Session 7: Multi-omics Data Analysis, Part II.

Hokeun Sun

Professor
Department of Statistics
Pusan National University

Contents

Contents

1. **Statistical Variable Selection and Cross-validation**

2. Analysis of High-dimensional Genomic Data I
3. Analysis of High-dimensional Genomic Data II
4. Analysis of High-dimensional DNA Methylation Data

Statistical Modeling

- Statistical learning is a set of tools for modeling and understanding complex datasets.
- Outcome measurement Y (also called dependent variable, response, phenotype outcome).
- Vector of p predictor measurements X (also called inputs, regressors).
  - Genomic features (SNP, gene exparession levels, ...)
  - Clinical/environmental features (age, gender, ...)
- In the regression problem, Y is quantitative (e.g numerical values of trait, blood pressure, cholesterol levels, ...)
- In the classification problem, Y takes values in a finite, unordered set (e.g., survived/died, cancer class of tissue sample, case/control, ...)

Estimation

- Now we write our model as

$$Y = f(X) + \epsilon$$

  where ϵ captures measurement errors and other discrepancies.
- With a good $f(\cdot)$ we can make predictions of Y at new points $X = x$.
- Depending on the complexity of $f(\cdot)$, we may be able to understand how each component X_j of X affects Y.
- The function f that connects the input variable to the output variable is in general unknown.
- There are relatively many approaches for estimating the function f.

Linear Regression

- If Y is quantitative, the linear regression is defined as

$$\begin{aligned} Y &= f(X_1, \ldots, X_p) + \epsilon \\ &= \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon, \end{aligned}$$

  where $X = (X_1, X_2, \ldots, X_p)$.
- Linear regression assumes that the dependence of Y on X_1, X_2, $\cdots$, X_p is linear.
- True regression functions are never linear!
- Although it may seem overly simplistic, linear regression is extremely useful both conceptually and practically.
- The linear model has distinct advantages in terms of its interpretability and often shows good predictive performance.

Least Square Estimate

- In linear regression, regression coefficients,

$$\boldsymbol{\beta} = \{\beta_0, \ldots, \beta_p\}$$

  are unknown and need to be estimated.
- The least square (LS) estimate for $\boldsymbol{\beta}$ minimizes the residual sum of squares. i.e.,

$$\hat{\boldsymbol{\beta}}^{LS} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\min}\, \mathsf{RSS}(\boldsymbol{\beta})$$

  where

$$\mathsf{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Limitation of Least Square Estimate

- Gauss-Markov Theorem: The LS estimates for $\boldsymbol{\beta}$ have the smallest variance among all linear unbiased estimates.
- Problems in multiple linear regression
  - Some coefficients in $\boldsymbol{\beta}$ are not well estimated and have large variance for correlated predictors, since correlated variables carry the same information regarding the response.
  - LS cannot be computed when $n < p$ (high-dimensional data) due to singularity.
- As an alternative, we can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero.
  - Ridge regression
  - Lasso (Least absolute shrinkage and selection operator).

Lasso Regularization

- Lasso coefficient estimates solve the problems

$$\underset{\beta}{\text{minimize}} \sum_{i}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

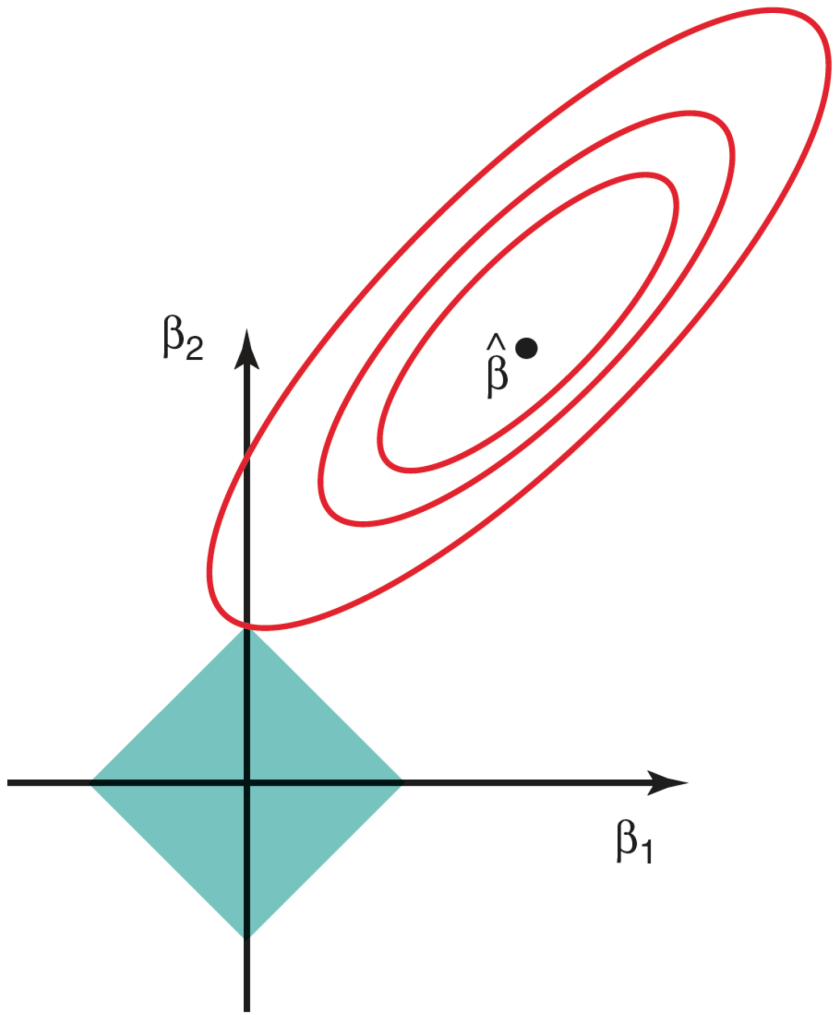
- The solution is equivalent to minimize the quantity

$$\text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|,$$

where $\lambda > 0$ is a tuning parameter to control sparsity.

The Geometric View of Lasso

- When $p = 2$, Lasso solution is



Lasso Regularization

- The l_1 penalty of the lasso has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, the lasso performs variable selection.
- The lasso yields sparse models - that is, models that involve only a subset of the variables.
- The tuning parameter λ controls the number of nonzero regression coefficients (degrees of freedom).
  - If λ increases, the degrees of freedom decrease.
  - If λ decreases, the degrees of freedom increase.
- Selecting an appropriate value of λ for the lasso is crucial.
- In general, the optimal λ value is chosen among a grid of λ values

Lasso Regularization

- For a grid of λ values such as

$$\lambda_{\text{max}} = \lambda_1 > \lambda_2 > \ldots > \lambda_{m-1} > \lambda_m = \lambda_{\text{min}},$$

  the number of nonzero regression coefficients (degrees of freedom: df) is

$$df(\hat{\boldsymbol{\beta}}_{\lambda_1}) = 0 \leq df(\hat{\boldsymbol{\beta}}_{\lambda_2}) \leq \ldots \leq df(\hat{\boldsymbol{\beta}}_{\lambda_{m-1}}) \leq df(\hat{\boldsymbol{\beta}}_{\lambda_m})$$

- The optimal value of λ should be chosen by a resampling method such as cross-validation.
- Numerical values of nonzero regression coefficients also rely on λ.

Training Set

- In general, two types of data are considered such as the training set and the test set.
- The main goal is to minimize the test error.
- Training set is used to train the model, i.e., to estimate the regression coefficients. Suppose that we have m different models

$$\hat{f}_{\lambda_1}(x), \hat{f}_{\lambda_2}(x), \hat{f}_{\lambda_3}(x), \ldots, \hat{f}_{\lambda_m}(x)$$

- The l-th model has $\hat{\boldsymbol{\beta}}_l = \{\hat{\beta}_{0,l}, \hat{\beta}_{1,l}, \ldots, \hat{\beta}_{p,l}\}$ such that

$$\hat{f}_{\lambda_l}(x) = \hat{\beta}_{0,l} + \hat{\beta}_{1,l}x_1 + \cdots + \hat{\beta}_{p,l}x_p$$

- $\hat{\boldsymbol{\beta}}_l$ should be estimated based only on the training set.

