KUBIG
Data Science and Machine Learning

Week 4.
Regularization

Review

- Logistic Regression Optimization

$$Loss[Y,\ \hat{Y}] = -\sum_{i=1}^{n}[\,y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \log(1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{argmin}\ L[Y,\ \hat{Y}]$$

$\Rightarrow$ Do not have an explicit solution!

Review

- Quadratic approximation (2nd order Taylor Expansion)

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}_0) + \nabla L(\boldsymbol{\theta}_0)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \mathbf{H}(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

where

$$\nabla L(\boldsymbol{\theta}_0) = \frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}) \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$$

$$\mathbf{H}(\boldsymbol{\theta}_0) = \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} L(\boldsymbol{\theta}) \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$$

Review

- Newton-Raphson Method

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)})\nabla L(\boldsymbol{\theta}^{(t)})$$

$$= \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)})\frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} L(\boldsymbol{\theta}^{(t)})$$

$$cf. \quad \theta^{(\mathrm{t}+1)} = \theta^{(\mathrm{t})} - \frac{f'(\theta^{(\mathrm{t})})}{f''(\theta^{(\mathrm{t})})}$$

Review

$$L[\boldsymbol{\beta}] = -\sum_{i=1}^{n}[y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \log(1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

$$\nabla L(\boldsymbol{\beta}) = \frac{\partial}{\partial\boldsymbol{\beta}}L(\boldsymbol{\beta}) = -\sum_{i=1}^{n}\left[y_i\mathbf{X}_i - \frac{\exp(\boldsymbol{\beta}^T\mathbf{X}_i)}{1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)}\mathbf{X}_i\right]$$

$$\mathbf{H}(\boldsymbol{\beta}) = \frac{\partial^2}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}L(\boldsymbol{\beta}) = \sum_{i=1}^{n}\left[\left(\frac{\exp(\boldsymbol{\beta}^T\mathbf{X}_i)}{1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)}\right)\left(\frac{1}{1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)}\right)\mathbf{X}_i\mathbf{X}_i^T\right]$$

Review

$$L[\boldsymbol{\beta}] = -\sum_{i=1}^{n}[\,y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \log(1+\exp(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

Update

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \mathbf{H}^{-1}\left(\boldsymbol{\beta}^{(t)}\right)\nabla L\left(\boldsymbol{\beta}^{(t)}\right)$$

until

$$||\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^{t}|| < \epsilon \qquad \text{for small } \epsilon > 0$$

Review

					Solvers
Penalties	**'liblinear'**	**'lbfgs'**	**'newton-cg'**	**'sag'**	**'saga'**
Multinomial + L2 penalty	no	yes	yes	yes	yes
OVR + L2 penalty	yes	yes	yes	yes	yes
Multinomial + L1 penalty	no	no	no	no	yes
OVR + L1 penalty	yes	no	no	no	yes
Behaviors					
Penalize the intercept (bad)	yes	no	no	no	no
Faster for large datasets	no	no	no	yes	yes
Robust to unscaled datasets	yes	yes	yes	no	no

Review

- Gradient Descent

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)})\nabla L(\boldsymbol{\theta}^{(t)})$$

$$\Rightarrow \quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)}\, \nabla L(\boldsymbol{\theta}^{(t)})$$

Review

- Gradient Descent

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)}) \nabla L(\boldsymbol{\theta}^{(t)})$$

$$\Rightarrow \quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)} \nabla L(\boldsymbol{\theta}^{(t)}) \quad \rightarrow$$ in Deep Learning

$$\Rightarrow \quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla L(\boldsymbol{\theta}^{(t)})$$

Review

- (Batch) Gradient Descent

$$\underbrace{\boxed{\quad\quad\quad\quad\quad\quad\quad\quad \boldsymbol{\theta} \quad\quad\quad\quad\quad\quad\quad\quad}}_{\text{1 epoch}}$$

- Batch size = n

Review

- Stochastic Gradient Descent

$$\underbrace{\begin{array}{|c|c|c|c|c|c|}\hline \theta^{(1)} & \theta^{(2)} & \theta^{(3)} & \cdots\cdots & \theta^{(n-1)} & \theta^{(n)} \\ \hline \end{array}}_{\text{1 epoch}}$$

