“데이터 사이언티스트를 위한 DMR-4”
“shrinkage : lasso, elastic net”

Jinseog Kim
Dongguk University
jinseog.kim@gmail.com

2016-11-08

Supervised learning

1. generalized linear models
    1. linear regression
    2. logistic regression
    3. poisson, gamma, ...
2. non-parametric models
    1. decision tree, knn,
3. neural network(1 hidden layer)
4. **penalized regression models**
    1. ridge - l_2 penalty
    2. lasso - l_1 penalty
    3. elastic net - $l_1 + l_2$ penalty
5. ensembles
    1. bagging
    2. various boosting
    3. random forest
6. deep learning : h2o

Predictive modeling workflow and R methods(functions)

1. fit model `fit <- knn(trainingData, outcome, k = 5)`
2. check or validation models `print`, `plot`, `summary`
3. predict using selected model & test data `predict(fit, newdata)`
4. performance evaluation `ROCR`, 'caret' packages

regularization methods & penalty functions

- regularization methods

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\sum_{j=1}^{p}J_\lambda(\beta_j) \tag{1}$$

1. L1-norm: $J_\lambda(z)=\lambda I(z\neq 0)$
2. L2-norm: $J_\lambda(z)=\lambda|z|^2$
3. LASSO 벌점함수(Tibshirani, 1996): $J_\lambda(z)=\lambda|z|$
4. elastic net(Zou, 2005): $\lambda\sum_{j=1}^{p}\left((1-\alpha)|\beta_j|+\alpha\beta_j^2\right),\alpha\in[0,1]$.
5. group lasso(Yuan et al, Kim et al.)
6. fused lasso, SCAD, MCP,...

R 패키지 및 함수

R패키지 및 함수명	옵션	함수/옵션 설명
lars::lars		lasso
	type	“lars”|“lasso”|“forward.swagewise”|“stepwise”
	s	조율모수의 값
lasso2::l1ce		lasso for lm
lasso2::gl1ce		lasso for glm
elasticnet::enet		elastic net, ridge, lasso
glmnet::glmnet		elastic net
plus::plus		scad, mcp, lasso
ElemStatLearn::spam		spam mail data

Lasso 예제: fit model using lars package

```
library(lars)
library(ElemStatLearn)
# sampling index of train data
tr.idx <- sample(nrow(spam), 0.7*nrow(spam))
# fit svm model
x <- as.matrix(spam[,-58]) # should be matrix
y <- as.integer(spam$spam =="spam") # integer valued
m.lars <- lars(x=x[tr.idx, ], y=y[tr.idx], type="lasso")
tail(summary(m.lars))
```

```
## LARS/LASSO
## Call: lars(x = x[tr.idx, ], y = y[tr.idx], type = "lasso")
##    Df    Rss     Cp
## 52 53 336.93 62.498
## 53 54 336.80 63.297
## 54 55 336.06 58.349
## 55 56 335.39 54.046
## 56 57 335.39 56.000
## 57 58 335.39 58.000
```

