍法を用いる。

H. Neural Network（NN）

NN は脳の神経細胞の働きを数理モデルで表現したものである。Rosenblatt がパーセプトロンのモデルを発表し、一度は Minsky らに実用性が疑問視された。しかし Rumelhart らが階層型ネットワークモデルと逆伝播学習の提案をして以来、研究が急速に発展している[16,17,18]。層の数が L 、第 k 層の次元が n_l であるとし、第 l 層のベクトルを $o^{(l)}=\{o_1^{(l)},...,o_{n_l}^{(l)}\}^T$ と表現すると、階層型ネットワークでは各層で次のように計算される。出力層の次元 n_L はクラスの数に等しくなる。

$$\text{入力層：}\ o^{(1)}=\phi^{(1)}\left(x\right)$$

$$\text{中間層：}\ o^{(l)}=\phi^{(l)}\left(\sum_{l'<l}W^{ll'}o^{(l')}-h^{(l)}\right),\ l=2,...,L-1$$

出力層：$o^{(L)} = \sum_{l'<L} W^{Ll'} o^{(l')} - h^{(L)}$

$W^{ll'}$は一つ前の層までの重みを結合したものであり、$h^{(l)}$はある閾値である。このとき関数$\phi^{(l)}$は次のようなシグモイド関数が使われることが多い。

$$\phi(z) = \frac{1}{(1+\exp(-z))}$$

これはロジット変換の逆変換である。クラスの予測をするときはこの出力層の値を使って次のようにソフトマックス関数と呼ばれる関数で計算し、最も高いクラスへと分類する。

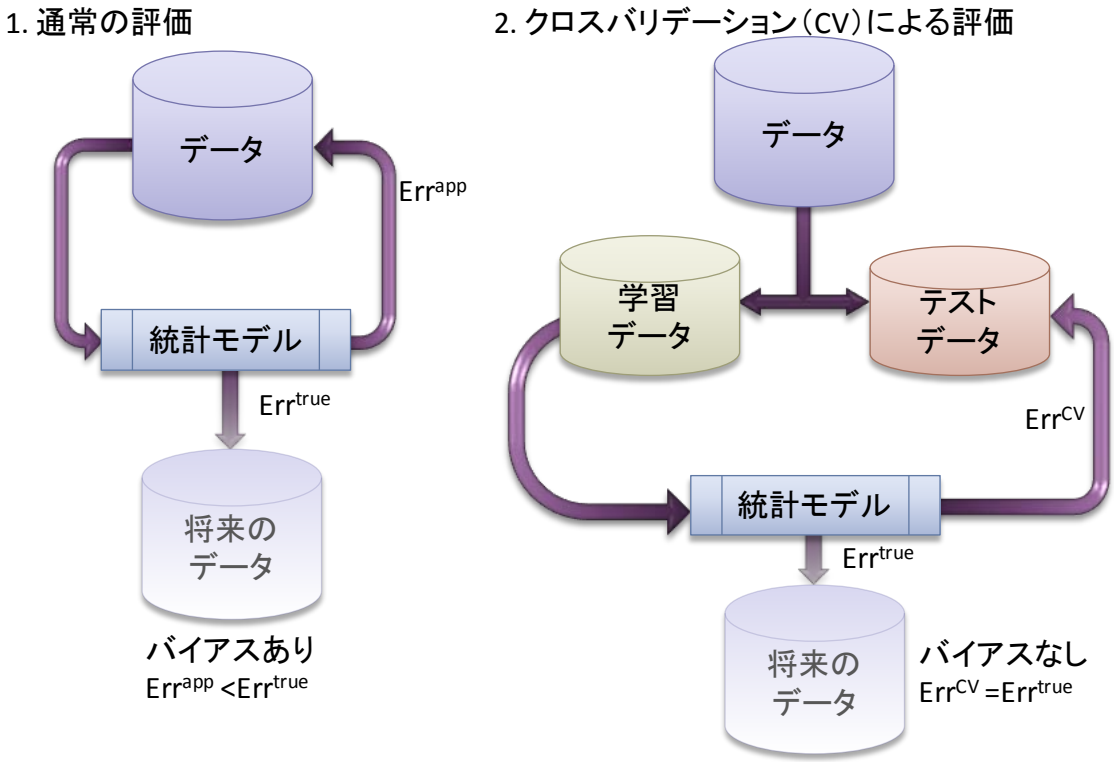
$$f_j(x_i;\theta) = \frac{\exp(o_j^{(L)})}{\sum_{j=1}^{n_L} \exp(o_j^{(L)})}$$

ここでθはすべての結合荷重$W^{ll'}$と閾値$h^{(l)}$をまとめて表現したものである。このθの推定に誤差逆伝播学習を利用することで、複雑なパラメータ推定を行うことができる。層の数は全部で 3 層にすることが多く、本研究でも 3 層の NN を行う。

3. 誤分類率の推定と検定

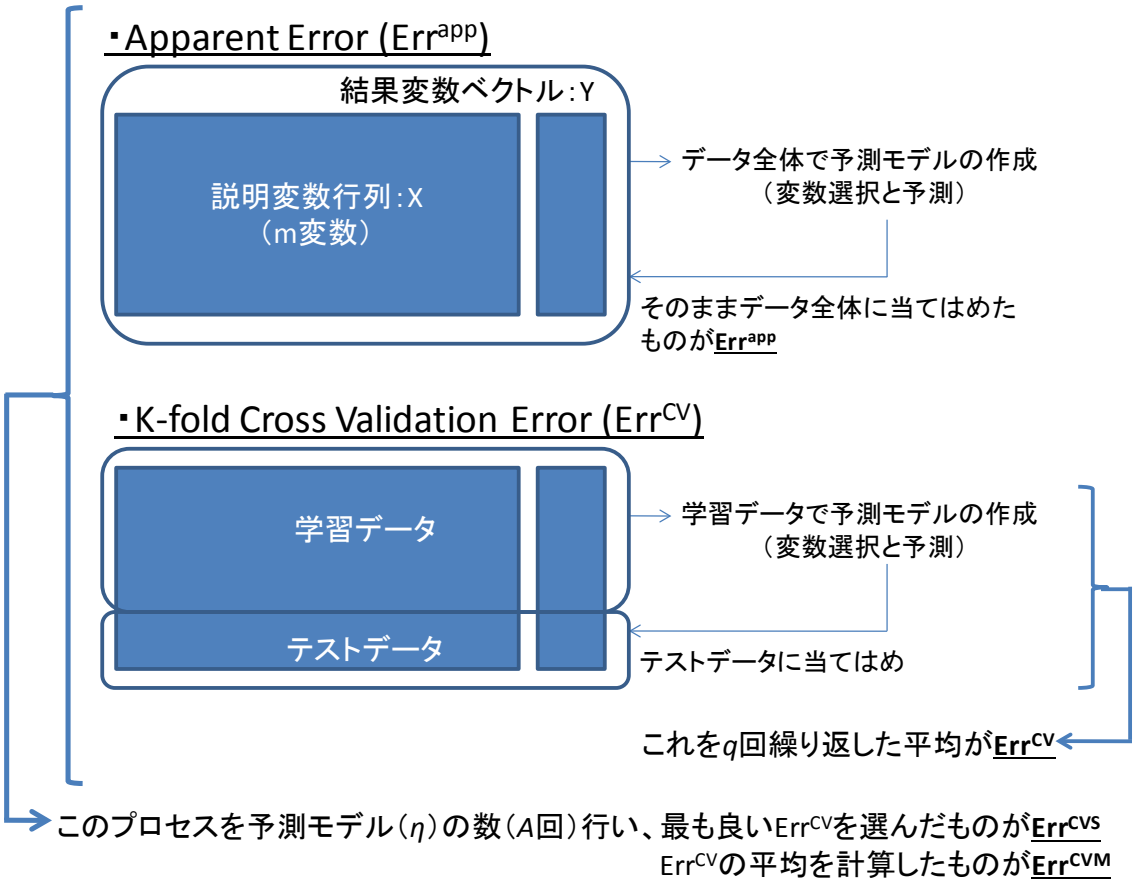
統計モデルを作成するデータを学習データ、作成した統計モデルの誤分類率を評価するデータを検証データと呼ぶ。誤分類率の推定は q-fold Cross Validation（q-fold CV）法で行われることが多いが、この方法はデータを q 個に等分割し

$n(q-1)/q$ 個の観測値を学習データ、残りの n/q 個の観測値を検証データとし、統計モデルの作成と評価を q 回行うものである。$q=n$ の場合は leave-one out CV（LOOCV）と呼ばれる。作成した統計モデルで将来の未知データを予測した場合の誤分類率が真の誤分類率であり、Err^{true} と表現する。データ全てを学習データかつ検証データとして利用した場合の誤分類率を Err^{app} 、q-fold CV で求めた誤分類率の q 個の平均を Err^{CV} とし、各 CV での誤分類率を Err_q^{CV} で表す。Err^{true} と Err^{app} 、Err^{CV} の関係を【図 2】に示す。通常はデータを全て使用して統計モデルを作成しそのまま性能を評価するが、これでは Err^{true} にバイアスが入る [4]。そこで CV を行うことでこのバイアスを解消することが必要となる [3,6]。



【図 2】 Err^{true} と Err^{app} 、Err^{CV} の関係

Err^{app} と Err^{CV} は統計モデルの試した数だけ推定され、添え字 $a(a=1,...,A)$ で表現する。つまり一回の統計モデルを適用し誤分類率を評価する場合、Err^{app} が 1 つ推定される。また q-fold CV で Err_q^{CV} が q 回計算されるのでその平均で Err^{CV} を求める。この一連の計算を統計モデルの回数だけ行うことになる。統計モデルを A 回試す場合は、Err^{app} と Err^{CV} は A 個ずつ推定される（Err_q^{CV} は $A\times q$ 個）。次に統計モデルを η で表現し、全データを使って作成した統計モデルを η_a^{app}、CV で作成した q 個の統計モデルを総合して η_a^{CV} と書く（各 CV で作成されたそれぞれの統計モデルを表現するときは添え字の q を付ける）。また今回行う CV は、全て序文で述べた変数選択も含めて行う CV2 である。以上の内容と、後述する Err^{CVS} と Err^{CVM} の関係を【図 3】に図示する。



【図 3】統計モデルの構築方法と各誤分類率の関係

ここで誤分類率を推定するため、i 番目の観測値 x_i を統計モデル η に代入して得られる予測値が、結果変数 y_i と等しいかどうかを表す指示関数を以下のように定義する。

$$L\left(y_i, x_i, \eta_a\right) = 0 \qquad if \quad \eta_a\left(x_i\right) = y_i$$

$$1 \qquad if \quad \eta_a\left(x_i\right) \neq y_i$$

このとき Err^{app} と Err^{CV} は次のように計算できる。

$$Err^{app} = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, x_i, \eta_a^{app}\right)$$

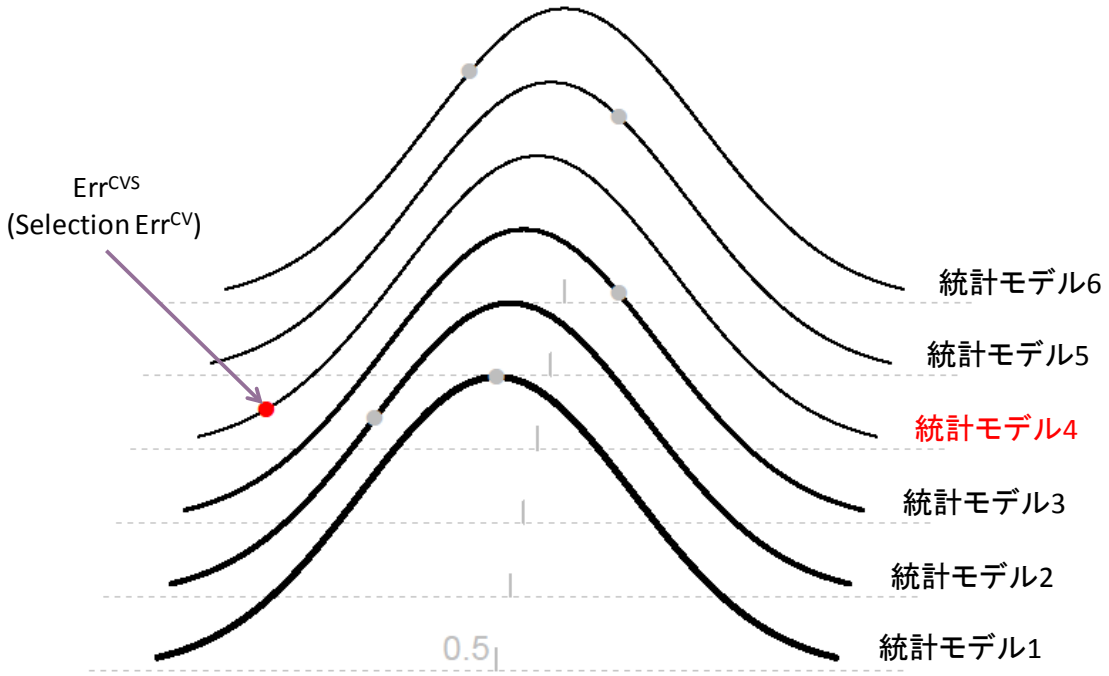
$$Err^{CV} = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, x_i, \eta_a^{CV}\right) = \frac{1}{Q}\sum_{q=1}^{Q}\frac{Q}{n}\sum_{i=1}^{n/Q} L\left(y_i, x_i, \eta_a^{CV}\right)_q \qquad (1)$$

$L(y_i, x_i, \eta_{aq}^{CV})$ と書いている場合、統計モデル η_{aq}^{CV} を作成するときの学習データに x_i は含まれていないものとする。Efron や Simon の研究によれば、Err^{CV} は真の誤分類率に対して平均的にバイアスがない。また 3.1 で述べるように、Err^{CV} は二項分布に従っている。

次に統計モデルをいくつか試し、その中で一番小さい Err^{CV} を選んだときの誤分類率を Err^{CVS}（CV selection error）と表現すると、Err^{CVS} は次のように表すことができる。

$$Err^{CVS} = \min_{a=1}^{A}\left(Err_a^{CV}\right) \qquad (2)$$

これは二項確率 Err^{CV} の極値分布になっている[19]。Err^{CV} 自体にバイアスがなくてもばらつきはあるため、その最小値を選択している Err^{CVS} は真の誤分類率を過小評価する。この状況を【図 4】に示す。それぞれの統計モデルの誤分類率はある真値（この図では 0.5）のまわりをばらつくが、いくつもモデルを試して最も良いものを選ぶと図に示すようにバイアスとなってしまう。



【図 4】統計モデルの選択によって起こるバイアス。誤分類率の真値が 0.5 の統計モデルを 6 回試した状況。この場合は統計モデル 4 が最も良い結果となっているが、それを選択するとバイアスとなる。

本研究では以下のように正規説明変数行列 X と結果変数 Y に全く関連が無い場合（Type I）と、関連がある場合（Type II）の 2 種の状況を考える。また CV は q=10 の 10-fold CV を行う。

3.1. Type I の状況での推定

まず Type I の状況を考える。どんな統計モデルを試しても、この状況では X が Y を説明することができないので、真の誤分類率は $Err^{true} = 0.5$ である。Err^{CV} は平均的にはバイアスは無く、Err^{true} を中心にランダムにばらつく。Y の予測が正解するかどうかはランダムであるが、サンプル同士で予測結果に相関があると

し、$L(y_i, x_i, \eta_a^{CV})$は確率 0.5 の相関のあるベルヌーイ分布に従っていると仮定する。

$$L\left(y_i, x_i, \eta^{CV}\right) \sim Be\left(0.5\right),\ r_{ij} = Co\ m\left(L\left(y_i, x_i, \eta^{CV}\right), L\left(y_j, x_j, \eta^{CV}\right)\right) \quad (3)$$

すると式(1)よりErr_a^{CV}の分布は以下のように二項分布で表現できる。

$$Err_a^{CV} \sim \frac{Bi\left(n, 0.5\right)}{n}$$

同じような性質の統計モデルを使うと似たような予測結果が推定されるため、η_aとη_bに方法論的な関連があればErr_a^{CV}とErr_b^{CV}に相関が生じる。Err^{CV}の相関と$L(y_i, x_i, \eta^{CV})$の相関は等しいので、次のように表現する。

$$r_a = Corr\left(Err_a^{CV}, Err_b^{CV}\right) = Corr\left(L\left(y_i, x_i, \eta_a^{CV}\right), L\left(y_i, x_i, \eta_b^{CV}\right)\right) \quad (4)$$

式(3)と式(4)より、$\{Err_1^{CV}, ..., Err_A^{CV}\}$は以下のような入れ子型の相関行列$R$を持つ多変量ベルヌーイ分布の和を$n$で割った二項確率となる。

$$R = \begin{pmatrix} R_1' & & & \\ & R_a' & R_{ab} & \\ & & R_b' & \\ & & & R_A' \end{pmatrix}, R_{ab} = \begin{pmatrix} r_{ab} & & & \\ & \ddots & r_{ab} & \\ & & \ddots & \\ & & & r_{ab} \end{pmatrix}, R_a' = \begin{pmatrix} r_{a_{11}} & & & \\ & r_{a_{ii}} & r_{a_{ij}} & \\ & & r_{a_{jj}} & \\ & & & r_{a_{nn}} \end{pmatrix}$$

統計モデル間相関係数行列R_{ab}、サンプル間相関係数行列R_a'は$n \times n$行列であり、全体の相関係数行列Rは$nA \times nA$行列である。確率 0.5、相関行列Rで発生させたnA個のベルヌーイ分布をn個ずつ足して、サンプルのn数で割ったものがそれ

