BUSS6002 Data Science in Business

Week 7: Model Evaluation and Selection

Dr. Jie Yin
Discipline of Business Analytics
University of Sydney Business School

Table of contents

- Generalisable Model
- Model Selection
- The Bias-Variance Decomposition
- Cross-Validation
- Regularisation

Introduction

Model Selection: estimate the performance of different models in order to choose the (approximate) best one. We select the model that is estimated to have the best predictive ability.

Model Evaluation: after choosing the "right" model, estimate its test error (generalisation error) on new data.

Generalisable Model

Recall

- **Supervised learning (SL) problem**: Given a training dataset $(x_i, y_i)_{i=1}^n$, we would like to 1) accurately predict unseen test cases, 2) understand which inputs affect the outcome and how, and 3) assess the quality of our predictions.

  Two SL methods considered:

  - **Linear regression**

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon$$

  - **Logistic regression** for classification

$$P(Y = 1|X) = \frac{\exp(\beta_0 + \sum_{i=1}^{p} \beta_i X_i)}{1 + \exp(\beta_0 + \sum_{i=1}^{p} \beta_i X_i)}.$$

Fundamental problem of supervised learning

- How to select the ideal regression/classification model?
- Predication accuracy vs interpretability
  - Interpretability can be an important consideration in addition to predictive accuracy. Highly flexible models tend to be less interpretable than simpler methods.
  - Increasing in flexibility and decreasing in interpretability
- Good fit vs overfit or underfit
  - How do we know when the fit is just right?
- Parsimony vs black-box.
  - A parsimonious model uncovers the relationship between X and Y.
  - A black box aims to yields high accuracy, regardless of the complexity of the model.

Assessing model accuracy

- Suppose we fit a model $\hat{f}(x)$ to some training data $Tr = \{x_i, y_i\}_1^N$, and we wish to see how well it performs. We could compute the average squared prediction error over Tr:

$$MSE_{Tr} = Ave_{i \in Tr}[y_i - \hat{f}(x_i)]^2.$$

- This is the RSS (a.k.a loss function) which we minimise to obtain the Least Squares fit – see Week 5.

Overfitting and Underfitting

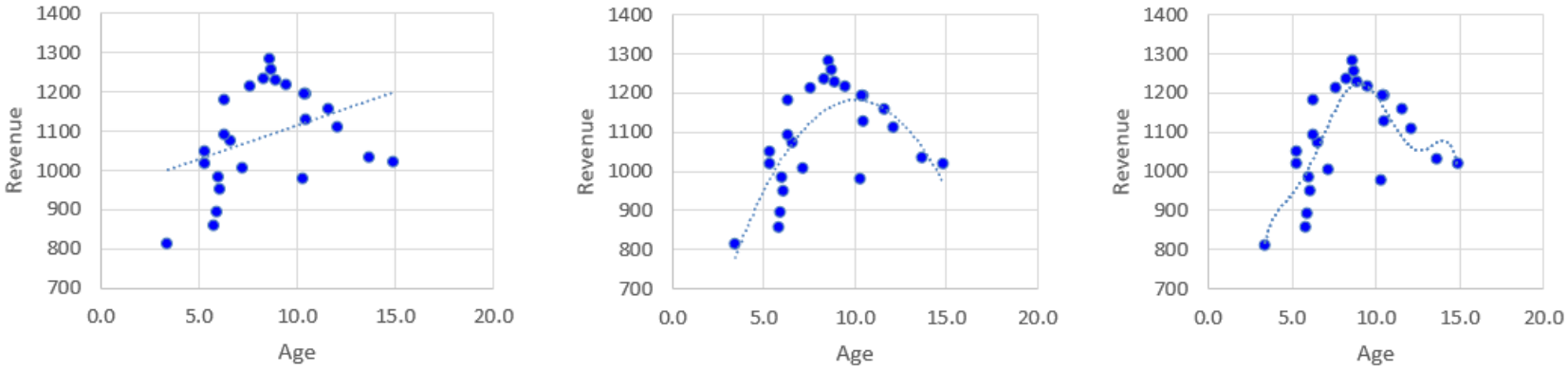


Figure: Underfitting, optimal and overfitting models

- Underfitting model: fitted values do not change much but are often off
- Overfitting model: fitted values change a lot but are less off

Why overfitting is not good?

