Методы выбора регрессоров, регуляризации и снижения размерности

ЦМФ

Цели методов

- повышение точности прогноза за счёт избавления от шума
- определение списка наиболее значимых факторов

Пошаговая процедура

Действия:

- исключение регрессоров
- добавление регрессоров

Критерии:

- p-value
- AIC
- R^2_{adj}

Исходные данные

```
statedata <- data.frame(state.x77, row.names = state.abb,
check.names = T)

head(statedata)
```

	Population	Income	Illiteracy	**Life.Exp**	Murder	HS.Grad	Frost	Area
AL	3615	3624	2.1	**69.05**	15.1	41.3	20	50708
AK	365	6315	1.5	**69.31**	11.3	66.7	152	566432
AZ	2212	4530	1.8	**70.55**	7.8	58.1	15	113417
AR	2110	3378	1.9	**70.66**	10.1	39.9	65	51945
CA	21198	5114	1.1	**71.71**	10.3	62.6	20	156361
CO	2541	4884	0.7	**72.06**	6.8	63.9	166	103766

Пошаговое исключение регрессоров (1/4)

исходная модель с полным списком регрессоров

```
ols <- lm(Life.Exp ~ ., data = statedata)
summary(ols); extractAIC(ols)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.094e+01  1.748e+00  40.586  < 2e-16 ***
Population   5.180e-05  2.919e-05   1.775   0.0832 .
Income      -2.180e-05  2.444e-04  -0.089   0.9293
Illiteracy   3.382e-02  3.663e-01   0.092   0.9269
Murder      -3.011e-01  4.662e-02  -6.459 8.68e-08 ***
HS.Grad      4.893e-02  2.332e-02   2.098   0.0420 *
Frost       -5.735e-03  3.143e-03  -1.825   0.0752 .
Area        -7.383e-08  1.668e-06  -0.044   0.9649
Adjusted R-squared:  0.6922
[1]   8.00000 -22.18462
```

Исключаем из модели регрессор, обладающий самым низким p-value (либо тот, без которого модель будет иметь более высокий R^2_{adj} или более низкий AIC)

Пошаговое исключение регрессоров (2/4)

исключение площади штата

```
ols <- update(ols, . ~ . - Area)
summary(ols); extractAIC(ols)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.099e+01  1.387e+00  51.165  < 2e-16 ***
Population   5.188e-05  2.879e-05   1.802   0.0785 .
Income      -2.444e-05  2.343e-04  -0.104   0.9174
Illiteracy   2.846e-02  3.416e-01   0.083   0.9340
Murder      -3.018e-01  4.334e-02  -6.963 1.45e-08 ***
HS.Grad      4.847e-02  2.067e-02   2.345   0.0237 *
Frost       -5.776e-03  2.970e-03  -1.945   0.0584 .

Adjusted R-squared:  0.6993 # вырос

[1]   7.00000 -24.18229 # снизился
```

Процедура повторяется до тех пор, пока не останутся только значимые регрессоры (либо R^2_{adj} перестанет расти, а AIC снижаться)

Пошаговое исключение регрессоров (3/4)

финальная версия модели

```
summary(ols); extractAIC(ols)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
Population   5.014e-05  2.512e-05   1.996  0.05201 .
Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
HS.Grad      4.658e-02  1.483e-02   3.142  0.00297 **
Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
Adjusted R-squared:  0.7126
[1]   5.00000 -28.16123
```

Пошаговое исключение регрессоров (4/4)

автоматизированная процедура на основе критерия AIC

```
ols <- lm(Life.Exp ~ ., data = statedata)
step.back <- step(ols, direction = "backward", trace = 0)

summary(step.back); extractAIC(step.back)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
Population   5.014e-05  2.512e-05   1.996  0.05201 .
Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
HS.Grad      4.658e-02  1.483e-02   3.142  0.00297 **
Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
Adjusted R-squared:  0.7126
[1]   5.00000 -28.16123
```