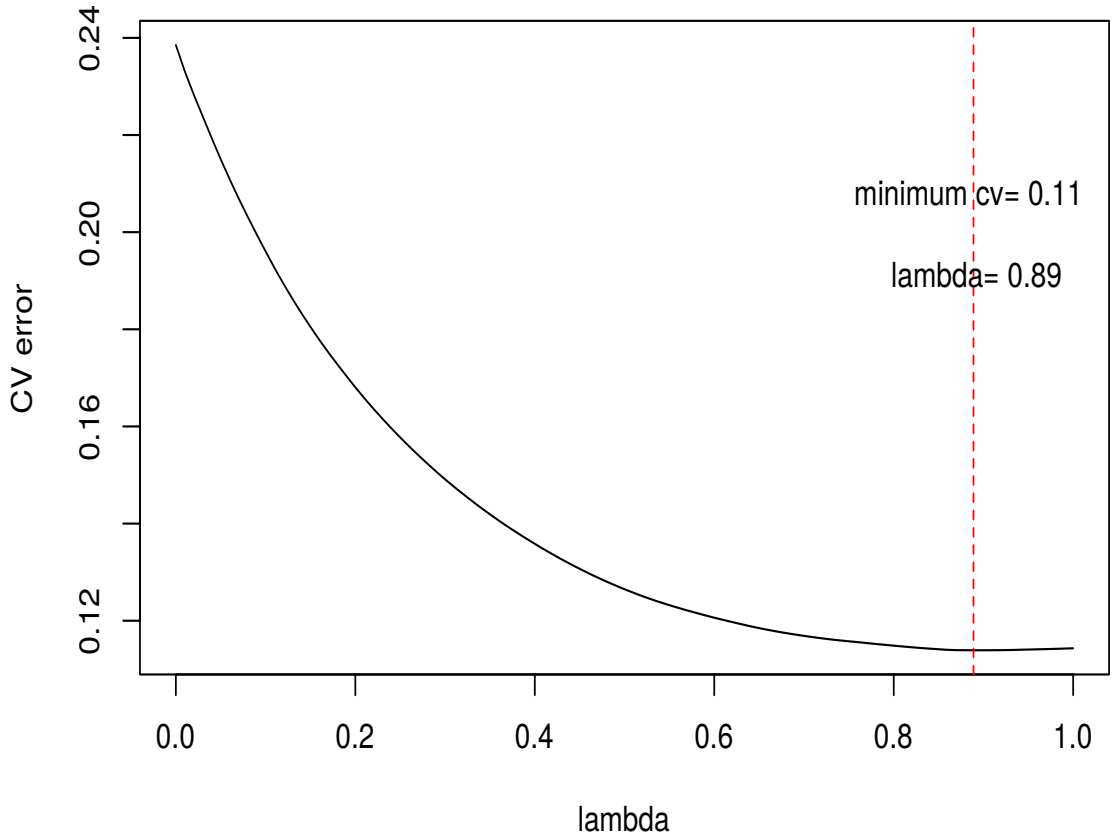
“‘ ## Lasso 예제: plot regularization paths

```
plot(m.lars)
```

Lasso 예제: choose of optimal tunning parameters

```
m2.lars <- cv.lars(x[tr.idx, ], y[tr.idx], K=5, type="lasso", plot.it = FALSE)
midx <- which.min(m2.lars$cv)
plot(m2.lars$index, m2.lars$cv, type="l", xlab="lambda", ylab="CV error")
abline(v=m2.lars$index[midx], lty=2, col=2)
text(m2.lars$index[midx], 0.2,
     paste("minimum cv=", round(m2.lars$cv[midx], 2),
           "\n\n lambda=", round(m2.lars$index[midx], 2))
     )
```

Lasso 예제: choose of optimal tunning parameters



lasso: predict for new test data

```
pred <- predict.lars(m.lars, newx=x[-tr.idx,],
                     s=m2.lars$index[midx], # optimal tunning param.
                     type = "fit", mode="fraction")
names(pred)
```

```
## [1] "s"        "fraction" "mode"     "fit"
```

```
# Confusion Matrix
table(pred[[4]]>0.5, y[-tr.idx])
```

```
##
##            0   1
##   FALSE 781 121
##   TRUE   50 429
```

glmnet

- glm model with panalty function

$$\lambda \sum_{i=1}^{p} \left((1-\alpha)\beta_j^2 + \alpha|\beta_j| \right)$$

  - lasso if $\alpha = 1$
  - ridge if $\alpha = 0$
  - elastic net if if $0 < \alpha < 1$: deal with grouped variables
- fast algorithm via coordinate descent (Friedman, 2010)