- Low training error, but high generalization error
- Overreacts to minor fluctuations in training data
- Poor predictive performance on unseen new data

Approaches to address overfitting:

- Drop some features
  - Manually select features to keep (feature selection)
  - Model selection algorithms
- Regularisation
  - Keep all features, but reduce the magnitude/values of parameters.

The bias-variance trade-off

- Increasing model complexity brings higher flexibility and therefore lower bias. But this comes at a cost of higher variance $\Longrightarrow$ Overfitting.
- Decreasing model complexity leads to lower variance. However, simpler models may not be sufficiently flexible to capture the underlying patterns in the data, leading to higher bias $\Longrightarrow$ Underfitting.

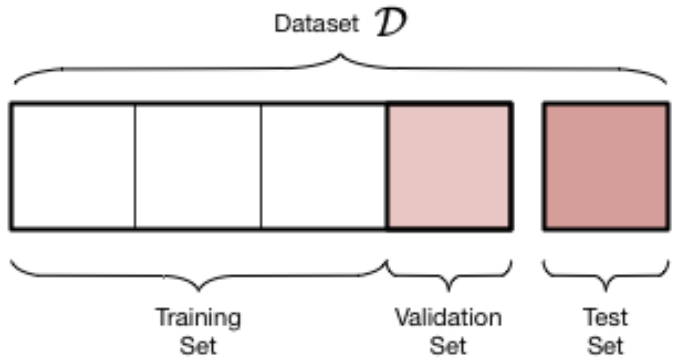
Examples

Linear regression. Adding predictors increases model complexity. Least squares estimates have high variability when the number of inputs is large, but excluding relevant predictors leads to bias.

Model Selection

Training, validation, and test split

In the **validation set** approach, we **randomly** split the training data into a training set, a validation set and a test set.



The validation set is used for selecting the optimal model complexity.

Typically, we use 50-80% of the data for the training set.

Training, validation, and test split

- **Training set**: for exploratory data analysis, model building, model estimation, etc.
- **Validation set**: for appropriate model selection.
- **Test set**: for model evaluation.

Training, validation, and test split

1. Estimate different models on the training data.
2. Predict the observations in the validation set.
3. Select the model with the best performance on the validation set.
4. **Re-estimate** the selected model by combining the training and validation sets.
5. Predict the test data with the selected model.

Validation set

The validation set approach has limitations when the size of the training data is not large. The model may not have enough data to train on, and there may not be enough cases in the validation set to reliably estimate generalisation performance.

We turn instead to cross validation that will be introduced in later sections.

The Bias-Variance Decomposition

Expected prediction error

- Suppose we have a model $\widehat{f}(X)$ fitted to the training data D_{Tr} and the true model with noise is $Y = f(X) + \varepsilon$, where we assume that $E(\varepsilon) = 0$ and $\text{Var}(\varepsilon) = \sigma^2$.
- We can define the **expected prediction error** for a new test observation $X = \boldsymbol{x}_0$ as

$$E\left[(Y_0 - \widehat{f}(\boldsymbol{x}_0)^2\right]$$

  where $Y_0 = f(\boldsymbol{x}_0) + \varepsilon_0$.
- The expectation is over ε_0 and the training sample, i.e., over the sampling distribution of $\widehat{f}(\cdot)$.

Expected prediction error decomposition

We can write expected generalization error as follows

$$
\begin{aligned}
E\left[(Y_0 - \widehat{f}(x_0)\right]^2 &= E\left[\left(f(\boldsymbol{x}_0) + \varepsilon_0 - \widehat{f}(\boldsymbol{x}_0)\right)^2\right] \\
&= \sigma^2 + E\left[\left(f(\boldsymbol{x}_0) - \widehat{f}(\boldsymbol{x}_0)\right)^2\right] \\
&= \text{Irreducible error} + \text{Reducible error}
\end{aligned}
$$

- The first term is the variance of the response around its true mean $f(\boldsymbol{x}_0)$. This source of error cannot be avoided and it puts an upper bound on the accuracy of the prediction.
- In selecting a model, our focus is the reducible error: we would like to minimise the estimation error $E\left[\left(f(\boldsymbol{x}_0) - \widehat{f}(\boldsymbol{x}_0)\right)^2\right]$.

