

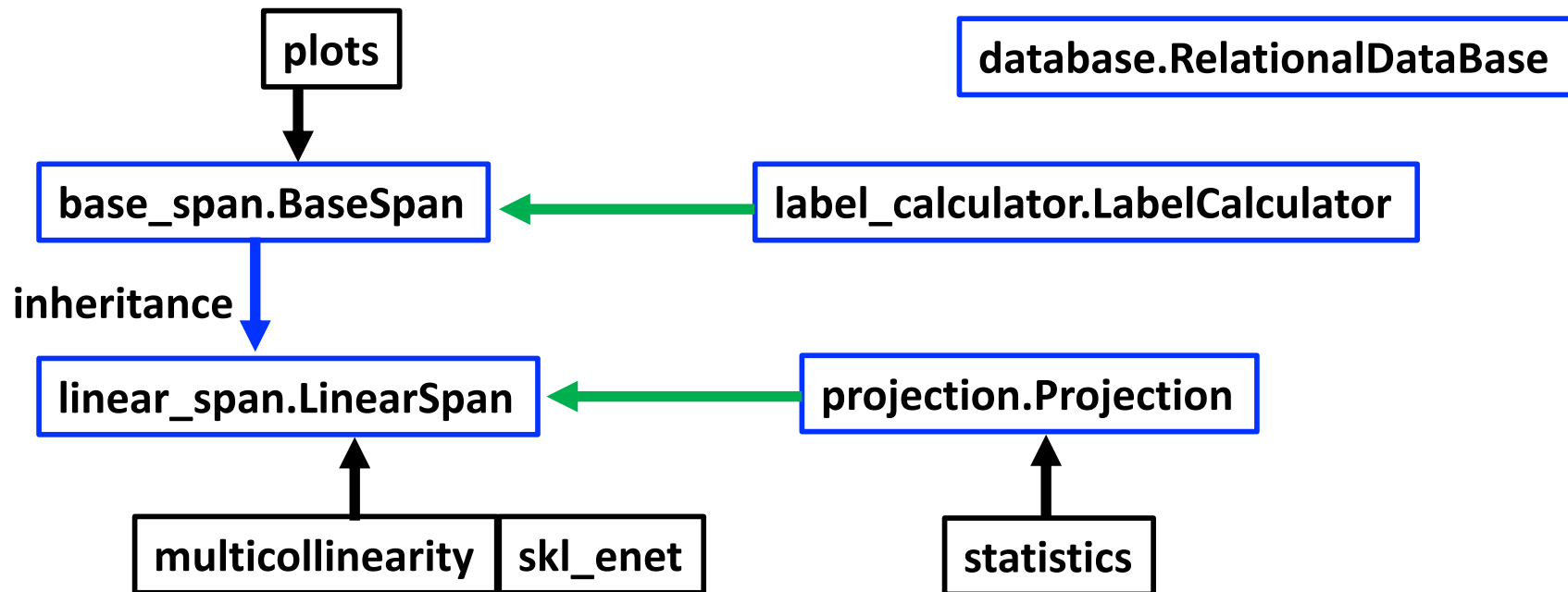
A basic manual of LIDG program

Hitoshi Fujii

Outline

- **Configuration of LIDG program**
- **Workflow of interpretable sparse modeling by LIDG**
- **Flowchart**
- **Term description**
- **Methods**
- **The meaning of subspace list**
- **Definition of criteria**
- **An example of backward selection method**