- Batch size = 1

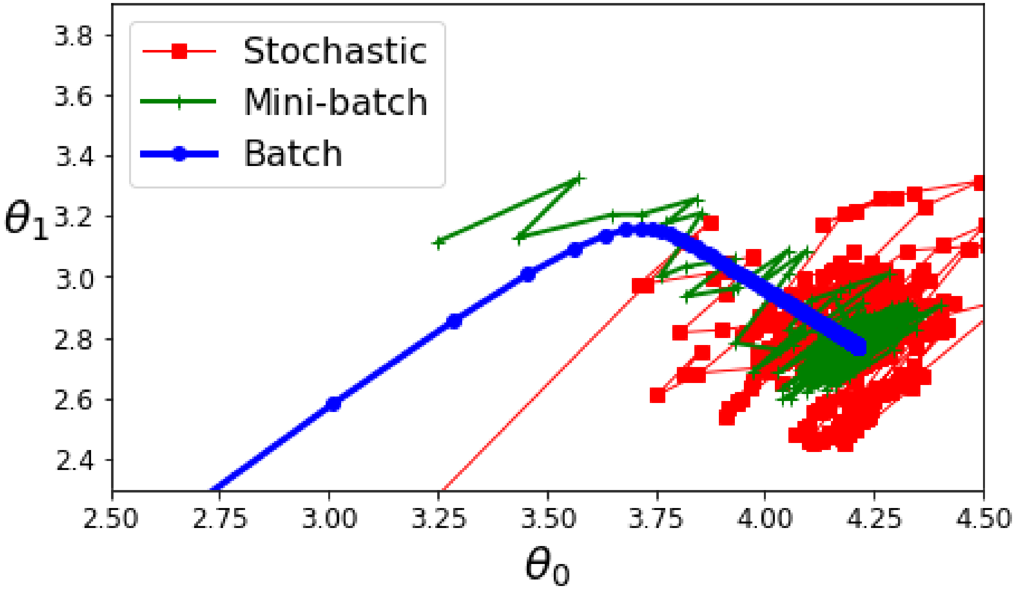
Review

- Mini-Batch Gradient Descent

$$\underbrace{\boxed{\theta^{(1)}}\boxed{\theta^{(2)}}\boxed{\quad\cdots\cdots\quad}\boxed{\theta^{(k)}}}_{\text{1 epoch}}$$

- Batch size = p , where p x k = n

Review



Stochastic
Mini-batch
Batch
θ_1
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
θ_0

Stein's Paradox

- Let $\mathbf{X} = [X_1, \cdots, X_p]^T \sim N_p(\boldsymbol{\theta}, I)$

- The UMVUE and MLE of $\boldsymbol{\theta}$ is

$$\hat{\boldsymbol{\theta}}_{MLE,UMVUE} = \mathbf{X}$$

- Using squared error loss, the risk of $\hat{\boldsymbol{\theta}}_{MLE,UMVUE}$ is

$$R(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_{UMVUE}) = E[||\mathbf{X} - \boldsymbol{\theta}||^2] = p$$

Stein's Paradox

- James and Stein (1961) Estimator

$$\hat{\boldsymbol{\theta}}_{JS} = \left(1 - \frac{p-2}{||\mathbf{X}||^2}\right)\mathbf{X}$$

- When $p \geq 3$,

$$R(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_{JS}) = p - (p-2)E\left(\frac{1}{||\mathbf{X}||^2}\right) < p$$

Steins Paradox

- Proof

$$R(\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}}_{JS}) = E\left[||\mathbf{X} - \boldsymbol{\theta} - \frac{(p-2)\mathbf{X}}{||\mathbf{X}||^2}||^2\right]$$

$$= p - 2(p-2)\sum_{j}^{p} E\left(\frac{X_j(X_j - \theta_j)}{||\mathbf{X}||^2}\right) + (p-2)^2 E\left(\frac{1}{||\mathbf{X}||^2}\right)$$

$$= p - (p-2)E\left(\frac{1}{||\mathbf{X}||^2}\right)$$

Since $\sum_{j}^{p} E\left(\frac{X_j(X_j-\theta_j)}{||\mathbf{X}||^2}\right) = (p-2)E\left(\frac{1}{||\mathbf{X}||^2}\right)$

Steins Paradox

- JS estimator shrinks each component of $\mathbf{X}$ towards the origin, and thus the biggest improvement comes when $\|\, \boldsymbol{\theta} \,\|$ is close to zero.
- Normality assumption is not critical, and similar results can be shown for a wide class of distributions.

Ridge Regression

- Normal Equation

$$(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} = \mathbf{X}^T\mathbf{Y}$$

- The OLS estimator

$$\widehat{\boldsymbol{\beta}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

Ridge Regression

- Normal Equation

$$(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} = \mathbf{X}^T\mathbf{Y}$$

- The OLS estimator

$$\widehat{\boldsymbol{\beta}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- What if $\mathbf{X}^T\mathbf{X}$ is not invertible?

Ridge Regression

- We can consider

$$\hat{\boldsymbol{\beta}}_{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

- Ridge estimator is

$$\hat{\boldsymbol{\beta}}_{Ridge} = \underset{\boldsymbol{\beta}}{argmin}\,(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

Lagrange Multiplier Theorem

Standard form problem (not necessarily convex)

$$\begin{aligned} \text{minimize } & f_0(x) \\ \text{subject to } & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned}$$

variable $x \in \mathbb{R}^n$, domain $\mathcal{D}$, optimal value p^*

Lagrange Multiplier Theorem

Standard form problem (not necessarily convex)

$$\begin{aligned} &\text{minimize } f_0(x) \\ &\text{subject to } f_i(x) \le 0, \; i = 1, \dots, m \\ &\qquad\qquad h_i(x) = 0, \; i = 1, \dots, p \end{aligned}$$

variable $x \in \mathbb{R}^n$, domain $\mathcal{D}$, optimal value p^*

$$\boxed{L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)}$$

Lagrange Multiplier Theorem

Definition: Lagrange dual function $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is the minimized Lagrangian with respect to x

$$
\begin{aligned}
g(\lambda, \nu) &= \inf_{x \in \mathbb{R}^n} L(x, \lambda, \nu) \\
&= \inf_{x \in \mathbb{R}^n} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)
\end{aligned}
$$

Lagrange Multiplier Theorem

Definition: Lagrange dual function $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is the minimized Lagrangian with respect to x

$$
\begin{aligned}
g(\lambda, \nu) &= \inf_{x \in \mathbb{R}^n} L(x, \lambda, \nu) \\
&= \inf_{x \in \mathbb{R}^n} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)
\end{aligned}
$$

$$
f_0(\tilde{x}) \geq L(\tilde{x}, \lambda, \nu) \geq \inf_{x \in \mathbb{R}^n} L(x, \lambda, \nu) = g(\lambda, \nu)
$$