ぞれの Err^{CV} となる。

よってこの A 個の二項分布 Err_a^{CV} を $P_a(n, 0.5, R'_a)$ とすると、式(2)より Err^{CVS} は次のようになる。

$$Err^{CVS} = \min_a \left(P_a(n, 0.5, R'_a) \right)$$

多くの論文ではいくつかの統計モデルを試し、最も良い結果のものを公表しているので、この誤分類率を推定し公表していることになる。次に以下のように、試した統計モデル全ての Err_a^{CV} の平均値を Err^{CVM}（CV mean error）とし、この推定値の性質も確認する。

$$Err^{CVM} = \frac{1}{A} \sum_{a=1}^{A} Err_a^{CV}$$

Err_a^{CV} には平均的にバイアスがないため、Err^{CVM} にもバイアスはない。以上の議論から分かることは、Err^{CVS} と Err^{CVM} は n の関数になっているが m の関数にはなっていない。そのため Type I の状況ではこれらの誤分類率は変数の数 m には依存せず、バイアスが入る大きさも、たとえ $n \ll m$ の状況であっても変わらない。

3.2. Type II の状況での推定

次に X と Y に関連がある Type II の場合の議論を行う。統計モデルが X と Y との関連を正しく特定していれば、誤分類率の推定値は Err^{true} を用いて次のように表現できる。

$$Err_a^{CV} \sim P_a\left(n, Err^{true}, R_a'\right),$$

$$Err^{CVS} = \min_a\left(Err_a^{CV}\right),$$

$$Err^{CVM} = \frac{1}{A}\sum_{a=1}^{A} Err_a^{CV}$$

しかし統計モデルには、説明変数と結果変数の関係を正しく特定できるものもあれば、できないものもある。よって X と Y の関連や XY 間のモデルを正しく特定できている統計モデルを superior algorithm（SA）、特定できていない統計モデルを inferior algorithm（IA）と区別する。また SA を使って予測したときの誤分類率を Err^{SA}、IA で予測したときの誤分類率を Err^{IA} とする。すると Err^{SA}、Err^{IA} と Err^{true} の関係は以下のようになる。

$$Err^{SA} = Err^{true},\ Err^{IA} > Err^{true}$$

X と Y の関連を正しく特定できる統計モデルで推定した誤分類率は真の誤分類率に等しく、正しく特定できていないもので推定した誤分類率は過大評価してしまう。つまり、SA と IA が混在している状況では、CV で推定した誤分類率は次の分布に従うことになる。

$$Err_{SA}^{CV} \sim \frac{Be\left(n, Err^{true}\right)}{n},\ Err_{IA}^{CV} \sim \frac{Be\left(n, Err^{IA}\right)}{n}$$

よって Err^{CVS} は次のような分布に従う。

$$Err^{CVS} = \min\left(P_1\left(n, Err^{true}\right), ..., P_{a_{SA}}\left(n, Err^{true}\right), P_1\left(n, Err_1^{IA}\right), ..., P_{a_{IA}}\left(n, Err_{a_{IA}}^{IA}\right)\right),$$

$$a_{SA} + a_{IA} = A$$

a_{SA} と a_{IA} はそれぞれ SA と IA である統計モデルの数である。

つまり Type II の状況では Err^{CVS} と Err^{CVM} の性質は SA と IA がどの程度含まれるのか、また IA の誤分類率 Err^{IA} がどの程度高いのかに依存することになる。もし SA の数（a_{SA}）が少なく、かつ Err^{IA} が高ければ Err^{CVS} にはほとんどバイアスが入らず、Err^{CVM} には IA のせいで過大評価のバイアスが入ることが予想される。反対に SA の数（a_{SA}）が多ければ Type I のときと同じように Err^{CVS} に過小評価のバイアスが入り、Err^{CVM} にはバイアスがないことが予想できる。以上の議論より予想される、Type I の状況と Type II の状況での Err^{CVS} と Err^{CVM} の性質を【表 1】にまとめた。

	algorithm	Err.CVS	Err.CVM
Type I		過小評価	バイアスなし
Type II	SAばかり	過小評価	バイアスなし
	IAを含む	過小評価～バイアスなし	過大評価
	IAばかり	過大評価	過大評価

【表 1】Type I と Type II の状況での Err^{CVS} と Err^{CVM} の性質

ある統計モデルが真の状況を特定できるかどうかは、サンプル数 n と変数の数 m に依存する。例えば説明変数のうち少数の変数が結果変数と線形な関連がある場合、t 検定でこれを選択するのが最も良い。しかし n が少なければ検出力は小さくなり、一方で m が大きくなれば間違って検出される変数が多くなる。このような状況では真のモデルを特定できず、t 検定で変数選択された変数で作成

する統計モデルも IA になってしまう。つまり SA を作るための条件は、正しい構造を数理的に表現できているか、サンプル数 n は多いか、変数の数 m は少ないか（または関連のない変数の数が少ないか）ということが挙げられる。つまり Err^{SA}、Err^{IA} の性質は数理的な正しさと n と m、またはその比（m/n）に依存する。

3.3. Err^{CVS} の検定

Err^{CVS} を検定するためには、帰無仮説（Type I）の状況での Err^{CVS} の期待値と標準誤差が分かれば良い。統計モデルによる予測は 2 値ベルヌーイ試行の繰り返しなので Err^{CV} は二項確率に従う。また二項分布は正規分布で近似できることは良く知られている事実である。Err^{CVS} の期待値を $E\left(Err^{CVS}\right)$、標準誤差を $\sigma_{Err^{CVS}}$ とすると、正規近似による片側有意水準 2.5%の検定方式は次のように表現することができる（誤分類率は過小方向のバイアスしか入らないので、検定は片側で行う）。

$$if \quad \hat{E}rr^{CVS} + 1.96 * \sigma_{Err^{CVS}} < E\left(Err^{CVS}\right) \quad then \quad reject \quad H_0$$

これは、推定された $\hat{E}rr^{CVS}$ に $1.96 * \sigma_{Err^{CVS}}$ を足しても $E\left(Err^{CVS}\right)$ よりも小さければ、帰無仮説（H_0）を棄却できるということである。H_0 を棄却するということは、得られたデータが Type II であり、説明変数と結果変数間の関連を統計モデルで

特定できているということを示す。ただしαエラー（第一種の過誤）は2.5%である。$\sigma_{Err^{CVS}}$は帰無仮説の状況でのErr^{CVS}の標準誤差なので、3.1で述べたように相関のある二項確率の極値分布から求めることができる。このような分布の分布関数はこれまで数式として明示されていないが、相関係数を適当な値に設定して、乱数発生により極値分布を発生させることによって、$\sigma_{Err^{CVS}}$を推定することができる。本論文では、乱数発生により推定した$\sigma_{Err^{CVS}}$を使ってErr^{CVS}の検定をすることを提案する。

3.4. K-sample Plot（K's Plot）による誤分類率の視覚的評価

3.1、3.2の議論により、Err^{CV}には平均的にバイアスが入らず、Err^{IA}の過大評価はサンプル数が増えると小さくなることがわかった（どんな劣った統計モデルでも、サンプル数が増えると幾分かは予測が成功するようになる）。つまり、Type Ⅰの状況ではnが変化してもErr^{CV}は常に$Err^{true}=0.5$のまわりをばらつき、Type IIの状況ではnが大きくなるに従ってErr^{IA}は$Err^{true}=Err^{SA}$に近づいていく。この性質を利用すると次に示すような図を確認することによって、統計モデルの誤分類率を視覚的に評価することができる。例えば、ある統計モデルの誤分類率をCVで$Err^{CV}=0.3$と推定したとき、この値が$Err^{true}=0.5$のばらつきから得られたものなのか、もっと小さいErr^{true}を過大評価しているのかを視覚的に確認

できる。

次のような手順で統計モデルの性能を視覚的に評価する図を、K-sample Plot（K's Plot）として提案する。n 個の観測値のうち、k_i 個を抜き出したデータを X^{k_i}、Y^{k_i} とし、このデータから CV によって推定した誤分類率を $Err^{CV_{k_i}}$ とする（本研究では 10-fold CV で推定している、k_i が小さく 10-fold CV が行えない場合は Leave-one-out CV を行っている）。k_i の数を変化させていくつか $Err^{CV_{k_i}}$ を計算し、$Err^{CV_{k_i}}$（y 軸）と k_i（x 軸）のプロットを作成する。するとこのプロットは、Type I の状況では k_i が大きくなっても $Err^{CV_{k_i}}$ は 0.5 のまわりをばらつき、Type II の状況では k_i が大きくなっていくに従って Err^{true} に漸近する曲線となる。結局は、この曲線は検出力が上がっていく様子を表現しているのであるが、複雑な統計モデルを適用するときは検出力を計算で求めることは難しい。具体的なサンプルの選び方は次のようなものを提案する。k_i は n の 10%～100%までの 10 回行う（偏りのないようにランダムに選ぶ）。プロットを描き $Err^{CV_{k_i}}$ が減少していく様子が見られないようであれば、1%～10%までの 10 回を追加してまたプロットを描く。これを解釈可能なプロットが得られるまで繰り返す。結果でも示すように、n が小さい場合は最初の 10 回で統計モデルの性質を評価できるが、n が大きい場合は k_i を十分に小さくしなければ評価が難しい。

次にこのプロットに以下の指数関数モデルを当てはめる（one-phase

exponential decay model、指数関数モデル）[20]。

$$\hat{Err}^{CV_{k_i}} = (0.5-\alpha)\exp(-\beta k_i)+\alpha$$

このモデルは、$k_i = 0$のとき$Err^{CV_{k_i}} = 0.5$から、k_iが大きくなってくるに従ってαに漸近する指数関数になっている。このモデルはサンプルが増えるに従って誤分類率が減っていくという$Err^{CV_{k_i}}$の性質を良く捉えている。パラメータのαとβは以下のように最小二乗法によって推定する。

$$\min_{\alpha,\beta} \sum_i \left(Err^{CV_{k_i}} - \hat{Err}^{CV_{k_i}} \right)^2 \qquad (5)$$

これによって推定されるαがErr^{true}の推定値になっており、βがErr^{true}に漸近する速さを表している。式(5)だけでは推定が不安定で上手くいかない場合は、以下の導関数も同時に最適化する。

$$\min_{\alpha,\beta} \frac{\partial}{\partial\alpha} \sum_i \left(Err^{CV_{k_i}} - \hat{\hat{Err}}^{CV_{k_i}} \right)^2 + \frac{\partial}{\partial\beta} \sum_i \left(Err^{CV_{k_i}} - Err^{CV_{k_i}} \right)^2$$

このとき、サンプルが増えるにつれて誤分類率が大きくなることはあり得ないので、α は 0.5 よりも大きくならないように制約を加えている。

CV や Bootstrap 法では全ての観測値を使った誤分類率しか推定できず、統計モデルの性能を視覚的に確認することは不可能であったが、この方法によっては可能となる。しかしnが少ないときはパラメータに過誤が起こりやすいことが予想される。例えば Type I の状況では$\alpha = 0.5$付近の値が推定されるはずである

が、データによっては$\alpha = 0.01$と推定されてしまうこともある。そのためサンプル数が少ない場合は、この 指数関数モデルで推定される α はErr^{true}の推定値というよりは 1 つの目安に留め、3.2 のErr^{CVS}の検定によって推論を行うべきだと考える。

結果

1. 誤分類率の推定と検定のシミュレーション

1.1. Type I の状況

説明変数行列Xは$n \times m$行列で、互いに独立な正規分布に従う。結果変数ベクトルYはランダムに 0 か 1 であるクラス変数であり、クラスの数はそれぞれ$n/2$で等しい。よってXとYには関連が全くないため、どのような統計モデルでも真の誤分類率は 0.5 である。統計モデルはこれまで述べた方法の組み合わせで 16 種類試している。このときのErr^{app}、Err^{CV}、Err^{CVS}、Err^{CVM}を【表 2】に示す。CV は 10-fold CV を行っており、設定 1：$n = 20, m = 100$、設定 2：$n = 20, m = 1000$、設定 3：$n = 200, m = 100$、設定 4：$n = 200, m = 1000$の 4 種の状況でシミュレーションしている。シミュレーション回数はどれも 2,000 回である。統計モデルは変数縮小と予測の組み合わせを表しており、たとえば BG は PCA で 5 変数に縮小した後にK近傍法で予測を行っている。表の Apparent、CV 、

CVS、CVM の値がそれぞれ Err^{app} 、Err^{CV} 、Err^{CVS} 、Err^{CVM} であり、Theoretical の値が 2.1 で述べた理論上の二項分布から乱数を発生させて計算した誤分類率を示している。Err^{T-CVS} と Err^{T-CVM} （“T-”は Theoretical の意）の平均と標準偏差は、相関のある二項確率を統計モデルの回数（ここでは 16 回）だけ発生させて最小値と平均値を計算し、この操作を 2,000 回行ったときの平均と標準偏差である。このとき、相関のある多変量二項分布を発生させる場合は、標準多変量正規分布をまず乱数発生させ、正の値であれば 1、負の値であれば 0 になるように 2 値に変換している。また多変量正規分布を発生させる際に、相関がある場合とない場合を設定している。また表の太字は Apparent の中で最も小さい値を示している。

設定 1 が本シミュレーションの結果を代表的に示しているので、この設定を主に確認する。Err^{app} は、CE～CH の PLS で次元縮小したものと、AG・BG・CG・DG の K 近傍法で予測を行ったときの統計モデルでほとんど 0 と予測されている。これらの方法を使った Err^{app} で誤分類率を評価するとかなりの過小評価が起こることが分かる。またその他の方法でも全体的に過小評価していることが分かる。それに対して Err^{CV} では 0.5 付近に推定されており、どの統計モデルでもバイアスが無い。

			設定1		設定2		設定3		設定4	
	n,m		20, 100		20, 1000		200, 100		200, 1000	
	model		Mean	SD	Mean	SD	Mean	SD	Mean	SD
Apparent	AE	(Ttest.lda)	0.095	0.087	0.030	0.040	0.356	0.040	0.251	0.023
	AF	(Ttest.svm)	0.127	0.091	**0.000**	0.004	0.306	0.084	0.123	0.020
	AG	(Ttest.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	AH	(Ttest.nnet)	0.038	0.065	0.000	0.003	0.292	0.066	0.170	0.048
	BE	(PCA.lda)	0.289	0.092	0.283	0.092	0.439	0.029	0.439	0.028
	BF	(PCA.svm)	0.164	0.069	0.165	0.067	0.347	0.030	0.347	0.030
	BG	(PCA.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	BH	(PCA.nnet)	0.192	0.078	0.230	0.089	0.380	0.029	0.384	0.032
	CE	(PLS.lda)	**0.000**	0.000	**0.000**	0.000	0.161	0.027	**0.000**	0.000
	CF	(PLS.svm)	0.015	0.032	0.014	0.032	0.141	0.026	**0.000**	0.000
	CG	(PLS.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	CH	(PLS.nnet)	**0.000**	0.000	0.001	0.006	0.084	0.033	**0.000**	0.001
	DE	(Clus.lda)	0.286	0.093	0.287	0.093	0.438	0.028	0.438	0.029
	DF	(Clus.svm)	0.197	0.068	0.208	0.069	0.349	0.029	0.355	0.028
	DG	(Clus.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	DH	(Clus.nnet)	0.171	0.089	0.173	0.102	0.395	0.060	0.397	0.070
CV	AE	(Ttest.lda)	0.504	0.166	0.500	0.138	0.501	0.052	0.500	0.052
	AF	(Ttest.svm)	0.500	0.144	0.497	0.106	0.500	0.044	0.500	0.050
	AG	(Ttest.knn)	0.499	0.151	0.497	0.147	0.500	0.038	0.501	0.040
	AH	(Ttest.nnet)	0.502	0.158	0.502	0.157	0.501	0.045	0.501	0.048
	BE	(PCA.lda)	0.501	0.135	0.498	0.138	0.498	0.043	0.501	0.042
	BF	(PCA.svm)	0.500	0.031	0.500	0.007	0.499	0.030	0.501	0.013
	BG	(PCA.knn)	0.501	0.095	0.501	0.062	0.500	0.035	0.500	0.033
	BH	(PCA.nnet)	0.498	0.111	0.499	0.114	0.499	0.035	0.499	0.034
	CE	(PLS.lda)	0.503	0.140	0.498	0.142	0.500	0.044	0.499	0.044
	CF	(PLS.svm)	0.504	0.074	0.500	0.011	0.500	0.043	0.499	0.041
	CG	(PLS.knn)	0.501	0.134	0.503	0.093	0.500	0.042	0.499	0.044
	CH	(PLS.nnet)	0.501	0.135	0.497	0.134	0.500	0.042	0.500	0.043
	DE	(Clus.lda)	0.501	0.125	0.500	0.115	0.501	0.040	0.499	0.038
	DF	(Clus.svm)	0.502	0.055	0.500	0.027	0.501	0.032	0.500	0.028
	DG	(Clus.knn)	0.503	0.104	0.501	0.080	0.500	0.036	0.500	0.035
	DH	(Clus.nnet)	0.504	0.099	0.498	0.075	0.499	0.027	0.499	0.023
CVS			0.321	0.092	0.324	0.086	0.440	0.026	0.440	0.027
CVM			0.502	0.065	0.500	0.048	0.500	0.019	0.500	0.018
Theoretical										
CVS	相関なし		0.305	0.061	0.306	0.060	0.438	0.019	0.439	0.019
	相関考慮		0.296	0.076	0.298	0.078	0.428	0.029	0.430	0.028
CVM	相関なし		0.500	0.028	0.501	0.027	0.500	0.009	0.500	0.009
	相関考慮		0.499	0.063	0.499	0.066	0.500	0.022	0.501	0.021