Configuration of LIDG program



lidg/___init___py

base_span.py

Basic method to handle DataFrame

label_calculator.py

Label handling, calculation of strings

linear_span.py

Main program

projection.py

Linear regression

multicollinearity.py

Find multicollinearities in span

statistics.py

Statistic calculations

skl_enet.py

Elastic net calculation by sklearn

plots.py

Plotting method

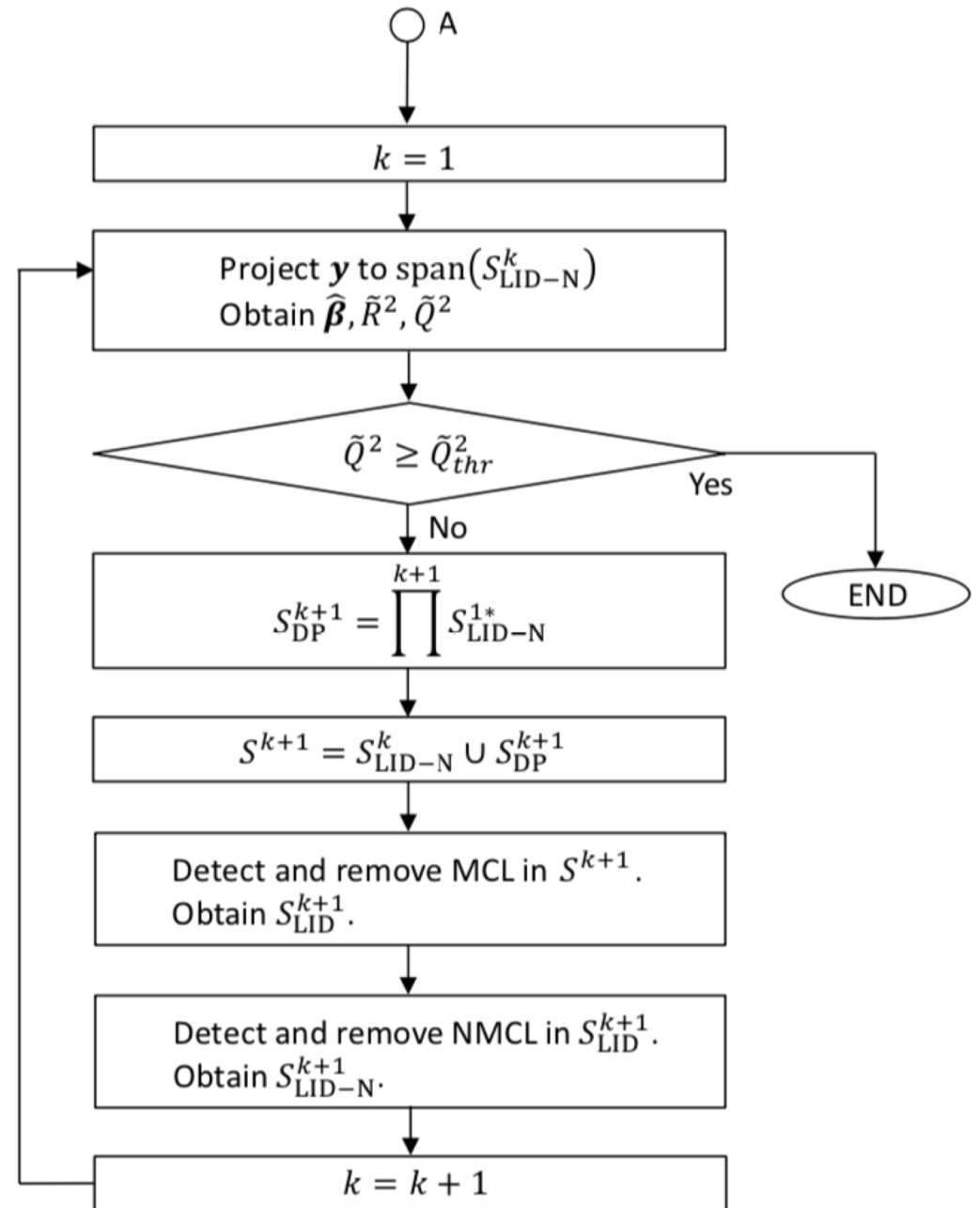
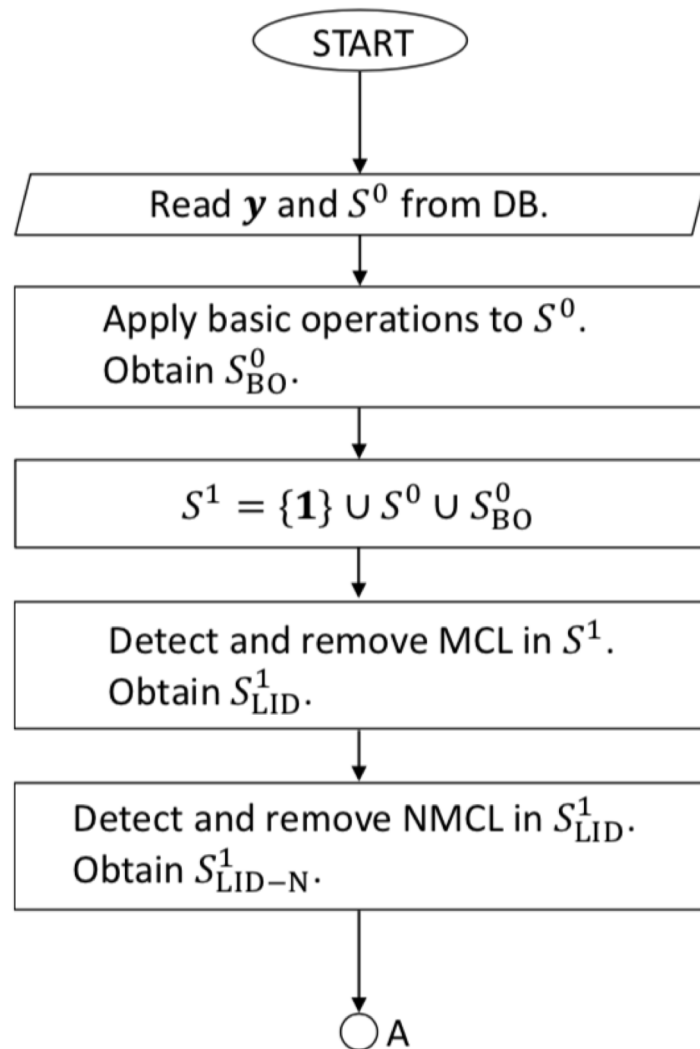
database.py