Lagrange Multiplier Theorem

$$\text{minimize } f_0(x) + \sum_i \lambda_i f_i(x) + \sum_i \nu_i h_i(x)$$

1. Lagrange dual is an *unconstrained* minimization of Lagrangian
2. Lagrangians are measure of ‘irritation’ or penalty associated with violation of constraints, but having less penalty or ‘smoothed’ irritation than the originally strict constraints

Lagrange Multiplier Theorem

- Primal Problem

$$\min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})$$

subject to $||\boldsymbol{\beta}||_2^2 \leq C,$ where $||\boldsymbol{\beta}||_2^2 = \boldsymbol{\beta}^T\boldsymbol{\beta} = \sum_j^p \beta_j^2$

- Dual Problem

$$\min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda(\, ||\boldsymbol{\beta}||_2^2 - C\,)$$

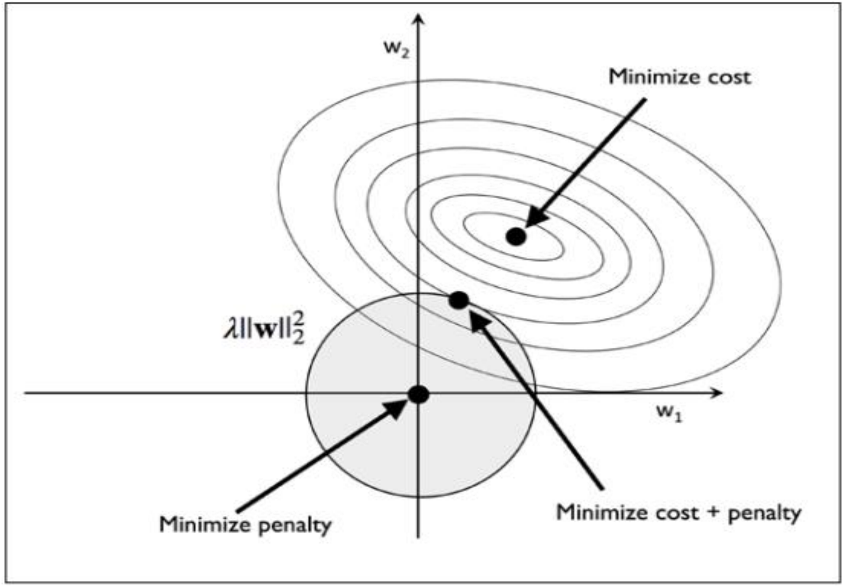
Lagrange Multiplier Theorem

$$(\widehat{\boldsymbol{\beta}}^{\lambda,2} =)\ \widehat{\boldsymbol{\beta}}_{Ridge} = \underset{\boldsymbol{\beta}}{argmin}\ (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\ ||\boldsymbol{\beta}||_2^2$$

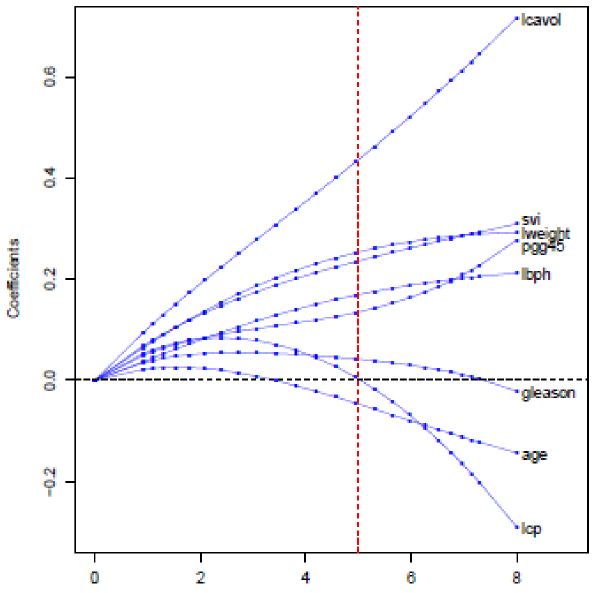
$$\Leftrightarrow \underset{\boldsymbol{\beta}}{argmin}\ (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda(\ ||\boldsymbol{\beta}||_2^2 - C\ )$$

```
# Logistic regression
from sklearn.linear_model import LogisticRegression
Logit = LogisticRegression(C=1e2, random_state=1023)  # C = 1/λ. 디폴트: L2, One-versus-Rest.
Logit.fit(X_train_std, y_train)
```

Ridge Regression



Ridge Regression



Bias-Variance Trade off

- Expected Prediction Error

$$E\left[(Y_0 - \hat{Y}_0)^2\right] = \quad \sigma^2 \quad + \quad E\left[(\mu_0 - \hat{Y}_0)^2\right]$$

Irreducible error　　model error

where $\quad Y_0 = \mu_0 + \epsilon_0 = \mathbf{X}_0^T \boldsymbol{\beta} + \epsilon_0$

and $\quad \hat{Y}_0 = \mathbf{X}_0^T \widehat{\boldsymbol{\beta}}$

Bias-Variance Trade off

- Model Error

$$E\left[(\mu_0 - \hat{Y}_0)^2\right] = E\left[(\mu_0 - E[\hat{Y}_0] + E[\hat{Y}_0] - \hat{Y}_0)^2\right]$$

$$= \underbrace{(\mu_0 - E[\hat{Y}_0])^2}_{\text{Bias}^2} + \underbrace{Var[\hat{Y}_0]}_{\text{variance}}$$