Test Set

- Test set is used to assess the model performance. The test set data should not used in the model building process.
- Given the test data $T = \{x_i, y_i\}$ for $i \in \{1, \ldots, N\}$, the mean squared error (MSE) of λ_l is

$$MSE_{\lambda_l} = \frac{1}{N} \sum_{i \in T} \left(y_i - \hat{f}_{\lambda_l}(x_i) \right)^2$$

- We can find the best model among m models as comparing the test error of m models.

$$\hat{\lambda} = \underset{\lambda_1, \ldots, \lambda_m}{\arg\min} \, MSE_{\lambda_l}$$

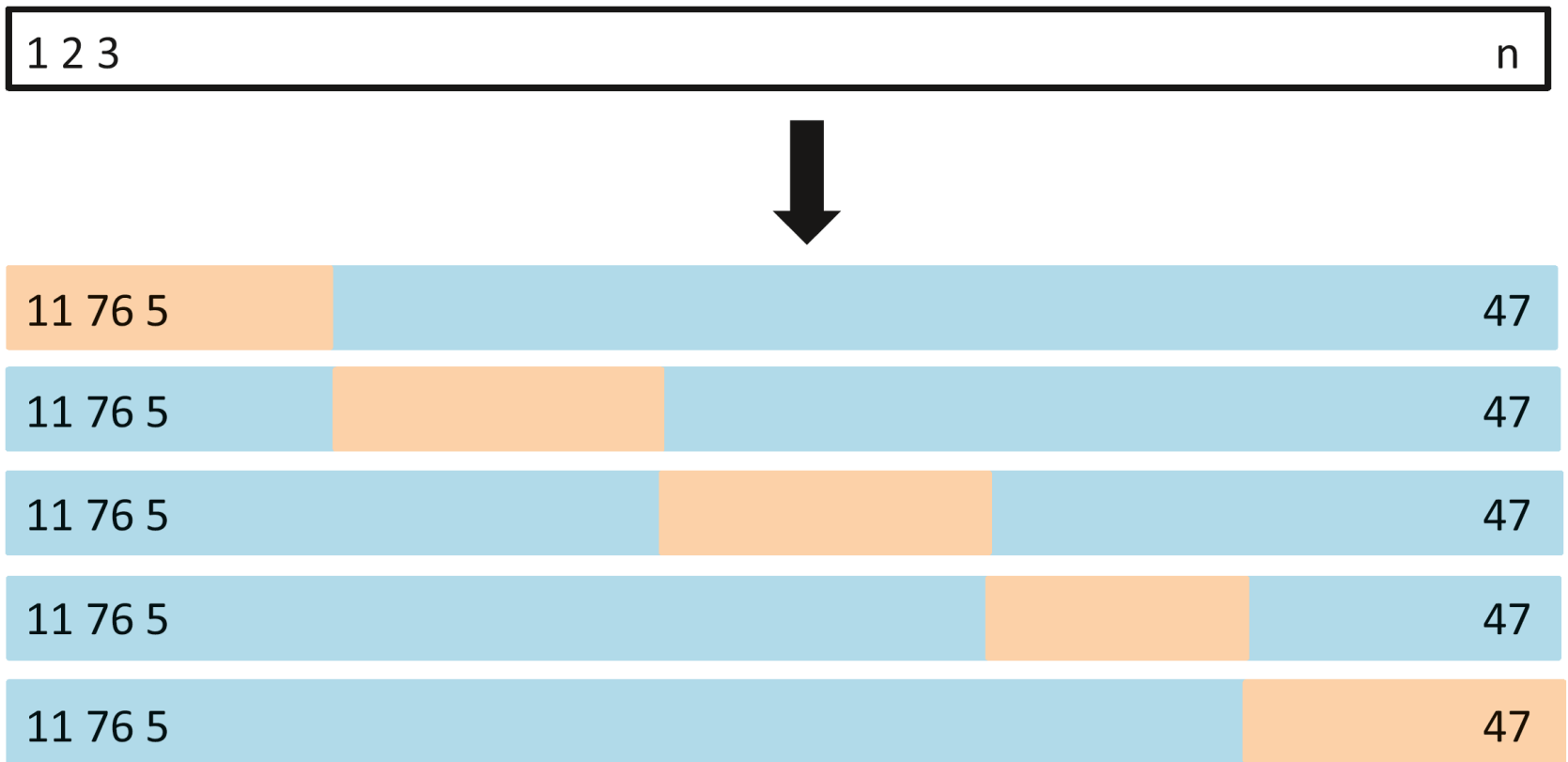
- Test set is often called validation set in tuning parameter selection.

K-fold Cross-validation

- In order to compute the test error, we need to have a large designated test set. However, it is not often available.
- The K-fold cross-validation (CV) is the most popular approach to estimate the test error.
- For each model,
  1. Randomly divide the data into K equal-sized parts.
  2. Leave out part k, and estimate the regression coefficients based on the other $K-1$ parts (training set).
  3. Compute MSE for the left-out kth part (test set).
  4. This is done in turn for each part $k = 1, 2, \cdots, K$.
  5. Final MSE value is then averaged over K parts.

K-fold Cross-validation

- Divide data into K roughly equal-sized parts ($K = 5$ here)



K-fold CV for Lasso

- Suppose that y_i is a quantitative value of the i-th individual.
- For lasso, the K-fold CV procedure:
  1. Randomly separate n samples into K folds: $C_1, C_2, \ldots, C_K$
  2. For each λ, compute regression coefficients based on C_{-k},

$$\hat{\boldsymbol{\beta}}_{\lambda_1}^{[-k]}, \hat{\boldsymbol{\beta}}_{\lambda_2}^{[-k]}, \ldots, \hat{\boldsymbol{\beta}}_{\lambda_{m-1}}^{[-k]}, \hat{\boldsymbol{\beta}}_{\lambda_m}^{[-k]}$$

     Note that C_{-k} is an observation set with part k removed.
  3. Compute cross-validation error (CVE) for each λ.

$$\text{CVE}(\lambda_l) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in C_k} \left(y_i - x_i^{\mathrm{T}} \hat{\boldsymbol{\beta}}_{\lambda_l}^{[-k]} \right)^2$$

     for $l = 1, 2, \ldots, m$.
  4. Pick up the optimal $\hat{\lambda}_l$ that minimizes $\text{CVE}(\hat{\lambda}_l)$.

CV Error for Binary Outcomes

- Suppose that $y_i = 1$ for a case and $y_i = 0$ for a control.
- CVE based on deviance

$$\text{CVE} = \frac{1}{n} \sum_{k=1}^{K} -2 \sum_{i \in C_k} \left(y_i \log \hat{p}_i^{[-k]} + (1 - y_i) \log(1 - \hat{p}_i^{[-k]}) \right),$$

where $\hat{p}_i^{[-k]}$ is the probability of $y_i = 1$ for the observation i, obtained from the data with part k removed.

- CVE based on classification error

$$\text{CVE} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in C_k} I \left(y_i - \hat{y}_i^{[-k]} \right),$$

where $\hat{y}_i^{[-k]}$ is the fit for the observation i, obtained from the data with part k removed.

Example: Breast Cancer Dataset

- An R package "`breastCancerMAINZ`" from the bioconductor contains the dataset "`mainz`", which consists of 22,283 features with 200 samples.
  - `exprs(mainz)` : Matrix containing gene expressions as measured by Affymetrix hgu133a technology (single-channel, oligonucleotides).
  - `pData(mainz)` : AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- We regarded a tumor size as the response variable.
- The main goal of the analysis is to identify genes associated with a tumor size.
- Reference
  - Schmidt et al. (2008) The humoral immune system has a key prognostic impact in node-negative breast. *Cancer Research* 68(13):5405-13.

