Mathematical techniques in data science

Lecture 11: Linear regression

Mathematical techniques in data sciences

- A short introduction to statistical learning theory
- SVM – the kernel trick
- Linear regression – regularization and feature selection
- Random forests — boosting and bootstrapping

Supervised learning: standard setting

- Given: a sequence of label data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ sampled (independently and identically) from an unknown distribution $P_{X,Y}$
- Goal: predict the label of a new instance x
- In a regression problem, the outputs are not categorical

Linear regression

Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)} + \epsilon$$

- p: number of variables ($X \in \mathbb{R}^p$)
- n: number of observations

Classical setting

Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)}$$

- $n \gg p$ (n much larger than p). With enough observations, we hope to be able to build a good model
- even if the true relationship between the variables is not linear, we can include transformations of variables

$$X^{(p+1)} = [X^{(1)}]^2, \quad X^{(p+2)} = X^{(1)} X^{(3)}, \ldots$$

- adding transformed variables can increase p significantly

Trade-off: complexity vs. interpretability

Linear model

$$Y = \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)} + \epsilon$$

- Higher values of p lead to more complex model $\rightarrow$ increases prediction power/accuracy
- Higher values of p make it more difficult to interpret the model: It is often the case that some or many of the variables regression model are in fact not associated with the response

Moderns settings

Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)} + \epsilon$$

- it is often the case that $n \ll p$
- requires supplementary assumptions (e.g. sparsity)
- can still build good models with very few observations.

Linear regression by least squares

Settings

- $Y \in \mathbb{R}^{n\times 1}, \quad X \in \mathbb{R}^{n\times(p+1)}$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} 1 & | & | & \dots & | \\ \dots & x^{(1)} & x^{(2)} & \dots & x^{(p)} \\ 1 & | & | & \dots & | \end{bmatrix}$$

  where $x^{(1)}, x^{(2)}, \dots, x^{(p)} \in \mathbb{R}^{n\times 1}$ are the observations of $X^{(1)}, X^{(2)}, \dots, X^{(p)}$.

- We want

$$Y = \beta^{(0)} + \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \dots \beta^{(p)}X^{(p)}$$

Settings

- We want

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)}$$

- Equivalent to

$$Y = X\beta, \qquad \beta = \begin{bmatrix} \beta^{(0)} \\ \beta^{(1)} \\ \\ \ldots \\ \beta^{(p)} \end{bmatrix}$$

Least squares

$$Y = X\beta$$

- In general, the system has no solution ($n \gg p$) or infinitely many solutions ($n \ll p$)
- The most popular estimation method is least squares, in which we pick the coefficients to minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Least squares