LASSO Regression

- Ridge Regression solves

$$\min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda ||\boldsymbol{\beta}||_2^2 \qquad (L2\ penalty)$$

- LASSO Regression solves

LASSO Regression

- Ridge Regression solves

$$\min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_2^2 \qquad (L2\ penalty)$$

- LASSO Regression solves

$$\min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1 \qquad (L1\ penalty)$$

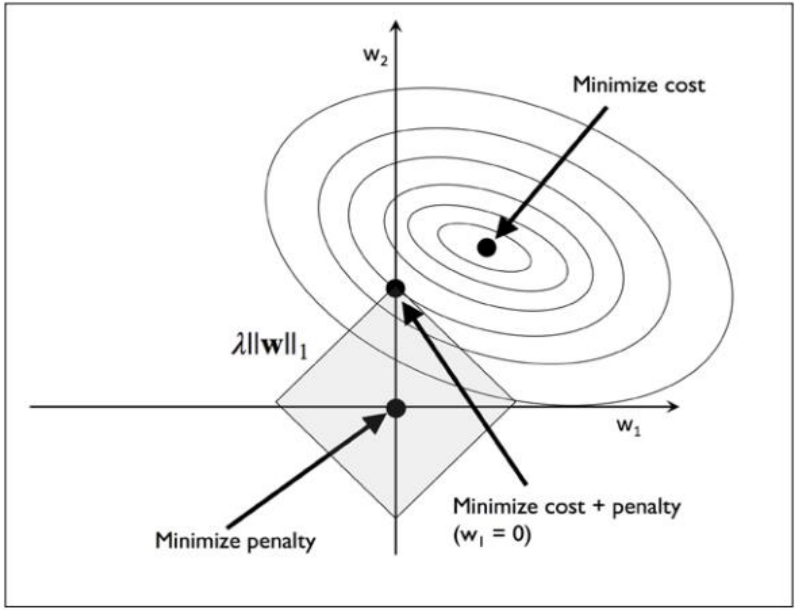
LASSO Regression

- LASSO (Least Absolute Shrinkage and Selection Operator)

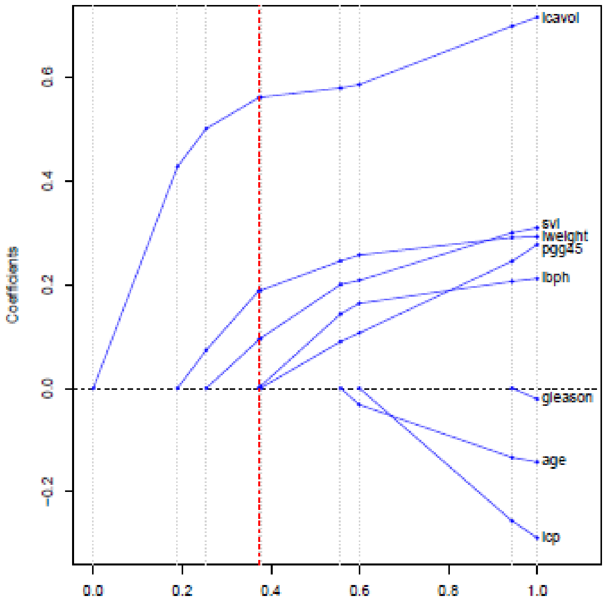
$$(\widehat{\boldsymbol{\beta}}^{\lambda,1} =)\ \widehat{\boldsymbol{\beta}}_{LASSO} = \underset{\boldsymbol{\beta}}{argmin}\ (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\ ||\boldsymbol{\beta}||_1$$

where $||\boldsymbol{\beta}||_1 = \sum_j^p |\beta_j|$

LASSO Regression



LASSO Regression



One-dimensional Case

- For simplicity, let $y_i = \beta x_i + \epsilon_i$, where $\sum_i^n x_i = 0$ and $\sum_i^n x_i^2 = n$

- Least Square estimator is

$$\hat{\beta}_{OLS} = \frac{1}{n}\sum x_i y_i$$

- Ridge estimator is

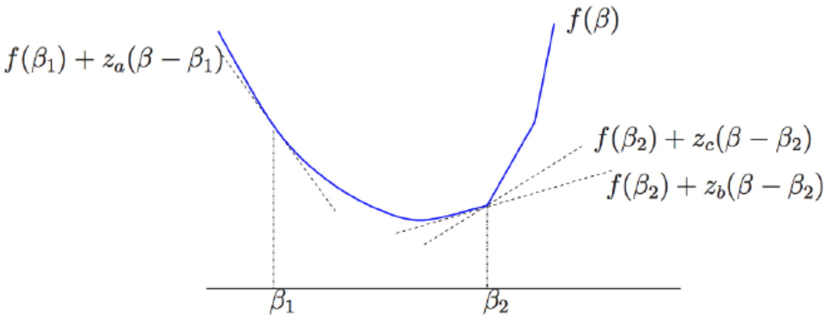
$$\hat{\beta}_{Ridge} = \frac{\hat{\beta}_{OLS}}{1+\lambda}$$

One-dimensional Case

Definition (Subgradient)

For a convex function $f : \mathbb{R}^p \to \mathbb{R}$, a vector $\mathbf{z} \in \mathbb{R}^p$ is to be a *subgradient* of f at $\boldsymbol{\beta}$ if

$$f(\boldsymbol{\beta}') \geq f(\boldsymbol{\beta}) + \mathbf{z}^T(\boldsymbol{\beta}' - \boldsymbol{\beta}) \qquad \text{for all } \beta' \in \mathbb{R}^p.$$