Example: Breast Cancer Data

```
library(kogo)
```

```
data(BC)
names(BC)
str(BC)
dim(BC$x)
BC$y
head(BC$geneID, 10)
BC$x[1:3, 1:10]
```

```
x <- BC$x
y <- BC$y
g1 <- glmnet(x, y)
```

```
plot(g1, "lambda")
names(g1)
```

Example: Breast Cancer Data

```
g1$df
g1$lambda
data.frame(lambda=g1$lambda, df=g1$df)
dim(g1$beta)
```

```
BC$path.kegg[1:10]
BC$geneID[1:100, 2]
path <- BC$path.kegg[[1]]
intersect(path, BC$geneID[,2])
genes <- intersect(path, BC$geneID[,2])
```

```
nx <- ngene <- NULL
for (i in 1:length(genes)) {
    ww <- which(BC$geneID[,2]==genes[i])
    ngene <- c(ngene, rep(genes[i], length(ww)))
    nx <- cbind(nx, x[,ww])
}
colnames(nx) <- ngene
```

Example: Breast Cancer Data

```
BC2 <- data.frame(nx, y=y)
str(BC2)
g0 <- lm(y~., BC2)
g0$coef
```

```
g2 <- glmnet(nx, y)
plot(g2, "lambda")
```

```
data.frame(s=colnames(g2$beta), lambda=g2$lambda, df=g2$df)
dim(g2$beta)
g2$beta[, c("s0", "s5", "s15", "s25", "s35")]
```

```
s40 <- as.numeric(coef(g2, s=g2$lambda[41]))
s20 <- as.numeric(coef(g2, s=g2$lambda[21]))
s10 <- as.numeric(coef(g2, s=g2$lambda[11]))
s2 <- as.numeric(coef(g2, s=g2$lambda[3]))
data.frame(LS=g0$coef, lasso.s40=s40, lasso.s20=s20,
           lasso.s10=s10, lasso.s2=s2)
```

Example: Breast Cancer Data

```
set.seed(12345)
gcv <- cv.glmnet(nx, y, nfolds=5)
plot(gcv)
```

```
names(gcv)
data.frame(lambda=gcv$lambda, df=gcv$nzero, CVE=gcv$cvm)
gcv$lambda[10]
gcv$lambda[gcv$cvm==min(gcv$cvm)]
gcv$lambda.min
```

```
g2 <- glmnet(nx, y)
plot(g2, "lambda")
abline(v=log(gcv$lambda.min), col="red", lty=2)
coef(gcv, s=gcv$lambda.min)
```

```
plot(g2, "lambda")
```

Example: Breast Cancer Data

```
set.seed(111)
V <- lam <- NULL
for (i in 1:10) {
    gcv <- cv.glmnet(nx, y, nfolds=5)
    abline(v=log(gcv$lambda.min), col="red", lty=2)
    lam[i] <- gcv$lambda.min
    cv <- as.numeric(coef(gcv, s=gcv$lambda.min))
    V <- cbind(V, cv)
}
```

```
data.frame(LS=g0$coef, V)
df <- apply(V, 2, function(t) sum(t!=0)-1)
data.frame(DF=df, lambda.opt=lam)
mean(lam)
```

```
g4 <- glmnet(nx, y, lambda=mean(lam))
coef(g4)
```

1. Statistical Variable Selection and Cross-validation

2. Analysis of High-dimensional Genomic Data I

3. Analysis of High-dimensional Genomic Data II
4. Analysis of High-dimensional DNA Methylation Data

Genetic Association Studies

- In genetic association studies, the goal is to identify SNP, genetic variants, genes or gene sets that are associated with diseases/traits.
- It is a statistically challenging problem due to
  - Problem of dimensionality, i.e., $p \gg n$
  - Highly correlated genomic data (LD)
  - Considering covariate effects (non-genetic effects)
  - Sparsity of true signals
- Univariate analysis where each variable (a group or variables) is tested one at a time cannot be accounted for correlation among genetic variables.
- Multiple linear regression cannot be applied because high variance of $\hat{\boldsymbol{\beta}}$ and $p \gg n$.
- Alternatives: regularization methods (penalized regression)!

Penalized Regressions in Recent Bioinformatics

BIOINFORMATICS ORIGINAL PAPER
Vol. 27 no. 24 2011, pages 3399–3406
doi:10.1093/bioinformatics/btr591

Gene expression

Optimized application of penalized regression methods to diverse genomic data

Levi Waldron[1,3], Melania Pintilie[2], Ming-Sound Tsao[3], Frances A. Shepherd[3], Curtis Huttenhower[1,*] and Igor Jurisica[3,*]

BIOINFORMATICS ORIGINAL PAPER
Vol. 25 no. 6 2009, pages 714–721
doi:10.1093/bioinformatics/btp041

Genome analysis

Genome-wide association analysis by lasso penalized logistic regression

Tong Tong Wu[1], Yi Fang Chen[2], Trevor Hastie[2,3], Eric Sobel[4] and Kenneth Lange[4,5,*]

BIOINFORMATICS ORIGINAL PAPER
Vol. 27 no. 19 2011, pages 2679–2685
doi:10.1093/bioinformatics/btr450

Systems biology

Advance Access publication July 30, 2011

Penalized regression elucidates aberration hotspots mediating subtype-specific transcriptional responses in breast cancer

Yinyin Yuan[1,2,*], Oscar M. Rueda[1,2], Christina Curtis[1,2,†] and Florian Markowetz[1,2,*]

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Sparse logistic regression with a $L_{1/2}$ penalty for gene selection in cancer classification

Yong Liang[1*], Cheng Liu[1], Xin-Ze Luan[1], Kwong-Sak Leung[2], Tak-Ming Chan[2], Zong-Ben Xu[3] and Hai Zhang[3]

Penalized Regressions in Recent Bioinformatics

Bioinformatics, 33(12), 2017, 1765–1772
doi: 10.1093/bioinformatics/btx064
Advance Access Publication Date: 6 February 2017
Original Paper

OXFORD

Genome analysis

pETM: a penalized Exponential Tilt Model for analysis of correlated high-dimensional DNA methylation data

Hokeun Sun[1], Ya Wang[2], Yong Chen[3], Yun Li[4,5,6] and Shuang Wang[2,*]

[1]Department of Statistics, Pusan National University, Busan 609-735, Korea, [2]Department of Biostatistics, Mailman School of Public Health, Columbia University, New York, NY 10032, USA, [3]Division of Biostatistics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA, [4]Department of Biostatistics, [5]Department of Genetics and [6]Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599, USA

*To whom correspondence should be addressed.
Associate Editor: John Hancock

Journal of Bioinformatics and Computational Biology
Vol. 16, No. 4 (2018) 1850010 (18 pages)

DOI: 10.1142/S0219720018500105

World Scientific
www.worldscientific.com

New variable selection strategy for analysis of high-dimensional DNA methylation data

Kim and Sun *BMC Bioinformatics* (2019) 20:510
https://doi.org/10.1186/s12859-019-3040-x

BMC Bioinformatics

Jiyun Choi, Kipoong Kim and Hokeun Sun*

Department of Statistics
Pusan National University
Busan 46241, Korea
**hsun@pusan.ac.kr*

METHODOLOGY ARTICLE Open Access

Incorporating genetic networks into case-control association studies with high-dimensional DNA methylation data

Kipoong Kim and Hokeun Sun*

Check for updates

Regularization Methods

- Regularization methods are based on a penalized likelihood

$$Q_\lambda(\boldsymbol{\beta}) = -l(\boldsymbol{\beta}) + p_\lambda(\boldsymbol{\beta})$$

- $l(\boldsymbol{\beta})$ is a log-likelihood function and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_p\}$ is the p-dimensional vector of regression coefficients.
- $l(\boldsymbol{\beta})$ is determined by different types of phenotype outcomes.
- $p_\lambda(\boldsymbol{\beta})$ is a penalty function, where a tuning parameter λ controls sparsity.
- Solution to regression coefficient $\hat{\boldsymbol{\beta}}$ can be obtained by

$$\hat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} Q_\lambda(\boldsymbol{\beta}),$$

for a fixed value of λ.

Penalized Likelihood for Quantitative Outcome

- For the i-th individual,
  - y_i : a quantitative outcome
  - $x_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}}$: the p-dimensional genomic data.
  - $z_i = (z_{i1}, \ldots, z_{iq})^{\mathrm{T}}$: the q-dimensional covariates.
- Linear regression model can be defined as

$$y_i = \beta_0 + z_i^{\mathrm{T}}\boldsymbol{\gamma} + x_i^{\mathrm{T}}\boldsymbol{\beta} + \epsilon_i,$$

  where $\boldsymbol{\gamma} = \{\gamma_1, \ldots, \gamma_q\}^{\mathrm{T}}$ and $\epsilon \sim N(0, \sigma^2)$.
- Least squared loss is used for the negative log-likelihood

$$\begin{aligned} Q_\lambda(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta}) = & -l(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta}) + p_\lambda(\boldsymbol{\beta}) \\ & = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \beta_0 - z_i^{\mathrm{T}}\boldsymbol{\gamma} - x_i^{\mathrm{T}}\boldsymbol{\beta}\right)^2 + p_\lambda(\boldsymbol{\beta}) \end{aligned}$$

Penalized Likelihood for Binary Outcome

- For the i-th individual,
  - y_i : a binary outcome ($y_i = 1$ for cases and $y_i = 0$ for controls)
- In case-control studies, the penalized likelihood is

$$Q_\lambda(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta}) = -l(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta}) + p_\lambda(\boldsymbol{\beta}).$$

