Predictive Modeling

Regularization Techniques and Feature Importance Analysis

Mirko Birbaumer

HSLU T&A

1. Ridge Regression
2. The Lasso
3. Boosting
4. Feature Importance Analysis

Regularization Methods - Ridge Regression and the Lasso

- **Subset selection methods**: involve least squares to fit a linear model that contains a **subset** of the predictor variables
- Alternative: we 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
- Shrinking the coefficient estimates can significantly **reduce their variance**
- Two best-known techniques for shrinking the regression coefficients towards zero are : **ridge regression** and the **lasso**

Ridge Regression

- **Least squares fitting** procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

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

- **Ridge regression** coefficient estimates are the $\widehat{\beta}^R$ values that minimize

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

where $\lambda \geq 0$ is a **tuning parameter**

Ridge Regression

Ridge Regression trades off two different criteria:

- As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small
- The second term $\lambda \sum_j \beta_j^2$ is called a **shrinkage penalty**
- Shrinkage penalty is small when $\beta_1, \ldots, \beta_p$ are close to zero, and so it has the effect of shrinking the estimates of β_j towards zero.

Ridge Regression: Tuning Parameter

- **Tuning parameter** λ serves to control the relative impact of these two terms on the regression coefficient estimates
  - $\lambda = 0$: the penalty term has no effect, and ridge regression will produce the least squares estimates
  - $\lambda \to \infty$: the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero
  - Please see examples `1.1` and `1.2` of the `Regularization Techniques` chapter

Ridge Regression: Disadvantages

- Unlike best subset, forward stepwise, and backward stepwise selection, which will generally select models that involve just a subset of the variables, ridge regression will include **all** p predictor variables in the final model

- Penalty $\lambda \sum_{j=1}^{p} \beta_j^2$ will shrink all of the coefficients **towards** zero, but it will not set any of them exactly to zero (unless $\lambda = \infty$)

- Challenge in **model interpretation** in settings in which the number of variables p is quite large

The Lasso

- **The lasso** overcomes the disadvantages of ridge regression
- The lasso coefficients, $\widehat{\beta}_\lambda^L$, minimize the quantity

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

- Only difference between ridge regression and the lasso is that the term β_j^2 in the ridge regression penalty has been replaced by $|\beta_j|$ in the lasso penalty

The Lasso

- 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, much like best subset selection, the lasso performs **variable selection** $\rightarrow$ **increased interpretability**

- Please check examples `2.2` and `2.3` of the `Regularization Techniques` chapter

The Lasso

- One can show that the lasso and ridge regression coefficient estimates solve the problems:
  1. $$\underset{\beta}{\text{minimize}} \quad \left\{ \sum_{i=1}^{p} \left(y_i - \beta_0 \sum_{j=1}^{p} \beta_j x_{ij} \right) \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| < s$$
  2. $$\underset{\beta}{\text{minimize}} \quad \left\{ \sum_{i=1}^{p} \left(y_i - \beta_0 \sum_{j=1}^{p} \beta_j x_{ij} \right) \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 < s$$

- **Goal**: find set of coefficient estimates that lead to smallest RSS, subject to the constraint that there is a **budget** s for how large $\sum_{i=1}^{p} |\beta_j|$ can be