One-dimensional Case

- For LASSO, the objective function is

$$f(\beta) = \frac{1}{n}\sum(y_i - \beta x_i)^2 + \lambda|\beta|$$

and its subdifferential is

$$\partial f(\beta) = \begin{cases} \beta - \hat{\beta}_{OLS} + \lambda & \text{if } \beta > 0 \\ \beta - \hat{\beta}_{OLS} + \lambda[-1,1] & \text{if } \beta = 0 \\ \beta - \hat{\beta}_{OLS} - \lambda & \text{if } \beta < 0 \end{cases}$$

One-dimensional Case

- LASSO estimator is

$$\hat{\beta}_{LASSO} = \begin{cases} \hat{\beta}_{OLS} - \lambda & \text{if } \hat{\beta}_{OLS} > \lambda \\ 0 & \text{if } |\hat{\beta}_{OLS}| \le \lambda \\ \hat{\beta}_{OLS} + \lambda & \text{if } \hat{\beta}_{OLS} < -\lambda \end{cases}$$

One-dimensional Case

- LASSO estimator is

$$\hat{\beta}_{LASSO} = \begin{cases} \hat{\beta}_{OLS} - \lambda & \text{if } \hat{\beta}_{OLS} > \lambda \\ 0 & \text{if } |\hat{\beta}_{OLS}| \leq \lambda \\ \hat{\beta}_{OLS} + \lambda & \text{if } \hat{\beta}_{OLS} < -\lambda \end{cases}$$

- Soft-thresholding operator

$$S_\lambda(x) = sign(x)\ (|x| - \lambda)_+$$

One-dimensional Case

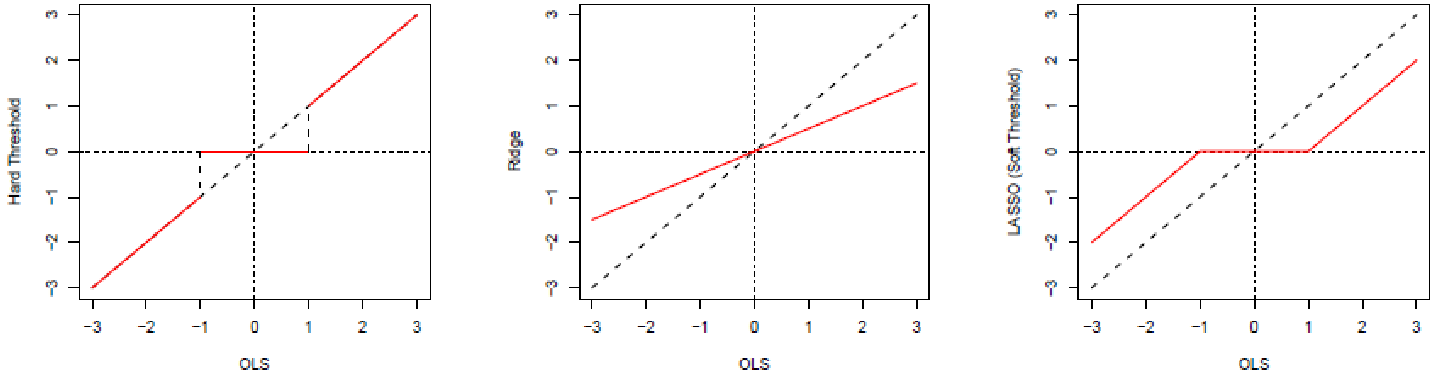
- LASSO estimator is

$$\hat{\beta}_{LASSO} = S_\lambda(\hat{\beta}_{OLS})$$

- Soft-thresholding operator

$$S_\lambda(x) = sign(x)\ (|x| - \lambda)_+$$

One-dimensional Case



(a) Hard Thresh. (b) Ridge Regression (c) Lasso (Soft Thresh.)

Feature Selection and Extraction

- LASSO (Least Absolute Shrinkage and Selection Operator)

$$(\hat{\boldsymbol{\beta}}^{\lambda,1} =)\ \hat{\boldsymbol{\beta}}_{LASSO} = \underset{\boldsymbol{\beta}}{argmin}\ (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\ ||\boldsymbol{\beta}||_1$$

where $||\boldsymbol{\beta}||_1 = \sum_j^p |\beta_j|$

Feature Selection and Extraction

- LASSO (Least Absolute Shrinkage and Selection Operator)

$$(\hat{\boldsymbol{\beta}}^{\lambda,1} =)\ \hat{\boldsymbol{\beta}}_{LASSO} = \underset{\boldsymbol{\beta}}{argmin}\ (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\ ||\boldsymbol{\beta}||_1$$

where $||\boldsymbol{\beta}||_1 = \sum_j^p |\beta_j|$

Feature Selection and Extraction

- LASSO (Least Absolute Shrinkage and Selection Operator)

- LASSO estimator gives a sparse solution

  ⇒ Thus, features are selected automatically!

Feature Selection and Extraction

- Feature Selection

  - Subset selection, Stepwise method, LASSO, Least Angle Regression etc..

- Feature Extraction (Dimension Reduction)

  - Principal Component Analysis, Partial Least Square, Discriminant Analysis, Factor Analysis, Latent Class Analysis, etc..