R functions

glmnet 함수명	옵션	옵션 설명
glmnet		
	x,y	input, output
	family	binomial|gaussian|poisson|cox
	family	binomial|gaussian|poisson|cox
	lambda	penalty1
	alpha	penalty2: [0,1]
cv.glmnet		k-fold CV for lambda
plot.glmnet		
plot.cv.glmnet		
predict		prediction

model fit & solution paths

```
library(glmnet)
# Fit models using spam data
fit.lasso <- glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=1)
fit.ridge <- glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=0)
fit.elnet <- glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=.8)
```

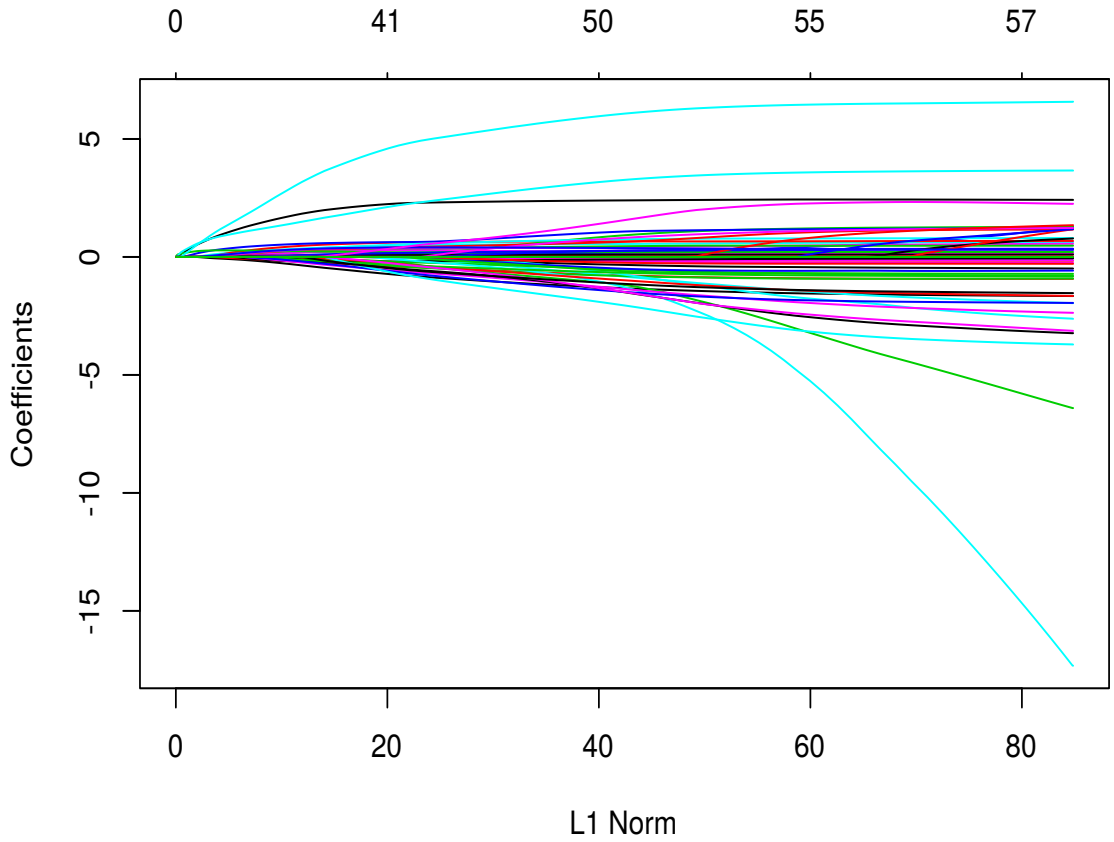
Plot solution paths:

```
plot(fit.lasso)
```