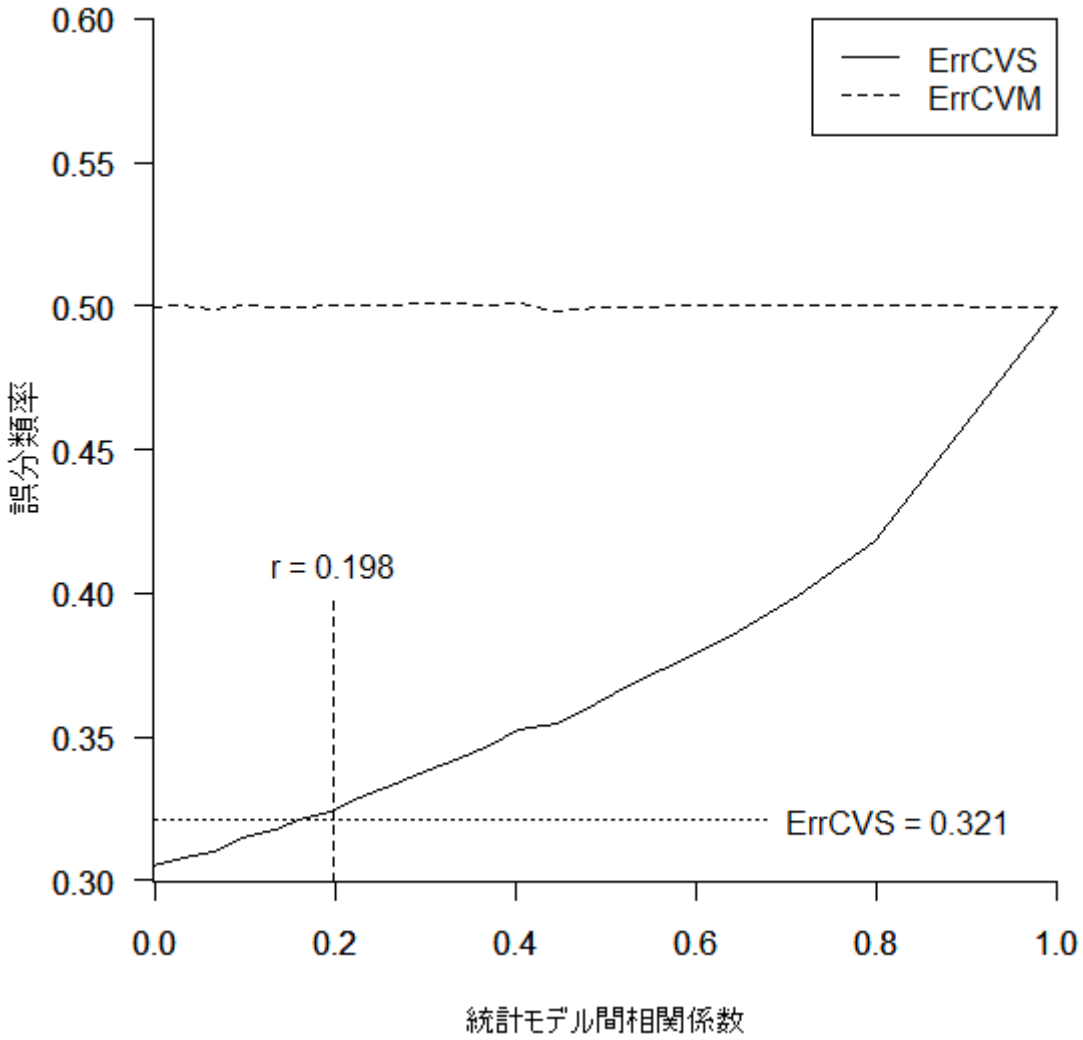
【表 2】Type I の状況での Err^{app} 、Err^{CV} 、Err^{CVS} 、Err^{CVM} のシミュレーション結果と、理論分布を発生させ計算した各誤分類率。太字は Apparent の中で最小値を示している。

このようにどのような統計モデルを使っても Err^{CV} は平均的にバイアスが無いことが示されたが、最も良い統計モデルを選んでいる Err^{CVS} の平均値は 0.321 となっており、真値を約 0.18 過小評価している。Err^{CV} にバイアスがなくても、

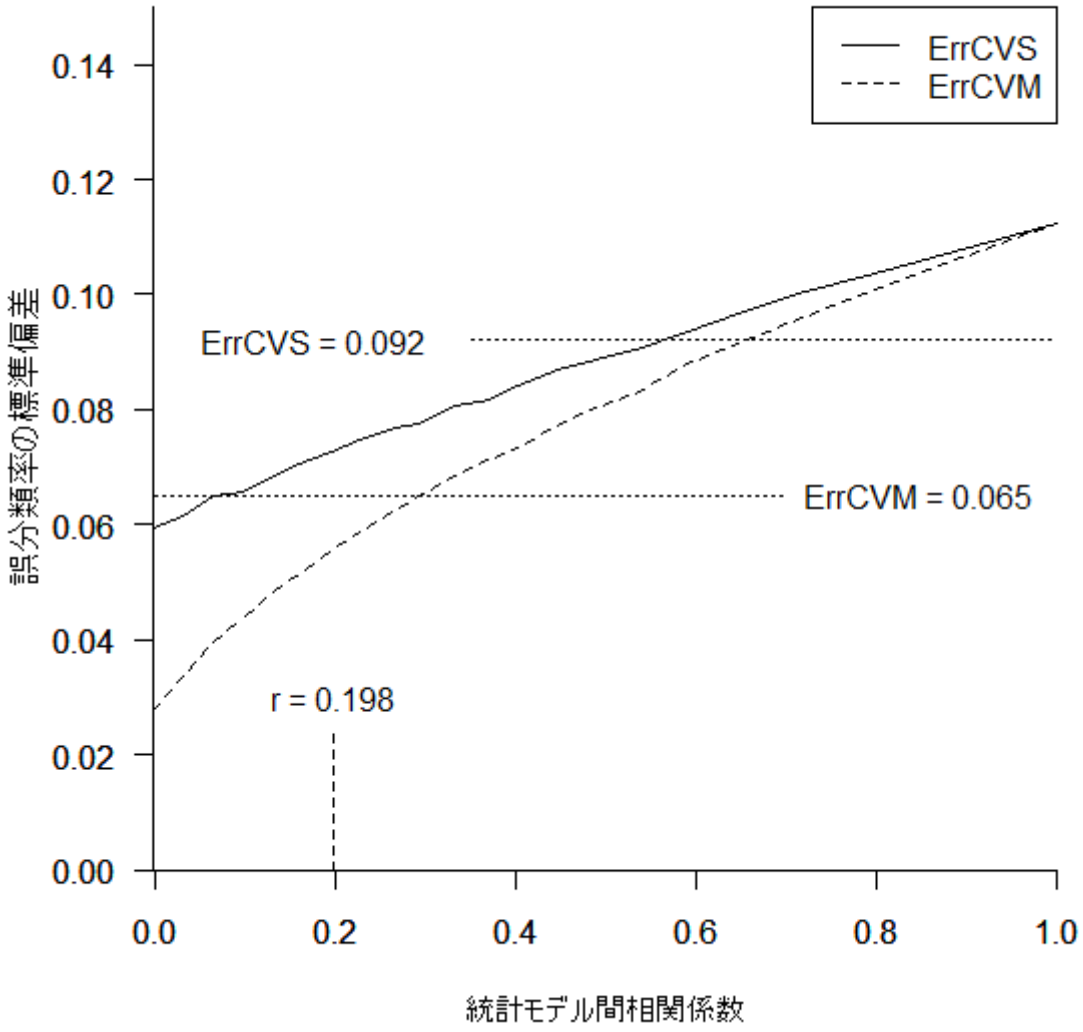
ばらついているためにたまたま良い予測が行われた結果を選択しているためである。また多くの統計モデルを試しているために、バイアスが大きくなっているとも言える。表には記載していないが、AE～AH の 4 種類だけを試した場合の Err^{CVS} は 0.426 であった。統計モデルを試す回数が大きくなるほど、このバイアスも大きくなる。

【図 5】は、統計モデル間相関のみを考慮して乱数発生による Err^{T-CVS} を図示したものである（シミュレーションの Err^{CVM} は 0.5 であり、図では割愛している）。真値が 0.5 であるので、無相関のときにバイアスが最も大きく、相関が大きくなるとバイアスが小さくなることが分かる。統計モデル同士に関連が全く無い場合は、Err^{CV} の相関もなくなりそれぞれ独立にばらつくため、Err^{CVS} のバイアスが大きくなる。反対に統計モデル同士に関連があると Err^{CV} は相関してばらつくので、最小値を選択してもバイアスは少ない。今回試した AE～DH の結果の相関行列を【表 3】に示す。同じ変数縮小を行っている統計モデル同士の Err^{CV} は相関が高いが（線で囲んでいるブロック）、その他の Err^{CV} 同士の相関は低い。また【図 5】で理論的な相関係数との関係を確認すると、ほぼ等しいことが分かる。シミュレーションでは相関係数の平均値が $\bar{r} = 0.198$ で $\bar{E}rr^{CVS} = 0.321$ であり、理論値では $r = 0.193$ のとき $Err^{T-CVS} = 0.325$ であった。【図 6】には標準偏差と相関係数の関係を示しているが、標準偏差は理論値よりも少

し大きくなっており、これはサンプル間相関を考慮していないためだと思われる。同様にして Err^{CVM} を確認すると、【図 5】より相関がに関わらず誤分類率にバイアスが入っていないことが分かる。また【図 6】より、標準偏差もほぼ理論値に等しい値になっている。



【図 5】設定 1 での乱数発生による理論分布の誤分類率期待値と統計モデル間相関係数。実線が Err^{CVS}、破線が Err^{CVM}。また点線で示している Err^{CVS} と r はシミュレーションの値。

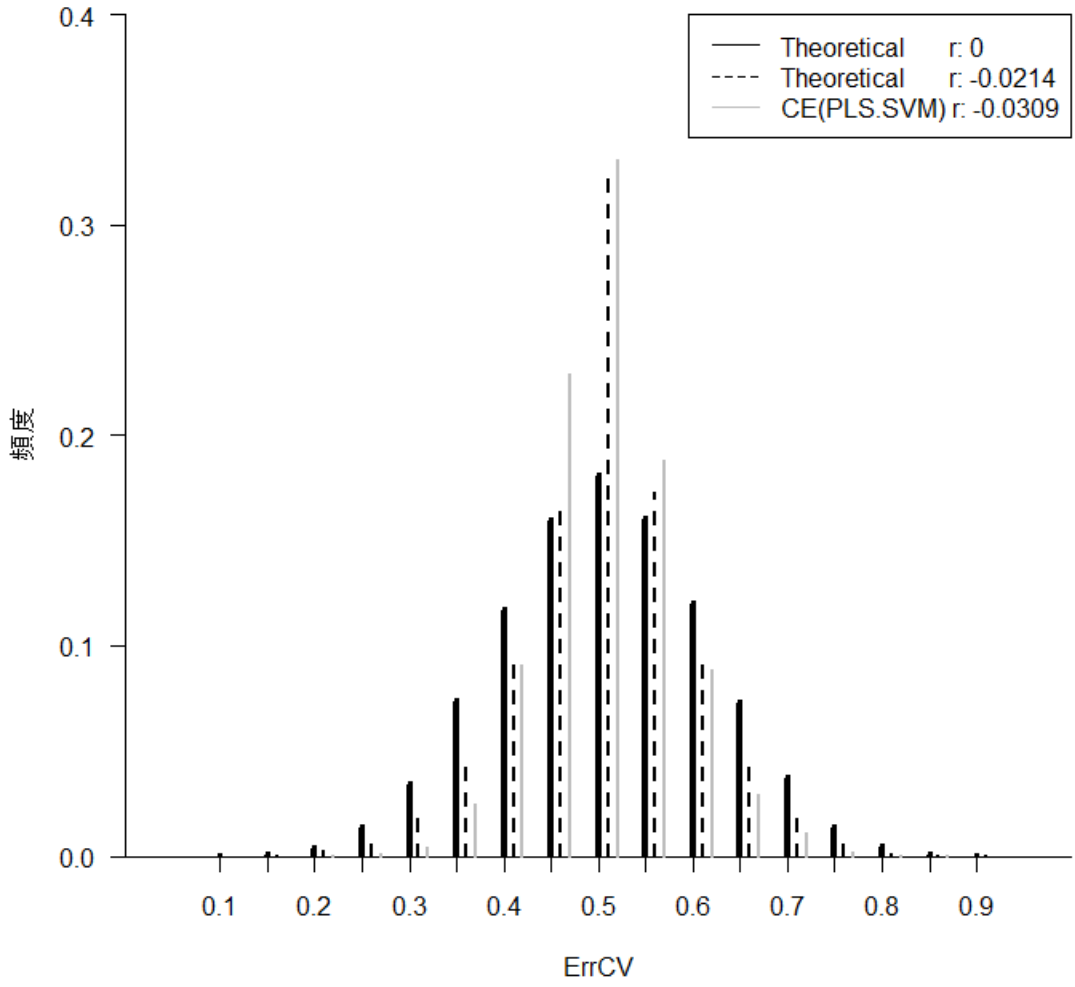


【図 6】設定 1 での乱数発生による理論分布の誤分類率標準偏差と統計モデル間相関係数。実線が Err^{CVS}、破線 が Err^{CVM}。また点線で示している Err^{CVS} と Err^{CVM}、r はシミュレーションの値。

	AE	AF	AG	AH	BE	BF	BG	BH	CE	CF	CG	CH	DE	DF	DG	DH
AE	1.00															
AF	0.79	1.00														
AG	0.75	0.72	1.00													
AH	0.84	0.75	0.75	1.00												
BE	0.13	0.14	0.14	0.13	1.00											
BF	0.03	0.03	0.04	0.03	0.19	1.00										
BG	0.03	0.04	0.04	0.03	0.22	0.04	1.00									
BH	0.09	0.10	0.10	0.09	0.53	0.14	0.24	1.00								
CE	0.24	0.24	0.24	0.24	0.33	0.10	0.08	0.22	1.00							
CF	0.16	0.16	0.14	0.16	0.17	0.06	0.03	0.12	0.49	1.00						
CG	0.23	0.23	0.21	0.22	0.32	0.10	0.08	0.21	0.82	0.47	1.00					
CH	0.24	0.24	0.23	0.23	0.38	0.12	0.09	0.25	0.81	0.45	0.73	1.00				
DE	0.10	0.10	0.10	0.09	0.26	0.07	0.06	0.18	0.22	0.13	0.20	0.24	1.00			
DF	0.06	0.07	0.06	0.06	0.13	0.07	0.04	0.10	0.11	0.05	0.12	0.13	0.24	1.00		
DG	0.06	0.05	0.05	0.05	0.12	0.04	0.08	0.10	0.09	0.03	0.08	0.10	0.21	0.16	1.00	
DH	0.08	0.09	0.08	0.08	0.24	0.07	0.06	0.19	0.18	0.11	0.17	0.21	0.56	0.23	0.25	1.00

【表 3】設定 1 の状況の Err^{CV} の統計モデル間相関

次に【図 7】でサンプル間相関をチェックする。図からも確認できるように、サンプル間相関を考慮しない乱数発生による理論 Err^{CV} の標準偏差（0.111）はシミュレーション CF の Err^{CV} の標準偏差（0.070）よりも大きかった。しかし、二項分布を作成するために多変量正規分布を発生させる際にサンプル間相関を考慮すると、【図 7】のように理論分布とシミュレーション分布がほぼ一致していることがわかる。



【図 7】統計モデル CE（PLS→SVM）で得られた Err^{CV} と乱数発生による理論分布のヒストグラム。縦軸が頻度（2,000 回分の割合）で横軸が Err^{CV} の値。実線：サンプル間相関なしの理論分布、点線：サンプル間相関を考慮した理論分布、薄実線：シミュレーションの CE で得られた Err^{CV} の分布。

【表 2】の理論分布の相関は、これらを踏まえて統計モデル間相関とサンプル間相関を同時に考慮したものである。シミュレーションで観測されたそれぞれの相関は 0.198 と 0.017 であったので、理論分布の相関が 0.199 と 0.019 になるように、多変量正規分布の相関係数を設定した。相関係数が完全一致しない

のは、相関行列が複雑であるため正定行列にするために計算によって微調整しなくてはならず、そのため事前に与えたパラメータ値と少しずれてしまうためである（付録 7 のプログラムを参考）。相関を同時に考慮すると平均はあまり変わらないが、標準偏差がシミュレーション値に近くなっている。

シミュレーションの他の設定を確認すると、n を固定して m を増やしても Err^{CVS} はあまり変わらない。Err^{CVS} は設定 1 と 2 のとき 0.32 付近で、設定 3 と 4 のとき 0.44 付近である。また n を増やすと、各 Err^{CV} は標準偏差が小さくなっていることが分かる（$n = 20$ のとき SD は約 0.03～0.166、$n = 200$ のとき約 0.27～0.052）。つまり Type I の状況では 2.1 で述べたように、Err^{CVS} は変数の数 m には依存せずサンプルの数 n のみに依存することが確認できた。

1.2. Type II の状況

説明変数 X の一部が結果変数 Y と線形な関連がある状況を考える。ここでの真の誤分類率は、統計モデルが正しく真の状況を表現できていた場合の誤分類率とする。

1.2.1. 一部の変数のみに関連がある状況（Type II-1）

まずは Simon が行ったシミュレーションのように、一部の説明変数が結果変

数と関連がある状況を考える[6]。$x_1,...,x_5$が独立に次の正規分布に従っているとする。

$$x_i \overset{i.i.d.}{\sim} N\left(\frac{1}{2}-y_i,1\right),\ i=1,...,5$$

このとき X と Y の関連は次の統計モデルで予測すると最も検出力が高い。

$$if \quad x_1+\cdots+x_5 \geq 0 \quad then \quad y=1$$

$\Pr\left(x_1+\cdots+x_5 \geq 0\right)=0.87$ であるので、$x_1+\cdots+x_5$ があらかじめ分かっていれば、このような統計モデルの検出力は 87%である（誤分類率は 0.13）。実際には t 検定によって説明変数を特定し、関連する説明変数の線形和で結果変数を推定できるので、統計モデル AE で変数選択を行った Err^{CV} が真のモデルに近くなる。【表 4】にシミュレーションの結果を示す。Type I のときと同様に、設定 1～設定 4 の 4 種類のシミュレーションを行っている。また Type II の状況では 3.2 で述べたように、検出力が統計モデルごとに存在しさらにサンプル数によっても変化するため、理論分布を設定することは困難である。しかし、Err^{CVS} の検定を行うためには帰無仮説（Type I）の状況での平均と標準偏差が分かれば十分であるので、Type II の理論分布は割愛する。