Пошаговое добавление регрессоров

начинаем с самой простой модели (только константа)

```
ols <- lm(Life.Exp ~ 1, data = statedata)
step.fwd <- step(ols, scope = list(lower = . ~ ., upper = . ~
Population + Income + Illiteracy + Murder + HS.Grad + Frost + Area),
direction = "forward", trace = 0)

summary(step.fwd); extractAIC(step.fwd)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
Population   5.014e-05  2.512e-05   1.996  0.05201 .
Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
HS.Grad      4.658e-02  1.483e-02   3.142  0.00297 **
Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
Adjusted R-squared:  0.7126
[1]   5.00000 -28.16123
```

Если в параметре «direction» указать "both", то на каждом шаге может происходить как добавление, так и исключение регрессора (в пределах, указанных в параметре «scope»)

Недостатки пошаговой процедуры

- из-за последовательного характера добавления и исключения рассматриваются не все возможные комбинации регрессоров
- значимость (равно как R^2_{adj} и AIC) не является надёжной характеристикой полезности регрессора; в частности, значимость меняется при добавлении и удалении переменных
- нет чёткого правила остановки, особенно в случае рассогласования критериев

Регуляризация

- Ridge regression
- Lasso regression

Ridge regression

Две эквивалентные формулировки:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{d} x_{i,j}\beta_j\right)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 \to \min_{\beta} \qquad (1)$$

$$\begin{cases} \sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{d} x_{i,j}\beta_j\right)^2 \to \min_{\beta} \\ \sum_{j=1}^{d} \beta_j^2 \leq t \end{cases} \qquad (2)$$

$$\hat{\beta}_{ridge} = (X'X + \lambda^* I)^{-1} X'\vec{y}, \qquad \text{где } \lambda^* \text{ — вектор } (0, \lambda)$$

Достоинства:

- нечувствительность к мультиколлинеарности
- матрица $X'X + \lambda I$ всегда имеет полный ранг

Lasso regression

Две эквивалентные формулировки:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{d} x_{i,j}\beta_j\right)^2 + \lambda \sum_{j=1}^{d}\left|\beta_j\right| \to \min_{\beta} \qquad (1)$$

$$\begin{cases} \sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{d} x_{i,j}\beta_j\right)^2 \to \min_{\beta} \\ \sum_{j=1}^{d}\left|\beta_j\right| \le t \end{cases} \qquad (2)$$

Нет единой формулы для расчёта $\hat{\beta}_{lasso}$

Достоинства:

- оценки коэффициентов при больших λ обнуляются, что позволяет осуществлять выбор набора регрессоров

Оценки $\hat{\beta}_{lasso}$

```
library(penalized)

lasso <- penalized(y, X, unpenalized = ~1, lambda1 = 10,
                   lambda2 = 0, model = "linear", trace = FALSE)
beta.lasso <- c(lasso@unpenalized, lasso@penalized)
```

Снижение размерности

- Principal components regression
- Partial least squares

Идея состоит в замене исходной матрицы регрессоров X на матрицу Z, столбцы которой являются линейными комбинациями столбцов X

Principal components regression

Z_{pc} — матрица главных компонент X:
$Z_{pc} = XV$,
V — матрица собственных векторов $X'X$ (по столбцам)

Новая матрица регрессоров Z составляется из p столбцов Z_{pc}, $p \in \{1; \dots; d\}$

$\hat{\beta}_{pcr} = (Z'Z)^{-1}Z'y$

PCR в R

```
X <- as.matrix(statedata[,-4]); y <- cbind(statedata[,4])
e <- eigen(t(X)%*%X)
Z.pc <- X %*% e$vectors
```