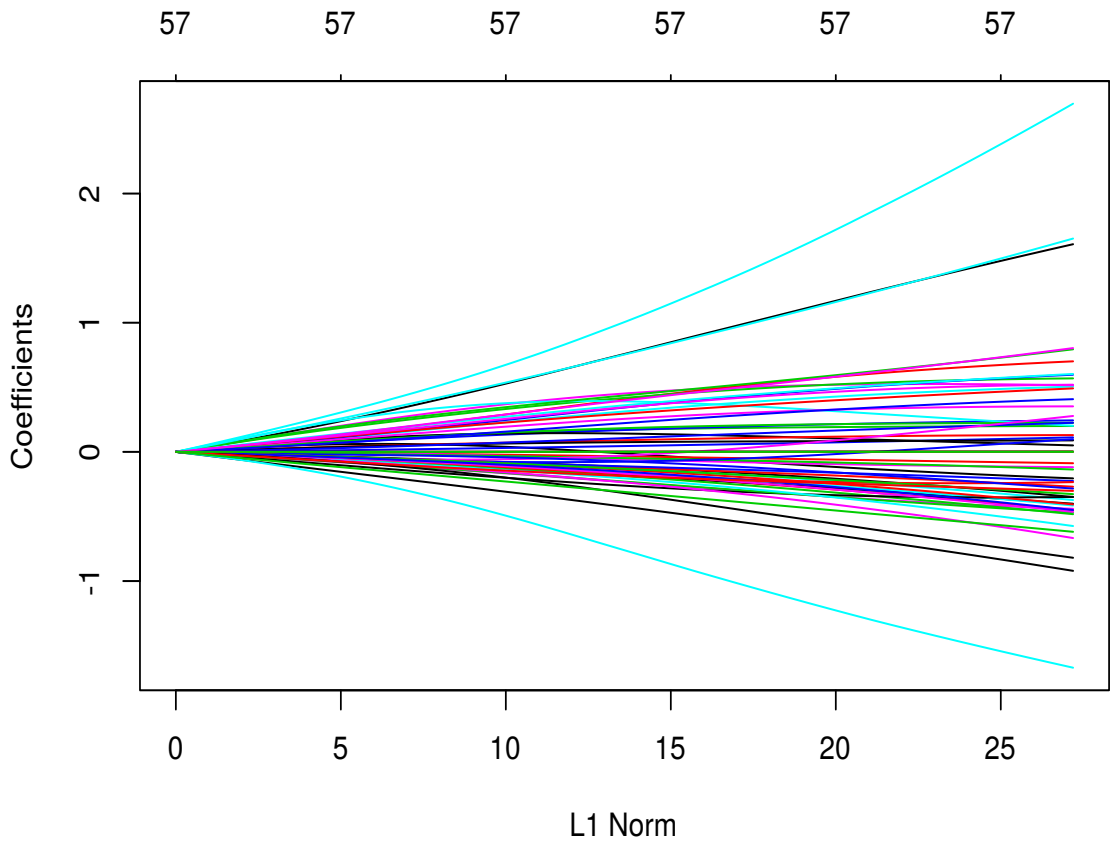
```
plot(fit.ridge)
```

```
plot(fit.elnet)
```

