NYC DATA SCIENCE ACADEMY

Python Machine Learning Class 3: Model Selection

NYC Data Science Academy

Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
    - Ridge regression
    - Lasso regression
    - ElasticNet regression
- Grid Search

Overview

- In this session we will discuss some general methods of improving model performance.
- We'll learn how to
  - Fine-tune machine learning models
  - Evaluate model performance using different metrics
  - Diagnose common problems of machine learning algorithms

Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
  - Ridge regression
  - Lasso regression
  - ElasticNet regression
- Grid Search

Training Error and Test Error

- Recall the difference between the accuracy on the training data set and test data set.
- We define *training error* and *test error* as:
  - T*raining error*: the error we get by applying the model to the same data from which we trained.
  - *Test error*: the error that we incur on new data.
    - The test error is a much stronger indication of how well we'll do on future data the model hasn't seen.
- Training error almost always underestimates test error, sometimes dramatically.

Training Error and Test Error

- The goal of learning is try to minimize the error for future prediction. However this is always unknown.
- The predicted error on a future dataset can be represented by the test error if we carefully choose the test dataset.

Solutions

To estimate the test error:

- A large test set is the best and simplest solution. Unfortunately, that is often unavailable.
- We estimate the test error by holding out a subset of the training observations from the fitting process, and then applying the learning method to those held out observations.

Validation

- Concretely, we randomly divide the available set of samples into two parts: a *training set* and a *validation set*.
- Then fit the model based on the training set, and predict the responses for the observations in the validation set.
- The error of the validation set is an estimate of the test error. Typically,
    - For a quantitative response (regression problem), it's the MSE (mean square error).
    - For a qualitative response (classification problem), it's the misclassification rate.

Drawbacks of Validation

- The test error of the validation set may be highly variable, depending on how we split the data set.
- Only a fraction of the observations are included in the training set used to fit the model, while the others are wasted.
- The validation set error may tend to overestimate the test error for the model fit on the entire data set, since some of the observations are not used in the fitting procedure.

Cross-Validation

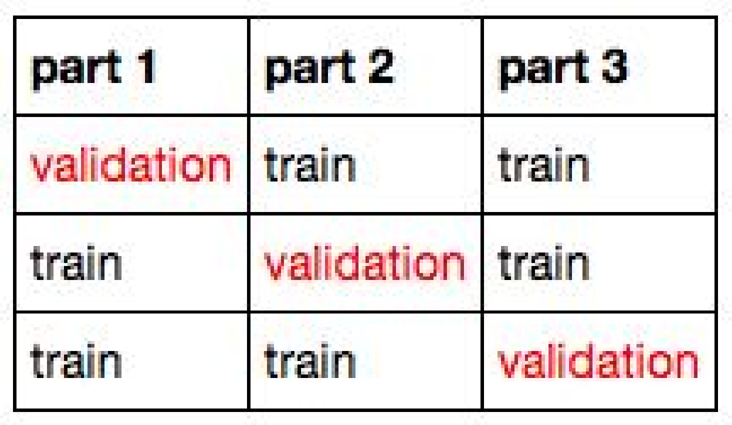
Instead of splitting the data set into two parts, divide the data into *K* equal-sized parts.

- We leave out part *k*, fit the model to the other *K* − 1 parts (combined), and then obtain predictions for the left-out *k*th part.
- This is done in turn for each part *k* = 1, 2, . . . *K*, and then the results are combined.

This is called the *k-fold cross-validation*, which is widely used to estimate the test error.