The bias-variance trade-off

$$\begin{aligned}\text{Estimation Error} &= E\left[\left(f(\boldsymbol{x}_0) - \widehat{f}(\boldsymbol{x}_0)\right)^2\right] \\ &= \left[E[\widehat{f}(\boldsymbol{x}_0)] - f(\boldsymbol{x}_0))\right]^2 + E\left(\left[\widehat{f}(\boldsymbol{x}_0) - E[f(\boldsymbol{x}_0)\right]^2\right) \\ &= \text{Bias}^2(\widehat{f}(\boldsymbol{x}_0)) + \text{Var}(\widehat{f}(\boldsymbol{x}_0))\end{aligned}$$

- Bias reflects the error between the average of our estimate differs from the true function.
- Variance reflects the expected squared deviation of $\widehat{f}(\boldsymbol{x}_0)$ around its mean.

The bias-variance trade-off

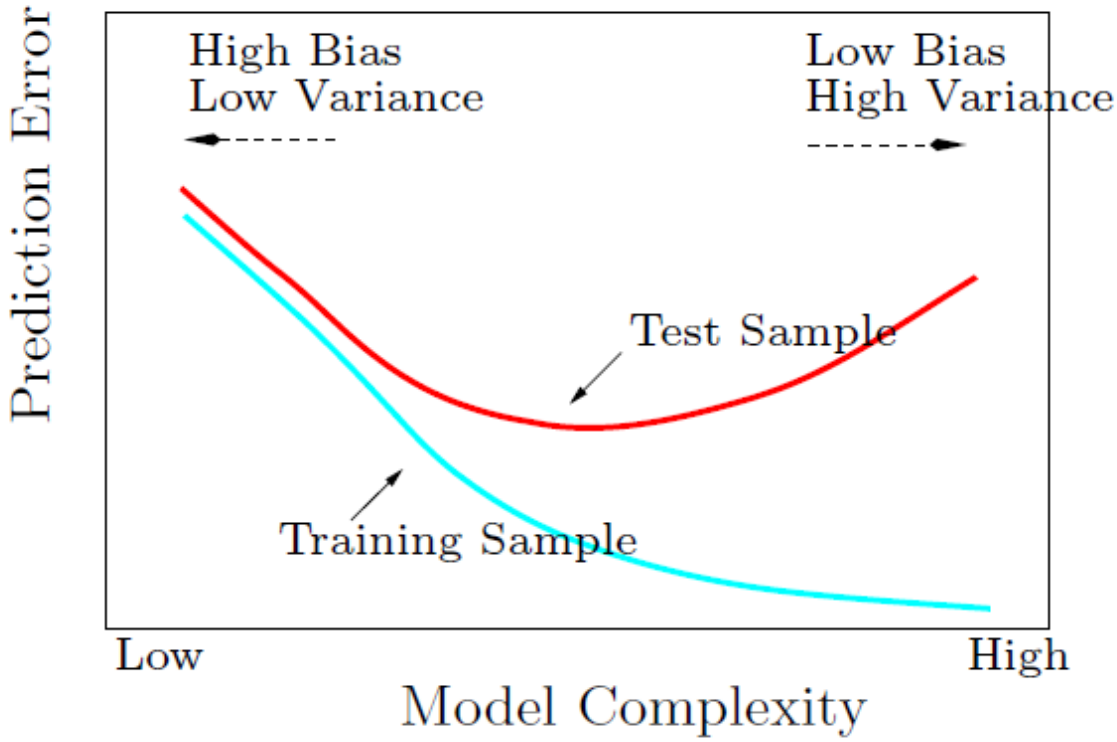
$$\text{Estimation Error} = \text{Bias}^2(\widehat{f}(\boldsymbol{x}_0)) + \text{Var}(\widehat{f}(\boldsymbol{x}_0))$$

Such a decomposition is known as the **bias-variance tradeoff**

- As model becomes more complex (more terms included), local complex structure can be picked up
- But coefficient estimates suffer from high variance as more terms are included in the model

Hence, we would like to find the optimal model complexity that minimises the expected loss.