- A logistic likelihood $l(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta})$ is

$$= \sum_{i=1}^{n} \left[y_i \log p_i(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta}) + (1 - y_i) \log(1 - p_i(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta})) \right],$$

where

$$p_i(\beta_0, \boldsymbol{\gamma}, \boldsymbol{\beta}) = \frac{e^{\beta_0 + z_i^{\mathrm{T}} \boldsymbol{\gamma} + x_i^{\mathrm{T}} \boldsymbol{\beta}}}{1 + e^{\beta_0 + z_i^{\mathrm{T}} \boldsymbol{\gamma} + x_i^{\mathrm{T}} \boldsymbol{\beta}}}.$$

Penalized Likelihood for Censored Survival Outcomes

- For the i-th individual,
  - t_i : Either survival or censoring time of the i-th individual
  - δ_i : the indicator of failure ($\delta_i = 1$) or censoring ($\delta_i = 0$)
- The penalized likelihood is

$$Q_\lambda(\boldsymbol{\gamma}, \boldsymbol{\beta}) = -\log L(\boldsymbol{\gamma}, \boldsymbol{\beta}) + p_\lambda(\boldsymbol{\beta}).$$

- A Cox partial likelihood can be written as

$$L(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \prod_{i=1}^{n} \left\{ \frac{\exp(z_i^{\mathrm{T}}\boldsymbol{\gamma} + x_i^{\mathrm{T}}\boldsymbol{\beta})}{\sum_{j \in R_i} \exp(z_i^{\mathrm{T}}\boldsymbol{\gamma} + x_j^{\mathrm{T}}\boldsymbol{\beta})} \right\}^{\delta_i},$$

  where R_i is the index set for observations that are at risk before time t_i.

Types of Penalty Functions

- Convex penalty functions
  - Lasso (Tibshirani, JRSS, 1996)
  - Fused lasso (Tibshirani *et al.* JRSS, 2005)
  - Adaptive lasso (Zou, JASA, 2006)
  - Elastic-net (Zou and Hastie, JRSS, 2005)
- Non-convex penalty functions
  - l_q-norm penalty with $0 < q < 1$
  - Smoothly clipped absolute deviation (SCAD) (Fan and Li, JASA, 2005)
  - Minimax concave penalty (MCP) (Zhang, AOS, 2010)
- Group structure penalty functions
  - Group lasso (Yuan and Lin, JRSS, 2006)
  - Graph-constrained regularization (Li and Li, AOAS 2010)
  - Sparse group lasso (Simon *et al.* JGCS 2013)
  - Group exponential lasso (Breheny, Biometrics 2015)

Lasso and Ridge

- Lasso regression uses a l_1-norm penalty

$$p_\lambda(\boldsymbol{\beta}) = \lambda \|\boldsymbol{\beta}\|_1 = \lambda \sum_{j=1}^{p} |\beta_j|$$

- Lasso can perform variable selection since
  - $\hat{\beta}_j = 0$ for most of $j \in \{1, 2, \ldots, p\}$.
  - $\hat{\beta} \neq 0$ for only a few j.
- Ridge regression uses a squared l_2-norm penalty

$$p_\lambda(\boldsymbol{\beta}) = \lambda \|\boldsymbol{\beta}\|_2^2 = \lambda \sum_{j=1}^{p} \beta_j^2$$

Lasso and Ridge

- Variable selection
  - Ridge regression shrinks the coefficient estimates towards zero, just like lasso.
  - But, ridge regression coefficients are not exactly equal to zero, unlike lasso.
  - Technically, ridge regression cannot perform variable selection.
- Correlated variables
  - Ridge regression is likely to shrink the coefficient estimates of correlated variables together.
  - Lasso regression often selects only one of correlated variables even though all of them are related with a response.
  - Lasso regression is not ideal for highly correlated predictors.
- Neither ridge nor lasso will universally dominate the other.

Limitations of Lasso

- Lasso penalty function (l_1-norm)
  - If $p > n$, lasso selects at most n variables. The number of selected genes is bounded by the number of samples.
  - Lasso is indifferent to highly correlated variables and tends to pick only one variable and ignore the rest.
- Ridge penalty function (squared l_2-norm)
  - It cannot perform variable selection. (No coefficients are zero.)
  - But, ridge regression shrinks correlated features to each other
- Elastic-net regularization
  - The elastic-net combines the l_1 and squared l_2 penalties
  - The l_1 part of the penalty generates a sparse model.
  - The l_2 part of the penalty encourages grouping effects.

Elastic-net Regularization

- The penalty function of elastic-net is

$$p_\lambda(\boldsymbol{\beta}) = \lambda \left(\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2 \right)$$
$$= \lambda\alpha \sum_{j=1}^{p} |\beta_j| + \lambda(1-\alpha) \sum_{j=1}^{p} \beta_j^2$$

  where $\alpha \in [0, 1]$ is a balance between lasso and ridge.
  - Lasso if $\alpha = 1$
  - Ridge regression if $\alpha = 0$.
- Need to pick up the optimal λ and α.
  - α is often fixed as either 0.1 or 0.5.
  - λ is chosen by cross-validation.

Regularization with Covariates

- Suppose that we have a q-dimensional covariate vector. The covariate of the i-individual is

$$z_i = (z_{i1}, z_{i2}, \ldots, z_{iq})^{\mathrm{T}}$$

- The lasso solution $\hat{\boldsymbol{\beta}}_\lambda$ minimizes the penalized likelihood

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{k=1}^{q}\gamma_k z_{ik} - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|,$$

  if y_i is a quantitative outcome.

- Note that the regression coefficients of the covariates are not penalized.

One-standard-error Rule in Variable Selection

- Choose a smaller model: one-standard-error rule
- Calculate the standard error of the estimated test MSE for each λ.

$$sd\left(\text{CVE}(\lambda_l)\right) = \sqrt{\frac{1}{n-1}\sum_{k=1}^{K}\left(\text{MSE}_k(\lambda_l) - \frac{1}{n}\sum_{k=1}^{K}\text{MSE}_k(\lambda_l)\right)^2}$$

where

$$\text{MSE}_k(\lambda_l) = \sum_{i \in C_k}\left(y_i - x_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\lambda_l}^{[-k]}\right)^2$$

- Select the smallest model (largest λ) for which the test error is within one standard error of the lowest point on the curve.

$$\left[\min(\text{CVE}) - sd\left(\text{CVE}\right), \min(\text{CVE}) + sd\left(\text{CVE}\right)\right]$$

Example: Breast Cancer Data

```
BC$geneID[1:100, ]
miss <- is.na(BC$geneID[,2])
sum(miss)
gene.all <- BC$geneID[!miss, 2]
length(gene.all)
length(unique(gene.all))
```

```
x2 <- x[, !miss]
colnames(x2) <- gene.all
dim(x2)
x2[1:3, 1:10]
```

```
z <- BC$z
summary(z)
z$age
x3 <- cbind(z$age, x2)
colnames(x3)[1] <- "age"
dim(x3)
x3[1:3, 1:10]
```

Example: Breast Cancer Data

```
gr <- glmnet(x3, y, alpha=0)
gl <- glmnet(x3, y, alpha=1)
```

```
par(mfrow=c(1,2))
plot(gr, "lambda")
plot(gl, "lambda")
```

```
pp <- rep(1, ncol(x3))
pp[1] <- 0
gp <- glmnet(x3, y, alpha=1, penalty.factor=pp)
```

```
dim(gp$beta)
gp$beta[1:4,]
data.frame(lambda=gp$lambda, df=gp$df)
```

Example: Breast Cancer Data

```
set.seed(123)
gcv <- cv.glmnet(x3, y, penalty.factor=pp, nfolds=5)
c(gcv$lambda.min, gcv$lambda.1se)
dev.off()
plot(gcv)
```

```
plot(gp, "lambda")
abline(v=log(gcv$lambda.min), lwd=2, lty=2, col="red")
abline(v=log(gcv$lambda.1se), lwd=2, lty=2, col="blue")
```

```
fit1 <- coef(gcv, s="lambda.min")[-1, ]
fit2 <- coef(gcv, s="lambda.1se")[-1, ]
c(sum(as.matrix(fit1)!=0), sum(as.matrix(fit2)!=0))
```

```
w1 <- which(as.matrix(fit1)!=0)
w2 <- which(as.matrix(fit2)!=0)
data.frame(coef=fit1[w1])
data.frame(coef=fit2[w2])
```