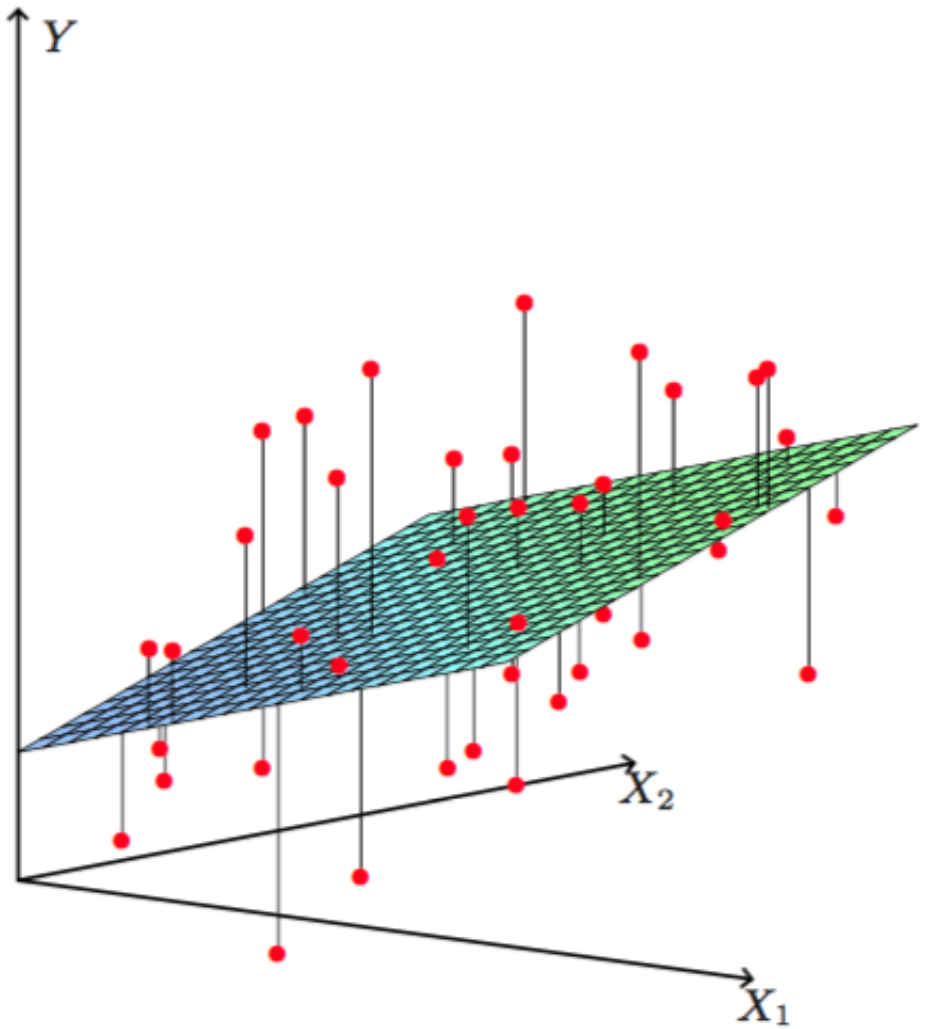


FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*

Least squares

- Minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- Or alternatively,

$$\hat{\beta} = \min_{\beta} \| Y - X\beta \|_2^2$$

Least squares

- Minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$
$$= \sum_{i=1}^{n} \left(y_i - \beta^{(0)} - \beta^{(1)} x_i^{(1)} - \beta^{(2)} x_i^{(2)} - \ldots - \beta^{(p)} x_i^{(p)} \right)^2$$

- Taking derivative

$$\frac{\partial RSS}{\partial \beta^{(j)}} = \sum_{i=1}^{n} 2(y_i - x_i \beta) x_i^{(j)} = 2[x^{(j)}]^T (Y - X\beta)$$

Least squares

- Set derivatives to zero

$$X^T(Y - X\beta) = 0$$

- If $X^T X$ is invertible

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

- Predicted values

$$\hat{Y} = X_{test}\hat{\beta} = X_{test}(X_{train}^T X_{train})^{-1} X_{train}^T Y_{train}$$

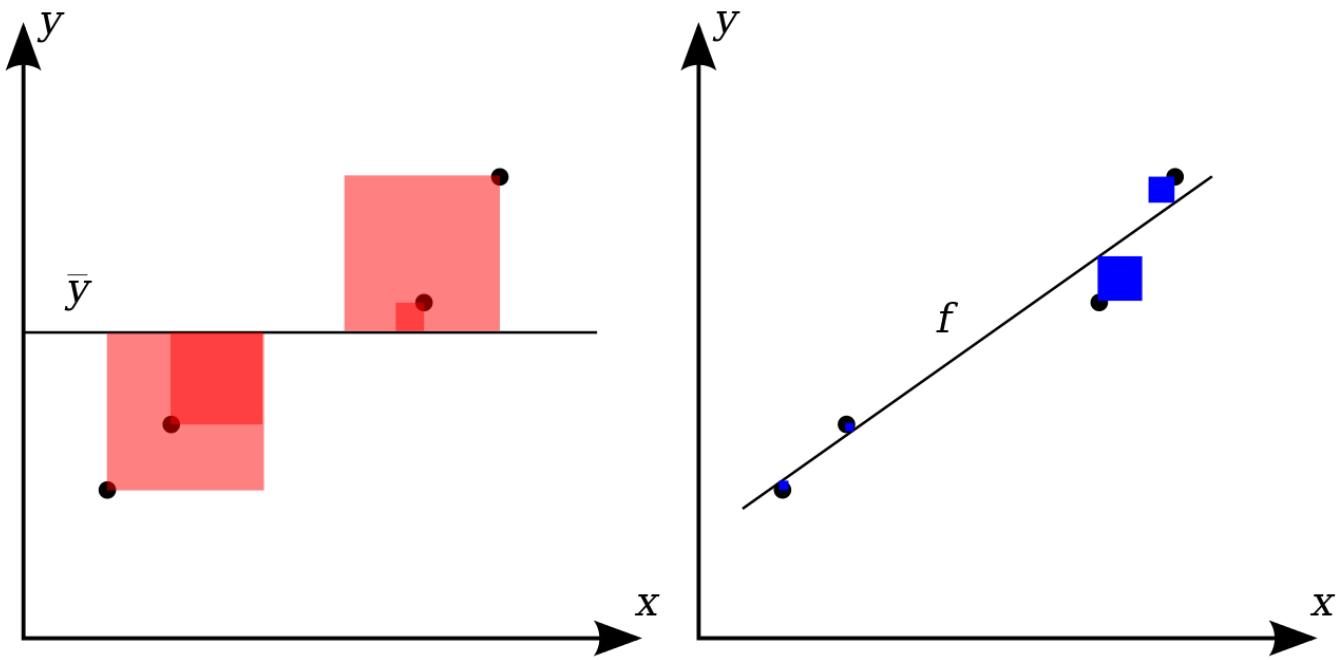
The coefficient of determination

- The coefficient of determination, called "R squared" and denoted by

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

  where $\bar{y}$ is the average of $y_1, \ldots, y_n$
- Often used to measure the quality of a linear model
- A model with a R^2 close to 1 fit the data well.