Handle a relational database

Workflow of interpretable sparse modeling by LIDG

1. Read target and descriptor vectors
2. Generate first order descriptors by basic operations (BO)
3. Find and remove multicollinearities (MCL)
4. Find and remove strong near multicollinearities (NMCL)
5. Perform OLS and check the accuracy of the model. If the accuracy is sufficient, then go to 8.
6. Generate higher order descriptors by direct product (DP)
7. Go to 3. (Repeat until the regression model reaches a sufficient accuracy)
8. Perform a descriptor selection, such as BS, LASSO, MCTS, etc.

Flowchart



Term description

線形包 (Linear span) または単にスパン:

数学用語。ベクトルの集合であり、それらが張るベクトル空間のこと。

サンプル空間、目的変数ベクトル、記述子ベクトル、記述子空間、記述子行列:

m 個のサンプル物質(の様々な物性値に対する値)を直交基底とした m 次元サンプル空間 V を考える。ある一つの物性値を目的変数し、各サンプル物質に対する目的変数の値の配列をサンプル空間上の m 次元ベクトルとみなす。これを目的変数ベクトル \mathbf{y} と呼ぶことにする。同様に、他の n 個の物性値(記述子)もサンプル空間上のベクトルと見なし、これら n 個の m 次元ベクトルを記述子ベクトルと呼ぶことにする。記述子ベクトルの集合 $S \equiv \{x_1, x_2, \dots, x_n\}$ を考え、それらが張る(生成する)空間を記述子空間 $W \equiv \text{Span}(S) \subset V$ と呼ぶことにする。記述子ベクトルを列ベクトルとした $m \times n$ 行列は記述子行列(計画行列) $X \equiv [x_1, x_2, \dots, x_n]$ と呼ばれる。この時 $\text{rank}(X) = \dim(W)$ が成り立つ。つまりスパンの基底の数(線型独立なベクトルの数)は記述子行列の階数から求まる。

最小二乗法 (OLS):

幾何学的には、スパン W (超平面)に目的変数ベクトル \mathbf{y} を射影(vector projection)したベクトルを見つけることに他ならない。 $(\hat{\mathbf{y}} = X\hat{\boldsymbol{\beta}})$

残差 (residual):

vector rejection、つまり、OLSでの推定値 $\hat{\mathbf{y}}$ から目的変数ベクトル \mathbf{y} へのベクトル($\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$)。

Methods (1/6) (basic methods)

`.read(str or df)`

スペース区切りテキストファイルもしくはpandasデータフレームを読み込む。

`.set_y(str)`

データフレームからラベル(str)で指定した目的変数を分離し、それをクラス変数self.yに格納する。

`.add_const()`

定数項ベクトルが付け加えられる。

`.change_labels(str)`

記述子のラベルを通し番号付きの指定したラベル(str)に変更。

`.show()`

スパン内の記述子ベクトルの数、それらのラベルを表示する。

`.show_save()`

バックアップ用スパン内の記述子ベクトルの数、それらのラベルを表示する。

スパンに関するクラス変数は3つあり、一つはバックアップ用のもの(self.x_save)で、もう一つは作業用のもの(self.x)である。実際の使用では、作業用スパン内の記述子を削除したり新たな記述子を追加して回帰精度などを逐一確認するというようなことが頻繁に行われるので、このように、いつでも変更前に戻れる様、バックアップのためのクラス変数を用意している。最後の3つ目のクラス変数はself.originであり、これは新たな記述子を作る際に使われる。.readで読み込まれた時点ではこれら3つは全て同じである。また、スパンをいくら変更しても、self.originが変更されることはない。(new()メソッドを除いて。)