Learning curve



Increasing model complexity will always reduce the training error, but there is an optimal level of complexity that minimises the test error.

Diagnosing learning curve

Suppose your training error is low, while validation/test error is high.

- Underfitting or overfitting problem?

Suppose your training error is high, while validation/test error is also high.

- Underfitting or overfitting problem?

Diagnosing learning curve

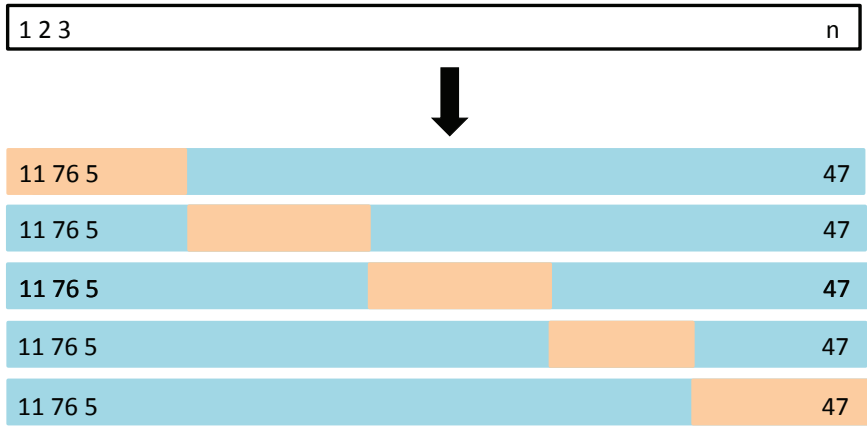
- **Underfitting**: training error is high, validation error is slightly $>$ training error;
- **Overfitting**: training error is low, validation error is significantly $>$ training error.

Cross Validation

Cross validation

Cross validation methods are based on multiple random training/validation set splits. Unlike in the validation set approach, each observation gets a turn at being predicted.

K-fold cross-validation



The idea of K-fold cross validation is simple:

1. We randomly split the training sample into K **folds** of roughly equal size.
2. For each fold $k \in \{1, \ldots, K\}$, we estimate the model on all other folds combined, and use k as the validation set.
3. The cross validation error is the average error across the K validation sets.

K-fold cross-validation

5-fold and 10-fold CV. $K = 5$ or $K = 10$ folds are common choices for cross validation.

Leave one out cross validation. If we set $K = N$, this is called leave one out cross validation, or **LOOCV**. For each observation i, we train the model on all other observations, and predict i.

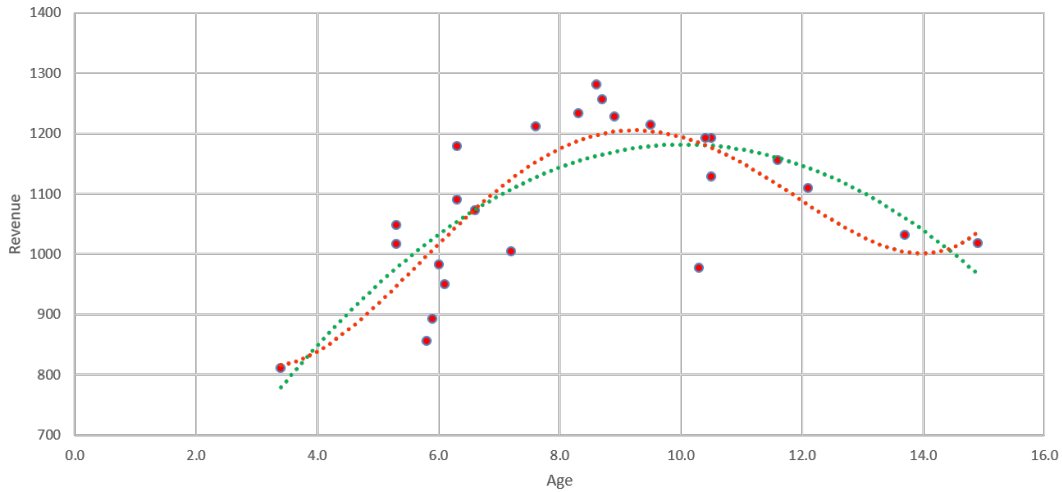
LOOCV vs K-fold cross-validation

LOOCV. Approximately unbiased estimator of the expected prediction error. However, it can have high variance in some settings (since the training sets are very similar for every prediction) and a high computational cost.