Example: Breast Cancer Data

```
set.seed(111)
g01 <- cv.glmnet(x3, y, alpha=0.1, penalty.factor=pp, nfolds=5)
g05 <- cv.glmnet(x3, y, alpha=0.5, penalty.factor=pp, nfolds=5)
```

```
par(mfrow=c(1, 2))
plot(g01)
plot(g05)
```

```
fit11 <- coef(g01, s="lambda.min")[-1, ]
fit12 <- coef(g01, s="lambda.1se")[-1, ]
c(sum(as.matrix(fit11)!=0), sum(as.matrix(fit12)!=0))
```

```
fit51 <- coef(g05, s="lambda.min")[-1, ]
fit52 <- coef(g05, s="lambda.1se")[-1, ]
c(sum(as.matrix(fit51)!=0), sum(as.matrix(fit52)!=0))
```

```
w5 <- which(as.matrix(fit51)!=0)
data.frame(genes=names(fit51[w5]), coef=fit51[w5])
```

Example: Breast Cancer Data

```
set.seed(1234)
foldid <- sample(rep(1:5, length=nrow(x3)))
table(foldid)
```

```
new.lam <- glmnet(x3, y, penalty.factor=pp)$lambda
alp <- seq(0.1, 1, 0.1)
cve <- df <- NULL
for (j in 1:length(alp)) {
    g <- cv.glmnet(x3, y, alpha=alp[j], penalty.factor=pp,
                   foldid=foldid, lambda=new.lam)
    cve[j] <- min(g$cvm)
    df[j] <- g$nzero[g$cvm==min(g$cvm)]
}
```

```
data.frame(alpha=alp, DF=df, min.CVE=cve)
dev.off()
plot(alp, cve, type="b", xlab="alpha", ylab="Minimum CVE")
```

1. Statistical Variable Selection and Cross-validation
2. Analysis of High-dimensional Genomic Data I

3. Analysis of High-dimensional Genomic Data II

4. Analysis of High-dimensional DNA Methylation Data

High-dimensional Genomic Data with a Group Structure

- Group structured genomic data
  - Single-nucleotide polymorphism (SNP) data: genes or genetic regions includes multiple SNPs.
  - DNA methylation data: genes or genetic regions contains multiple CpG sites.
  - Gene expression data: genetic pathways or signaling pathways consist of multiple genes.
- Genomic measurements within the same group are likely to be correlated with each other.
- Penalty functions encouraging grouping effects
  - Group lasso (only group selection)
  - Group exponential lasso (both group and individual selection)
  - Sparse group lasso (both group and individual selection)
  - Network-based regularization (both group and individual selection)

Group Lasso

- Group lasso performs group selection
- Suppose that the p predictors are divided into K disjoint groups of sizes q_k with corresponding coefficients $\boldsymbol{\beta}_k$.
- Group lasso solves the convex optimization problem.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[-l(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{K} \sqrt{q_k} \|\boldsymbol{\beta}_k\|_2 \right]$$

- $\|\boldsymbol{\beta}_k\|_2 = 0$ if and only if $\boldsymbol{\beta}_k = (0, \ldots, 0)$, thus entire groups are shrunken to 0.
- Examples:
  - SNPs/variants/CpG sites in the same genes.
  - Genes grouped into functional pathways
  - Indicator variables for the levels of a categorical variable

Group Lasso in Bioinformatics

BMC Bioinformatics

BioMed Central

Research

Open Access

Identifying main effects and epistatic interactions from large-scale SNP data via adaptive group Lasso

Can Yang*[1], Xiang Wan[1], Qiang Yang[2], Hong Xue[3] and Weichuan Yu*[1]

Bioinformatics, 34(2), 2018, 278–285
doi: 10.1093/bioinformatics/btx594
Advance Access Publication Date: 18 September 2017
Original Paper

OXFORD

Bioimage informatics

A novel SCCA approach via truncated ℓ_1-norm and truncated group lasso for brain imaging genetics

BMC Bioinformatics

BioMed Central

Methodology article

Open Access

Supervised group Lasso with applications to microarray data analysis

Shuangge Ma*[1], Xiao Song[2] and Jian Huang[3]

Example: Breast Cancer Data

```
BC$geneID[1:100, ]
miss <- is.na(BC$geneID[,2])
sum(miss)
gene.all <- BC$geneID[!miss, 2]
length(gene.all)
length(unique(gene.all))
```

```
x2 <- x[, !miss]
colnames(x2) <- gene.all
dim(x2)
```

```
group.id <- as.numeric(factor(gene.all))
ss <- as.numeric(table(group.id))
table(ss)
```

```
k <- 13
w <- which(ss==13)
x2[1:5, group.id==w]
```

Example: Breast Cancer Data

```
oo <- order(group.id)
gr <- group.id[oo]
sx <- x2[ ,oo]
```

```
colnames(sx)[1:50]
gr[1:50]
length(gr)
max(gr)
length(unique(gene.all))
```

```
gg <- gglasso(sx, y, group=gr, dfmax=100)
data.frame(lambda=gg$lambda, df=gg$df)
plot(gg)
```

```
k <- 30
abline(v=log(gg$lambda[k]), col="red", lty=2, lwd=1.5)
```

Example: Breast Cancer Data

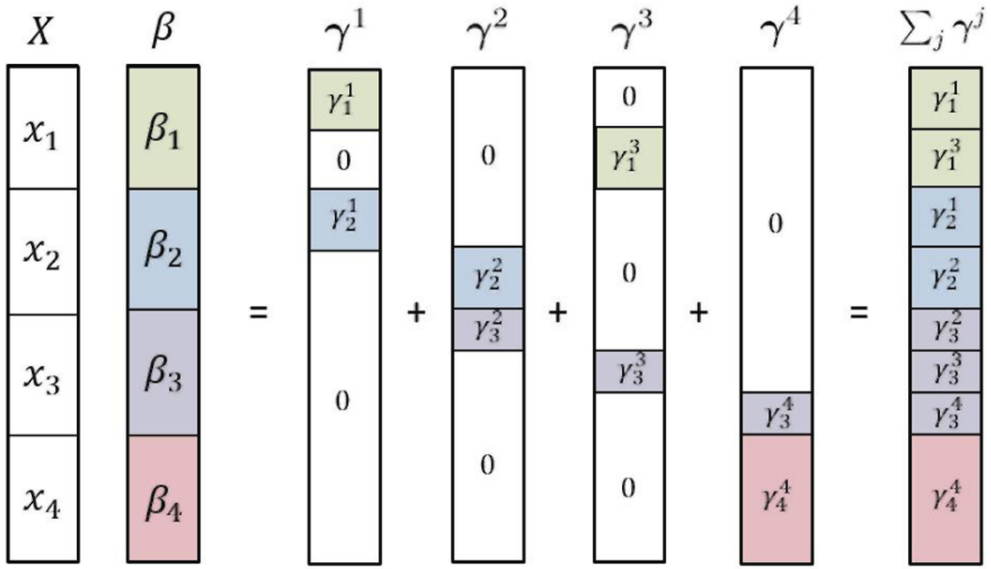
```
dim(gg$beta)
bk <- gg$beta[,k]
as.matrix(bk[bk!=0], ncol=1)
```

```
sel.gene <- names(bk[bk!=0])
length(sel.gene)
table(sel.gene)
table(table(sel.gene))
gg$beta[sel.gene, c("s0", "s5", "s15", "s20", "s25")]
```

```
######## Do not Run ####################################
# set.seed(123)
# ggcv <- cv.gglasso(sx, y, group=gr, dfmax=100, nfolds=5)
# plot(ggcv)
# ggcv$lambda.min
# ggcv$lambda.1se
########################################################
```

Overlapping Group Lasso

- If K groups are not disjoint, then overlapping group lasso can be applied.
- There are 4 variables such as x_1, x_2, x_3 and x_4 and 4 groups such as $\{x_1, x_2\}$, $\{x_2, x_3\}$, $\{x_1, x_3\}$ and $\{x_3, x_4\}$,



$$\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}\sum_{j=1}^{4}\gamma^j = \tilde{\boldsymbol{X}}\boldsymbol{\gamma},$$

where $\tilde{\boldsymbol{X}} = (x_1, x_1, x_2, x_2, x_3, x_3, x_3, x_4)$.