			設定1		設定2		設定3		設定4	
	n,m		20, 100		20, 1000		200, 100		200, 1000	
	model		Mean	SD	Mean	SD	Mean	SD	Mean	SD
Apparent	AE	(Ttest.lda)	0.029	0.047	0.001	0.008	0.112	0.023	0.028	0.014
	AF	(Ttest.svm)	0.063	0.062	0.020	0.033	0.062	0.024	**0.000**	0.000
	AG	(Ttest.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	AH	(Ttest.nnet)	0.006	0.022	0.000	0.003	0.068	0.034	0.018	0.013
	BE	(PCA.lda)	0.186	0.094	0.262	0.095	0.190	0.034	0.374	0.042
	BF	(PCA.svm)	0.134	0.069	0.161	0.068	0.166	0.031	0.314	0.034
	BG	(PCA.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	BH	(PCA.nnet)	0.115	0.083	0.207	0.089	0.106	0.037	0.317	0.050
	CE	(PLS.lda)	**0.000**	0.000	**0.000**	0.000	0.033	0.015	**0.000**	0.000
	CF	(PLS.svm)	0.014	0.031	0.014	0.032	0.029	0.014	**0.000**	0.000
	CG	(PLS.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	CH	(PLS.nnet)	0.000	0.000	**0.000**	0.005	0.005	0.007	**0.000**	0.001
	DE	(Clus.lda)	**0.218**	0.095	**0.270**	0.098	0.161	0.053	**0.410**	0.035
	DF	(Clus.svm)	0.176	0.071	0.199	0.074	0.141	0.047	0.341	0.031
	DG	(Clus.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	DH	(Clus.nnet)	0.107	0.084	0.160	0.101	0.098	0.061	0.365	0.077
CV	AE	(Ttest.lda)	0.358	0.155	0.466	0.142	**0.156**	0.028	**0.270**	0.045
	AF	(Ttest.svm)	0.362	0.139	0.460	0.104	0.172	0.031	**0.270**	0.039
	AG	(Ttest.knn)	0.346	0.152	0.430	0.151	0.241	0.038	0.349	0.040
	AH	(Ttest.nnet)	0.346	0.160	0.424	0.155	0.195	0.036	0.273	0.041
	BE	(PCA.lda)	0.363	0.140	0.459	0.141	0.211	0.037	0.426	0.050
	BF	(PCA.svm)	**0.487**	0.039	**0.500**	0.007	0.257	0.043	**0.489**	0.018
	BG	(PCA.knn)	0.458	0.110	0.498	0.063	**0.332**	0.050	0.492	0.036
	BH	(PCA.nnet)	0.401	0.129	0.478	0.118	0.265	0.045	0.452	0.042
	CE	(PLS.lda)	**0.338**	0.139	**0.436**	0.140	0.250	0.038	0.331	0.045
	CF	(PLS.svm)	0.452	0.068	0.499	0.011	0.256	0.038	0.361	0.039
	CG	(PLS.knn)	0.350	0.132	0.468	0.095	0.250	0.039	0.332	0.045
	CH	(PLS.nnet)	0.342	0.132	0.441	0.135	0.255	0.039	0.332	0.043
	DE	(Clus.lda)	0.407	0.123	0.484	0.115	0.181	0.050	0.456	0.038
	DF	(Clus.svm)	0.477	0.062	0.499	0.027	0.207	0.055	0.484	0.030
	DG	(Clus.knn)	0.461	0.107	0.498	0.085	0.256	0.064	0.492	0.036
	DH	(Clus.nnet)	0.431	0.107	0.493	0.078	0.227	0.057	0.482	0.028
CVS			0.205	0.097	0.279	0.091	0.148	0.027	0.250	0.037
CVM			0.399	0.071	0.471	0.050	0.232	0.030	0.393	0.020

【表 4】Type II-1 の状況での Err^{app} 、Err^{CV} 、Err^{CVS} 、Err^{CVM} のシミュレーション結果。太字は Apparent・CV の中で最小値と最大値を示している。

まず設定 1 の結果を確認する。Err^{app} は 0.13 よりも小さいものが多いが、Err^{CV} はどれも 0.13 より大きくなっている。最も良いであろう AE でさえ、0.358 であった。変数は 100 個あるのに対して観測値は 20 個しかないため、$x_1,...,x_5$ をうまく特定することができていない。そのためどの統計モデルでも Err^{true} を過大評価しており、結果的に Type I の状況とは異なり Err^{CVS} でも Err^{true} を過大評価してい

る。それに対して、設定 3 の状況では、Err^{CVS} は Err^{true} =0.13 を少しの過大評価で収まっている。統計モデル AE、AF、DE の Err^{CV} が真値と近い。その他の統計モデルも設定 1 のときに比べて予測性能が上がっている。3 種の統計モデルが真のモデルを表現できているが、他の統計モデルは表現できていないと考えると、この場合の Err^{CVS} は次の分布に従っていると思われる。

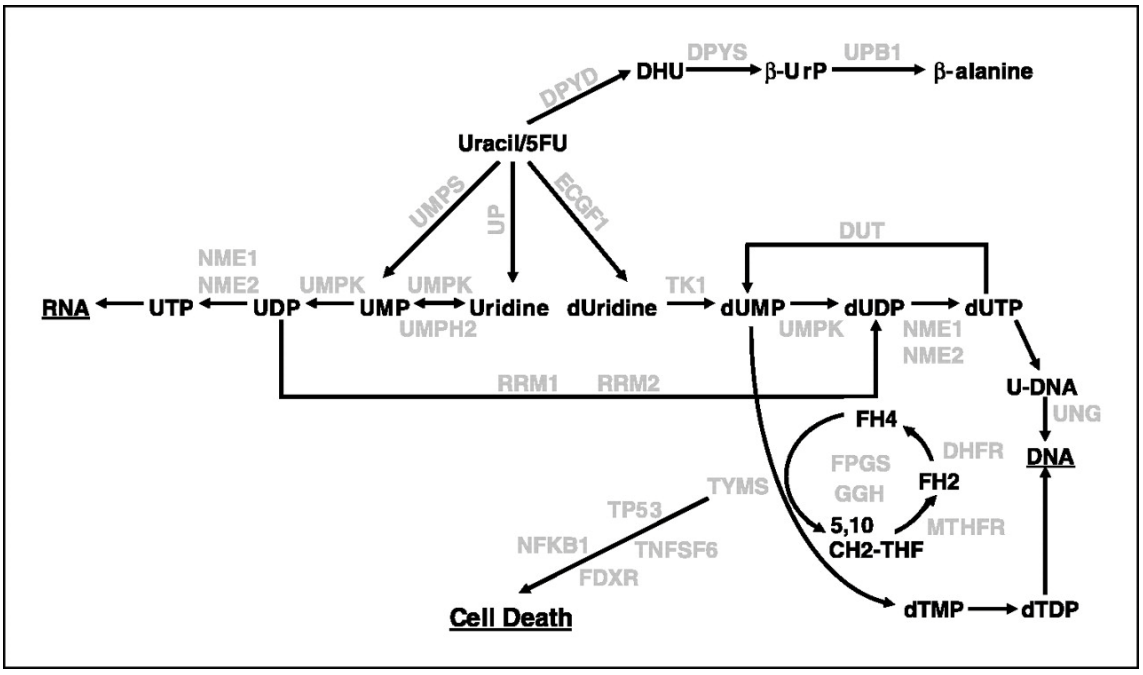
$$Err^{CVS} = \min\left(P_1\left(n, Err^{true}\right), ..., P_3\left(n, Err^{true}\right), P_1\left(n, Err_1^{IA}\right), ..., P_{13}\left(n, Err_{13}^{IA}\right)\right)$$

統計モデルに少し sperior algorithm が含まれている分、Err^{CVS} の過大評価は抑えられているが、Err^{CVM} は依然として過大評価になっている。また設定 1 の場合、全ての統計モデルが superior algorithm（Err^{CV}=0.13）であった場合の理論分布を発生させると、Err^{CVS}=0.015 である（相関は考慮していない）。バイアスは 0.13-0.015=0.115 の過小評価である。設定 3 の場合は理論分布での推定値が 0.089 であったので 0.041 のバイアスであった。

設定 2 のときは変数の数が多いので統計モデルが真のモデルを特定することがさらに難しく、誤分類率は過大評価する傾向にあった。また、設定 4 の状況では設定 3 よりは検出力が落ちるものの、設定 1 よりは高かった。つまり検出力の高い順に設定を並べると、設定 3＞設定 4＞設定 1＞設定 2 となった。

1.2.2. 変数のグループに関連のある状況（Type II-2）

遺伝子データでは、あるグループ変数がまとまって up-regulate したり down-regulate したりする状況がある。例えば【図 8】は 5-FU の代謝パスウェイを示しており[21]、これらの遺伝子を測定すると相関が高くなるはずである。そこで結果変数 *y*=1 の群のみ、10 変数のグループ（10 変量正規分布）を 5 つ作り、3 グループがまとまって up-regulate、残りの 2 グループが down-regulate する状況を考える。グループ内の相関係数は 0.8 で、それぞれのグループの平均値は±0.5 とする。また *y*=0 の群では説明変数は変化しない（平均 0）。



【図 8】5-FU の代謝パスウェイ

結果を【表 5】に示す。大まかな結果は Type II-1 の状況と似ているが、設定 3 と設定 4 の検出力が同じくらいである。どちらの設定でも、クラスタリングで

変数縮小する統計モデルの性能が良い。設定 1 と設定 2 ではこの統計モデルの性能が良くないが、サンプル数が少ないためであろう。

			設定1		設定2		設定3		設定4	
	n,m		20, 100		20, 1000		200, 100		200, 1000	
	model		Mean	SD	Mean	SD	Mean	SD	Mean	SD
Apparent	AE	(Ttest.lda)	0.027	0.051	0.001	0.007	0.125	0.026	0.022	0.014
	AF	(Ttest.svm)	0.077	0.073	0.025	0.038	0.079	0.027	0.004	0.006
	AG	(Ttest.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	AH	(Ttest.nnet)	0.011	0.029	**0.000**	0.003	0.077	0.036	0.042	0.029
	BE	(PCA.lda)	**0.137**	0.084	0.202	0.097	0.152	0.024	**0.189**	0.030
	BF	(PCA.svm)	0.133	0.071	0.143	0.069	0.182	0.023	**0.189**	0.025
	BG	(PCA.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	BH	(PCA.nnet)	0.077	0.068	0.168	0.102	0.047	0.041	0.090	0.038
	CE	(PLS.lda)	**0.000**	0.000	**0.000**	0.000	0.076	0.020	**0.000**	0.000
	CF	(PLS.svm)	0.040	0.048	0.018	0.035	0.093	0.021	**0.000**	0.000
	CG	(PLS.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	CH	(PLS.nnet)	**0.000**	0.000	**0.000**	0.005	0.028	0.022	**0.000**	0.000
	DE	(Clus.lda)	0.129	0.081	**0.246**	0.099	0.155	0.026	0.151	0.023
	DF	(Clus.svm)	0.161	0.062	0.193	0.071	**0.183**	0.024	0.181	0.024
	DG	(Clus.knn)	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	DH	(Clus.nnet)	0.054	0.061	0.138	0.094	0.059	0.052	0.056	0.049
CV	AE	(Ttest.lda)	0.410	0.142	0.449	0.146	0.287	0.037	0.390	0.048
	AF	(Ttest.svm)	0.376	0.134	0.451	0.104	0.202	0.030	0.246	0.034
	AG	(Ttest.knn)	0.349	0.148	0.419	0.155	**0.367**	0.041	0.333	0.043
	AH	(Ttest.nnet)	0.381	0.149	0.439	0.151	0.277	0.039	0.325	0.043
	BE	(PCA.lda)	**0.241**	0.118	0.368	0.140	0.163	0.025	0.202	0.032
	BF	(PCA.svm)	**0.448**	0.065	**0.499**	0.008	0.280	0.026	**0.458**	0.020
	BG	(PCA.knn)	0.326	0.133	0.481	0.075	0.128	0.025	0.327	0.048
	BH	(PCA.nnet)	0.283	0.131	0.415	0.129	0.134	0.028	0.232	0.038
	CE	(PLS.lda)	0.322	0.146	**0.367**	0.139	0.311	0.043	0.296	0.046
	CF	(PLS.svm)	0.420	0.068	0.499	0.011	0.310	0.038	0.350	0.028
	CG	(PLS.knn)	0.302	0.139	0.425	0.104	0.222	0.037	0.290	0.045
	CH	(PLS.nnet)	0.335	0.141	0.383	0.133	0.268	0.039	0.310	0.046
	DE	(Clus.lda)	0.264	0.105	0.453	0.111	0.168	0.027	0.163	0.025
	DF	(Clus.svm)	0.420	0.061	0.495	0.031	0.261	0.030	0.255	0.029
	DG	(Clus.knn)	0.265	0.104	0.481	0.086	**0.111**	0.025	**0.108**	0.025
	DH	(Clus.nnet)	0.272	0.109	0.468	0.081	0.138	0.031	0.132	0.029
CVS			0.159	0.081	0.253	0.094	0.104	0.022	0.105	0.023
CVM			0.338	0.080	0.443	0.057	0.227	0.021	0.276	0.023

【表 5】Type II-2 の状況での Err^{app} 、Err^{CV} 、Err^{CVS} 、Err^{CVM} のシミュレーション結果。太字は Apparent・CV の中で最小値と最大値を示している。

1.3. Err^{CVS} の検定

2.3 で述べた方法で Err^{CVS} の検定を行う。シミュレーションで算出した 2,000 個の Err^{CVS} を、理論分布から求めた Err^{CVS} の平均と標準偏差を用いて検定する。

例えば設定 1 の状況で相関を考慮しない検定を行う場合は、平均が 0.305 で標準偏差が 0.061 である。つまりこの場合は、シミュレーションの Err^{CVS} が 0.305-1.96*0.061=0.185 よりも小さければ、片側 2.5%水準で有意となる。Type II の状況で検定する場合も同じ値を使う。

結果を【表 6】に示す。Type I ではαエラーの結果を、Type II では検出力の結果を示している。Type I の場合、検定の名義水準は 2.5%であるので、αエラーはこれより小さければ良い。相関を考慮しない場合は、どの検定結果も有意水準よりも大きいαエラーとなっている。しかし、相関を考慮すると設定 2 を除いて有意水準を満たす結果となった。Type II の場合は、検出力の大きい順に設定 3、4＞設定 1＞設定 2 となっている。また相関を考慮すると検出力が小さくなっているが、これは Type I で有意水準を保っているためである。

	αエラー（名義水準は2.5%）			
TypeI	設定1	設定2	設定3	設定4
相関なし	6.8%	5.7%	8.9%	9.8%
相関考慮	2.5%	2.0%	1.2%	2.3%

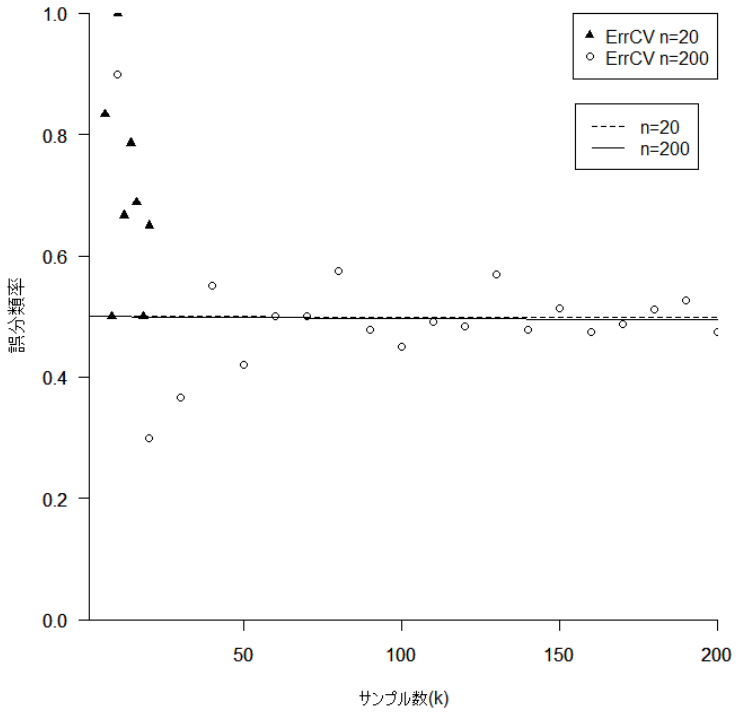
	検出力（1−βエラー）			
TypeII-1	設定1	設定2	設定3	設定4
相関なし	39.6%	13.5%	100.0%	100.0%
相関考慮	21.4%	5.0%	100.0%	100.0%
TypeII-2	設定1	設定2	設定3	設定4
相関なし	58.7%	21.6%	100.0%	100.0%
相関考慮	37.0%	9.0%	100.0%	100.0%