проверим, насколько хорошо каждая из главных компонент
объясняет дисперсию **y**

```
d <- ncol(X)
tss <- sum((y - mean(y))^2)
beta <- vector("list", d); r.sq <- numeric(d)
for (i in 1:d) {
  Z <- cbind(1, Z.pc[,i])
  beta[[i]] <- solve(t(Z)%*%Z)%*%t(Z)%*%y
  y.hat <- Z %*% cbind(beta[[i]])
  rss <- sum((y - y.hat)^2)
  r.sq[i] <- 1 - rss/tss
}
```

PCR в R

```
r.sq
[1] 0.0115228286 0.0006716743 0.0450917017 0.0171580799 0.1030782448
[6] 0.5798264601 0.0005978996
```

Порядок главных компонент Z_{pc} отражает степень их влияния на дисперсию признаков, но их влияние на дисперсию объясняемой переменной может быть иным

Переставим столбцы Z_{pc} в соответствии со значениями R^2

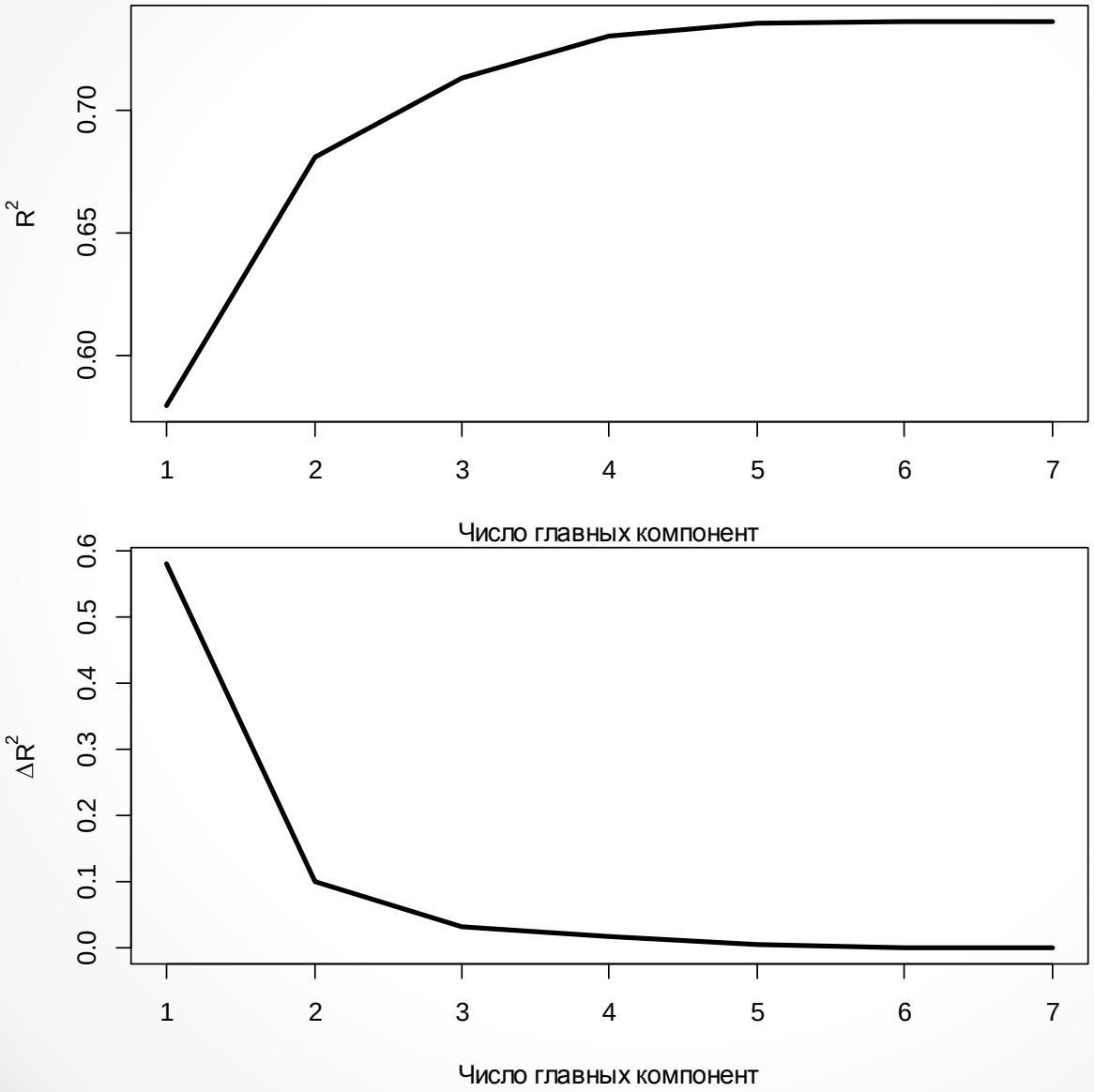
```
z <- order(r.sq, decreasing = TRUE)
Z.pc <- Z.pc[,z]
```