Duplication of Overlapped Variables

- Suppose that the observed values of the i-individual at p variables by

$$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^{\mathrm{T}}$$

- If the l-th variable are overlapped by k_l different groups, we can duplicate the corresponding l-th variables k_l times, i.e.,

$$\tilde{\boldsymbol{x}}_i = (\overbrace{x_{i1}, \ldots, x_{i1}}^{k_1}, \overbrace{x_{i2}, \ldots, x_{i2}}^{k_2}, \ldots\ldots, \overbrace{x_{ip}, \ldots, x_{ip}}^{k_p})^{\mathrm{T}}$$

- If the l-th variables belong to only one group, $k_l = 1$.
- The p-dimensional vector $\boldsymbol{x}_i$ is converted to the q-dimensional vector $\tilde{\boldsymbol{x}}_i$, where

$$q = \sum_{l=1}^{p} k_p.$$

Overlapping Group Lasso in Pathway Analysis

- Overlapping group lasso is a simple extension from group lasso, where overlapped variables are duplicated.
- It has been recently applied to a genetic pathway selection problem as an alternative method of gene set enrichment analysis (GSEA).

Cancer Inform. 2016; 15: 179–187.
Published online 2016 Sep 15. doi: 10.4137/CIN.S40043
PMCID: PMC5026200
PMID: 27679461

Overlapping Group Logistic Regression with Applications to Genetic Pathway Selection

Yaohui Zeng and Patrick Breheny

- In multiple genetic pathways, many genes are overlapped.
- Suppose that each pathway is regarded as a group and an individual gene is a member of group. Then, overlapping group lasso can identify genetic pathways associated with a phenotype outcome.

Example: Breast Cancer Data

```
gene.kegg <- BC$path.kegg
gene.kegg[1:10]
length(gene.kegg)
lapply(gene.kegg, length)
sz <- as.numeric(lapply(gene.kegg, length))
sort(sz)
```

```
whh <- which(sz > 10 & sz < 100)
sel.path <- gene.kegg[whh]
length(sel.path)
lapply(sel.path, length)
```

```
all.kegg <- unique(unlist(sel.path))
length(all.kegg)
ngene <- unique(colnames(sx))
length(ngene)
length(intersect(ngene, all.kegg))
```

Example: Breast Cancer Data

```
xp <- ig <- gid <- gname <- NULL
for(i in 1:length(sel.path)) {
    ig <- intersect(sel.path[[i]], ngene)
    k <- 0
    gsy <- NULL
    for (j in 1:length(ig)) {
        wh <- which(ig[j]==colnames(sx))
        xp <- cbind(xp, sx[,wh,drop=FALSE])
        k <- k + length(wh)
        gsy <- c(gsy, rep(ig[j], length(wh)))
    }
    gid <- c(gid, rep(i, k))
    gname <- c(gname, gsy)
}
colnames(xp) <- gname
```

```
dim(xp)
c(length(gid), length(gname))
c(length(unique(gname)), max(gid))
```

Example: Breast Cancer Data

```
ogl <- gglasso(xp, y, group=gid, dfmax=15)
plot(ogl)
```

```
wu <- unique(ogl$df)[2:10]
grd <- NULL
pathseq <- vector(mode="list", length=length(wu))
for (i in 1:length(wu)) {
    wj <- min(which(ogl$df==wu[i]))
    wk <- unique(gid[ogl$beta[,wj]!=0])
    if (i==1) wb <- wk
    else wb <- wk[!(wk %in% grd)]
    grd <- c(grd, wk)
    gs <- NULL
    for (k in 1:length(wb)) {
        gs <- c(gs, names(sel.path[wb[k]]))
    }
    pathseq[[i]] <- gs
}
pathseq
```

Example: Breast Cancer Data

```
K <- 4
wuk <- wu[1:K]
path <- vector(mode="list", length=length(wuk))
names(path) <- pathseq[1:K]
path
```

```
grd <- gg <- NULL
for (i in 1:length(wuk)) {
     wj <- min(which(ogl$df==wuk[i]))
     wk <- which(ogl$beta[,wj]!=0)
     if (i==1) wb <- wk
     else wb <- wk[!(wk %in% grd)]
     grd <- c(grd, wk)
     path[[i]] <- colnames(xp[ ,wb])
}
path
```

Bi-level Selection

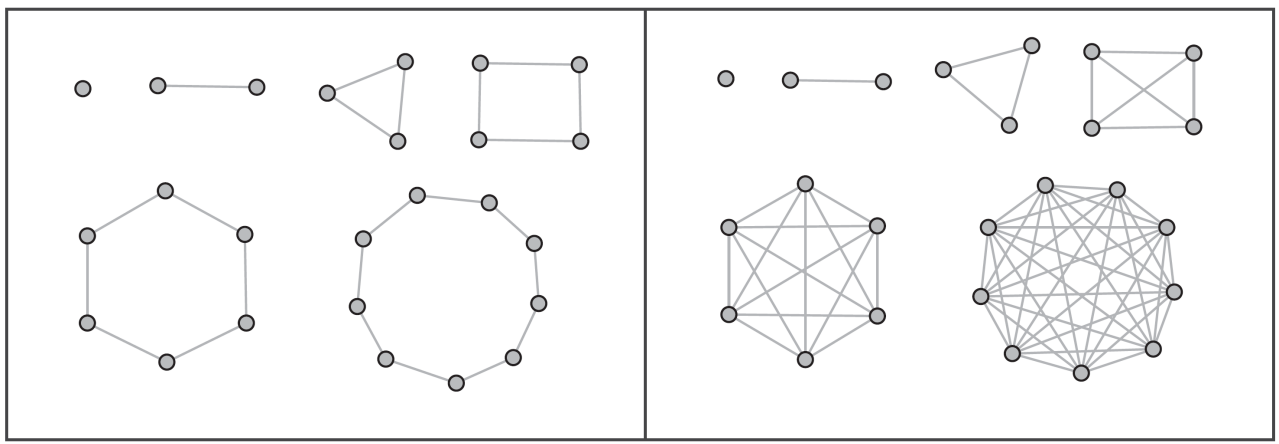
- Group lasso introduced the idea of grouped variables and the idea of selecting important groups of variables, rather than individual variables.
- There are often situations where we might be interested in selection at both the individual and group levels, or bi-level selection.
- Grouping made sense here: if a group is unimportant to the response, we don't want to select any members from it.
- However, selecting individual members also makes sense: just because a group is important to the response doesn't mean that every single member is important.
- Even if a group is important, a group can contain unimportant members. In this case, group lasso fails to identify unimportant members.

Bi-level Selection

- Sparse group lasso (SGL)
  - Additive penalty with lasso and group lasso
  - Computationally expensive
  - R package : "`SGL`"
- Group exponential lasso (GEL)
  - Multiplicative penalty with lasso and an exponential function
  - Computationally efficient
  - R package : "`grpreg`"
- Network-based regularization
  - Elastic-net penalty with a Laplaician matrix
  - It can be applied to either a network-structured data or a group-structured data
  - Computationally efficient
  - R package : "`pclogit`"

Network-based Regularization

- Network-based regularization was used for group structured data (Sun and Wang, 2012).
- Imaginary network structure for each gene
  - Ring network: adjacent sites/vartiants are only connected with each other
  - Fully connected (F.con) network: all sites/variants are fully connected with each other



Ring F.con

- Laplacian matrix has a block diagonal matrix

Example: Breast Cancer Data

```
g01 <- grpreg(sx, y, group=gr, penalty="gel")
g02 <- SGL(list(x=sx, y=y), index=gr, alpha=0.5)
```

```
par(mfrow=c(1,2))
plot(g01)
title("GEL")
matplot(rev(log(g02$lambdas)), t(g02$beta), type="l", lty=1,
        col="red", xlab="log.lambda", ylab="Coefficients",
        main="SGL")
```

```
set.seed(111)
gcv <- cv.grpreg(sx, y, group=gr, penalty="gel")
dev.off()
plot(gcv)
```

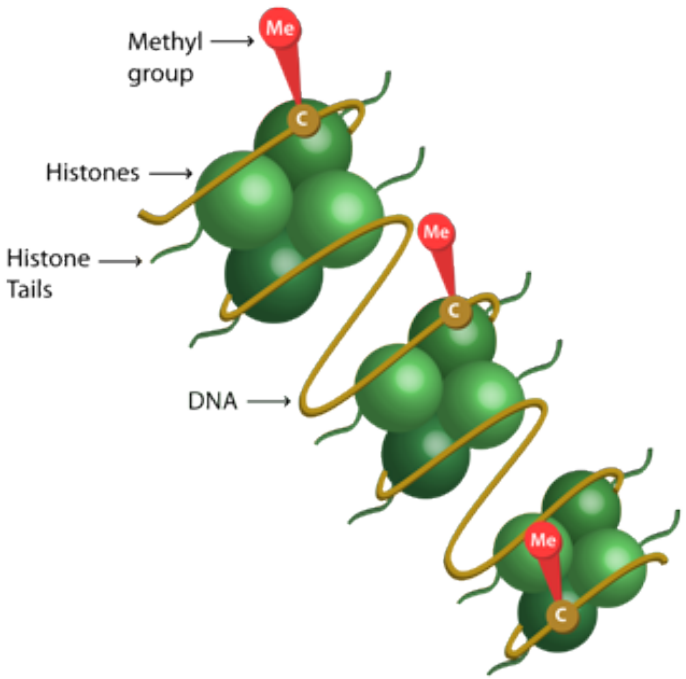
```
gcv.min <- coef(gcv)
as.matrix(gcv.min[gcv.min!=0], ncol=1)
```

1. Statistical Variable Selection and Cross-validation
2. Analysis of High-dimensional Genomic Data I
3. Analysis of High-dimensional Genomic Data II

4. Analysis of High-dimensional DNA Methylation Data

DNA Methylation

- DNA methylation is the addition of a methyl group to the 5' position of cytosine in the context of a CpG dinucleotide.
- Adding methyl groups change the appearance and structure of DNA.