【表 6】それぞれの状況での検定結果

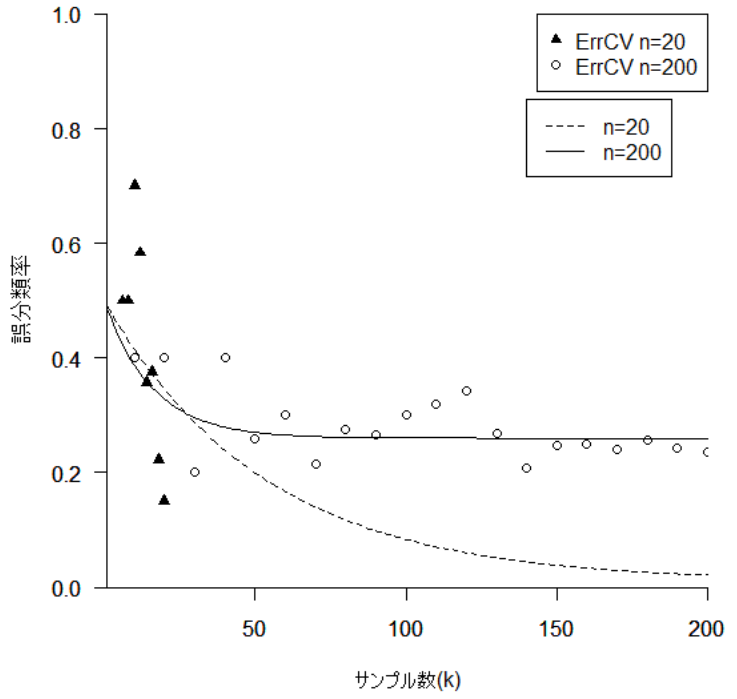
1.4. K's Plot による誤分類率の評価

Type I と Type II-1 の状況でシミュレーションデータの K's Plot を描いた結果を【図 9】、【図 10】に示す。それぞれの状況で、設定 1 と設定 3 の 2 種の設定で 1 回だけ発生させたデータの結果である。統計モデルは AE を用いた。設定 1 のときは $k_i = (6,8,...,20)$ の 8 回、設定 3 のときは $k_i = (10,20,...,200)$ の 20 回 Err^{CV} を推定している。また、推定された Err^{CV} に指数関数モデルを当てはめた結果を曲線で示している。まず【図 9】の Type I と【図 10】の Type II-1 で大きく異なるのは、推定された曲線の傾きである。Type I のときの α は共に 0.49 と推定されたが、Type II-1 では 0.01、0.26 と推定された。この値が Err^{true} の参考値になっているが、この場合は観測値が 20 個しかない状況でも、指数関数モデルによって推定がうまくいっている。

このようにプロットをすることで、統計モデルの性能を目視することができる。検定を行うだけではなく、K's Plot によって観測値が増えるに従って誤分類率がどのように変化しているのかを確認できる。またプロットを利用して Type II の状況で Err^{true} を推定するためのサンプルサイズの目安を得ることも考えられる。



【図 9】Type I の状況でのサンプル数 k_i（横軸）と Err^{CV}（縦軸）の関係。▲が設定 1、〇が設定 3、点線と実線が指数モデルで推定した曲線を示す。



【図 10】Type II-1 の状況でのサンプル数 k_i（横軸）と Err^{CV}（縦軸）の関係。▲が設定 1、〇が設定 3、点線と実線が指数モデルで推定した曲線を示す。

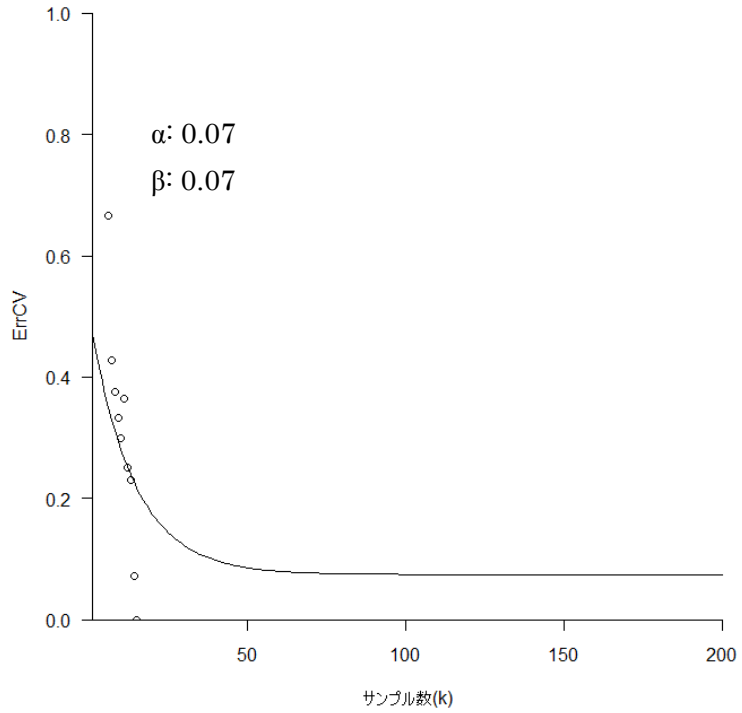
2. 実データ

2.1. 白血病のデータ

まず Ross らの急性リンパ性白血病（acute lymphoblastic leukemia: ALL）のデータを利用する[22]。このデータは 128 個の ALL サンプルをマイクロアレイで測定したものである。マイクロアレイで測定できる値は、遺伝子の一部分をプローブとして抜き出した発現量である。ALL はいくつかのサブタイプに分かれており、今回は ALL/AF4 群と E2A/PBX1 群の判別を行うことを考える。サンプル数はそれぞれ 10 個、5 個でありプローブ数は 12,625 個測定している。これらのプローブのうち、少なくともどちらかの群の平均発現量が 100 を超えている 3,762 プローブを解析対象とする。このデータで統計モデル AE～DH を行った際の誤分類率を【表 7】に示し、サンプル数 k_i を変化させ統計モデル AE を適用したときの K's Plot を【図 11】に示す。

	AE	AF	AG	AH	BE	BF	BG	BH	CE	CF	CG	CH	DE	DF	DG	DH
ErrApp	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.33	0.00	0.00
ErrCV	0.00	0.07	0.00	0.33	0.00	0.33	0.00	0.20	0.00	0.33	0.00	0.00	0.07	0.33	0.60	0.33

【表 7】白血病データでの Err^{app} と Err^{CV}



【図 11】白血病データに統計モデル AE を当てはめた場合の K's Plot。α は Err^{CV} の漸近値、β は漸近する早さ。

Err^{CV} はほとんど小さい値を示している。各統計モデルの CV で推定された結果の相関を平均すると 0.20 であった。統計モデル AH、BF、CF、DF、DH の Err^{CV} がどれも 0.33 であり、E2A/PBX1 群が全て誤分類されていたので、これらは inferior algorithm であることが示唆される。AE を当てはめたときの Err^{CV} に指数関数をあてはめると、α=0.07、β=0.07 と推定された。観測値が 15 しかないので α が極端に小さい値に推定されやすいが、ALL データは Type II のデータであり、統計モデル AE での分類が可能であることが伺える。またサンプル数 15、統計モデル数 16 での Err^{CVS} の理論平均と標準偏差は、相関なしの状況で 0.276 と 0.67、相関ありの状況で 0.290 と 0.079 であった。相関を考慮しても、誤分類率が

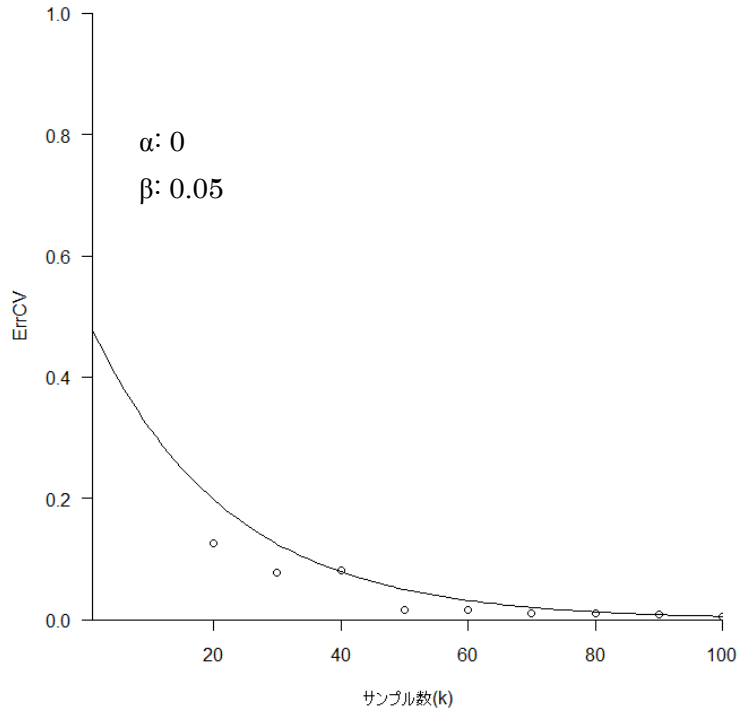
0.290-1.96*0.079=0.135 よりも小さければ有意になるので、【表 7】で得られた統計モデルの誤分類率は有意である。

2.2. がんの予測

Kurahashi らは原発不明がんの予測を行う研究で、公共データベース Gene Expression Omnibus（GEO[23]）から約 2,500 サンプルの原発がんマイクロアレイデータを入手している [15]。彼らは全てのサンプルを使って原発不明がんの原発巣を予測するための統計モデルを作成しているが、本論文では一部のサンプルを使って、マイクロアレイデータでの統計モデルの性能を確認する。利用するデータは膀胱がんと悪性脳腫瘍で、共に 80 サンプルである。遺伝子（プローブセット）の数は 22,215 のうち 1,000 個のみ利用する。またクラスタリングを利用した統計モデルは使わず、残りの 12 モデルを評価した。結果を【表 8】に示す。最も良いモデルは BG（主成分分析→k-近傍法）であり、検定を行っても有意であった。また最も良かった統計モデル BG の K's Plot の結果を【図 12】に示す。サンプル数が増えるに従って、誤分類率が減少し検出力が増えていることが目視できる。

	AE	AF	AG	AH	BE	BF	BG	BH	CE	CF	CG	CH
ErrApp	0.05	0.03	0.00	0.02	0.02	0.02	0.00	0.02	0.00	0.00	0.00	0.00
ErrCV	0.16	0.12	0.24	0.17	0.03	0.23	0.01	0.07	0.03	0.25	0.04	0.04

【表 8】原発がんデータでの Err^{app} と Err^{CV}



【図 12】原発がんデータの統計モデル BG（PCA→k-近傍法）の K's Plot。α は Err^{CV} の漸近値、β は漸近する早さ。

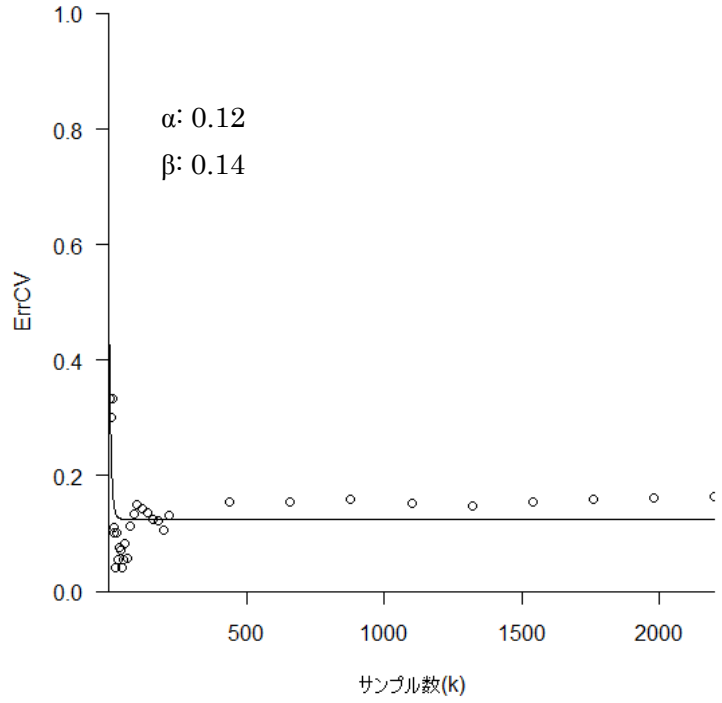
2.3. 健康診断データ

平成 20 年度より特定健康診査（特定健診）が実施され、40～74 歳の被保険者に対して一律の健康診断が行われている[24]。これらのデータはほぼ同一のフォーマットで各健保に保存されており、一律の解析が出来るため疫学研究等への有効活用が期待されている。健康診断データはサンプル数が数万～数十万で変数が数十～百程度であるので、これまで議論していた $n \ll m$ の状況には合致しない。しかしある統計モデルを当てはめた際に、そのモデルの予測性能を視覚的に確認することは有意義であると思われる。そこで本論文では、この特定健診データを用いて腹囲周囲径の統計モデルを作り、その性能を K's Plot で確認する。特

定健診データはある健保から借用したものである。欠測を全て取り除き、男性のみを対象としたため、対象者は 22,342 人であった。また腹囲を予測するために利用する変数は、身長、体重、BMI、収縮期血圧、拡張期血圧、中性脂肪、HDL コレステロール、LDL コレステロール、空腹時血糖、HbA1c、GOT、GPT、γ-GTP、血色素量、年齢の 15 変数とした。また、腹囲が特定健診の法定階層化の基準値となる 85cm を超える対象者を 1 として、これを結果変数とする。統計モデルは 15 変数のうち t 検定で P 値が上位 5 つの変数に選択した後、判別分析で行った（統計モデル AE に準じたモデルである）。

この統計モデルで推定した各誤分類率は、$Err^{App} = 0.163$、$Err^{CV} = 0.164$ であったため、Err^{App} のバイアスはほとんど見られなかった。Err^{App} の計算の際に T 検定で選ばれた変数は、身長、BMI、中性脂肪、HDL コレステロール、GPT の 5 変数であった。また K's Plot によってモデルの予測性能を確認すると、【図 13】のようになった。サンプル数 k_i は 6～22,000 の範囲で変化させているが、グラフに表示しているのは 2,000 サンプルまでである。k_i が約 200 までは Err^{CV} が大きくばらついているが、200 よりも大きくなるとほぼ一定の値を示している。k_i が 200 以上の部分での誤分類率の平均値は 0.159 であった。それに対して、指数関数モデルの推定値は 0.12 となっており、小さく推定された。これは k_i が 0～100 の範囲で、Err^{CV} が 0 付近と推定されているので、この値に引っ張られているた

めだと考えられる。



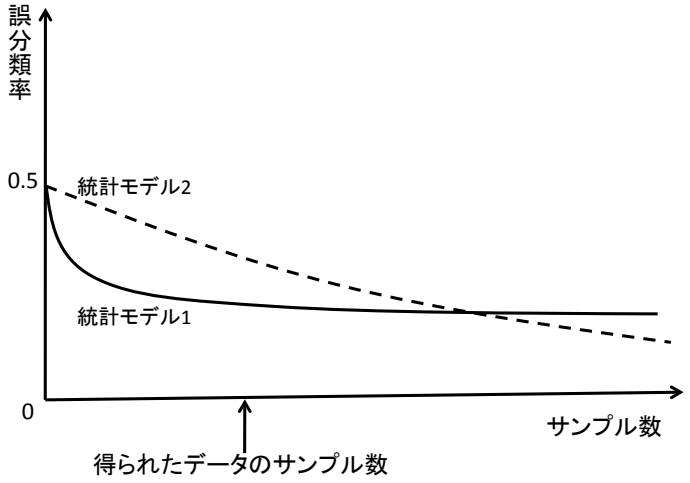
【図 13】特定健診データの腹囲統計モデルの K's Plot。α は Err^{CV} の漸近値、β は漸近する早さ。

考察

1. 本研究の内容の利用可能性

本研究では、多数の統計モデルから 1 つの最も良いモデルを選択する場合にどのようなバイアスが入るかを示し、その解決方法を提案した。多くの統計モデルからどのモデルを選択すれば良いのかという問題は、統計家にとって常に頭を悩ませる問題である。最良のモデルを選べばデータへの over-fitting（過適合）が起こっているのではないかと考えることも多い。本稿で提案した検定と K's Plot は統計モデルを選択するときの 1 つの根拠になることができる。

まず検定によって確認できることは、選択した統計モデルの性能が偶然誤差によるものなのかどうかである。いくつの統計モデルを試したかを記録しておき、その情報をもとに検定を行う。有意になれば統計モデルの性能は偶然ではないことが示唆される。次に K's Plot によって、個々の統計モデルの性能を視覚的に評価する。特に有意になった統計モデルが多数存在する場合に、それぞれ K's Plot を描き、性能を比較することができる。ここでは単純に誤分類率の大きさではなく、K's Plot の曲線の形に注目することが重要である。例えば【図 14】のようなプロットが得られるかもしれない。この場合は現在得られているデータでは統計モデル 1 の誤分類率の方が低いが、サンプル数を増やすことで統計モデル 2 の性能が上回ることが示唆されている。



【図 14】複数の統計モデルでの K's Plot の確認

このように誤分類率の検定と K's Plot は統計モデルの選択を行う上で非常に有用なツールに成り得る。例えば臨床試験によってバイオマーカーで統計モデ

ルを作る場面などでは、非常に綿密に練られたプロトコルが必要となる。しかしそのような研究ではデータにどんなモデルが適合するか事前に分からず、どのような統計手法を使ってモデル構築をするかを記述するのは難しい。そのような場合でも、本研究のような検定と K's Plot を利用して統計手法を全て評価すると記載をしていれば、統計的に間違った推論を起こす可能性は低いし、様々な手法を試すことが可能となる。