The coefficient of determination



In some sense, the R^2 measures how much better is the prediction compared to a constant prediction

The adjusted coefficient of multiple determination

- It is desirable to adjust R^2 to take account of the fact that its value may be quite high just because many predictors were used relative to the amount of data
- The adjusted coefficient of multiple determination

$$R_a^2 = 1 - \frac{\frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y}$ is the average of $y_1, \ldots, y_n$

sklearn.linear_model.LinearRegression

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0000...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

sklearn.preprocessing.PolynomialFeatures

```
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
>>> poly = PolynomialFeatures(interaction_only=True)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.],
       [ 1.,  2.,  3.,  6.],
       [ 1.,  4.,  5., 20.]])
```

Questions

- Is at least one of the predictors $X_1, X_2, \ldots, X_p$ useful in predicting the response?
- Do all the predictors help to explain Y, or is only a subset of the predictors useful?

Subset selection

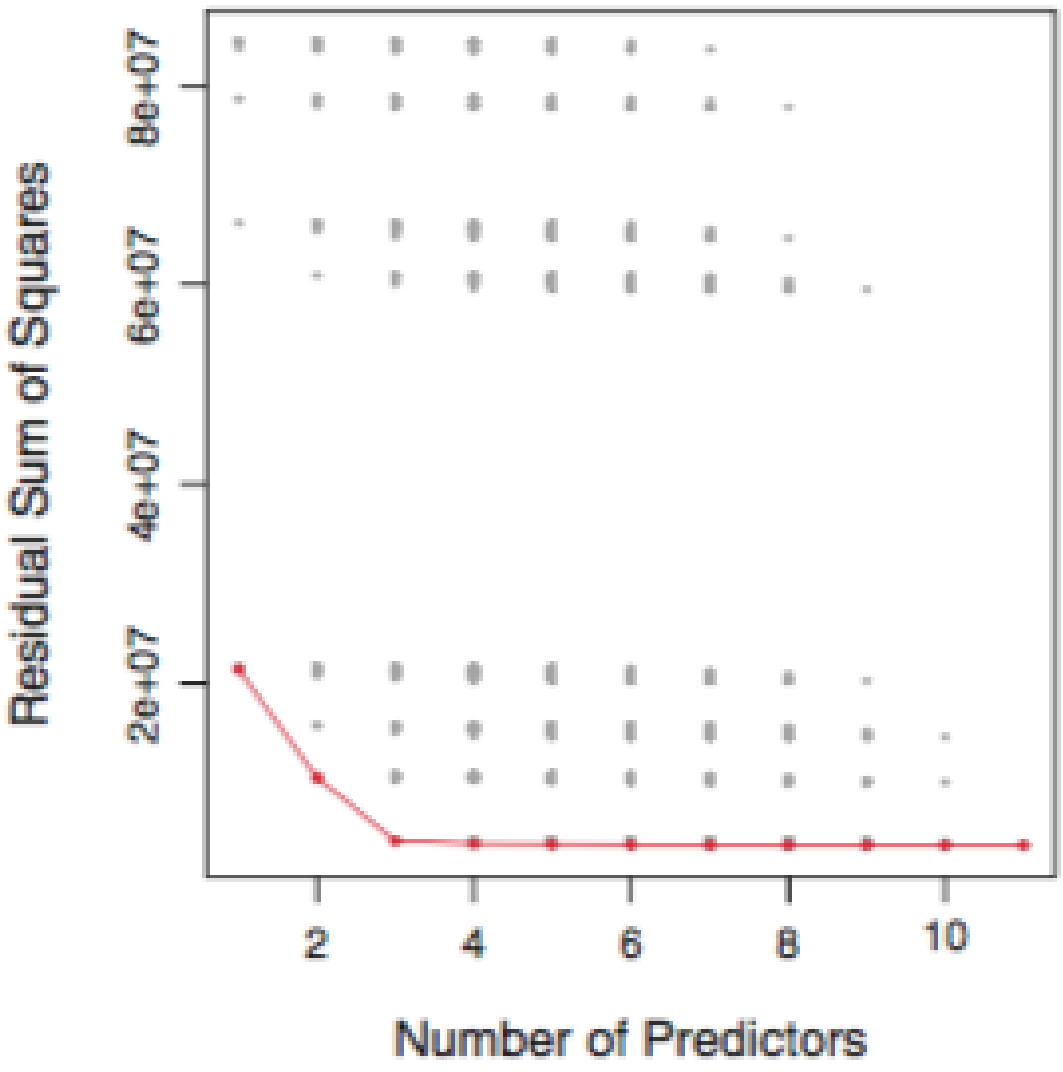
Trade-off: complexity vs. interpretability