K-fold. Lower computational cost and may have lower variance. However, it is subject to bias since the training sets are smaller than N.

Regularisation

Regularisation intuition



$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 \Rightarrow \text{Just right}$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4 \Rightarrow \text{Overfitting}$$

How do we penalize parameters β_3 and β_4 to be close to 0, so that the model is approximately $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$?

Regularisation intuition

- Consider the Penalized Residual Sum of Squares (PRSS) :

$$PRSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda\beta_3^2 + \lambda\beta_4^2.$$

- If λ is large, e.g., 10,000, then parameters β_3 and β_4 are heavily penalized, e.g., close to 0, since we minimize the loss function.
- Note that the sign of β_3 and β_4 does not matter, negative and positive values of equal magnitude are penalized equally for both parameters.

Regularisation methods

Regularisation (also called **shrinkage**) methods fit a model involving all predictors, but shrink the coefficients towards zero. Depending on the type of shrinkage, some estimated coefficients may be zero, in which case the method also performs variable selection.

Regularisation for linear regression

Regularisation, or shrinkage, methods for linear regression follow the general framework of regularised empirical risk minimisation.

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})) + \lambda C(\boldsymbol{\theta}) \right]$$

We will use the squared error loss, and the complexity function will involve a norm of the regression coefficient vector $\boldsymbol{\beta}$.

Ridge regression

- Least Squares method estimates $\beta = (\beta_0, \beta_1, \cdots, \beta_p)$ using the values that minimise

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

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

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

Ridge regression

The ridge regression method solves the following problem:

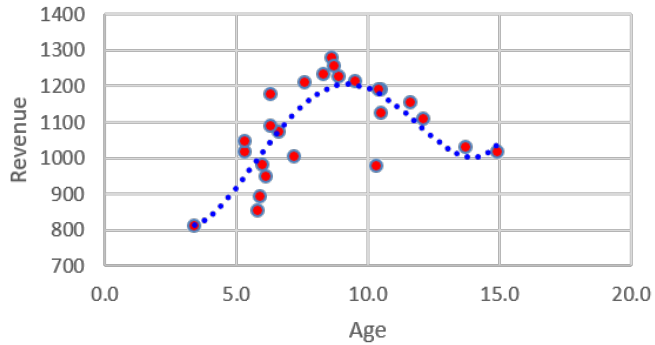
$$\widehat{\boldsymbol{\beta}}_{\text{ridge}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- $\lambda \geq 0$ is a *tuning hyper-parameter* that controls the amount of shrinkage.
- The criterion penalizes $||\boldsymbol{\beta}||^2 = \sum_{j=1}^{p} \beta_j^2$, which has the effect of shrinking the OLS coefficients towards zero. It is also called the squared Euclidean norm (a.k.a. the L_2 norm) of $\boldsymbol{\beta}$.
- This procedure is also referred to as L_2 **regularisation**.

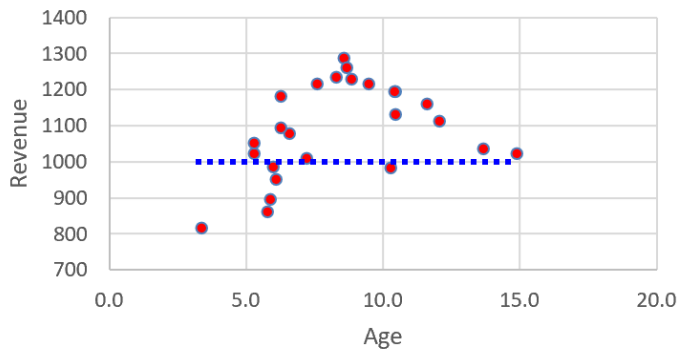
The effect of λ

Ridge regression cannot zero out a specific coefficient. The model either includes all the coefficients in the model, or none of them