2. Err^{CVS} の性質と検定

データからある統計モデルを構築するとき、いくつものモデルを試すことがあるが、実際に利用する又は公表する統計モデルはその中で最も良いものである。例えば乳がんの予後予測を行う Oncotype DX というキットの統計モデルは、相関の高い遺伝子の選択→主成分分析等によるスコア化が主な手順になっている [13]。しかしこの統計モデルに辿り着くまでに、他の統計モデルは試さなかったであろうか。本論文では 16 個の統計モデルは敢えてあまり似ている手法は選ばなかった。そのため【表 3】のように統計モデル間の相関はあまり高くなかった。しかし、いくつかの統計モデルを試すということは、本論文のように手法自体が異なるものを試すだけではない。例えば変数選択の際にステップワイズ法を行う場合に P 値を基準にするのか、AIC を基準にするのかで異なった結果が得られる。また遺伝子の分野では特に様々な手法が提案されており、デー

タクリーニングの際の正規化をどのように行うかで結果が変わってくる[25,26]。

より良い統計モデルを構築するためには、いくつもの解析手法を試し試行錯誤することが不可欠である。しかし 2 値変数の予測の場合に統計モデルの評価を行う誤分類率は、異なる統計モデルを試す度にばらついてしまう。そのため最も良い統計モデルを選択すると、意図せずとも選択バイアスの含んだ結果となってしまう。特にシミュレーションでも明らかになったように、マイクロアレイなどの分野のようにサンプル数が少ないとバイアスが大きくなる。本論文で提案した方法で検定すれば、選んだ統計モデルの性能が偶然によるものなのかどうかを調べることができる。検定に最低限必要なパラメータは、サンプル数と試した統計モデルの数だけであるので、簡単に行うことができる。R で行う場合は、付録に付けている CvsCvm.Theor 関数で計算される期待値と標準偏差を使うことで、検定することができる。

また今回は【図 1】の CV3 のシミュレーションは行っていないが、CV3 まで行うことが出来れば誤分類率にバイアスが入ることはない。今後コンピュータや解析ソフトの発展によって、CV3 を手軽に出来るようになれば本論文のような議論の必要性は減っていく。しかし現段階ではいくつもの手法を組み込んで CV を行うことはコンピュータの処理能力の問題のため現実的ではないし、新たに提案された統計手法と既に試した統計手法を同時に CV するパッケージを作

ることが必要になる。

3. K's Plot の解釈

誤分類率を推定する場合は、CV や Bootstrap を行えばバイアスは入らない[4]。また統計モデルの選択を含めた CV を行うことが出来なくても、本論文で提案した検定を行えばバイアスのない推論を行うことができる。しかしこれらの手法で得られる誤分類率は 1 つの統計モデルに対して 1 つである。そのため選んだ統計モデルが、データに対してどのような性能を持っているのかを知ることが困難である。そこで、統計モデルの性能を視覚的に確認できるツールとして、本論文では K's Plot を提案した。サンプル数が多くなれば統計モデルの性能が上がり検出力が大きくなるという事実は常識的なことである。だが機械学習の分野が発展してアルゴリズムが複雑化し、計算方法がブラックボックス化していく中で、数理計算で検出力を算出することは難しくなってきている。

K's Plot は検出力を、実データからシミュレーションのような形で推定する。これによって、本論文で試した腹囲推定のように、本来であれば予測できることが当たり前のようなモデルでも、検出力の変化を確認できた。統計モデルの構築に何万人もの対象者を解析する必要はなく、数百人も居れば十分であることが示唆された。このようにデータをだた単に解析するのではなく、視覚的に確認することで新しい見方ができることは、データの視覚化という観点からも

重要である[27,28]。

4. 本研究の限界と今後の課題

本研究では誤分類率の推定は全て 10-fold CV で行っている。これは計算時間と CV の性質を考えると、実用的には 10-fold CV を行う場面が多いためである。しかし今回評価した誤分類率のバイアスは、CV の性質に依存している可能性もある。これまでシミュレーション結果を調べた経験からするとサンプル間相関のみが変化し、Err^{CVS} の期待値は変わらず標準誤差が少し変化するだけではないかと予想しているが、今後 CV や Bootstrap を何種類かシミュレーションしてみる必要があるかもしれない。

誤分類率のバイアスや検定、K's Plot を連続量や生存時間に応用することも考えられる。検定は 2 項分布の極値分布を主軸にして提案をしているので、これを正規分布やワイブル分布等に応用ができる。しかし多くの予測問題は 2 値に集約することができるため、本研究の検定を広い分野で利用することができる。例えば Oncotype DX は、統計モデルの開発は生存時間解析などを使っているが、最終的な予測は 2 値で行っている [13]。また K's Plot は「データの部分集団で統計モデルの評価を行う」という操作を行っているので、そのまま連続量の予測に利用することが可能である。例えば、線形モデルで予測を行う場合は横軸にサンプル数、縦軸に残差をプロットすれば、連続値での K's Plot を描くことができ

る。その他にも、AIC 等のモデル評価の指標をプロットして確認することも可能である。

本研究では誤分類率という、予測問題では最も単純で根本的なものを扱った。しかし現実に 2 値予測で統計モデルを評価する場合は、感度や特異度の両方を用いることも多い。その場合、2 群のサンプル数のバランスも問題となる。これらの問題を考えるときも、結局は片群ごとに 2 項分布を仮定してそれぞれの群で誤分類率を計算すれば良い。つまり本論文の検定方式を片群ずつに応用することで、推定と検定を行うことが可能である。

結論

本研究で提案した選択バイアスのある状況での誤分類率の検定では、有意水準を保って検定を行うことが出来ることが、シミュレーションによって示された。この検定を行えば統計モデル選択全体を含めた複雑な Cross Validation を行わなくても、バイアスのない推論を行うことが可能である。また実データによって、K-sample Plot を描くことで個々の統計モデルの性質を視覚的に評価できた。

謝辞

ここに、大変貴重な健康診断データをご提供して下さった、株式会社ヘルスケアコミッティー様に感謝の意を表させて頂きます。また本研究を進めるにあたり、ご指導を頂いた大橋靖雄教授に深謝いたします。日常の議論を通じて多くの知識や示唆を頂いた疫学・生物統計学研究室の皆様に感謝します。協力していただいた皆様へ心から感謝の気持ちと御礼を申し上げたく、謝辞にかえさせていただきます。

参考文献

[1] Miller R. Simultaneous statistical inference. Springer. (1981)

[2] Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression. Annals of Statistics, 32:407-99. (2004)

[3] Efron B. Estimating the error rate of a prediction rule: improvement on cross-validation. Journal of the American Statistical Association, 382:316-31. (1983)

[4] Efron B. How biased is the apparent error rate of a prediction rule? Journal of the American Statistical Association, 394:461-70. (1986)

[5] Efron B. Imprivements on cross-validation: the .632+ bootstrap method. Journal of the American Statistical Association, 438:548-60. (1997)

[6] Simon R, Radmacher MD, Dobbin K, et al. Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification. Journal of the National Cancer Institute, 95:14-8. (2003)

[7] Andreopoulos B, An A, Wang X, et al. A roadmap of clustering algorithms: finding a match for a biomedical application. Briefings in Bioinformatics, 10:297-314. (2009)

[8] R-organization. 参照日: 2010 年 11 月 1 日, 参照先: http://cran.r-project.org/

[9] Gentleman R, Carey V, Irizarry R, et al. Bioinformatics and computational biology solutions using R and bioconductor. Springer. (2005)

[10] Simon R, Lam A, Li MC, et al. Analysis of gene expression data using BRB-array tools. Cancer Informatics, 2:11-7. (2007)

[11] Kuhn M. Package 'caret'. 参照日: 2010 年 11 月 1 日, 参照先: http://caret.r-forge.r-project.org/

[12] Golub TR, Slonim DK, Tamayo P, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science, 286:531-7. (1999)

[13] Paik S, Shak S, Tang G, et al. A Multigene Assay to Predict Recurrence of Tamoxifen-Treated, Node-Negative Breast Cancer. N Engl J Med, 351:2817-2826. (2004)

[14] Tothill RW, Kowalczyk A, Rischin D, et al. An expression-based site of origin diagnostic method designed for clinical application to cancer of unknown origin. Cancer

Research, 65:4031-40. (2005)

[15] Kurahashi I, Nakagawa K, Nishio K, et al. An algorithm of prediction for primary site (in Japanese). Molecular Target Therapy of Cancer, 7:14-9. (2009)

[16] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65:386-408. (1958)

[17] Minsky M, Papert S. Perceptrons. Cambridge: MIT Press. (1969)

[18] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by backpropagationg errors. Nature, 323:533-6. (1986)

[19] Close S. An introduction to statistical modeling of extreme values. Springer. (2001)

[20] Bacon CJ, Woo J, Lau EM, et al. Effects of 25-hydroxyvitamin D level and its change on parathyroid hormone in premenopausal Chinese woman. Osteiporosis International, 21:1935-41. (2010)

[21] Kidd EA, Yu J, Li X, et al. Variance in the Expression of 5-Fluorouracil Pathway Genes in Colorectal Cancer. Clinical Cancer Research, 11:2612-9. (2005)

[22] Ross ME, Zhou X, Song G, et al. Gene expression profiling of pediatric acute myelogenous leukemia. Blood, 102:2951-9. (2004)

[23] NCBI. Gene Expression Omnibus. 参照日: 2010 年 11 月 1 日, 参照先: http://www.ncbi.nlm.nih.gov/geo/

[24] 厚生労働省　保健局. 標準的な健診・保健指導プログラム（確定版）. 参照日: 2010 年 11 月 1 日, 参照先:
http://www.niph.go.jp/soshiki/jinzai/koroshoshiryo/kenshin/data/zentai.pdf

[25] Simon MR, Korn EL, McShane LM, et al. Design and Analysis of DNA Microarray Investigations (Statistics for Biology and Health). Springer. (2004)

[26] 倉橋一成, 伊藤陽一, 松山裕ら. cDNA マクロアレイデータ解析における正規化手法の性能評価. 147-63. (2007)

[27] Tukey J. Exploratory data analysis. University Microfilms International. (1988)

[28] Sarker D. Lattice: Multivariate Data Visualization with R (Use R). Springer. (2008)

付録

【表】R の caret パッケージで利用可能な統計モデル

手法のグループ	手法(関数)	手法のグループ	手法(関数)
Generalized linear model	glm	Penalized linear models	penalized
(一般化線形モデル)	glmStepAIC	(制限付き線形回帰)	enet
Generalized additive model	gam		lars
(一般化加法モデル)	gamLoess		lars2
	gamSpline		enet
Recursive partitioning	rpart		foba
(再帰的分割)	ctree	Supervised principal components	superpc
	ctree2	(教師付き主成分分析)	
Boosted trees	gbm	Quantile Regression Forests	qrf
	blackboost	Linear discriminant analysis	lda
	ada	(線形判別分析)	Linda
Boosted regression models	glmboost	Quadratic discriminant analysis	qda
(ブースティング)	gamboost	(2次判別分析)	QdaCov
	logitBoost	Stabilized linear discriminant analysis	slda
	rf	Heteroscedastic discriminant analysis	hda
	parRF	Stepwise discriminant analysis	stepLDA
	cforest	(ステップワイズ判別分析)	stepQDA
Bagging	treebag	Stepwise diagonal discriminant analysis	sddaLDA
(バッギング)	bag		sddaQDA
	logicBag	Shrinkage discriminant analysis	sda
Other Trees	nodeHarvest	(縮小判別分析)	
	partDSA	Sparse linear discriminant analysis	sparseLDA
Logic Regression	logreg	Regularized discriminant analysis	rda
(ロジスティック回帰)		Mixture discriminant analysis	mda
Elastic net (glm)	glmnet	(混合判別分析)	
Neural networks	nnet	Sparse mixture discriminant analysis	smda
(ニューラルネットワーク)	neuralnet	Penalized discriminant analysis	pda
	pcaNNet	(制限付き判別分析)	pda2
Projection pursuit regression	ppr	Stabilised linear discriminant analysis	slda
Principal component regression	pcr	High dimensional discriminant analysis	hdda
(主成分回帰)		(高次元判別分析)	
Independent component regression	icr	Flexible discriminant analysis (MARS)	fda
(独立成分回帰)		Bagged FDA	bagFDA
Partial least squares	pls	Logistic/multinomial regression	multinom
Sparse partial least squares	spls	(ロジスティック、多項回帰)	
Support vector machines	svmLinear	Penalized logistic regression	plr
(サポートベクターマシン)	svmRadial	(制限付きロジスティック回帰)	
	svmPoly	Rule-based classification	J48
Relevance vector machines	rvmLinear		OneR
	rvmRadial		PART
	rvmPoly		JRip
Least squares support vector machines	lssvmRadial	Logic Forests	logforest
Gaussian processes	guassprLinearl	Bayesian multinomial probit model	vbmpRadial
(ガウス過程)	guassprRadial	k nearest neighbors	knn3
	guassprPoly	(k-近傍法)	
Linear least squares	lm	Nearest shrunken centroids	pam
	lmStepAIC		scrda
Robust linear regression	rlm	Naive Bayes	nb
(ロバスト回帰)		(ナイーブベイズ)	
Multivariate adaptive regression splines	earth	Generalized partial least squares	gpls
(多項加法スプライン)		Learned vector quantization	lvq
Bagged MARS	bagEarth	ROC Curves	rocc
Rule Based Regression	M5Rules		

以下に本シミュレーションを行うにあたって作成した R プログラムを添付する。

1. 16 種の統計モデル

関数名　：Prediction
内容　：16 種の統計モデルで予測を行う関数
パラメータ：TrainData（学習データ）、TestData（検証データ）
返り値　：TrainData で作成した 16 モデルの予測式を TestData に当てはめた正誤結果の行列

```
#------//予測の部分を関数化
   #---TrainData で作成した予測式を TestData に当てはめた正誤結果を返す
   Prediction <- function(TrainData, TestData){
   #------予測 E：判別分析
   Data.lda          <- lda(y ~ ., data=TrainData)
   Data.lda.P        <- ifelse(TestData$y != predict(Data.lda, TestData)$class, 1, 0)

   #------予測 F：SVM
   Data.svm          <- svm(y ~ ., data=TrainData, kernel="polynomial", degree=3)
   Data.svm.P        <- ifelse(TestData$y != predict(Data.svm, TestData), 1, 0)

   #------予測 G：K 近傍
   #---予測式を作るときにもテストセットを使っている
   Data.knn          <- knn(data.frame(TrainData[, c(1:ncol(TrainData)-1)]),
                data.frame(TestData[,   c(1:ncol(TestData )-1)]), TrainData$y, k=1)
   Data.knn.P        <- ifelse(TestData$y != Data.knn, 1, 0)

   #------予測 H：ニューラルネット
   Data.nnet.pre <- nnet(y ~ ., data=TrainData, size=5, maxit=1)
   Data.nnet         <- nnet(y ~ ., data=TrainData, size=5, maxit=1000,
                    Wts=rep(1, length(Data.nnet.pre$wts)))
   Data.nnet.P       <- ifelse(TestData$y != predict(Data.nnet, TestData, type="class"), 1, 0)

   #---それぞれのサンプルの予測結果
   rbind(Data.lda.P, Data.svm.P, Data.knn.P, Data.nnet.P)
   }
#------予測の関数//
```

2. Err^{app} を計算するための 2 値予測

関数名　：Apparent

内容　：16 種の統計モデルでの Err^{app} を予測するための関数

パラメータ:UseDataXY（利用するデータ全体）、UseDataX（データの説明変数部分）、UseDataY（データの結果変数部分）、M（説明変数の数）、N2（サンプルの数）