Linear model

$$Y = \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \ldots \beta^{(p)}X^{(p)} + \epsilon$$

- Higher values of p lead to more complex model $\rightarrow$ increases prediction power/accuracy
- Higher values of p make it more difficult to interpret the model
- Ideally, we would like to try out a lot of different models, each containing a different subset of the predictors, then select the best model
- Problem: there are 2^p models that contain subsets of p variables

Best subset selection



Forward stepwise selection

Algorithm 6.2 *Forward stepwise selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p - k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Backward stepwise selection

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Hybrid approach

- Hybrid versions of forward and backward stepwise selection are available
- variables are added to the model sequentially
- after adding each new variable, the method may also remove any variables that no longer provide an improvement in the model fit

Adjusted training errors

- Adjusted R^2
- Mallow's C_p

$$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$$

where $\hat{\sigma}^2$ is an estimate of the variance of the error, d is the number of predictors

- AIC (Akaike information criterion)

$$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2)$$

- BIC (Bayesian information criterion)

$$BIC = \frac{1}{n\hat{\sigma}^2}(RSS + \log(n)\hat{\sigma}^2)$$

sklearn does not support subset selection

sklearn.feature_selection.RFE

class `sklearn.feature_selection.` **`RFE`** (*estimator*, *n_features_to_select=None*, *step=1*, *verbose=0*) [source]

Feature ranking with recursive feature elimination.

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a `coef_` attribute or through a `feature_importances_` attribute. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Shrinkage methods

Settings

$$Y \in \mathbb{R}^{n \times 1}, \quad X \in \mathbb{R}^{n \times (p+1)}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} 1 & | & | & \dots & | \\ \dots & x^{(1)} & x^{(2)} & \dots & x^{(p)} \\ 1 & | & | & \dots & | \end{bmatrix}$$

Linear model: settings

- Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)} + \epsilon$$

- Equivalent to

$$\mathbf{Y} = \mathbf{X}\beta, \qquad \beta = \begin{bmatrix} \beta^{(0)} \\ \beta^{(1)} \\ \ldots \\ \beta^{(p)} \end{bmatrix}$$

- Least squares regression

$$\hat{\beta}^{LS} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2$$

ℓ_0 regularization

- ℓ_0 regularization

$$\hat{\beta}^0 = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{i=1}^{p} \mathbf{1}_{\beta^{(i)} \neq 0}$$

  where $\lambda > 0$ is a parameter
- pay a fixed price λ for including a given variable into the model
- variables that do not significantly contribute to reducing the error are excluded from the model (i.e., $\beta_i = 0$)
- problem: difficult to solve (combinatorial optimization). Cannot be solved efficiently for a large number of variables.

ℓ_2 (Tikhonov) regularization

- Ridge regression/ Tikhonov regularization

$$\hat{\beta}^{RIDGE} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} [\beta^{(j)}]^2$$

  where $\lambda > 0$ is a parameter
- shrinks the coefficients by imposing a penalty on their size
- penalty is a smooth function.
- easy to solve (solution can be written in closed form)
- can be used to regularize a rank deficient problem ($n < p$)

ℓ_2 (Tikhonov) regularization

$$\frac{\partial \left(\|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \|\beta\|^2 \right)}{\partial \beta} = 2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\beta) + 2\lambda\beta$$

- The critical point satisfies

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\beta = \mathbf{X}^T\mathbf{Y}$$

- Note: $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ is positive definite, and thus invertible
- Thus

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

ℓ_2 (Tikhonov) regularization

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

- When $\lambda > 0$, the estimator is defined even when $n < p$
- When $\lambda = 0$ and $n > p$, we recover the usual least squares solution

The Lasso

Lasso

- The Lasso (Least Absolute Shrinkage and Selection Operator)

$$\hat{\beta}^{lasso} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta^{(j)}|$$

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero
- However, the ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when λ is sufficiently large
- the lasso performs variable selection $\rightarrow$ models are easier to interpret