K-Fold Cross Validation

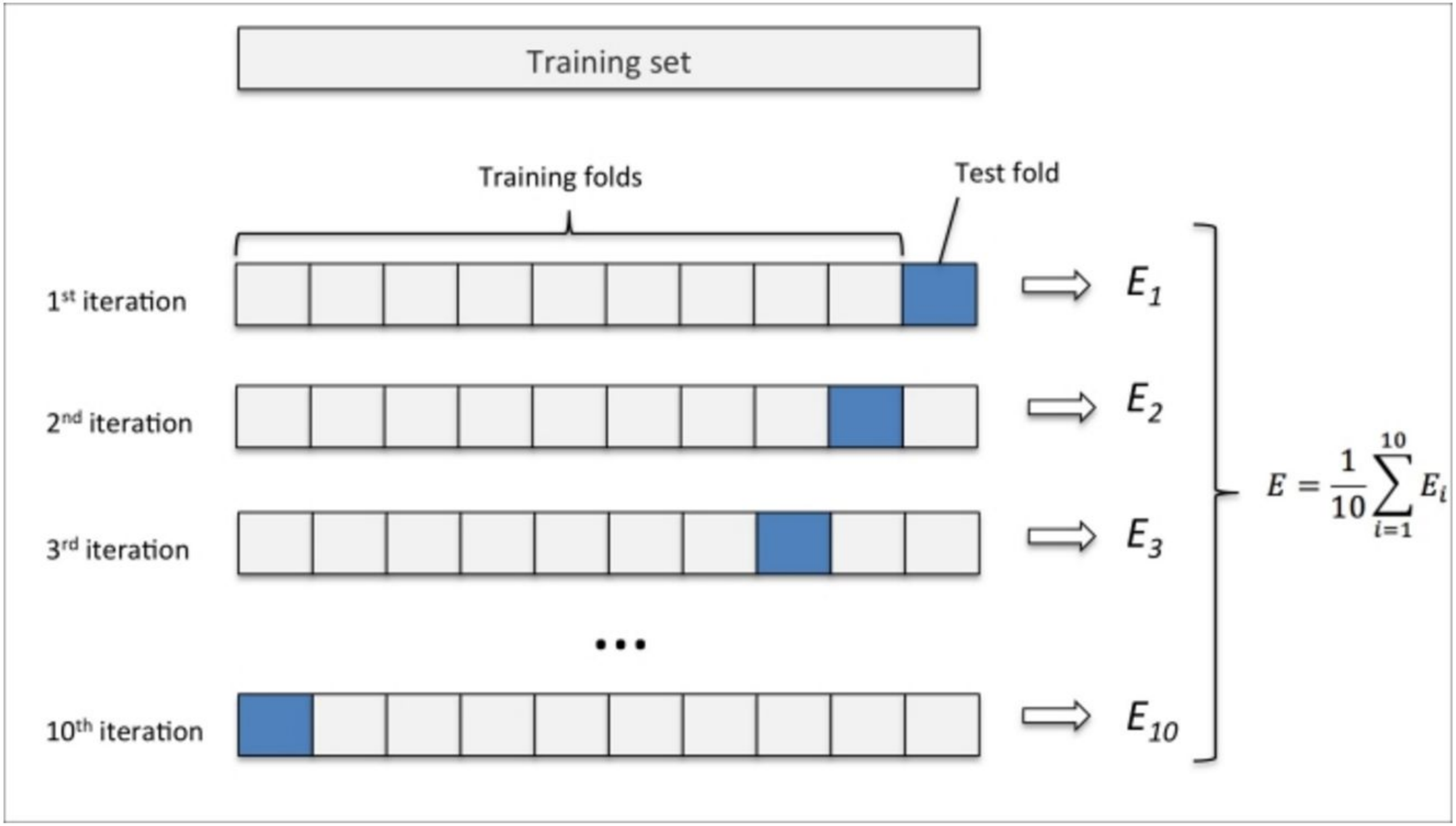
Here is simple example of dividing data into 3 equal-sized parts:



part 1	part 2	part 3
validation	train	train
train	validation	train
train	train	validation

K-Fold Cross Validation

Here is another example with K=10:



Cross Validation for Regression

- After dividing the data into K parts, let C_i denotes the i^{th} part of the observations.

- Let n_i denote the number of observations in C_i, let n denote the total number of observations. If all the parts are equal-sized, then $n_i = n/k$ for $i = 1, 2, 3, \dots k$.

- The final test error is:

$$CV_k = \sum_{i=1}^{k} \frac{n_i}{n} MSE_i$$

where $MSE_i = \sum_{j \in c_j} \frac{1}{n_i}(y_j - \hat{y}_j)^2$. $\hat{y}_j$ refers to the prediction of observation j.