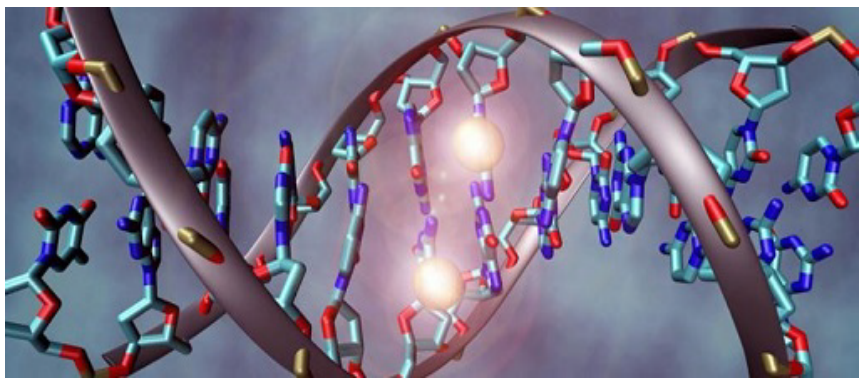


Illustration of a DNA molecule that is methylated at the cytosines.

High-dimensional DNA Methylation Data

- CpG (Cytosine-phosphate-Guanine) rich regions are frequently hyper-methylated in cancer tissues, but not in normal tissues.
- DNA methylation data have recently been generated from high-throughput DNA methylation platforms
  - Illumina Infinium HumanMethylation450 BeadChip array.
  - Illumina Infinium MethylationEPIC BeadChip array.

Design Markers | **Process BeadChip** | Analyze Data

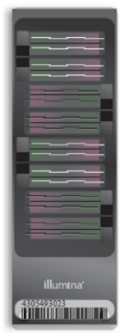
illumina®

Infinium® MethylationEPIC BeadChip

Affordable methylome analysis meets cutting edge content.

Highlights

- **Unique Combination of Coding Region and Enhancer-Wide Coverage**
  Over 850,000 methylation sites per sample at single-nucleotide resolution
- **High Assay Reproducibility**
  >98% reproducibility for technical replicates
- **Simple Workflow**
  PCR-free protocol with the powerful Infinium HD Assay
- **Compatible with FFPE Samples**
  Protocol available for methylation studies on FFPE samples



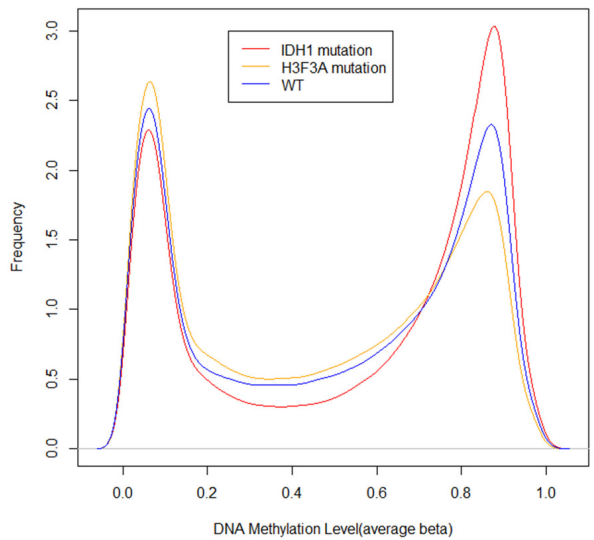
Methylation β-values

- The methylation array data of the CpG sites are a sequence of β-values.
- For each locus (CpG site)

$$\beta = \frac{\max(M, 0)}{\max(U, 0) + \max(M, 0) + 100},$$

where U is the fluorescent signal from an unmethylated allele and M is that from a methylated allele.

- This β-value ranges continuously from 0 (unmethylated) to 1 (completely methylated)
- The methylation levels are enriched at 0 and 1.



Analysis of High-dimensional DNA Methylation Data

- Research goal
  - Identify differentially methylated CpG sites and genes between cancer cases and healthy controls over entire genomes.
  - (Matched) case-control association study with high-dimensional DNA data
- Statistical problem
  - Variable selection for high-dimensional data
  - Methylation values of multiple CpG sites within the same genes are highly correlated with each other.
- Regularization methods have been proposed.
  - Sun and Wang (2012, *Bioinformatics*)
  - Sun and Wang (2013, *Stat. Med.*)
  - Sun *et al.* (2017, *Bioinformatics*)
  - Choi *et al.* (2018, *J. Bioinform. Comput. Biol.*)
  - Kim and Sun (2019, *BMC Bioinformatics*)
  - Kim *et al.* (2020, *J. Stat. Comput. Simul.*)

Ovarian Cancer Study from 27K DNA Methylation Data

- The data is available at the NCBI Gene Expression Omnibus.
- Investigated differentially methylated CpG sites between ovarian cancer cases and healthy controls.
  - Teschendorff, A. *et al.* (2010, *Genome Ressearch*)
  - Wang, S. (2011, *Genetic Epidemiology*)
  - Sun and Wang (2012, *Bioinformatics*)
  - Huang, H. *et al.* (2013, *BMC Bioinformatics*)
  - Chen, Y. *et al.* (2014, *Genetic Epidemiology*)
- Data from Illumina Infinium HumanMethylation27K Beadchip.
- After data processing, we ended up with 20,461 CpG sites from 12,770 genes.
- 12,770 genes have from 1 to 22 CpG sites.
- 152 healthy controls and 123 ovarian cancer cases.

Ovarian Cancer Data

```
data(ovarian27k)
str(ovarian27k)
```

```
head(ovarian27k$genename)
tail(ovarian27k$genename)
```

```
gsize <- ovarian27k$group
c(length(gsize), sum(gsize))
table(gsize)
```

```
ww <- which.max(gsize)
cc <- cumsum(gsize)[(ww-1):ww]
ovarian27k$genename[(cc[1]+1):cc[2], c(2,3)]
```

Ovarian Cancer Data

```
y <- ovarian27k$y
y
length(y)
table(y)
```

```
x <- ovarian27k$x
dim(x)
summary(as.numeric(x))
```

```
xm <- log2(x/(1-x))
summary(as.numeric(xm))
```

```
par(mfrow=c(1,2))
hist(x, nclass=100, col="orange",
     main=expression(paste(beta,"-values")))
hist(xm, nclass=100, col="orange", main="M-values")
```

Univariate Analysis

```
pval.t <- apply(xm, 2, function(t) t.test(t~y)$p.val)
pval.w <- apply(xm, 2, function(t) wilcox.test(t~y)$p.val)
```

```
alpha <- 0.1
oo.t <- order(pval.t)
bt <- p.adjust(pval.t, method="bonferroni")
st <- sum(bt < alpha)
ot <- oo.t[1:st]
gt <- ovarian27k$genename[ot, c(2,3)]
data.frame(gt, p.value=pval.t[ot], Bonferroni=bt[ot])
```

```
oo.w <- order(pval.w)
bw <- p.adjust(pval.w, method="bonferroni")
sw <- sum(bw < alpha)
ow <- oo.w[1:sw]
gw <- ovarian27k$genename[ow, c(2,3)]
data.frame(gw, p.value=pval.w[ow], Bonferroni=bw[ow])
```