Lasso: alternative form

Alternative form of lasso (using the Lagrangian and min-max argument)

$$\min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2$$

$$\text{subject to } \sum_{j=1}^{p} |\beta^{(j)}| \leq s$$

Lasso: alternative form

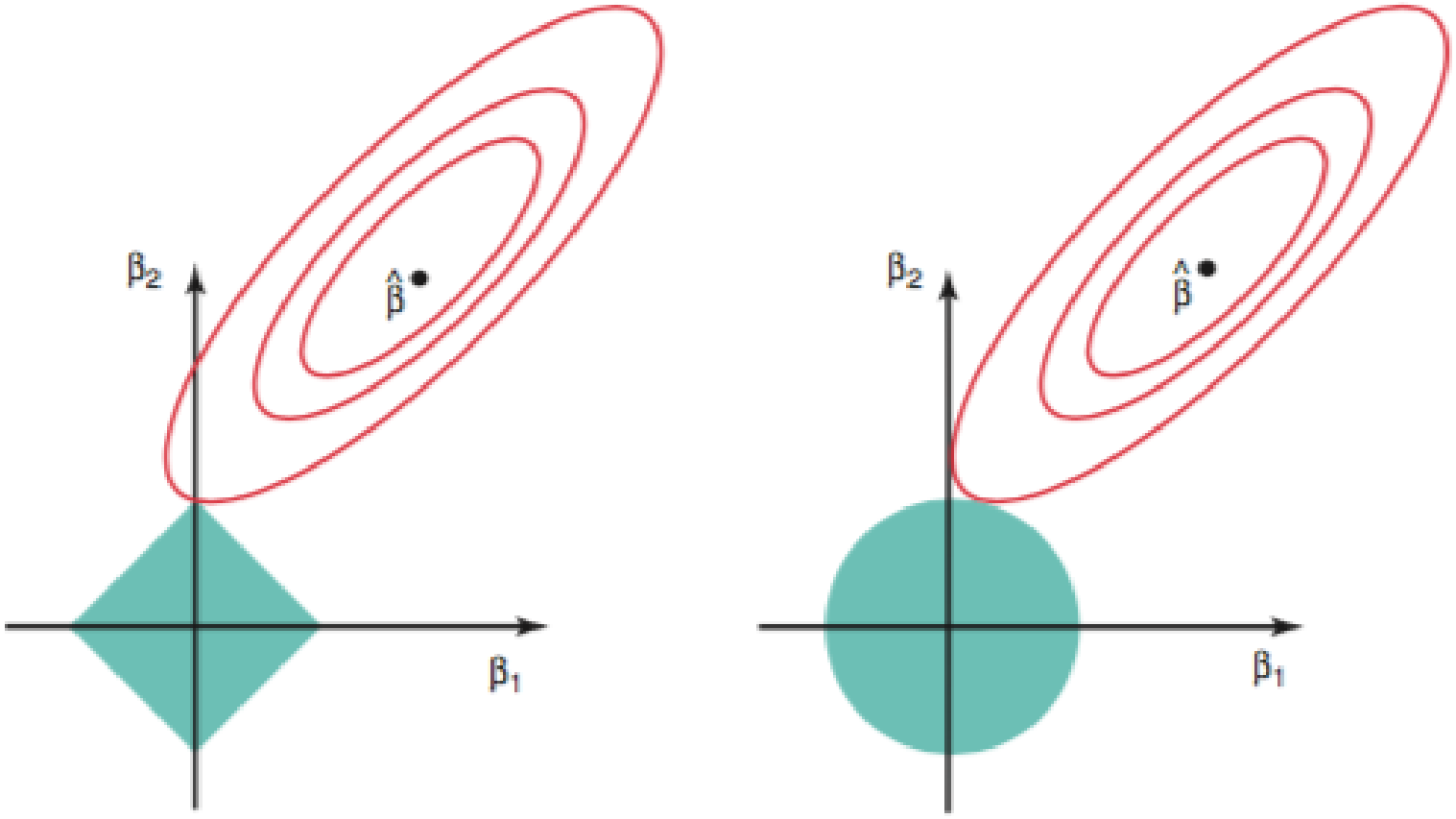


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Lasso

- The Lasso:

$$\hat{\beta}^{lasso} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta^{(j)}|$$

- More "global" approach to selecting variables compared to previously discussed greedy approaches
- Can be seen as a convex relaxation of the $\hat{\beta}^0$ problem
- No closed form solution, but can solved efficiently using convex optimization methods.
- Performs well in practice
- Very popular. Active area of research

Other shrinkage methods

- ℓ_q regularization ($q \geq 0$):

$$\hat{\beta} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} [\beta^{(j)}]^q$$