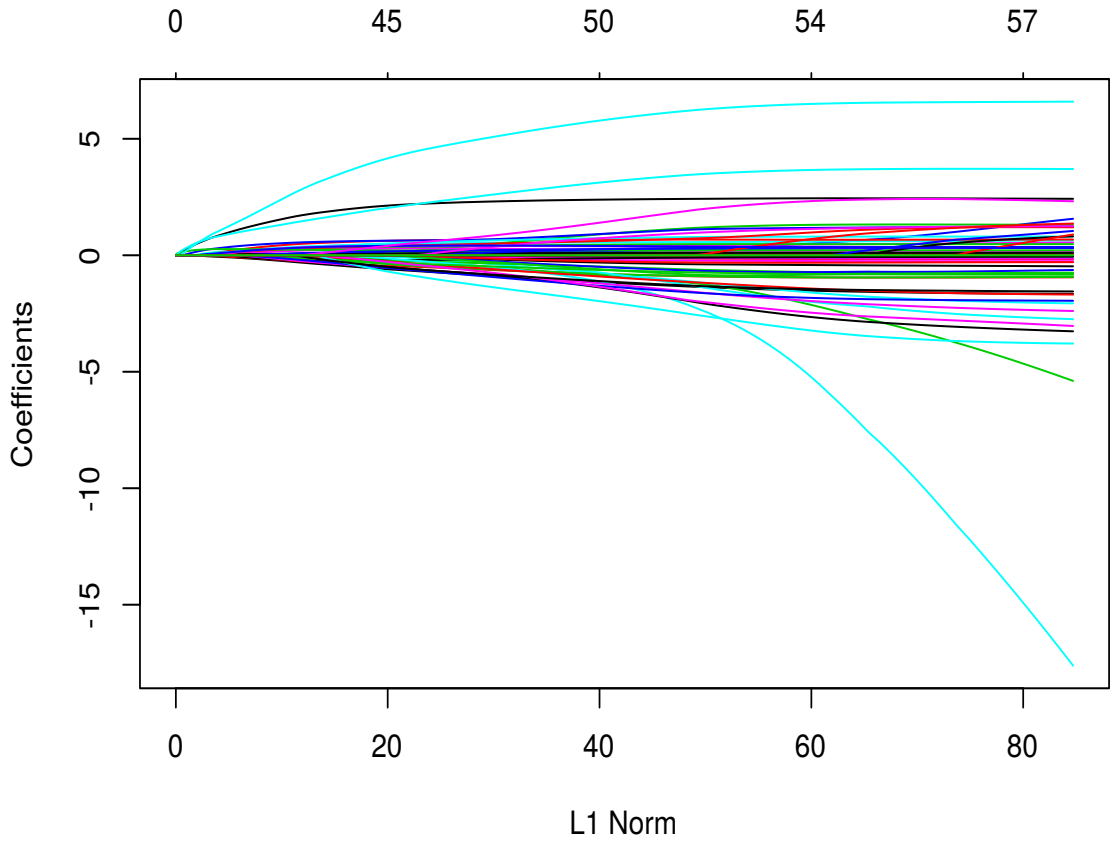
Plot solution paths: lasso



Plot solution paths: ridge



Plot solution paths: elastic net



CV.glmnet: find optimal λ

```
cfit.lasso <- cv.glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=1)
cfit.ridge <- cv.glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=0)
cfit.elnet <- cv.glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=.8)
names(cfit.elnet)
```

```
##  [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"
##  [6] "nzero"      "name"       "glmnet.fit" "lambda.min" "lambda.1se"
```

```
cfit.lasso$lambda.min
```

```
## [1] 0.0004037223
```

```
cfit.ridge$lambda.min
```

```
## [1] 0.02056617
```