Univariate Analysis

```
par(mfrow=c(2, 1))
cc <- rep("gray", length(pval.t))
cc[bt < alpha] <- "darkblue"
plot(-log10(pval.t), type="p", pch=20, col=cc, xlab="CpG sites",
     ylab=expression(paste("-log"[10],"(P)")),
     main="Independent two sample T-test")
abline(h=-log10(max(pval.t[bt < alpha])), col="darkblue",
       lty=3, lwd=1.5)
```

```
cw <- rep("gray", length(pval.w))
cw[bw < alpha] <- "darkblue"
plot(-log10(pval.w), type="p", pch=20, col=cw, xlab="CpG sites",
     ylab=expression(paste("-log"[10],"(P)")),
     main="Wilcoxon Rank Sum Test")
abline(h=-log10(max(pval.w[bw < alpha])),col="darkblue",
       lty=3, lwd=1.5)
```

Lasso: Ovarian Cancer Data

```
g1 <- glmnet(xm, y, alpha=1, family="binomial")
dev.off()
plot(g1, "lambda")
g1$df
```

```
set.seed(1111)
gv1 <- cv.glmnet(xm, y, alpha=1, family="binomial", nfolds=5)
plot(gv1)
```

```
fit11 <- coef(gv1, s="lambda.min")[-1, ]
fit12 <- coef(gv1, s="lambda.1se")[-1, ]
c(sum(as.matrix(fit11)!=0), sum(as.matrix(fit12)!=0))
```

```
w11 <- which(as.matrix(fit11)!= 0)
w12 <- which(as.matrix(fit12)!= 0)
gn11 <- ovarian27k$genename[w11, c(2,3)]
gn12 <- ovarian27k$genename[w12, c(2,3)]
```

Lasso: Ovarian Cancer Data

```
data.frame(gn11, beta=fit11[w11])
data.frame(gn12, beta=fit12[w12])
```

```
plot(g1, "lambda")
abline(v=log(gv1$lambda.min), lty=2, col=2)
abline(v=log(gv1$lambda.1se), lty=2, col=4)
```

```
par(mfrow=c(2,1))
plot(abs(fit11), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 1.1),
     main="Lasso using lambda.min")
plot(abs(fit12), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 1.1),
     main="Lasso using lambda.1se")
```

Elastic-net: Ovarian Cancer Data

```
g21 <- glmnet(xm, y, alpha=0.1, family="binomial")
g22 <- glmnet(xm, y, alpha=0.5, family="binomial")
```

```
par(mfrow=c(1,2))
plot(g21, "lambda")
plot(g22, "lambda")
```

```
set.seed(1234)
gv2 <- cv.glmnet(xm, y, alpha=0.5, family="binomial", nfolds=5)
dev.off()
plot(gv2)
```

```
fit21 <- coef(gv2, s="lambda.min")[-1, ]
fit22 <- coef(gv2, s="lambda.1se")[-1, ]
c(sum(as.matrix(fit21)!=0), sum(as.matrix(fit22)!=0))
```

```
w21 <- which(as.matrix(fit21)!= 0)
w22 <- which(as.matrix(fit22)!= 0)
```

Elastic-net: Ovarian Cancer Data

```
gn21 <- ovarian27k$genename[w21, c(2,3)]
gn22 <- ovarian27k$genename[w22, c(2,3)]
data.frame(gn21, beta=fit21[w21])
data.frame(gn22, beta=fit22[w22])
```

```
plot(g22, "lambda")
abline(v=log(gv2$lambda.min), lty=2, col=2)
abline(v=log(gv2$lambda.1se), lty=2, col=4)
```

```
par(mfrow=c(2,1))
plot(abs(fit21), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 1.1),
     main="Elastic-net using lambda.min")
plot(abs(fit22), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 1.1),
     main="Elastic-net using lambda.1se")
```

Elastic-net: Ovarian Cancer Data

```
set.seed(111)
foldid <- rep(1:5, length=length(y))
tapply(y, foldid, table)
new.lam <- glmnet(xm, y, family="binomial")$lambda
```

```
alp <- seq(0.1, 1, 0.1)
cve <- df <- NULL
for (j in 1:length(alp)) {
    gv2 <- cv.glmnet(xm, y, alpha=alp[j], family="binomial",
                     foldid=foldid, lambda=new.lam)
    cve[j] <- min(gv2$cvm)
    df[j] <- gv2$nzero[gv2$cvm==min(gv2$cvm)]
}
```

```
data.frame(alpha=alp, DF=df, min.CVE=cve)
dev.off()
plot(alp, cve, type="b", xlab="alpha", ylab="Minimum CVE")
```

Group Lasso: Ovarian Cancer Data

```
## library(gglasso)
```

```
grp <- rep(1:length(gsize), gsize)
ry <- y
ry[y==0] <- -1
```

```
##### Do not run ##########################################
# g3 <- gglasso(xm, ry, group=grp, loss="logit")
###########################################################
```

```
ur1 <- url("https://github.com/statsun78/kogo/blob/main/
            g35.Rdata?raw=true")
load(ur1)
```

```
g3$df
plot(g3)
```

Group Lasso: Ovarian Cancer Data

```
##### Do not run ###############################################
# set.seed(1111)
# gv3 <- cv.gglasso(xm, ry, group=grp, loss="logit", nfolds=5)
################################################################
```

```
fit31 <- coef(gv3, s="lambda.min")[-1]
fit32 <- coef(gv3, s="lambda.1se")[-1]
c(sum(as.matrix(fit31)!=0), sum(as.matrix(fit32)!=0))
```

```
w31 <- which(as.matrix(fit31)!= 0)
w32 <- which(as.matrix(fit32)!= 0)
gn31 <- ovarian27k$genename[w31, c(2,3)]
gn32 <- ovarian27k$genename[w32, c(2,3)]
```

```
data.frame(gn31, beta=fit31[w31])
data.frame(gn32, beta=fit32[w32])
```

Group Lasso: Ovarian Cancer Data

```
plot(g3)
abline(v=log(gv3$lambda.min), lty=2, col=2)
abline(v=log(gv3$lambda.1se), lty=2, col=4)
```

```
par(mfrow=c(2,1))
plot(abs(fit31), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 1.1),
     main="Group Lasso using lambda.min")
plot(abs(fit32), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 1.1),
     main="Group Lasso using lambda.1se")
```

GEL: Ovarian Cancer Data

```
g4 <- grpreg(xm, y, group=grp, family="binomial", penalty="gel")
dev.off()
plot(g4)
```

```
set.seed(12)
gv4 <- cv.grpreg(xm, y, group=grp, family="binomial",
                 penalty="gel")
abline(v=gv4$lambda.min, lty=2, col=2)
plot(gv4)
```

```
fit41 <- coef(gv4)[-1]
w41 <- which(fit41!=0)
gn41 <- ovarian27k$genename[w41, c(2,3)]
data.frame(gn41, beta=fit41[w41])
```

```
plot(abs(fit41), type="h", col="darkgreen", xlab="CpG sites",
     ylab=expression(paste("|",hat(beta),"|")), ylim=c(0, 3),
     main="Group Exponential Lasso")
```

Network-based Regularization: Ovarian Cancer Data

```
library(pclogit)
```

```
g5 <- pclogit(xm, y, alpha=0.5, group=gsize)
matplot(log(g5$lambda), t(g5$beta), type="l", lty=1, col="red",
        xlab=expression(log(lambda)), ylab="Coefficient",
        main="Network-based regularization")
data.frame(lambda=g5$lambda, df=g5$df)
```

```
##### Do not run ##########################################
# set.seed(1212)
# gv5 <- sel.pclogit(xm, y, alpha=0.5, group=gsize, K=200)
###########################################################
```

```
top <- 10
w51 <- as.numeric(gv5$maxsel[1:top,1])
gn51 <- ovarian27k$genename[w51, c(2,3)]
data.frame(gn51, SP=gv5$maxsel[1:top, 2])
```

Summary: Ovarian Cancer Data

```
ooo <- c(ot, ow, w11, w12, w21, w22, w31, w32, w41, w51)
ov <- rep(0, length=nrow(ovarian27k$genename))
for (i in 1:length(ooo)) {
    oi <- ooo[i]
    ov[oi] <- ov[oi] + 1
}
```

```
length(ov)
table(ov)
```

```
wo <- which(ov > 1)
gn6 <- ovarian27k$genename[wo, c(2,3)]
ff <- data.frame(gn6, Freq=ov[wo])
ok <- order(ov[wo], decreasing=TRUE)
Final <- ff[ok, ]
rownames(Final) <- seq(1, nrow(Final))
Final
```