返り値　：　16 モデルの予測結果行列

```
#------//Apparent 予測の計算
Apparent <- function(UseDataXY, UseDataX, UseDataY, M, N2){
  #---Clus で plot が必要なため空の図を描いておく
  plot(1)

  DataXY <- UseDataXY
  DataX  <- UseDataX
  DataY  <- UseDataY



  #------//A：t 検定
    #---変数選択
    Pval <- rep(99, M)
    for(i in 1:M){
      Ttest     <- t.test(DataXY[DataXY$y==1, i], DataXY[DataXY$y==0, i])          #ウェルチの t 検定
      Pval[i] <- Ttest$p.value
    }
    names(Pval) <- c(1:M)

    #---P 値が 0.05 未満の変数を抜き出す
    DataXY.Ttest <- DataXY[, c(as.numeric(names(Pval[Pval < 0.05])), M+1)]  #M+1 番目は結果変数
    DataXY.Ttest <- DataXY[, c(as.numeric(names(Pval[Pval < 0.05])), M+1)]

    #---0.05 未満の変数が無い場合は P 値の最も小さい変数を抜き出す
    if(length(Pval[Pval < 0.05]) == 0){
      DataXY.Ttest <- DataXY[, c(as.numeric(names(Pval[Pval == min(Pval)])), M+1)]
      DataXY.Ttest <- DataXY[, c(as.numeric(names(Pval[Pval == min(Pval)])), M+1)]
    }
```

```
  #---予測
  Ttest.P                  <- Prediction(DataXY.Ttest, DataXY.Ttest)
#------t 検定//


#------//B：PLS
  #---トレーニングデータの作成
  if(nrow(DataX) <    5) PLS <- plsr(as.numeric(DataY) ~ DataX)
  if(nrow(DataX) >= 5) PLS <- plsr(as.numeric(DataY) ~ DataX, ncomp=5)

  DataXY.PLS               <- data.frame(predict(PLS, type="scores"), y=DataY)

  #---検証データの作成
  if(length(DataX) != M){
    DataXY.PLS <- data.frame(predict(PLS, type="scores",
              newdata=DataX), y=DataY)
  }
  #---サンプルが 1 つしか無い場合はテストベクトルを転置する
  if(length(DataX) == M){
    DataX2         <- t(DataX)
    DataXY.PLS <- data.frame(predict(PLS, type="scores",
              newdata=DataX2), y=DataY)
  }

  #---予測
  PLS.P                    <- Prediction(DataXY.PLS, DataXY.PLS)
#------PLS//


#------//C：PCA
  #---トレーニングデータの作成
  if(nrow(DataX) <    5) PCA <- pcr(as.numeric(DataY) ~ DataX)
  if(nrow(DataX) >= 5) PCA <- pcr(as.numeric(DataY) ~ DataX, ncomp=5)

  DataXY.PCA <- data.frame(predict(PCA, type="scores"), y=DataY)
```

```
  #---検証データの作成
  if(length(DataX) != M){
    DataXY.PCA <- data.frame(predict(PCA, type="scores",
                newdata=DataX), y=DataY)
  }
  if(length(DataX) == M){
    DataX2         <- t(DataX)
    DataXY.PCA <- data.frame(predict(PCA, type="scores",
                newdata=DataX2), y=DataY)
  }


  #---予測
  PCA.P                  <- Prediction(DataXY.PCA, DataXY.PCA)
#------PCA//


#------//D：クラスタリング
  Clus                    <- hclust(dist(t(DataX)), method="ward")
  Clus.group              <- rect.hclust(Clus, k=5)

  #---トレーニングデータの作成
  DataXY.Clus <- data.frame(
       clus1 = rowMeans(DataX[, rownames(data.frame(Clus.group[1]))]),
       clus2 = rowMeans(DataX[, rownames(data.frame(Clus.group[2]))]),
       clus3 = rowMeans(DataX[, rownames(data.frame(Clus.group[3]))]),
       clus4 = rowMeans(DataX[, rownames(data.frame(Clus.group[4]))]),
       clus5 = rowMeans(DataX[, rownames(data.frame(Clus.group[5]))]),
       y        = DataY)

  #---検証データの作成
  if(length(DataX) != M){
    DataXY.Clus <- data.frame(
       clus1 = rowMeans(DataX[, rownames(data.frame(Clus.group[1]))]),
       clus2 = rowMeans(DataX[, rownames(data.frame(Clus.group[2]))]),
       clus3 = rowMeans(DataX[, rownames(data.frame(Clus.group[3]))]),
       clus4 = rowMeans(DataX[, rownames(data.frame(Clus.group[4]))]),
```

```
            clus5 = rowMeans(DataX[, rownames(data.frame(Clus.group[5]))]),
            y     = DataY)
        }
        if(length(DataX) == M){
          DataXY.Clus <- data.frame(
            clus1 = mean(DataX[rownames(data.frame(Clus.group[1]))]),
            clus2 = mean(DataX[rownames(data.frame(Clus.group[2]))]),
            clus3 = mean(DataX[rownames(data.frame(Clus.group[3]))]),
            clus4 = mean(DataX[rownames(data.frame(Clus.group[4]))]),
            clus5 = mean(DataX[rownames(data.frame(Clus.group[5]))]),
            y     = DataY)
        }

        #---予測
        Clus.P              <- Prediction(DataXY.Clus, DataXY.Clus)
    #------クラスタリング//

    #---16 アルゴリズムの予測結果（16×サンプル数の行列）
    rbind(Ttest.P, PLS.P, PCA.P, Clus.P)
}
#------Apparent 予測の計算//
```

3. Err^{CV} を計算するための 2 値予測

関数名　：CrossValidate

内容　：16 種の統計モデルでの Err^{CV} を予測するための関数

パラメータ:UseDataXY（利用するデータ全体）、UseDataX（データの説明変数部分）、UseDataY（データの結果変数部分）、M（説明変数の数）、N2（サンプルの数）、F.Fold（CV の回数）

返り値　：CV を行った 16 モデルの予測結果行列

```
#------//16 アルゴリズムのクロスバリデーションを関数化（16 アルゴリズムの予測結果が返り値）
CrossValidate <- function(UseDataXY, UseDataX, UseDataY, M, N2, N.Fold){
    #---Clus で plot が必要なため空の図を描いておく
    plot(1)

    #---CV のための指示変数を作りデータに付ける
    CV          <- rep(c(1:N.Fold), rep(N2/N.Fold, N.Fold))
```

```
  DataXY.CV <- data.frame(UseDataXY, CV = CV)

  #---//CV の実行
    for(n.cv in 1:N.Fold){
    #---トレーニングセットとテストセットの準備
    DataXY.train <- DataXY.CV[CV != n.cv, ]
    DataXY.test  <- DataXY.CV[CV == n.cv, ]

    DataX.train  <- UseDataX[CV != n.cv, ]
    DataX.test   <- UseDataX[CV == n.cv, ]
    DataY.train  <- UseDataY[CV != n.cv]
    DataY.test   <- UseDataY[CV == n.cv]



    #------//A：t 検定
      #---変数選択
      Pval <- rep(99, M)
      for(i in 1:M){
        Ttest    <-  t.test(DataXY.train[DataXY.train$y==1,  i],  DataXY.train[DataXY.train$y==0,  i])
#ウェルチの t 検定
        Pval[i] <- Ttest$p.value
      }
      names(Pval) <- c(1:M)

      #---P 値が 0.05 未満の変数を抜き出す
      DataXY.Ttest.train <- DataXY.train[, c(as.numeric(names(Pval[Pval < 0.05])), M+1)]  #M+1 番
目は結果変数
      DataXY.Ttest.test  <- DataXY.test[,  c(as.numeric(names(Pval[Pval < 0.05])), M+1)]

      #---0.05 未満の変数が無い場合は P 値の最も小さい変数を抜き出す
      if(length(Pval[Pval < 0.05]) == 0){
        DataXY.Ttest.train <- DataXY.train[, c(as.numeric(names(Pval[Pval == min(Pval)])), M+1)]
        DataXY.Ttest.test  <- DataXY.test[,  c(as.numeric(names(Pval[Pval == min(Pval)])), M+1)]
      }

      #---予測
```

```
  Ttest.P0                    <- Prediction(DataXY.Ttest.train, DataXY.Ttest.test)
  if(n.cv == 1) Ttest.P <- Ttest.P0
  if(n.cv != 1) Ttest.P <- cbind(Ttest.P, Ttest.P0)
#------t 検定//


#------//B：PLS
  #---トレーニングデータの作成
  if(nrow(DataX.train) <    5) PLS <- plsr(as.numeric(DataY.train) ~ DataX.train)
  if(nrow(DataX.train) >= 5) PLS <- plsr(as.numeric(DataY.train) ~ DataX.train, ncomp=5)

  DataXY.PLS.train            <- data.frame(predict(PLS, type="scores"), y=DataY.train)

  #---検証データの作成
  if(length(DataX.test) != M){
    DataXY.PLS.test <- data.frame(predict(PLS, type="scores",
            newdata=DataX.test), y=DataY.test)
  }
  #---サンプルが 1 つしか無い場合はテストベクトルを転置する
  if(length(DataX.test) == M){
    DataX.test2       <- t(DataX.test)
    DataXY.PLS.test <- data.frame(predict(PLS, type="scores",
            newdata=DataX.test2), y=DataY.test)
  }

  #---予測
  PLS.P0                    <- Prediction(DataXY.PLS.train, DataXY.PLS.test)
  if(n.cv == 1) PLS.P <- PLS.P0
  if(n.cv != 1) PLS.P <- cbind(PLS.P, PLS.P0)
#------PLS//


#------//C：PCA
  #---トレーニングデータの作成
  if(nrow(DataX.train) <    5) PCA <- pcr(as.numeric(DataY.train) ~ DataX.train)
  if(nrow(DataX.train) >= 5) PCA <- pcr(as.numeric(DataY.train) ~ DataX.train, ncomp=5)
```

```
    DataXY.PCA.train <- data.frame(predict(PCA, type="scores"), y=DataY.train)

    #---検証データの作成
    if(length(DataX.test) != M){
      DataXY.PCA.test <- data.frame(predict(PCA, type="scores",
                newdata=DataX.test), y=DataY.test)
    }
    if(length(DataX.test) == M){
      DataX.test2         <- t(DataX.test)
      DataXY.PCA.test <- data.frame(predict(PCA, type="scores",
                newdata=DataX.test2), y=DataY.test)
    }


    #---予測
    PCA.P0                    <- Prediction(DataXY.PCA.train, DataXY.PCA.test)
    if(n.cv == 1) PCA.P <- PCA.P0
    if(n.cv != 1) PCA.P <- cbind(PCA.P, PCA.P0)
#------PCA//


#------//D：クラスタリング
    Clus                      <- hclust(dist(t(DataX.train)), method="ward")
    Clus.group              <- rect.hclust(Clus, k=5)

    #---トレーニングデータの作成
    DataXY.Clus.train <- data.frame(
          clus1 = rowMeans(DataX.train[, rownames(data.frame(Clus.group[1]))]),
          clus2 = rowMeans(DataX.train[, rownames(data.frame(Clus.group[2]))]),
          clus3 = rowMeans(DataX.train[, rownames(data.frame(Clus.group[3]))]),
          clus4 = rowMeans(DataX.train[, rownames(data.frame(Clus.group[4]))]),
          clus5 = rowMeans(DataX.train[, rownames(data.frame(Clus.group[5]))]),
          y         = DataY.train)

    #---検証データの作成
    if(length(DataX.test) != M){
      DataXY.Clus.test <- data.frame(
```

```
            clus1 = rowMeans(DataX.test[, rownames(data.frame(Clus.group[1]))]),
            clus2 = rowMeans(DataX.test[, rownames(data.frame(Clus.group[2]))]),
            clus3 = rowMeans(DataX.test[, rownames(data.frame(Clus.group[3]))]),
            clus4 = rowMeans(DataX.test[, rownames(data.frame(Clus.group[4]))]),
            clus5 = rowMeans(DataX.test[, rownames(data.frame(Clus.group[5]))]),
            y     = DataY.test)
        }
        if(length(DataX.test) == M){
          DataXY.Clus.test <- data.frame(
            clus1 = mean(DataX.test[rownames(data.frame(Clus.group[1]))]),
            clus2 = mean(DataX.test[rownames(data.frame(Clus.group[2]))]),
            clus3 = mean(DataX.test[rownames(data.frame(Clus.group[3]))]),
            clus4 = mean(DataX.test[rownames(data.frame(Clus.group[4]))]),
            clus5 = mean(DataX.test[rownames(data.frame(Clus.group[5]))]),
            y     = DataY.test)
        }

        #---予測
        Clus.P0                 <- Prediction(DataXY.Clus.train, DataXY.Clus.test)
        if(n.cv == 1) Clus.P <- Clus.P0
        if(n.cv != 1) Clus.P <- cbind(Clus.P, Clus.P0)
      #------クラスタリング//
      }
    #---CV の実行//

    #---16 アルゴリズムの予測結果（16×サンプル数の行列）
    rbind(Ttest.P, PLS.P, PCA.P, Clus.P)
}
#------CV の関数//
```

4. シミュレーションを行う関数

4.1. Type I の状況

関数名　：TypeI

内容　：TypeI の状況のデータを発生し 16 モデルの CV を指定された回数シミュレーションする。

パラメータ：N（片群のサンプル数）、M（説明変数の数）、Start.Seed（乱数の種の最初の値）、
　End.Seed（乱数の種の最後の値）、OutCsv（保存するデータのフルパス名）
返り値　：予測の正誤結果

```
#---------//シミュレーションの関数(TypeI)
TypeI <- function(N, M, Start.Seed, End.Seed, OutCsv){
  #---パラメータの作成
  N2 <- 2*N
  #------Start から End までシミュレーションを実行(Pred がシミュレーション結果)
  for(simu in Start.Seed : End.Seed){
  #------//データ作成
    #------TypeI
    #---乱数の種を設定する
    set.seed(simu)

    #---独立な多変量正規説明変数ベクトルを発生させる
    norm                <- rnorm(N2*M)

    #---行列形式に整形する
    DataX               <- matrix(norm, N2, M, byrow=T)

    #--列名と行名を付ける
    rownames(DataX) <- paste("n", c(1:N2), sep="")
    colnames(DataX) <- paste("x", c(1:M), sep="")

    #---説明変数ベクトルと独立な結果変数ベクトル Y を作成し説明変数行列 X と付ける（結果変数は 0 と 1 が交互）
    DataY               <- rep(c("0", "1"), N)
    DataXY              <- data.frame(DataX, y = DataY)
  #------データ作成//

  #------Apparent とクロスバリデーション（10-fold Cross Validation で各サンプルの正誤データを作成する）
  Pred.App    <- Apparent(DataXY, DataX, DataY, M, N2, 10)
  Pred.Cross <- CrossValidate(DataXY, DataX, DataY, M, N2, 10)

  #---//シミュレーション結果を下に付けていく
```

```
    Pred0          <- c(Ttest.lda = Pred.Cross[1, ],  Ttest.svm = Pred.Cross[2, ],  Ttest.knn =
Pred.Cross[3, ],  Ttest.nnet = Pred.Cross[4, ],
                   PLS.lda     = Pred.Cross[5, ],  PLS.svm     = Pred.Cross[6, ],  PLS.knn     =
Pred.Cross[7, ],  PLS.nnet     = Pred.Cross[8, ],
                   PCA.lda     = Pred.Cross[9, ],  PCA.svm     = Pred.Cross[10, ], PCA.knn     =
Pred.Cross[11, ], PCA.nnet     = Pred.Cross[12, ],
                   Clus.lda    = Pred.Cross[13, ], Clus.svm    = Pred.Cross[14, ], Clus.knn    =
Pred.Cross[15, ], Clus.nnet    = Pred.Cross[16, ])
    if(simu == Start.Seed) Pred <- Pred0
    if(simu != Start.Seed) Pred <- rbind(Pred, Pred0)
  #---シミュレーション結果//

  #---10 回毎にシミュレーション結果を書き込み
  if(simu %% 10 == 0) write.csv(Pred, outcsv, row.names=F)
  }
  Pred
}
#---------シミュレーションの関数(TypeI)//
```

4.2. Type II-1 の状況

関数名　：TypeII.1
内容　：TypeII-1 の状況のデータを発生し 16 モデルの CV を指定された回数シミュレーションする。
パラメータ：N（片群のサンプル数）、M（説明変数の数）、Start.Seed（乱数の種の最初の値）、End.Seed（乱数の種の最後の値）、OutCsv（保存するデータのフルパス名）
返り値　：予測の正誤結果

```
#---------//シミュレーションの関数(TypeII.1)
TypeII.1 <- function(N, M, Start.Seed, End.Seed, OutCsv){
  #---パラメータの作成
  N2 <- 2*N
  #------Start から End までシミュレーションを実行(Pred がシミュレーション結果)
  for(simu in Start.Seed : End.Seed){
  #------//データ作成
    #------TypeI
```

```
    #---乱数の種を設定する
    set.seed(simu)

    #---独立な多変量正規説明変数ベクトルを発生させる
    norm                <- rnorm(N2*M)

    #---行列形式に整形する
    DataX               <- matrix(norm, N2, M, byrow=T)

    #--列名と行名を付ける
    rownames(DataX) <- paste("n", c(1:N2), sep="")
    colnames(DataX) <- paste("x", c(1:M), sep="")

    #---説明変数ベクトルと独立な結果変数ベクトル Y を作成し説明変数行列 X と付ける（結果変数は 0 と 1 が交互）
    DataY               <- rep(c("0", "1"), N)

    #---X の 1～5 変数に y と関連付ける（N(±0.5,1)が 5 変数）
    DataX[, c(1:5)] <- DataX[, c(1:5)] + (as.numeric(DataY) - 0.5)



    DataXY              <- data.frame(DataX, y = DataY)
  #------データ作成//

  #------Apparent とクロスバリデーション（10-fold Cross Validation で各サンプルの正誤データを作成する）
  Pred.App   <- Apparent(DataXY, DataX, DataY, M, N2, 10)
  Pred.Cross <- CrossValidate(DataXY, DataX, DataY, M, N2, 10)

  #---//シミュレーション結果を下に付けていく
    Pred0          <- c(Ttest.lda = Pred.Cross[1, ],  Ttest.svm = Pred.Cross[2, ],  Ttest.knn = Pred.Cross[3, ],  Ttest.nnet = Pred.Cross[4, ],
                        PLS.lda = Pred.Cross[5, ],  PLS.svm = Pred.Cross[6, ],  PLS.knn = Pred.Cross[7, ],  PLS.nnet = Pred.Cross[8, ],
                        PCA.lda = Pred.Cross[9, ],  PCA.svm = Pred.Cross[10, ],  PCA.knn = Pred.Cross[11, ], PCA.nnet = Pred.Cross[12, ],
                        Clus.lda = Pred.Cross[13, ],  Clus.svm = Pred.Cross[14, ],  Clus.knn =
```

```
Pred.Cross[15, ], Clus.nnet    = Pred.Cross[16, ])
      if(simu == Start.Seed) Pred <- Pred0
      if(simu != Start.Seed) Pred <- rbind(Pred, Pred0)
   #---シミュレーション結果//

   #---10 回毎にシミュレーション結果を書き込み
   outcsv <- paste("L:/論文（D 論）/資料/2000 回 CVTypeII1_", Number, ".csv", sep="")
   if(simu %% 10 == 0) write.csv(Pred, outcsv, row.names=F)
   }
   Pred
}
#---------シミュレーションの関数(TypeII.1)//
```

4.3. Type II-2 の状況

関数名　：TypeII.2

内容　：TypeII-2 の状況のデータを発生し 16 モデルの CV を指定された回数シミュレーションする。

パラメータ：N（片群のサンプル数）、M（説明変数の数）、Start.Seed（乱数の種の最初の値）、End.Seed（乱数の種の最後の値）、OutCsv（保存するデータのフルパス名）