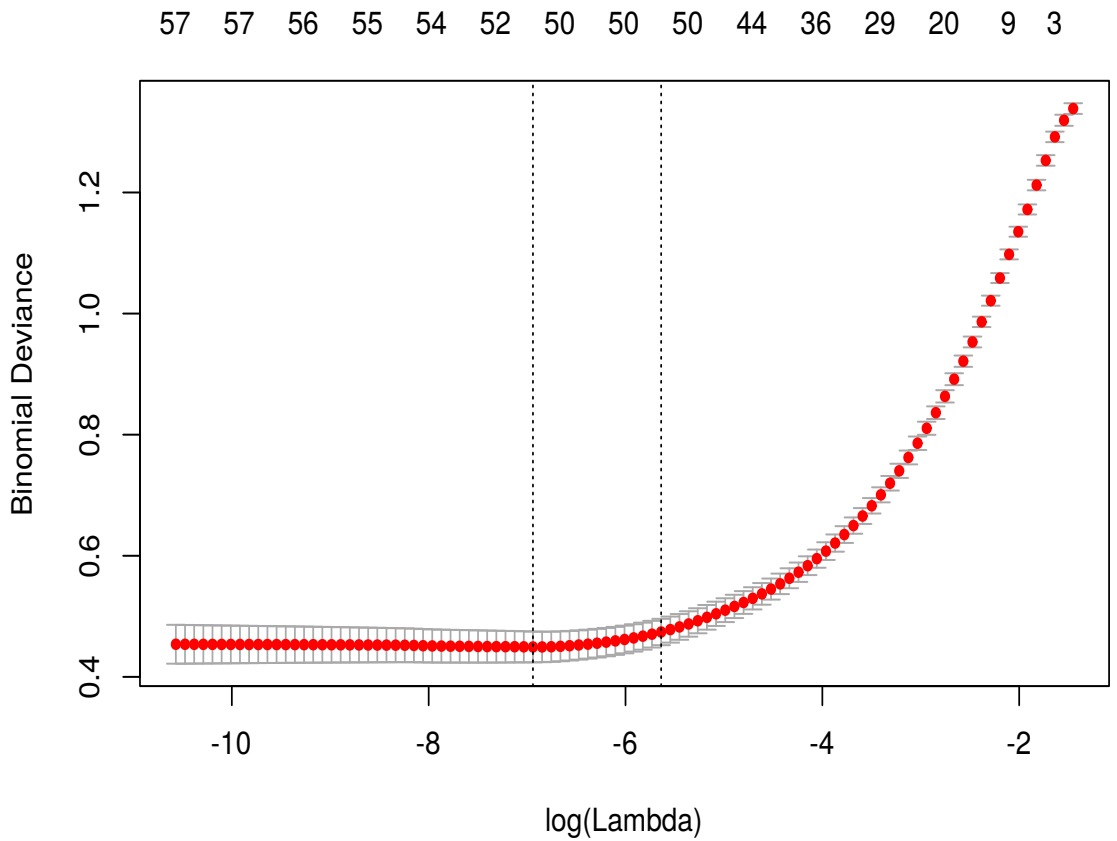
```
cfit.elnet$lambda.min
```

```
## [1] 0.0009678788
```

plot

```
plot(cfit.elnet)
```

```
#plot(cfit.lasso)
```



refit with optimal lambda

```
fit2.lasso <- glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=1, lambda=
fit2.ridge <- glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=0, lambda=
fit2.elnet <- glmnet(x=x[tr.idx, ], y=y[tr.idx], family="binomial", alpha=.8, lambda
names(fit2.elnet)
```

```
##  [1] "a0"         "beta"       "df"         "dim"        "lambda"
##  [6] "dev.ratio"  "nulldev"    "npasses"    "jerr"       "offset"
## [11] "classnames" "call"       "nobs"
```

```
head(coef(fit2.elnet))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) -1.5911856
## A.1         -0.3659260
## A.2         -0.1341301
## A.3          0.1623993
## A.4          0.2301302
## A.5          0.5623278
```

prediction

```
pred.lasso <- predict(fit2.lasso, newx=x[-tr.idx, ], type="response") #type="class"
pred.ridge <- predict(fit2.ridge, newx=x[-tr.idx, ], type="response") #type="class"
pred.elnet <- predict(fit2.elnet, newx=x[-tr.idx, ], type="response") #type="class"
```

evaluation

```
# Confusion Matrix
table(pred.lasso>0.5, y[-tr.idx])
```

```
##
##             0   1
##   FALSE 784  58
##   TRUE   47 492
```

```
table(pred.ridge>0.5, y[-tr.idx])
```

```
##
##             0   1
##   FALSE 779  80
##   TRUE   52 470
```

```
table(pred.elnet>0.5, y[-tr.idx])
```

```
##
##             0   1
##   FALSE 785  60
##   TRUE   46 490
```