Methods (2/6) (basic methods)

`.save()`

作業用スパンの記述子 (`self.x`) をバックアップ用スパン (`self.x_save`) へコピーする。

`.load(str or list)`

バックアップ用スパン (`self.x_save`) 内の指定した記述子を作業用スパン (`self.x`) へ追加する。

`.remove(str or list)`

指定した記述子を作業用スパンから削除する。バックアップはそのまま。

オプション “const”: 定数ベクトルを見つけて削除する。

“inf”: 発散した要素を持つ記述子ベクトルを見つけて削除する。

“all”: 全ての記述子を削除する。

`.add(str or list)`

文字列(式)を与えると、`self.origin`内の記述子に基づいて記述子を新たに作り、スパンに追加する。変更したラベルで式を書いても、内部で一旦全てのラベルをオリジナルラベルまで戻してから式の値を計算するので、**ラベルの変更を気にせず記述可能**。(`self.origin` に登録されている記述子以外を使った演算は計算不可能。ただし、実数は自由に使える。)

`.add_rh()`

一様乱数ベクトルをスパンに追加。色々なテストに使える。

`.add_rg()`

ガウシアンベクトルをスパンに追加。色々なテストに使える。

`.plot(df,series)`, `.plot_vs(series1,series2)`: `pandas.DataFrame`、`pandas.Series`のプロット

Methods (3/6) (generation, symmetrization)

.new(renew=True): return span

元のスパンから新たなスパンを作る。renew=Trueとすると、新たなスパンのself.xがself.originへコピーされる。つまり、ラベルを変更していた場合、それがoriginへ反映されてしまい、もはや最初のラベルの情報が失われる。しかし、これにより、一番最初のラベルにいちいち戻って記述子を計算しなくてよくなるので、計算時間を短縮できるようになる。

.sub(str or list): return span

指定した部分スパンを返す。

.join(str or list): return span

指定したスパンを追加 (concat) したスパンを返す。

.gen_bo(str or list): return span

基本演算により記述子を生成する。

"r"は逆数、"a"は絶対値、"b"は絶対値の逆数。

"r+" ("r-")は記述子の和(差)の逆数。全ての組み合わせを計算。"a+","b+"も同様。

新たな基本演算を自分で定義したい場合は、label_calculator.pyの中の、op1_dicに追加すれば良い。

例えば、平方根をとる演算子なら"u"、指数関数なら"e"などと、演算子を一文字だけで表す様にする。

.gen_dp(int): return span

直積により記述子を生成する。(高次記述子生成)

.sym(list): return span

対称化した記述子を返す。これを使うには、記述子のラベルがx(0,1,2,..)の様に識別子が明記されていないといけない。引数には[(0,1,2,..),(1,0,2,..),(0,2,1,..),...]の様に置換の形で恒等操作を含めた全ての対称操作を与える。これらが群を成してないとエラーとなる。

Methods (4/6) (multicollinearity)

.lid(normalize=False,tole1,tole2): return span

線型独立化したスパンを返す。normalize=Trueとすると多重共線性関係の出力が規格化した記述子に対するものになる。(内部では数値安定性のため、全ての記述子は一旦規格化される)

tole1が小さいと、なるべく初期の記述子の順番を守りながら多重共線性関係を見つけるようになる。もし、左から順に単純な形の記述子を並べている場合(推奨)、tole1を十分小さくすることで、単純な形の記述子を優先的に残すことができる。多重共線性関係がある場合、統計学的には、どの記述子が良い記述子かという判断ができなくなるのであるが、解釈可能なモデリングという観点から、単純な形の記述子は複雑なものより価値が高いと判断する。

tole2を大きくすると検出する線形独立(と見なされる)記述子の数が減る。つまり、余分に記述子が削られることになり、結果として、強い準多重共線性をも排除することができる。よほど自明な多重共線性関係でない限り、多重共線性と準多重共線性との違いは(数値的には)曖昧なのである。