The Variable Selection Property of the Lasso

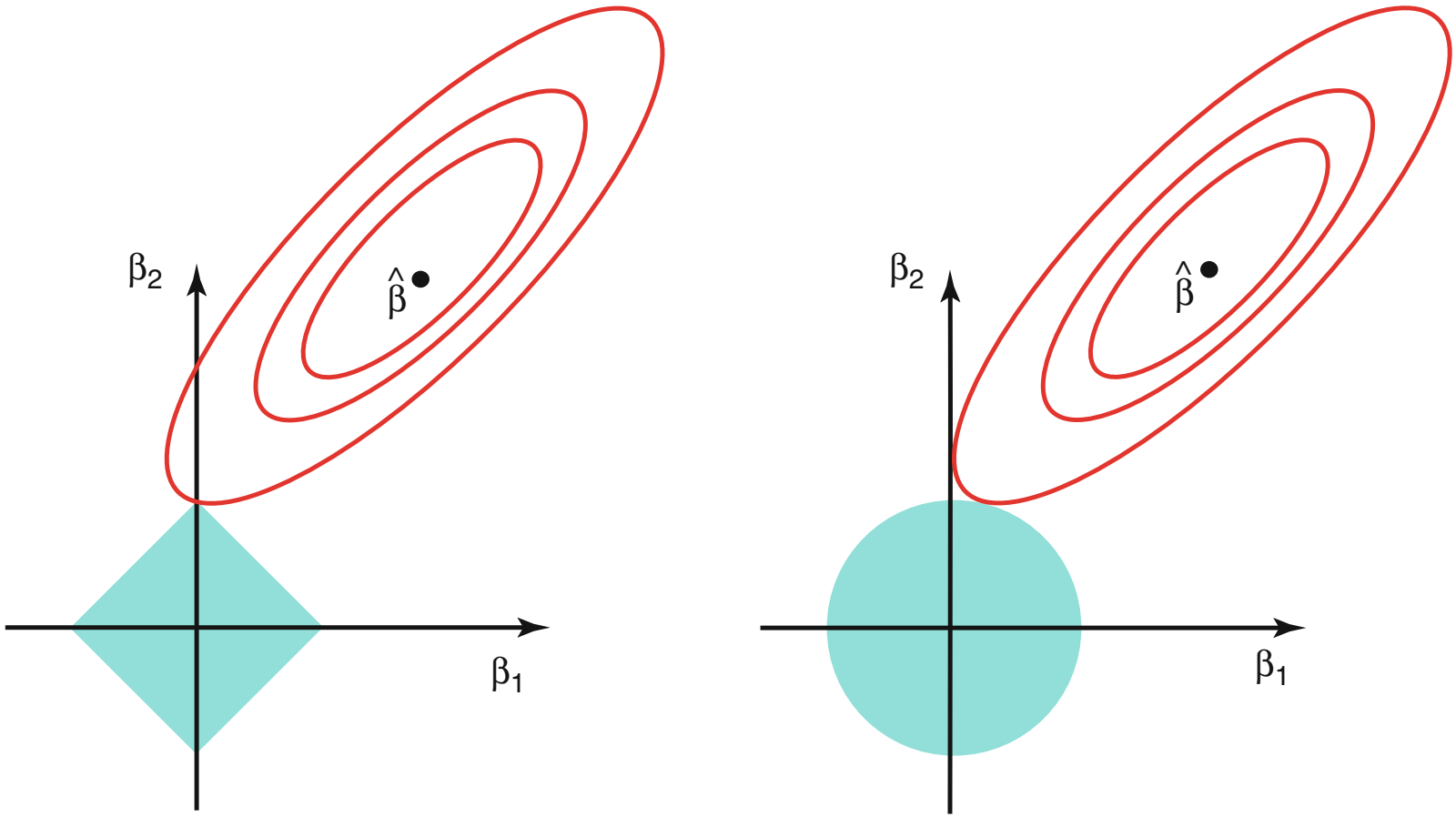

Abbildung: Contours of the error and constraint functions for the lasso (*left*) and ridge regression (*right*). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

Selecting the Tuning Parameter λ

- **Cross-validation** provides a simple way to tackle this problem
- We choose a grid of λ values, and compute the cross-validation error for each value of λ
- We then select the tuning parameter value for which the cross-validation error is smallest
- Please check examples `2.6` and `2.7` of the `Regularization Techniques` chapter

Boosting

- Like bagging, **boosting** is a general approach that can be applied to many statistical learning methods for regression or classification
- **Boosting** for decision trees: the trees are grown sequentially: each tree is grown using information from previously grown trees
- Each tree is fit on a modified version of the original data set