PCR в R

Выбирая p первых столбцов переставленной матрицы Z_{pc}, можно объяснить значительную часть дисперсии **y**

```
beta <- vector("list", d)
r.sq <- numeric(d); delta.r <- numeric(d)
for (p in 1:d) {
  Z <- cbind(1, Z.pc[,1:p])
  beta[[p]] <- solve(t(Z)%*%Z)%*%t(Z)%*%y
  y.hat <- Z %*% cbind(beta[[p]])
  rss <- sum((y - y.hat)^2)
  r.sq[p] <- 1 - rss/tss
  delta.r[p] <- ifelse(p == 1, r.sq[p], r.sq[p] - r.sq[p-1])
}
```

PCR в R


R²
0.60
0.65
0.70
1
2
3
4
5
6
7
Число главных компонент
ΔR²
0.0
0.1
0.2
0.3
0.4
0.5
0.6
1
2
3
4
5
6
7
Число главных компонент

Построение прогноза

Между столбцами матриц X и Z_{pc} существует функциональная линейная зависимость:

$Z_{pc} = XB$, где

матрица коэффициентов B — это переставленная матрица V

```
p <- 4
B <- e$vectors[,z][,1:p]
Z.test <- cbind(1, (X.test %*% B))
y.hat <- Z.test %*% cbind(beta[[p]])
```

$$\hat{y} = Z_{pc}\hat{\beta} = XB\hat{\beta} = X\hat{\beta}_{pcr}, \quad \hat{\beta}_{pcr} = B\hat{\beta}$$

```
beta.pcr <- c(beta[[p]][1], B %*% cbind(beta[[p]][-1]))
y.hat <- cbind(1, X.test) %*% cbind(beta.pcr)
```

Partial Least Squares

Новая матрица регрессоров Z составляется в ходе итеративной процедуры:

$X_0 := X, Y_0 := Y$

$i \in \{1; \dots; rank(X)\}$ {

$X_0' Y_0 Y_0' X_0 = UDV'$ — single value decomposition (SVD)

$z_i := X_0 U[,1]$ — i-я латентная переменная

$z^* := \frac{z_i}{z_i' z_i}$ — нормировка

$p := X_0' z^*, \hat{X} := z^* p'$ — дефлятор матрицы X_0

$X_0 := X_0 - \hat{X}, Y_0 := \vec{0}$

}

$Z := (z_1, \dots, z_{rank(X)})$

$Y = ZB + \varepsilon$

Partial Least Squares в R

```
library(pls)
pls <- plsr(Life.Exp ~ ., data = statedata, ncomp = 7)

summary(pls)
Data:   X dimension: 50 7
        Y dimension: 50 1
Fit method: kernelpls
Number of components considered: 7
TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X          99.722   99.993   100.00   100.00   100.00   100.00   100.00
Life.Exp 1.156     2.346     20.24     23.19     56.74     73.61     73.62

b <- coef(pls)
b0 <- pls$fitted[1] - sum(pls$model[1,-1] * b)
beta.pls <- c(b0, b)
```