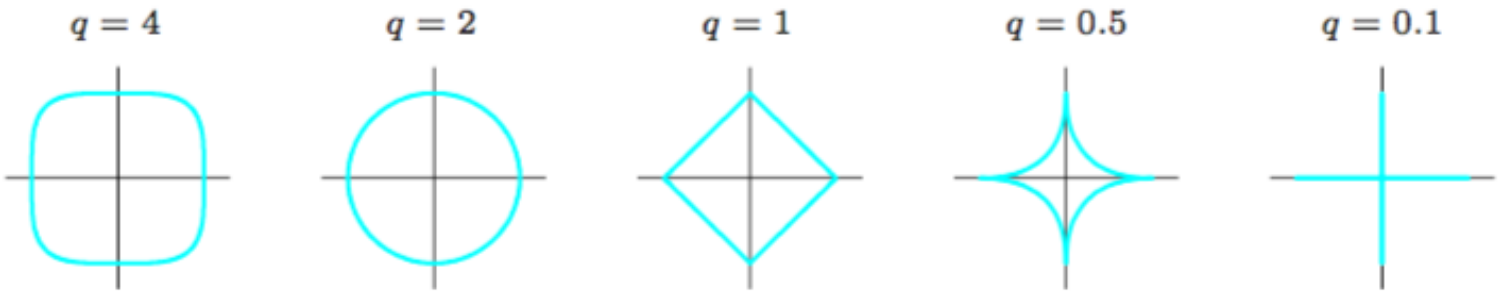


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

Other shrinkage methods

- Elastic net

$$\lambda \sum_{j=1}^{p} \alpha[\beta^{(j)}]^2 + (1-\alpha)|\beta^{(j)}|$$

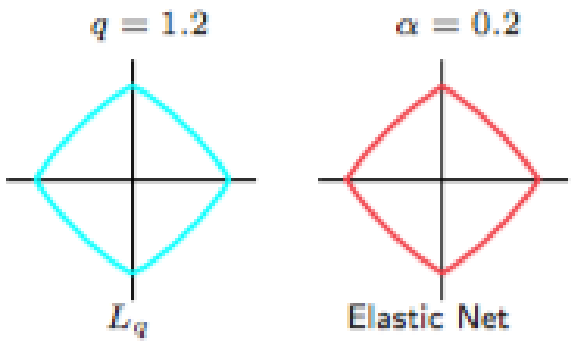


FIGURE 3.13. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for* $q = 1.2$ *(left plot), and the elastic-net penalty* $\sum_j (\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$ *for* $\alpha = 0.2$ *(right plot). Although visually very similar, the elastic-net has sharp (non-differentiable) corners, while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

Lasso: alternative form

Alternative form of lasso (using the Lagrangian and min-max argument)