Elastic Net

- Elastic Net solves

$$\min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \left[\alpha ||\boldsymbol{\beta}||_1 + \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2 \right]$$

⇒ middle ground of LASSO and Ridge penalty

Elastic Net

- L_{p} penalty

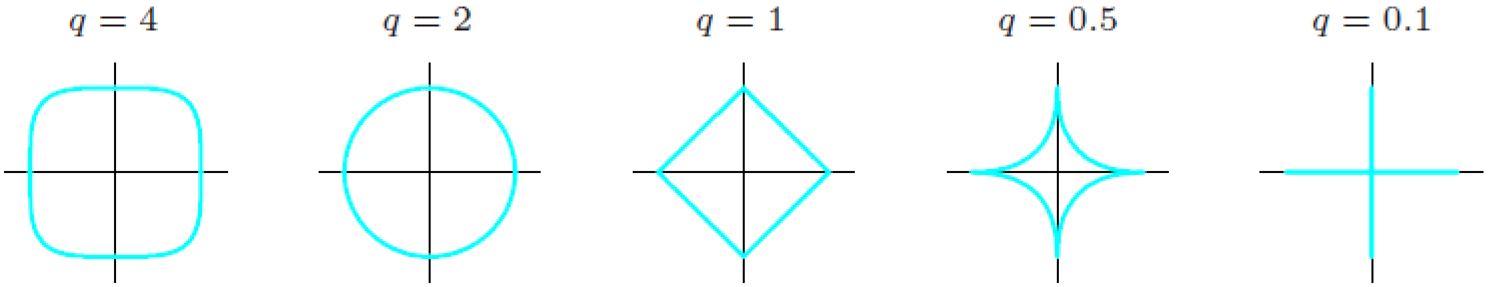


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

Elastic Net

- Elastic Net

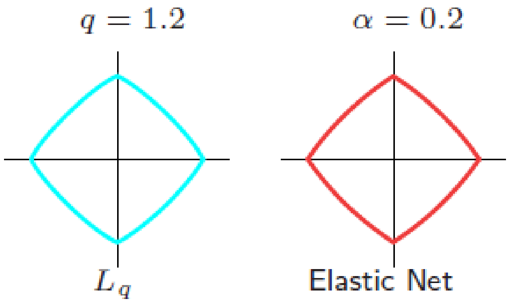


FIGURE 3.13. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for* $q = 1.2$ *(left plot), and the elastic-net penalty* $\sum_j(\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$ *for* $\alpha = 0.2$ *(right plot). Although visually very similar, the elastic-net has sharp (non-differentiable) corners, while the* $q = 1.2$ *penalty does not.*

Elastic Net

- Elastic Net

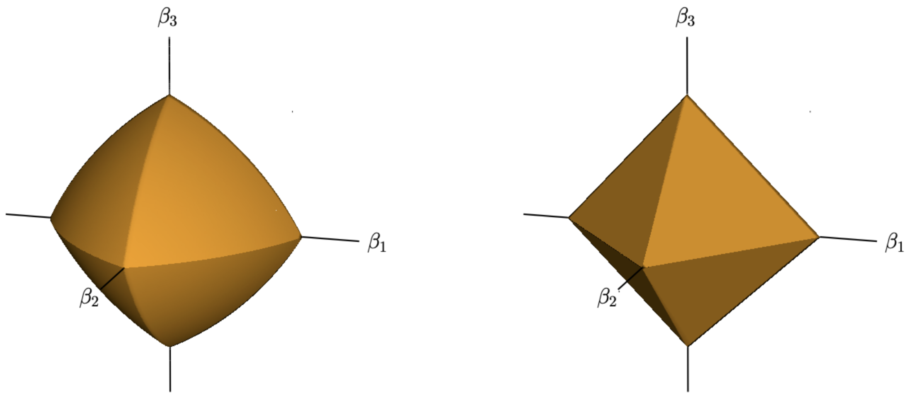


Figure 4.2 *The elastic-net ball with* $\alpha = 0.7$ *(left panel) in* $\mathbb{R}^3$*, compared to the* ℓ_1 *ball (right panel). The curved contours encourage strongly correlated variables to share coefficients (see Exercise 4.2 for details).*

Elastic Net

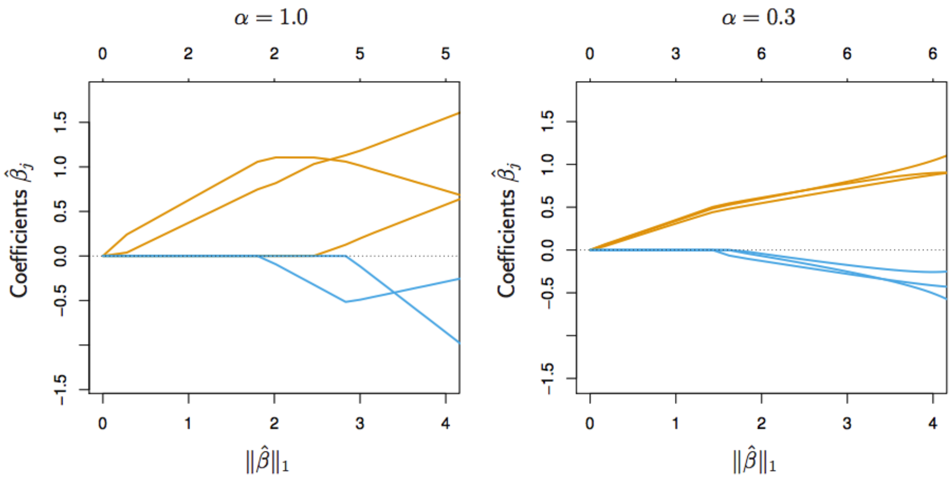


Figure 4.1 *Six variables, highly correlated in groups of three. The lasso estimates ($\alpha = 1$), as shown in the left panel, exhibit somewhat erratic behavior as the regularization parameter λ is varied. In the right panel, the elastic net with ($\alpha = 0.3$) includes all the variables, and the correlated groups are pulled together.*