多重共線性がある場合の記述子選択の指標としては「単純さ」の他に、「超体積の大きさ」、「 y との相関の強さ」などが考えられる。

.volume_comb(defi_list,cand_list,k,ntop=None)

defi_listで必ず使用する記述子を指定し、cand_listの中からk個の候補を選び、それらの記述子を稜とする超体積を計算し、上位ntopの組み合わせを出力する。

.xty()

各記述子と y との相関 $X^T y$ を計算する。

Methods (5/6) (near multicollinearity)

`.project(sort=None, normalize=False)`

OLSを行う。normalize=Trueとすると、規格化した記述子ベクトルに対する回帰係数を出力する。これにより、回帰係数の大きさをフェアに比較することができるようになる。推定値がself.hyに格納される。

推定精度(フィッティング精度)の指標

e2: 残差平方和、mse: mean square error (e2をサンプル数で割ったもの)、rmse: mseの平方根

R2: 決定係数

予測精度(汎化精度)の指標

eq2: 予測残差、mse_q: 予測残差をサンプル数で割ったもの、rmse_q: mse_qの平方根

Q2: 予測に対する決定係数 (leave-one-out cross validationの結果と同じになる。)

AIC: 赤池情報量基準

TR2, TQ2: それぞれ、自然なR2、自然なQ2。これらは、yの分散(にmをかけたもの)の代わりに、yのノルム二乗でe2、qe2を割ったもの。このLIDGプログラムオリジナルの指標であり、R2、Q2と似た振る舞いをする。

b, |b|: 回帰係数とその絶対値。

p_val, -log₁₀(p): 回帰係数のT検定のp値と、その対数を取ったもの。-log₁₀(p)が大きいほど重要な記述子と判断できる。

G2: 記述子の重要度の指標(LIDGオリジナル)。規格化した記述子に対する|b|や、-log₁₀(p)と同じ振る舞いをする期待される。

TRi2: 準多重共線性の強さを表し(LIDGオリジナル)、VIFなどと同様の振る舞いをする。定数項を扱える。

Methods (6/6)

.rij(ntop)

ピアソンの対相関係数(標準化した記述子同士の内積)の大きいペアからntopまで出力。

.nij(ntop)

規格化した記述子同士の内積を計算し、大きいものからntopまで出力。

.gij(ntop)

一般化対相関係数(オリジナルの指標)。ピアソンの対相関係数では準共線性しか検出できないが、この一般化対相関係数は準多重共線性も考慮した指標となっている。

.gik(str,ntop)

ある一つの記述子を指定し、それとの一般化対相関係数を計算する。

.corr(ntop)

.rij()、.nij()、.gij()をまとめて出力。

.outlier()

外れ値の計算。テコ値の他、内部スチューデント化残差、外部スチューデント化残差、DFFITSを計算。

.enet(almax=2,almin=-4,l1r=1.0,nf=10)

SklearnによるElastic Netの計算。

almin-almax: Hyperparameterの範囲の指定。

l1r = 1.0 for LASSO, l1r=0.0 for Ridge regression

nf: fold number of cross validation

The meaning of subspace list

検出された多重共線性関係

$$\vec{x}_k = c_i^k \vec{x}_i + c_j^k \vec{x}_j$$

$$\vec{x}_l = c_i^l \vec{x}_i + c_j^l \vec{x}_j$$

$$\vec{x}_m = c_i^m \vec{x}_i + c_j^m \vec{x}_j$$



以下のようにリスト表示することにする

$$[\vec{x}_k, \vec{x}_l, \vec{x}_m, [\vec{x}_i, \vec{x}_j]]$$

Temporary basis

この場合、この5つの記述子が張る空間
 $\text{span}(\vec{x}_i, \vec{x}_j, \vec{x}_k, \vec{x}_l, \vec{x}_m)$
の基底の数は2つだけで良いことがわかる。

5つから2つの基底を自由に選んで良い(空間の持つ情報は何も失われない)

Select \vec{x}_i, \vec{x}_j and remove $\vec{x}_k, \vec{x}_l, \vec{x}_m$

or

select \vec{x}_i, \vec{x}_k and remove $\vec{x}_j, \vec{x}_l, \vec{x}_m$

or

select \vec{x}_l, \vec{x}_m and remove $\vec{x}_i, \vec{x}_j, \vec{x}_k$

...