- $\lambda = 0 \Rightarrow$ gives the usual OLS solution; greatest complexity, smallest bias, highest variance



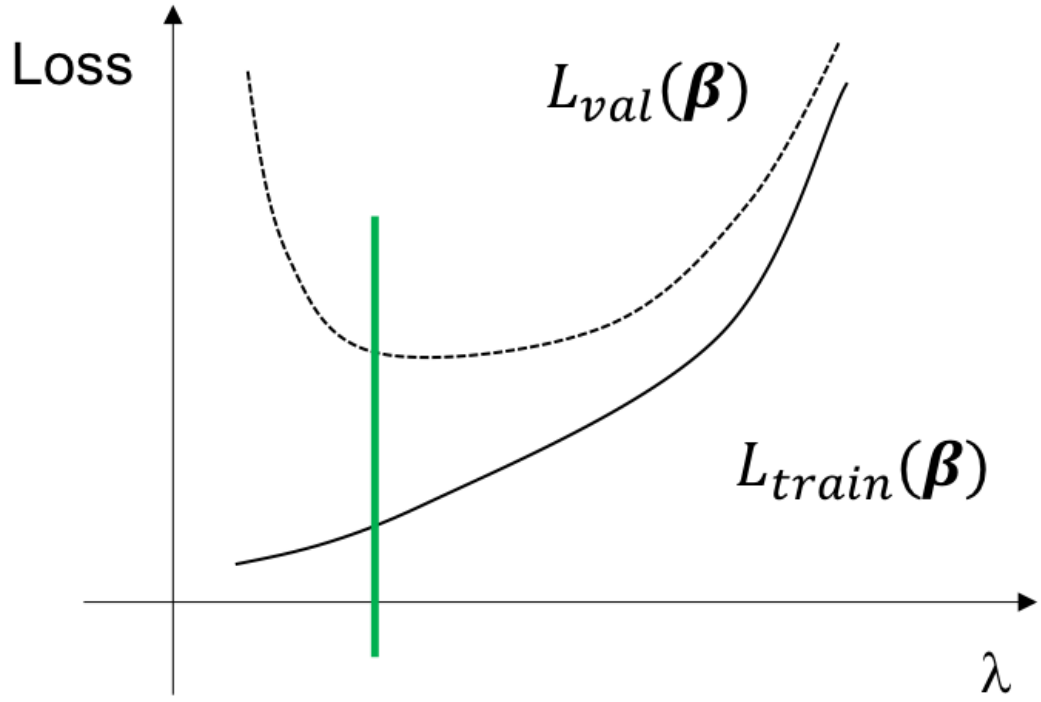
- $\lambda \to \infty \Rightarrow$ approaches a horizontal line through the data; lowest complexity, smallest variance, highest bias.



Learning curve

Tend to **Overfitting**.
High variance.

Tend to **underfitting**.
High bias.



Best Model

Lasso

The Lasso (least absolute shrinkage and selection operator) method solves the following problem:

$$\widehat{\beta}_{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- λ is a tuning hyper parameter.
- The criterion penalizes $\sum_{j=1}^{p} |\beta_j|$, which is L_1 norm of β, denoted by $||\beta||_1$.
- Thus, lasso performs L_1 **regularisation**.

Lasso: shrinkage and variable selection

- **Shrinkage**. As with ridge regression, the lasso shrinks the coefficients towards zero as Ridge. However, the nature of this shrinkage is different, as we discuss further below.
- **Variable selection**. In addition to shrinkage, the lasso also performs variable selection. With λ being sufficiently large, some estimated coefficients will be exactly zero, leading to sparse models, which are easier to interpret. This is a key difference between lasso and ridge.

Ridge vs Lasso

Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero?