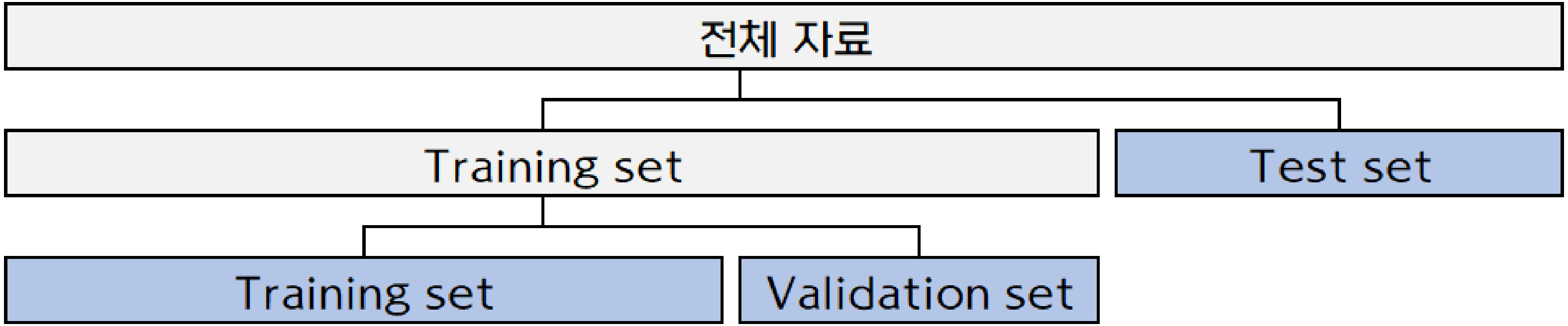
Elastic Net

```python
from sklearn.linear_model import LogisticRegression

lr2_1 = LogisticRegression(penalty='l2', C=1.0)    # L2 with C(=1/λ)=1
lr2_0_1 = LogisticRegression(penalty='l2', C=0.1)  # L2 with C(=1/λ)=0.1

lr1_1 = LogisticRegression(penalty='l1', C=1.0)    # L1 with C(=1/λ)=1
lr1_0_1 = LogisticRegression(penalty='l1', C=0.1)  # L1 with C(=1/λ)=0.1

lre_1 = LogisticRegression(penalty='elasticnet', C=1.0, l1_ratio=0.2)  # Elasticnet with C(=1/λ)=1.0
lre_0_1 = LogisticRegression(penalty='elasticnet', C=0.1, l1_ratio=0.2)  # Elasticnet with C(=1/λ)=0.1
```

```r
library(glmnet)
```

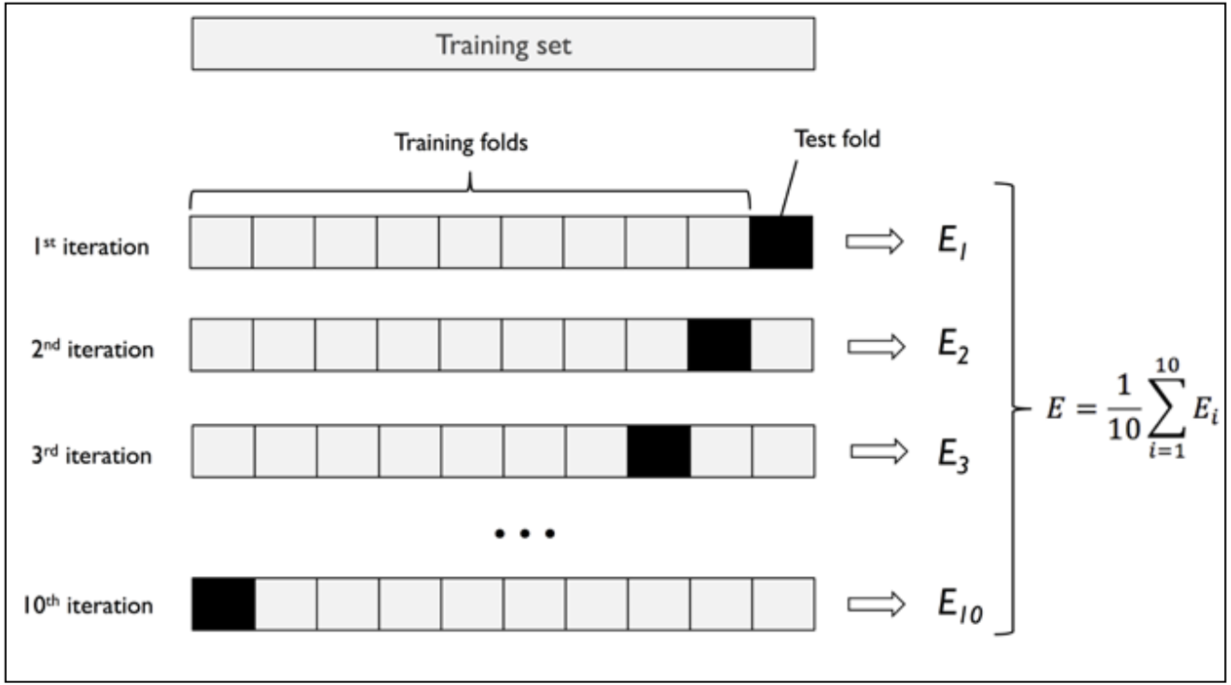
```r
fit = glmnet(x, y, alpha = 0.2, weights = c(rep(1,50),rep(2,50)), nlambda = 20)
```