$$\min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2$$

$$\text{subject to } \sum_{j=1}^{p} |\beta^{(j)}| \leq s$$

Lasso: alternative form

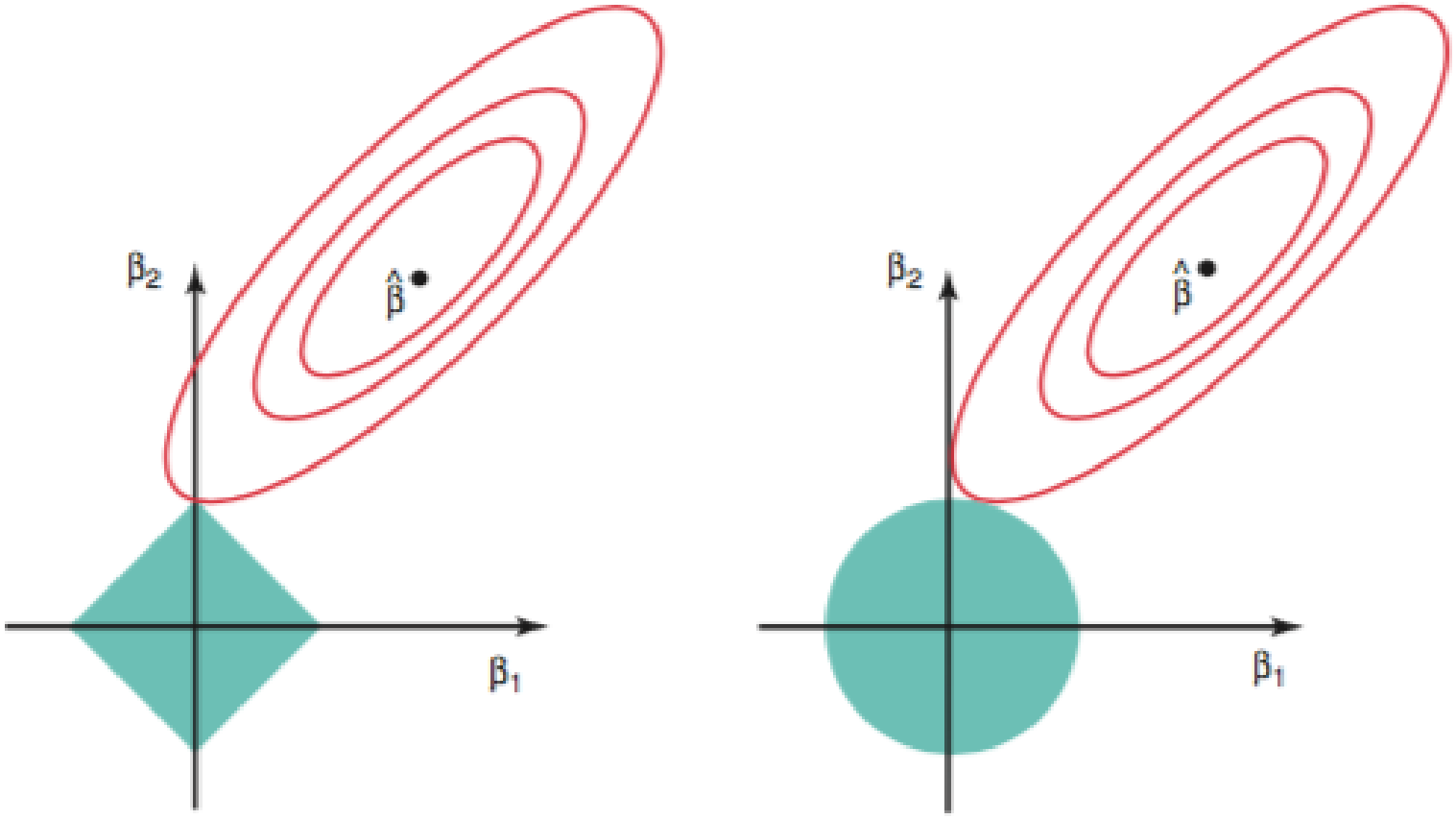


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Linear model: settings

- Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)} + \epsilon$$

- Equivalent to

$$\mathbf{Y} = \mathbf{X}\beta, \qquad \beta = \begin{bmatrix} \beta^{(0)} \\ \beta^{(1)} \\ \ldots \\ \beta^{(p)} \end{bmatrix}$$

Trade-off: complexity vs. interpretability

Linear model

$$Y = \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \ldots \beta^{(p)}X^{(p)} + \epsilon$$

- Higher values of p lead to more complex model $\rightarrow$ increases prediction power/accuracy
- Higher values of p make it more difficult to interpret the model

Regularization

- ℓ_0 regularization

$$\hat{\beta}^0 = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{i=1}^{p} \mathbf{1}_{\beta^{(i)} \neq 0}$$

- Ridge regression/Tikhonov regularization

$$\hat{\beta}^{RIDGE} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} [\beta^{(j)}]^2$$

- Lasso

$$\hat{\beta}^{lasso} = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta^{(j)}|$$

Choosing parameters: cross-validation

- ℓ_0, ridge, lasso have regularization parameters λ
- We obtain a family of estimators as we vary the parameter(s)
- optimal parameters needs to be chosen in a principled way
- cross-validation is a popular approach for rigorously choosing parameters.

K-fold cross-validation

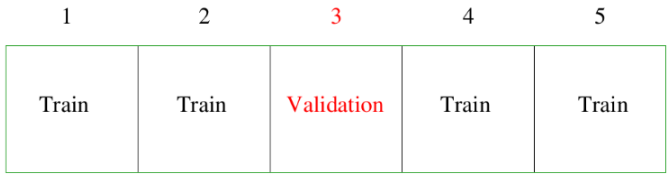
K-fold cross-validation:

Split data into K equal (or almost equal) parts/folds at random.
for each parameter λ_i **do**
  for $j = 1, \ldots, K$ **do**
    Fit model on data with fold j removed.
    Test model on remaining fold $\rightarrow$ j-th test error.
  end for
  Compute average test errors for parameter λ_i.
end for
Pick parameter with smallest average error.

K-fold cross-validation

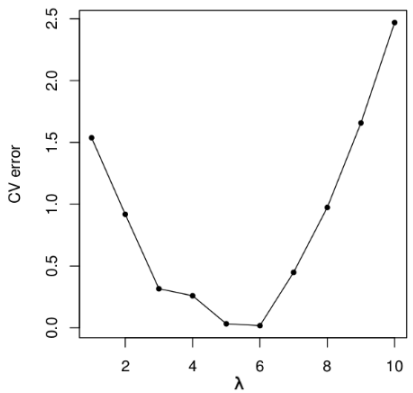
More precisely,

- Split data into K folds $F_1, \ldots, F_K$.