Boosting for Regression Trees

1. Set $\widehat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, ..., B$, repeat:
   1. Fit a tree $\widehat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r)
   2. Update $\widehat{f}$ by adding in a shrunken version of the new tree:

$$\widehat{f}(x) \leftarrow \widehat{f}(x) + \lambda \widehat{f}^b(x)$$

   3. Update the residuals,

$$r_i \leftarrow r_i - \lambda \widehat{f}^b(x_i)$$

3. Output the boosted model,

$$\widehat{f}(x) = \sum_{b=1}^{B} \lambda \widehat{f}^b(x)$$

Boosting

- Given the current model, we fit a decision tree to the **residuals** from the model - instead of outcome Y, tree is fit to **current residuals**
- We then **add** this new decision tree into the fitted function in order to update the residuals
- Each of these trees can be rather small, with just a few terminal nodes - determined by parameter d in the algorithm. By fitting small trees to the residuals, we slowly improve $\widehat{f}$ in areas where it does not perform well
- **Shrinkage parameter** λ slows the process down even further, allowing more and different shaped trees to attack the residuals
- Please check examples `4.1` and `4.2` of the `Regularization Techniques` chapter

Feature Importance Analysis: Interpretable Models

- *Examples of interpretable models:* Best subset selection, Decision Trees, and the Lasso
- *Examples of model-specific interpretation methods:* Mean Decrease of Gini index in random forests and boosting
- *Model-agnostic feature importance methods:* If we separate the feature importance measures from the machine learning model

Feature Importance Analysis: Interpretable Models

- *Model flexibility:* The interpretation method can work with any machine learning model, such as random forests and deep neural networks.
- *Explanation flexibility:* You are not limited to a certain form of explanation. In some cases it might be useful to have a linear formula, in other cases a graphic with feature importances.
- *Representation flexibility:* The explanation system should be able to use a different feature representation as the model being explained.

Model-agnostic feature importance: Permutation feature importance

- A feature is **important** if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction

- A feature is **unimportant** if shuffling its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction.

Permutation feature importance

Permutation feature importance

Input: Trained model $\widehat{f}$, measurements x_i in rows of data matrix X, response variable y, error measure $L(y, \widehat{f}(X))$ (e.g. mean squared error)

1. Estimate the original model error $L(y, \widehat{f}(X))$
2. For each feature $j \in \{1, 2, \ldots, p\}$ do:
   1. Generate data matrix X_{perm} by permuting feature j in the data matrix X. This breaks the association between feature j and the true outcome y.
   2. Estimate error $e_{\text{perm}} = L(y, \widehat{f}(X_{\text{perm}}))$ based on the predictions of the permuted data.
   3. Calculate permutation feature importance as quotient $FI_j = e_{\text{perm}}/e_{\text{orig}}$ or as difference $FI_j = e_{\text{perm}} - e_{\text{orig}}$
3. Sort features j by descending FI_j

Should We Compute Feature Importance on Training or Test Data?

- Answering the question about training or test data touches the fundamental question of what feature importance is.

- See `Example 5.1` in the `Regularization Techniques and Feature Importance Analysis` chapter

Should We Compute Feature Importance on Training or Test Data?

- Answering the question about training or test data touches the fundamental question of what feature importance is.

- See `Example 5.1` in the `Regularization Techniques and Feature Importance Analysis` chapter

Should We Compute Feature Importance on Training or Test Data? The Case for Test Data

- General principle in machine learning: If we measure the model error (or performance) on the same data on which the model was trained, the measurement is usually too optimistic

- The feature importance based on training data makes us mistakenly believe that features are important for the predictions, when in reality the model was just overfitting and the features were not important at all

Should We Compute Feature Importance on Training or Test Data? The Case for Training Data

- In practice, we want to use all our data to train our model to get the best possible model in the end.
- We are interested in how much the trained model's predictions are influenced by a feature.

Partial Dependence Plots

- The *partial dependence plot* (short PDP) shows the marginal effect one or two features have on the predicted outcome of a machine learning model.
- A partial dependence plot can show whether the relationship between the response variable and a predictor variable is linear, monotonic or more complex.
- *Example:* When applied to a linear regression model, partial dependence plots always show a linear relationship.
- *Motivation:* a flat PDP indicates that the predictor variable is not important
- See **`Example 5.2`** in the **`Regularization Techniques and Feature Importance Analysis`** chapter