Cross Validation



K-fold Cross Validation

- K = 10



K-fold Cross Validation

```
### Pipeline Streaming: 표준화 → PCA → Logistic Regression ###
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
pipe_lr = make_pipeline(StandardScaler(),
                        PCA(n_components=4),
                        LogisticRegression(random_state=1, solver='lbfgs'))  # 적용 순서대로 나열
pipe_lr.fit(X_train, y_train)            # 표준화(fit → transform) → PCA(fit → transform) → Logistic Reg fit의 순서로 처리
```

K-fold Cross Validation

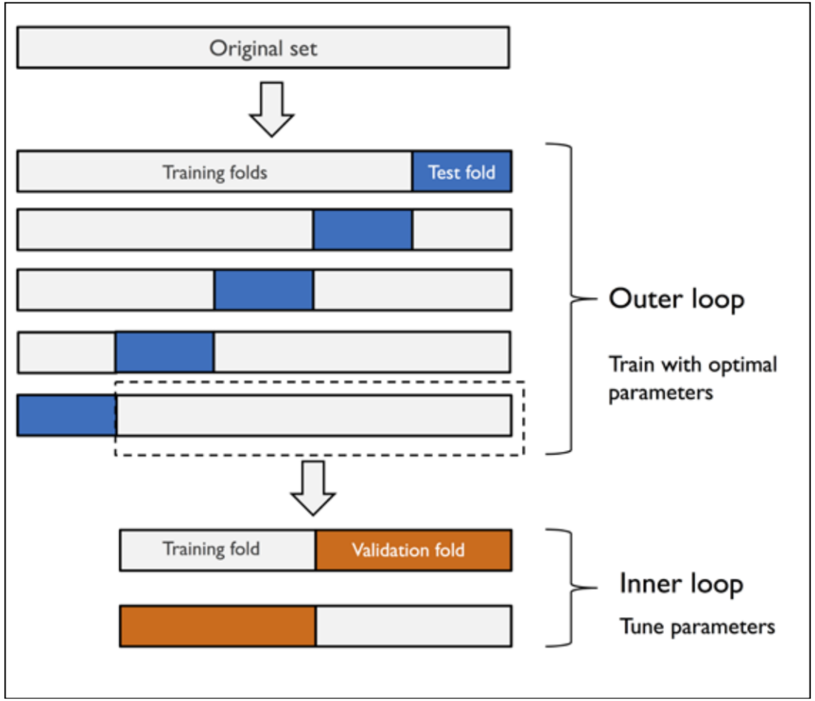
```
[ ]  ### K-fold cross-validation using pipeline ###
     from sklearn.model_selection import cross_val_score
     scores = cross_val_score(estimator=pipe_lr, X=X_train, y=y_train, cv=10)  # Accuracy scores
     print('CV accuracy scores: %s' % scores)

     import numpy as np
     print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

```
CV accuracy scores: [0.97826087 0.95652174 0.95652174 0.95652174 0.91304348 0.95555556
 0.97777778 0.97777778 1.         0.97777778]
CV accuracy: 0.965 +/- 0.022
```

Nested Cross Validation

- $K_1 = 5$

  $K_2 = 2$



Grid Search CV

```
[ ]  # Decision tree
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.model_selection import GridSearchCV
     from sklearn.model_selection import KFold
     inner_cv=KFold(n_splits=3, shuffle=True, random_state=0)
     outer_cv=KFold(n_splits=5, shuffle=True, random_state=0)
     gs = GridSearchCV(estimator=DecisionTreeClassifier(random_state=0),
                       param_grid=[{'max_depth': [1, 2, 3, 4, 5, 6, 7, None]}],
                       scoring='accuracy', cv=inner_cv)
     scores = cross_val_score(gs, X, y, scoring='accuracy', cv=outer_cv)
     print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

```
CV accuracy: 0.942 +/- 0.012
```

Grid Search CV

cv.glmnet {glmnet}   R Documentation

Cross-validation for glmnet

Description

Does k-fold cross-validation for glmnet, produces a plot, and returns a value for `lambda` (and `gamma` if `relax=TRUE`)

Usage

```
cv.glmnet(x, y, weights = NULL, offset = NULL, lambda = NULL,
  type.measure = c("default", "mse", "deviance", "class", "auc", "mae",
  "C"), nfolds = 10, foldid = NULL, alignment = c("lambda",
  "fraction"), grouped = TRUE, keep = FALSE, parallel = FALSE,
  gamma = c(0, 0.25, 0.5, 0.75, 1), relax = FALSE, trace.it = 0, ...)
```

reference

자료

19-2 STAT424 통계적 머신러닝 - 박유성 교수님

21-1 COSE423 볼록최적화입문 백승준 교수님

교재

파이썬을 이용한 통계적 머신러닝 (2020) – 박유성

ISLR (2013) - G. James, D. Witten, T. Hastie, R. Tibshirani

The elements of Statistical Learning (2001) - J. Friedman, T. Hastie, R. Tibshirani

Statistical Learning with Sparsity: The Lasso and Generalizations(2015)
 - R. Tibshirani, , T. Hastie, M. Wainwright