- Let $L(y, \hat{y})$ be a *loss function*. For example, $L(y, \hat{y}) = \|y - \hat{y}\|_2^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$.
- Let $f_\lambda^{-k}(\mathbf{x})$ be the model fitted on all, but the k-th fold.
- Let

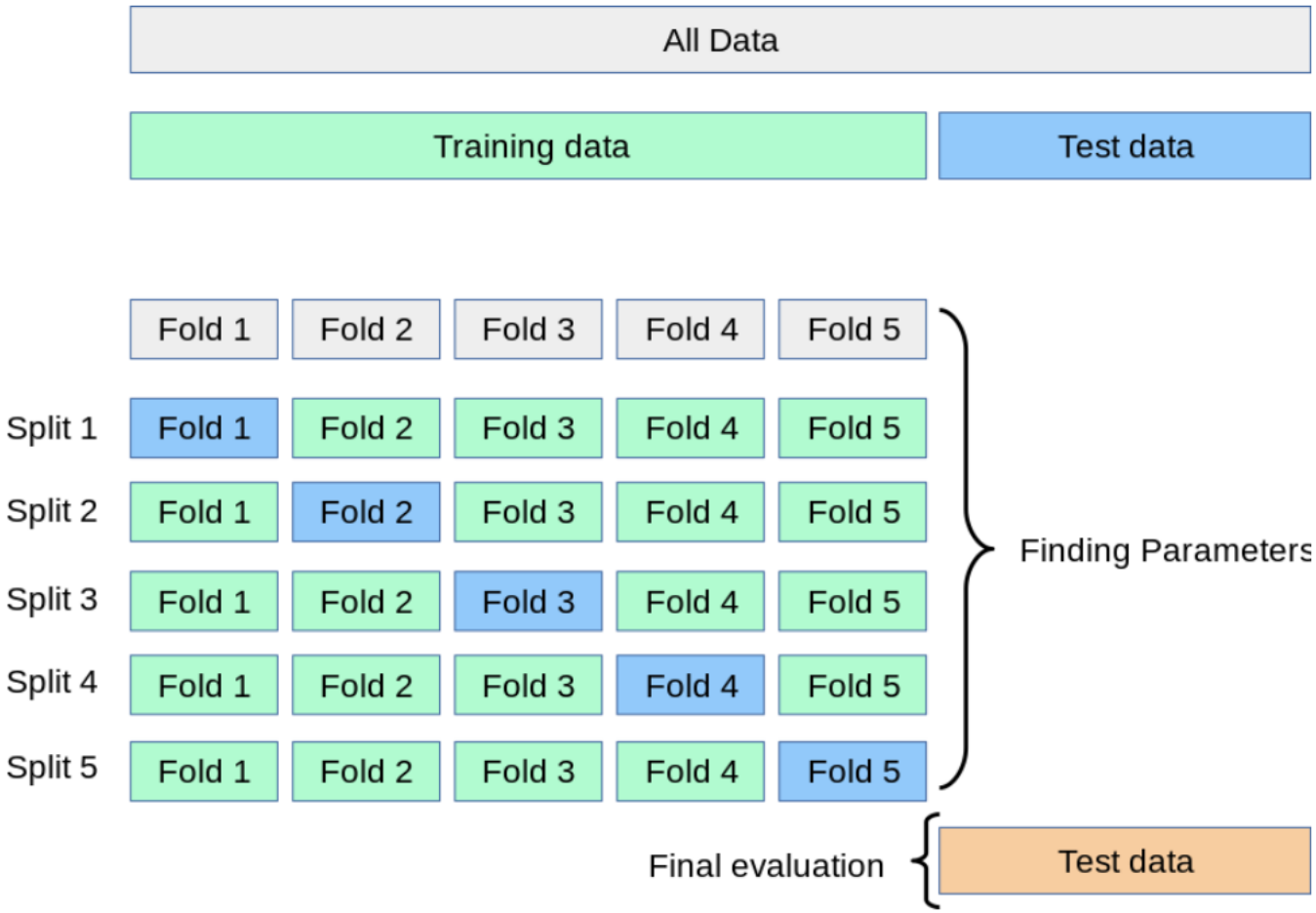
$$CV(\lambda) := \frac{1}{n} \sum_{k=1}^{n} \sum_{i \in F_k} L(y_i, f_\lambda^{-i}(\mathbf{x}_i))$$



- Pick λ among a *relevant* set of parameters

$$\hat{\lambda} = \underset{\lambda \in \{\lambda_1, \ldots, \lambda_m\}}{\operatorname{argmin}} \; CV(\lambda)$$

K-fold cross-validation



Demo: Cross-validation with Lasso