(選択基準は色々あるが、基本的には残したいものを残すようにすれば良い。)

このようにして、回帰精度を全く落とすことなく記述子の数を減らすことができる。

Definition of criteria

TABLE I. Summary of conventional statistical criteria and newly proposed criteria for empirical-law discovery (ELD). DDR and TDR stand for descriptor-descriptor regression and target-descriptor regression, respectively.

Problem	Criterion	
	Ordinal statistics	This study (ELD)
Estimation accuracy of DDR for \mathbf{x}_i	$R_i^2 = 1 - \frac{ \mathbf{e}_i ^2}{ \mathbf{J}_0 \mathbf{x}_i ^2}$ $\text{VIF} = \frac{1}{1 - R_i^2}$	$\tilde{R}_i^2 = 1 - \frac{ \mathbf{e}_i ^2}{ \mathbf{x}_i ^2}$
Importance of \mathbf{x}_i in DDR for \mathbf{x}_j	-	$G_{ij}^2 = 1 - \frac{ \mathbf{e}_j ^2}{ \mathbf{e}_j^{(i)} ^2} = g_{ij}^2$
Pairwise correlation between \mathbf{x}_i and \mathbf{x}_j	$r_{ij} = \frac{\mathbf{J}_0 \mathbf{x}_i}{ \mathbf{J}_0 \mathbf{x}_i } \cdot \frac{\mathbf{J}_0 \mathbf{x}_j}{ \mathbf{J}_0 \mathbf{x}_j }$	$g_{ij} = \frac{\mathbf{J}^{(i,j)} \mathbf{x}_i}{ \mathbf{J}^{(i,j)} \mathbf{x}_i } \cdot \frac{\mathbf{J}^{(i,j)} \mathbf{x}_j}{ \mathbf{J}^{(i,j)} \mathbf{x}_j }$
Estimation accuracy of TDR	$R^2 = 1 - \frac{ \mathbf{e}_y ^2}{ \mathbf{J}_0 \mathbf{y} ^2}$	$\tilde{R}^2 = 1 - \frac{ \mathbf{e}_y ^2}{ \mathbf{y} ^2}$
Prediction capability of TDR	$Q^2 = 1 - \frac{ \mathbf{q}_y ^2}{ \mathbf{J}_0 \mathbf{y} ^2}$ $\text{AIC} = -2\ln L + 2k$	$\tilde{Q}^2 = 1 - \frac{ \mathbf{q}_y ^2}{ \mathbf{y} ^2}$
Importance of \mathbf{x}_i in TDR	p value of T -test	$G_{iy}^2 = 1 - \frac{ \mathbf{e}_y ^2}{ \mathbf{e}_y^{(i)} ^2}$

TRi2はこれのこと

G2のこと

$$\begin{aligned} \text{AIC} &= -2\ln L + 2k \\ &= m \left(\ln(|\mathbf{e}_y|^2) + \ln\left(\frac{2\pi}{m}\right) + 1 \right) + 2k \end{aligned}$$

Jはrejection operator
 \mathbf{J}_0 は定数ベクトルからのrejection、
 $\mathbf{J}^{(i,j)}$ は \mathbf{x}_i 、 \mathbf{x}_j を除いたスパンからの
rejectionを意味する。

注意:

この右側のカラムにある指標は全て、この研究のために勝手に作ったものです。

An example of backward selection method

LIDG法では、線形独立な記述子からなる線形包(スパン)が得られ、推定精度の高いモデルが得られる。しかし、得られるモデルは一般にスパースではない。解釈可能なスパースモデル(経験則のようなもの)を得るには、改めて記述子選択を行う必要がある。

後退選択法の指針:

1. 記述子の回帰における重要度を意味する G^2 が小さい記述子を積極的に取り除くようにする。
2. 回帰係数(絶対値)がなるべく小さい記述子を取り除くようにする。
3. 除去することで、推定精度、 R^2 、 TR^2 があまり下がらないような記述子を除去する。
4. 除去することで、予測(汎化)精度 Q^2 、 TQ^2 が上昇し、AICが下がるような記述子を除去する。

このように、各統計指標を統合的に判断し、 TQ^2 が最大になるまで削除し続ける。

(結構、泥臭い作業となる。)

あるいは、ある指定した個数になるまで続ける。