Cross Validation for Classification

- After dividing the data into K parts, let C_i denote the i^{th} part of the observations.
- Let n_i denote the number of observations in C_i, let n denote the total number of observations. If all the parts are equal-sized, then $n_i = n/k$ for i = 1, 2, 3, ... k.
- The final test error is:

$$CV_k = \sum_{i=1}^{k} \frac{n_i}{n} Error_i$$

where $Error_i = \sum_{j \in c_j} \frac{1}{n_i} (y_j ! = \hat{y}_j)$. $\hat{y}_j$ refers to the prediction of observation j.

Cross Validation For Classification

- The estimated standard deviation of CV_K is:

$$SD(CV_K) = \sqrt{\frac{1}{K-1}\sum_{i=1}^{K}(Error_i - \bar{Error})^2}$$

Hands-on Session

- Please go to **"Cross Validation in Scikit-Learn"** in the lecture code.

Outline

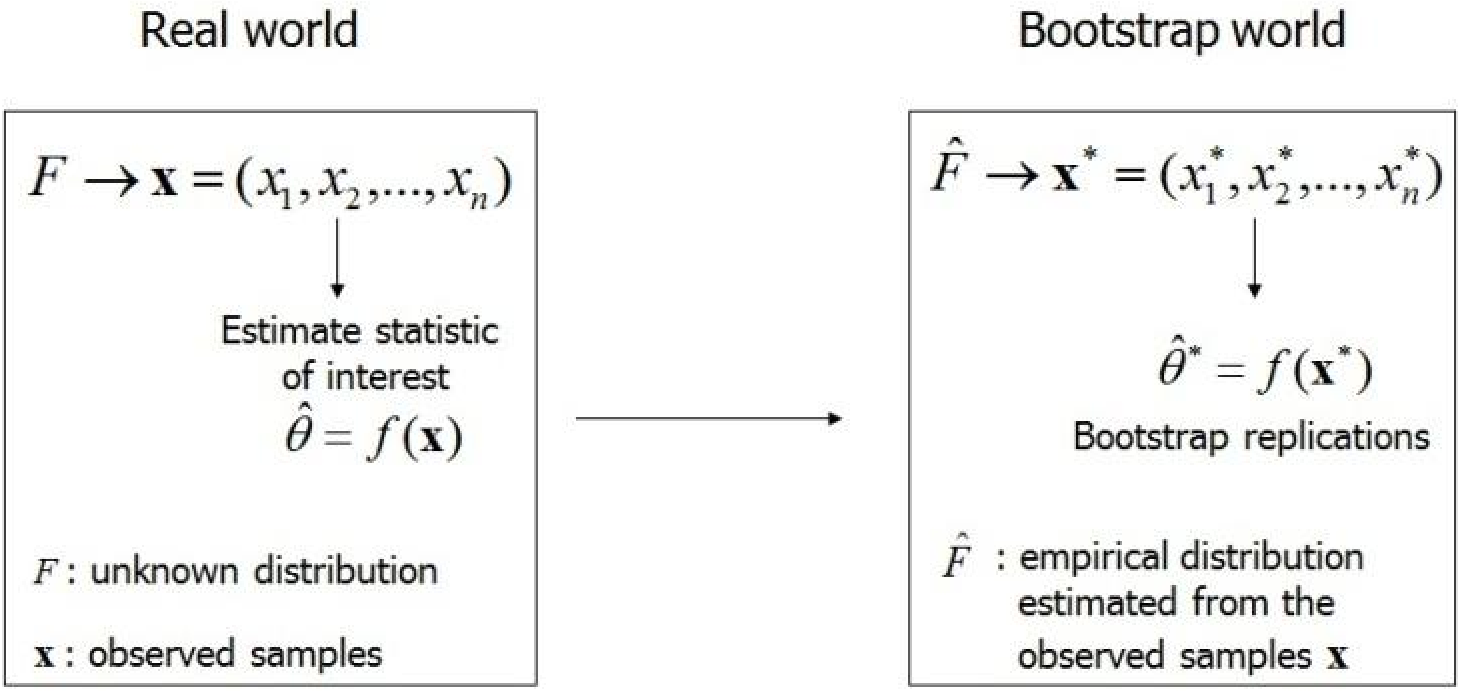
- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
  - Ridge regression
  - Lasso regression
  - ElasticNet regression
- Grid Search

Bootstrap

Bootstrap is another resampling method which:

- resamples from the data set with replacement, not just division.
- can be used to quantify the statistics associated with a given estimator or learning algorithm.
- can provide an estimate of the standard error or confidence interval of a coefficient.