Each can be shown to solve a constrained minimisation problem

- Lasso:

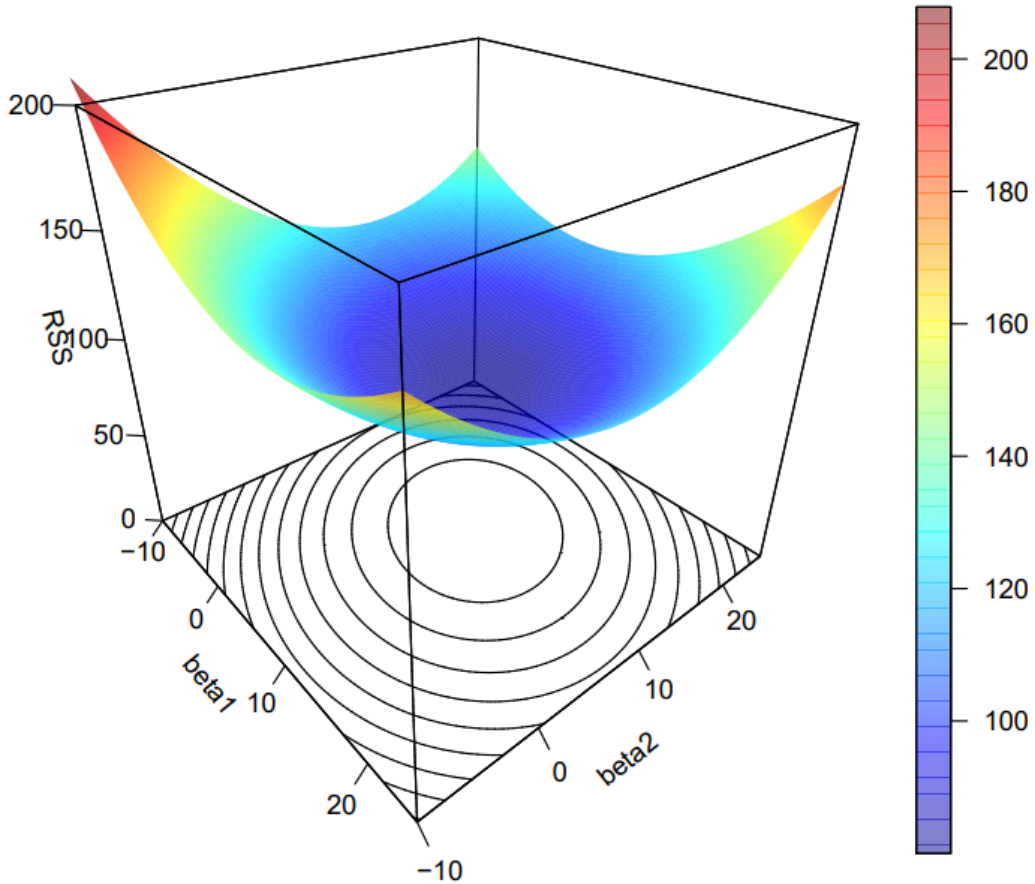
$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq t$$

- Ridge:

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

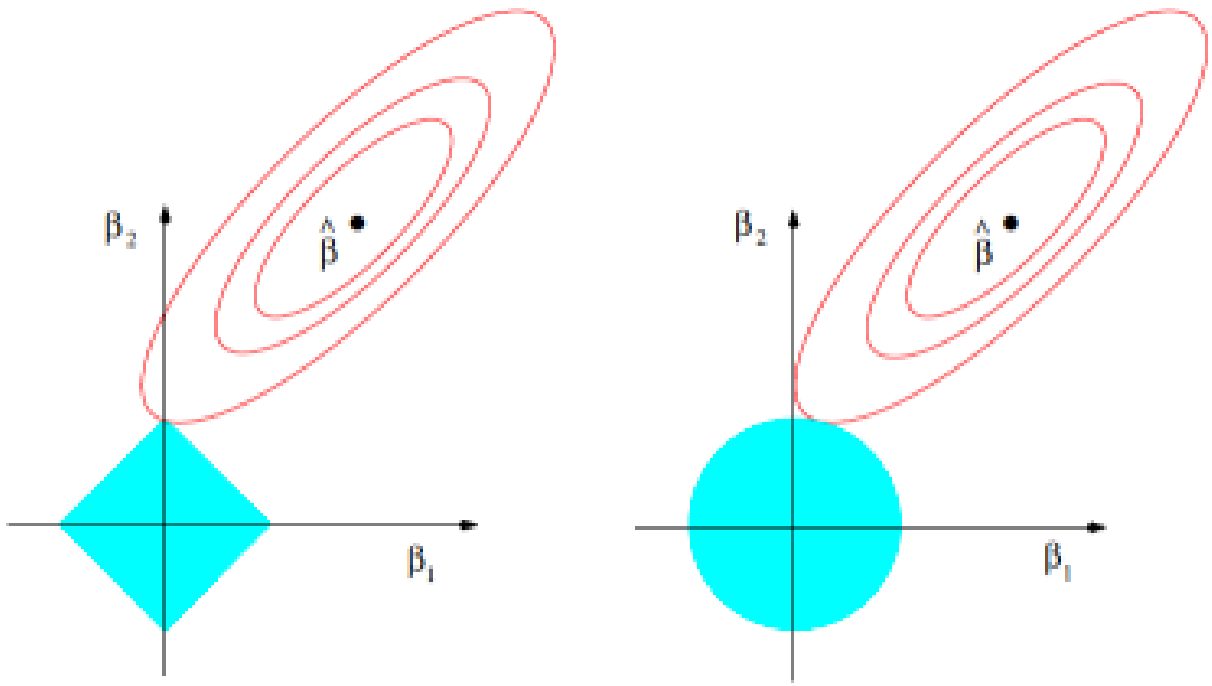
Geometry of Convex Optimization in $\mathbb{R}^2$