返り値　：予測の正誤結果

```
#---------//シミュレーションの関数(TypeII.2)
TypeII.2 <- function(N, M, Start.Seed, End.Seed, Number){
   #---パラメータの作成
   N2 <- 2*N
   #------Start から End までシミュレーションを実行(Pred がシミュレーション結果)
   for(simu in Start.Seed : End.Seed){
   #------//データ作成
      #------TypeI
      #---乱数の種を設定する
      set.seed(simu)

      #---独立な多変量正規説明変数ベクトルを発生させる
      norm                  <- rnorm(N2*M)
      #---行列形式に整形する
```

```
DataX                <- matrix(norm, N2, M, byrow=T)

#--列名と行名を付ける
rownames(DataX) <- paste("n", c(1:N2), sep="")
colnames(DataX) <- paste("x", c(1:M), sep="")

#---説明変数ベクトルと独立な結果変数ベクトル Y を作成し説明変数行列 X と付ける（結果変数は 0 と 1 が交互）
DataY                <- rep(c("0", "1"), N)

#---相関のある変数群を作る
r     <- 0.8
sig <- matrix(c(1, r, r, r, r, r, r, r, r, r,
      r, 1, r, r, r, r, r, r, r, r,
      r, r, 1, r, r, r, r, r, r, r,
      r, r, r, 1, r, r, r, r, r, r,
      r, r, r, r, 1, r, r, r, r, r,
      r, r, r, r, r, 1, r, r, r, r,
      r, r, r, r, r, r, 1, r, r, r,
      r, r, r, r, r, r, r, 1, r, r,
      r, r, r, r, r, r, r, r, 1, r,
      r, r, r, r, r, r, r, r, r, 1
      ), 10, 10)
#---10 変数同士の相関が 0.8、アップレギュレート群が 3 つ、ダウンレギュレート群が 2 つ
set.seed(simu)
CorX1 <- mvrnorm(N, mu=rep(0.5, 10), Sigma=sig)
set.seed(simu + 10000)
CorX2 <- mvrnorm(N, mu=rep(0.5, 10), Sigma=sig)
set.seed(simu + 20000)
CorX3 <- mvrnorm(N, mu=rep(0.5, 10), Sigma=sig)
set.seed(simu + 30000)
CorX4 <- mvrnorm(N, mu=rep(-0.5, 10), Sigma=sig)
set.seed(simu + 40000)
CorX5 <- mvrnorm(N, mu=rep(-0.5, 10), Sigma=sig)

CorX    <- cbind(CorX1, CorX2, CorX3, CorX4, CorX5)
```

```
    #---X の 1～50 変数に y と関連付ける（Y=1 のサンプルで説明変数の変数群が相関がある）
    DataX[as.character(DataY) == "1", c(1:50)] <- CorX


    DataXY              <- data.frame(DataX, y = DataY)
  #------データ作成//

  #------Apparent とクロスバリデーション（10-fold Cross Validation で各サンプルの正誤データを作成する）
  Pred.App   <- Apparent(DataXY, DataX, DataY, M, N2, 10)
  Pred.Cross <- CrossValidate(DataXY, DataX, DataY, M, N2, 10)

  #---//シミュレーション結果を下に付けていく
    Pred0          <- c(Ttest.lda = Pred.Cross[1, ],  Ttest.svm = Pred.Cross[2, ],  Ttest.knn = Pred.Cross[3, ],  Ttest.nnet = Pred.Cross[4, ],
                        PLS.lda = Pred.Cross[5, ],  PLS.svm = Pred.Cross[6, ],  PLS.knn = Pred.Cross[7, ],  PLS.nnet = Pred.Cross[8, ],
                        PCA.lda = Pred.Cross[9, ],  PCA.svm = Pred.Cross[10, ], PCA.knn = Pred.Cross[11, ], PCA.nnet = Pred.Cross[12, ],
                        Clus.lda = Pred.Cross[13, ], Clus.svm = Pred.Cross[14, ], Clus.knn = Pred.Cross[15, ], Clus.nnet = Pred.Cross[16, ])
    if(simu == Start.Seed) Pred <- Pred0
    if(simu != Start.Seed) Pred <- rbind(Pred, Pred0)
  #---シミュレーション結果//

  #---10 回毎にシミュレーション結果を書き込み
  outcsv <- paste("L:/論文（D 論）/資料/2000 回 CVTypeII2_", Number, ".csv", sep="")
  if(simu %% 10 == 0) write.csv(Pred, outcsv, row.names=F)
  }
  Pred
}
#---------シミュレーションの関数(TypeII.2)//
```

5. 誤分類率

5.1. 誤分類率の算出

関数名　：ErrCal
内容　：予測の正誤結果から、誤分類率の計算をする。
パラメータ：UsePred（予測の正誤結果）、N2（サンプル数）
返り値　：16 モデルの誤分類率行列

```
#---------//予測データから誤分類率を計算する関数
ErrCal <- function(UsePred, N2){
   cbind(Ttest.lda    = rowMeans(UsePred[, 1              : N2        ]),
   Ttest.svm    = rowMeans(UsePred[, (N2     +1) : (N2*2) ]),
   Ttest.knn    = rowMeans(UsePred[, (N2*2 +1) : (N2*3) ]),
   Ttest.nnet = rowMeans(UsePred[, (N2*3 +1) : (N2*4) ]),
   PLS.lda       = rowMeans(UsePred[, (N2*4 +1) : (N2*5) ]),
   PLS.svm       = rowMeans(UsePred[, (N2*5 +1) : (N2*6) ]),
   PLS.knn       = rowMeans(UsePred[, (N2*6 +1) : (N2*7) ]),
   PLS.nnet      = rowMeans(UsePred[, (N2*7 +1) : (N2*8) ]),
   PCA.lda       = rowMeans(UsePred[, (N2*8 +1) : (N2*9) ]),
   PCA.svm       = rowMeans(UsePred[, (N2*9 +1) : (N2*10)]),
   PCA.knn       = rowMeans(UsePred[, (N2*10+1) : (N2*11)]),
   PCA.nnet      = rowMeans(UsePred[, (N2*11+1) : (N2*12)]),
   Clus.lda      = rowMeans(UsePred[, (N2*12+1) : (N2*13)]),
   Clus.svm      = rowMeans(UsePred[, (N2*13+1) : (N2*14)]),
   Clus.knn      = rowMeans(UsePred[, (N2*14+1) : (N2*15)]),
   Clus.nnet    = rowMeans(UsePred[, (N2*15+1) : (N2*16)]))
}
#---------分類率計算関数//
```

5.2. Err^{app} の平均と標準偏差の算出

関数名　：ErrMSApp
内容　：Err^{App} の平均と標準偏差を計算する
パラメータ：UseErr（16 モデルの誤分類率行列）
返り値　：Err^{App} の平均と標準偏差の行列

```
#---------//App 誤分類率の平均と標準偏差を計算する関数
ErrMSApp <- function(UseErrApp){
  #---ErrApp の平均と標準偏差
  Mean.App   <- apply(UseErrApp, 2, mean)
  SD.App     <- apply(UseErrApp, 2, function(data) sqrt(var(data)))

  #---各誤分類率の平均と標準偏差の行列
  cbind(Mean.App, SD.App)
}
#---------App 誤分類率の平均と標準偏差計算関数//
```

5.3. 各種 Err^{CV} の平均と標準偏差の算出

関数名　：ErrMSCV
内容　：Err^{CV}、Err^{CVM}、Err^{CVS} の平均と標準偏差を計算する
パラメータ：UseErr（16 モデルの誤分類率行列）
返り値　：Err^{CV}、Err^{CVM}、Err^{CVS} の平均と標準偏差の行列

```
#---------//CV 誤分類率の平均と標準偏差を計算する関数
ErrMSCV <- function(UseErrCV){
  #---ErrCV の平均と標準偏差
  Mean.cv   <- apply(UseErrCV, 2, mean)
  SD.cv     <- apply(UseErrCV, 2, function(data) sqrt(var(data)))

  #---ErrCVS の平均と標準偏差
  Err.Cvs   <- apply(UseErrCV, 1, min)
  Mean.Cvs <- mean(Err.Cvs)
  SD.Cvs    <- sqrt(var(Err.Cvs))

  #---ErrCVM の平均と標準偏差
  Err.Cvm   <- apply(UseErrCV, 1, mean)
  Mean.Cvm <- mean(Err.Cvm)
  SD.Cvm    <- sqrt(var(Err.Cvm))

  #---各誤分類率の平均と標準偏差の行列
  rbind(cbind(Mean.cv, SD.cv), c(Mean.Cvs, SD.Cvs), c(Mean.Cvm, SD.Cvm))
```

```
}
#---------CV 誤分類率の平均と標準偏差計算関数//
```

6. K's Plot

6.1. k サンプル抽出したときの Err^{CV} の計算と K's Plot の描画

関数名　：KsErr.TLda
内容　：T 検定で関連の高い上位 5 変数に変数選択し、判別分析で予測するという統計モデルを利用した際の、K's Err の計算と K's Plot の描画
パラメータ：UseDataXY（データの一列目に結果変数を持ってきて、変数名は“y”とする)、Ks（サンプル数 k_i の最小値、この値から Ks×10 までの 10 回のサンプリングを行う）
返り値　：k_i と Err^{CV} の行列、K's Plot の図

```
#------//T 検定→判別分析の K's Plot
#---UseDataXY の一列目に結果変数 Y を持ってくる
KsErr.TLda <- function(UseDataXY, Ks){
  N.Err0 <- c(99, 99)
  for(NSample in seq(Ks, Ks*10, by=Ks)){
    Ttest.lda.cv0 <- 99
    set.seed(1)

    #---結果変数のバランスがとれるようにサンプリングする
    #CubeX   <- UseDataXY[, 1]
    #p          <- rep(NSample/nrow(UseDataXY), nrow(UseDataXY))
    #sample <- samplecube(CubeX, p, 1, FALSE)

    DataXY <- UseDataXY[sample(1:nrow(UseDataXY), NSample), ]
    DataX   <- DataXY[, -1]
    DataY   <- DataXY[,   1]
    CV        <- rep(1:10, -floor(-NSample/10))[1:NSample]
    m <- ncol(DataX)

    for(ncv in 1:10){
      DataXY.test   <- DataXY[CV == ncv, ]
      DataXY.train <- DataXY[CV != ncv, ]
      DataX.test     <- DataX[CV   == ncv, ]
```

```
        DataX.train    <- DataX[CV    != ncv, ]
        DataY.test     <- DataY[CV    == ncv    ]
        DataY.train    <- DataY[CV    != ncv    ]
      if(length(DataY.test)!=0){
          #---------t 検定
          Pval <- 99                          #初期値
          for(i in 1:m){
            Ttest  <-  t.test(DataX.train[DataXY.train$y  ==  0,  i],  DataX.train[DataXY.train$y==1,  i])
#ウェルチの t 検定
            Pval    <- c(Pval, Ttest$p.value)
          }
          Pval <- Pval[-1]                        #P 値のベクトル
          names(Pval) <- c(1:m)

          #------P 値の上位 5 つの変数を抜き出し（小さいものから 5 つを抜き出す）
          DataXY.Ttest.train <- DataXY.train[, c(as.numeric(names(Pval[rank(Pval) <= 5]))+1, 1)]
          DataXY.Ttest.test    <- DataXY.test[,    c(as.numeric(names(Pval[rank(Pval) <= 5]))+1, 1)]

          #------予測
          #---判別分析
          Ttest.lda         <- lda(y~., data=DataXY.Ttest.train)
          Ttest.lda.cv              <-      length(DataXY.test$y[DataXY.test$y       !=       predict(Ttest.lda,
DataXY.Ttest.test)$class])    #CV error count
          Ttest.lda.cv0 <- c(Ttest.lda.cv0, Ttest.lda.cv)
        }
      }
      Ttest.lda.cv <- sum(Ttest.lda.cv0[-1])/NSample
      N.Err             <- c(NSample, Ttest.lda.cv)
      N.Err0            <- rbind(N.Err0, N.Err)
    }
    plot(N.Err0[-1, ], xlim=c(1, Ks*10), ylim=c(0, 1), xlab="K-Sample", ylab="CV Error")
    N.Err0[-1, ]
}
#------T 検定→判別分析の K's Plot//
```

6.2. K's Plot への指数関数モデルの当てはめ

関数名　：KsPlot
内容　：K's Plot に指数関数モデルを当てはめた結果を描画する
パラメータ：UseData（K's Err の行列）、XLim（描画の x 軸の最大値）、Default（指数関数モデルの初期値、この値で推定値が大きく変化する）
返り値　：K's Plot に指数関数モデルを当てはめた図

```
KsPlot <- function(UseData, XLim, Default){
  ExpDecay <- function(par){
    k          <- par[1]
    plateau <- par[2]
    x          <- UseData[, 1]
    y          <- UseData[, 2]
    yhat       <- (0.5-plateau)*exp(-k*x) + plateau
    sum((y - yhat)**2)
  }
  #---微分の関数
  ExpDecay.grad     <- function(par){
    k          <- par[1]
    plateau <- par[2]
    x          <- UseData[, 1]
    y          <- UseData[, 2]

c(sum((-x*(0.5-plateau)*exp(-k*x))*y+(-2*x*(0.25-plateau+plateau**2)*exp(-k*x*2))+2*x*(plateau**2)*exp(-k*x)),
        sum((-exp(-k*x)+1)*y+(1-exp(-k*x*2)+2*(exp(-k*x*2)-2*exp(-k*x)+1)*plateau)))
  }

  ParExpDecay       <- constrOptim(Default, ExpDecay, ExpDecay.grad, ui=rbind(c(0, -1), c(0, 1)), ci=c(-0.5, 0),
                    method="BFGS", outer.eps=1e-07)

  plot(UseData, xlim=c(1, XLim), ylim=c(0, 1), xlab="K-Sample", ylab="CV Error")
  Ks.Est                    <- (0.5-ParExpDecay$par[2])*exp(-ParExpDecay$par[1]*c(1:XLim)) + ParExpDecay$par[2]
  lines(c(1:XLim), Ks.Est)
```

```
  text(XLim*0.1,  0.8,  paste("k:",  round(ParExpDecay$par[1],  digits=2),  "  Default:",
Default[1]), adj=0)
  text(XLim*0.1, 0.7, paste("plateau:", round(ParExpDecay$par[2], digits=2), "  Default:", Default[2]),
adj=0)
}
```

7. 乱数による Err^{CVS} の理論論分布の発生

関数名　：CvsCvm.Theor

内容　：モデル間相関とサンプル間相関を考慮した、Err^{CVS} と Err^{CVM} の理論論分布を発生し、その平均と標準偏差を計算する

パラメータ：N.Simu（シミュレーション回数）、N.Alg（試した統計モデルの数 ）、N2（サンプル数）、rm（多変量正規分布のモデル間相関係数）、rs（多変量正規分布のサンプル間相関係数）、P（真の誤分類率）、Seed（乱数の種）

返り値　：Err^{CVS} と Err^{CVM} の平均と標準偏差と、ベルヌーイ分布のモデル間相関係数とサンプル間相関係数

```
#---------//モデル間相関（rm）とサンプル間相関（rs）を考慮した理論分布
CvsCvm.Theor <- function(N.Simu, N.Alg, N2, rm, rs, P, Seed){
  #------サンプル間相関行列
  Rs                    <- diag(1, N2)
  Rs[upper.tri(Rs)] <- -rs      #ここと次を-rs にしたら正と負が半々の相関行列ができる
  Rs[lower.tri(Rs)] <- -rs

  Rs[((N2/2)):N2, 1:(N2/2)] <- rs
  Rs[1:(N2/2), ((N2/2)):N2] <- rs

  #------モデル間相関行列
  #Rm                    <- diag(rm, N2)

  Rm          <- Rs*rm
  diag(Rm) <- rm

  #------全体の相関行列
  L0 <- rep(99, N2*N.Alg)
  for(j in 1:N.Alg){
    for(i in 1:N.Alg){
    if(j == 1){
```

```
      if(i == 1) L <- Rs
      if(i != 1) L <- cbind(L, Rm)
    }
    if(j != 1){
      if(i == 1)              L <- Rm
      if(i !=1 & i != j) L <- cbind(L, Rm)
      if(i == j)              L <- cbind(L, Rs)
    }
    }
  L0 <- rbind(L0, L)
  }
  L <- L0[-1, ]

  L                       <- make.positive.definite(L)
  eigen(L)$values

  #------サンプルを発生させる
  #---正規分布の発生（N.Simu×N2*N.Alg の行列）
  set.seed(Seed)
  SampleNorm   <- mvrnorm(N.Simu, mu=rep(0, N2*N.Alg), Sigma=L)

  #---ベルヌーイ分布に変換（N.Simu×N2*N.Alg の行列、累積確率を入れることで真の誤分類率を表現する）
  SampleBern   <- ifelse(SampleNorm <= qnorm(P), 0, 1)

  #-----ベルヌーイ分布の相関チェック
  RsRm <- Bern.Cor(SampleBern, N2, N.Alg)


  #------二項分布の計算（N.Simu×N.Alg の行列）
  for(i in 1:N.Alg){
    if(i == 1) SampleBinom <- rowMeans(SampleBern[, 1:N2])
    if(i >= 2) SampleBinom <- cbind(SampleBinom, rowMeans(SampleBern[, (N2*(i-1)+1):(N2*i)]))
  }

  #------ErrCVS、ErrCVM の計算
  ErrCVS <- apply(SampleBinom, 1, min)
```

```
    ErrCVM <- apply(SampleBinom, 1, mean)

    #------ErrCVS、ErrCVM の期待値と分散
    ErrCVS.MeanSd <- c(mean(ErrCVS), sd(ErrCVS))
    ErrCVM.MeanSd <- c(mean(ErrCVM), sd(ErrCVM))

    round(rbind(ErrCVS = ErrCVS.MeanSd,
                ErrCVM = ErrCVM.MeanSd,
                RsRm   = RsRm), digits=3)
}
```