- Ridge and Lasso minimises the Least Squares objective.



Ridge vs Lasso

Lasso (left) and Ridge (right) minimise the least squares subject to different constraints.



The contours of the sum of squares function (which we are minimizing) are in red, and the constraint regions are in blue.

Which method to use?

- Neither Ridge regression nor the Lasso universally outperforms the other. The choice of the method should be data driven.
- Generally, the Lasso should perform better when a few predictors have large coefficients, while the remaining predictors zero or small.
- Ridge regression tends to perform better when all predictors have coefficients of similar size.
- The Lasso has better interpretability as it produces sparse solutions (i.e. with some coefficients set to zero).

Elastic Net

Elastic Net is a regularized regression method that linearly combines the penalties of the Lasso and Ridge methods.

$$\widehat{\boldsymbol{\beta}}_{\text{elastic}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_1 \sum_{j=1}^{p} \beta_j^2 \right\}$$

- Due to the shared L_1/L_2 regularisation, Elastic-Net does not aggressively prune features like Lasso.
- In practice it often performs well when used for regression prediction.

How to choose λ?

- Ridge regression and Lasso lead to a range of models for different values of λ.

λ	β	Validation Loss
0.001	Est set 1	
0.01	Est set 2	
0.02	Est set 3	
0.04	Est set 4	
0.08	Est set 5	Smallest
...		
100	Est set 10000	

- Test a large number of different λ value, e.g., 10000 values between 0.0001 and 100, denoted by λ_j, $j = 1, 2, 3, , 10000$.

Cross validation with regularisation

- Suppose we have 1000 data points and wish to find the best ridge or lasso regression. That is to find an appropriate λ.
- Under K-fold CV, we are going to test a large number of different λ_j values, e.g., $j = 1, 2, 3, \ldots, 10000$.
- If $K = 5$, randomly divide 1000 data points into 5 groups, each with 200 data points
- Run the 5-fold CV for each λ_j:
  - For each j, calculate the mean validation error over 5 runs
- Select the λ_j for which the error is minimal, say λ_5.
- The final model should use that value of λ.
  E.g., for Ridge: $RSS + \lambda_5 \sum_{j=1}^{p} \beta_j^2$

Regularised logistic regression

Regularisation can also be applied to logistic regression that aims to minimise the following loss function:

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \left(y_i \log(p(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - p(\boldsymbol{x}_i))\right) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- Recall that the negative log-likelihood is known as the cross-entropy loss function
- Thus, now we want to minimise this regularised loss function.

Concluding remarks

- Model selection is an important tool in your data analysis process, but should not be a replacement to model building through EDA, diagnostics, and domain knowledge.
- Model selection methods we discussed, e.g., cross-validation, are amenable to parallel computing, e.g., MapReduce, which is what we look at next.

Recommended reading

- Chapter 6 of *An introduction to statistical learning : with applications in R* by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer, 2014. `http://www-bcf.usc.edu/~gareth/ISL/`