Bootstrap



Real world

$F \rightarrow \mathbf{x} = (x_1, x_2, \ldots, x_n)$

Estimate statistic of interest

$\hat{\theta} = f(\mathbf{x})$

F : unknown distribution

$\mathbf{x}$: observed samples

Bootstrap world

$\hat{F} \rightarrow \mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$

$\hat{\theta}^* = f(\mathbf{x}^*)$

Bootstrap replications

$\hat{F}$: empirical distribution estimated from the observed samples $\mathbf{x}$

Bootstrap

To quantify the uncertainty associated with a given estimator α:

- Generate the first bootstrap data denoted by B^{*1}; use B^{*1} to produce a new bootstrap estimate for α, denoted by α^{*1}.
- Repeat the first step K times, so that we have K bootstrap data B^{*1}, B^{*2}, … B^{*K} as well as K estimates α^{*1}, α^{*2}, … α^{*K}.

We estimate the standard error of these estimates by the formula:

$$\sigma_\alpha = \sqrt{\frac{1}{K-1}\sum_{i=1}^{K}(\alpha^{*i} - \bar{\alpha^*})^2}$$

Bootstrap

Can bootstrap be used to estimate the test error?

- One way is to fit models based on the bootstrap data and test on the original data.
- Since there are many *overlaps* between the bootstrap data and the original data, it will seriously underestimate the test error.
- In cross validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: there is no overlap. This is crucial for its success.

Bootstrap

To estimate the test error using bootstrap, we can only use predictions for those observations that did not occur in the current bootstrap sample.

- Generate a bootstrap data B^{*i}, fit a model M^{*i}.
- Make a prediction on the observations that are not in B^{*i}.
- Repeat the first two steps K times.

This is called *Out Of Bag Error* (OOB Error).

Bootstrap

- The OOB error is complicated to calculate.
- Since each bootstrap data in randomly sampled from the original data, we have no idea how many *observations out of bag* left for testing. So we would lose control of the test size.
- Cross validation is a simpler and more natural way to estimate the validation scores.

Hands-on Session

- Please go to **"Bootstrap in Scikit-Learn"** in the lecture code.

Outline

- Cross-Validation
- Bootstrap
- **Feature Selection**
- Regularization
  - Ridge regression
  - Lasso regression
  - ElasticNet regression
- Grid Search

Feature selection

- Feature selection is the process of selecting a subset of relevant features for use in model construction. It's also known as variable selection.
- Many areas including text processing of internet documents, gene expression array analysis, and combinatorial chemistry often have tens or hundreds of thousands of variables available. Feature selection is crucial.
- The goals of feature selection are:
  - improving prediction performance
  - providing predictors with better computation speed and cost
  - providing a better understanding of the underlying process that generated the data

Feature selection

- In practice, feature selection often needs some background knowledge to determine which predictors to select.
- We will introduce some basic methods from a mathematical perspective:
  - Removing features with low variance.
  - Univariate feature selection
  - L1-based feature selection
  - Tree-based feature selection

Removing Features with low variance

- If a predictor has the same value in all observations, it's useless.
- In linear regression, if a predictor is constant, $X^T X$ will be singular.
- The `VarianceThreshold()` function is used to remove features with low variance.

Univariate feature selection

- Univariate feature selection is based on the univariate statistical tests.
- The tests we are using can be set manually.
  - For regression problems, the most commonly used test is the F test.
  - For classification problems, the most commonly used tests are the chi-square test and the F test.
- Function `chi2` is used to perform a chi-square test.
- Function `f_regression` is used to perform an F test for regression problems.
- Function `f_classifier` is used to perform an F test for classification problems.

Hands-on Session

- Please go to **"Removing Features with Low Variance"** in the lecture code.

Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- **Regularization**
  - Ridge regression
  - Lasso regression
  - ElasticNet regression
- Grid Search

Regularization

- Fit a model using a technique that regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero.
- Shrinking the coefficient estimates can significantly reduce their variances.
- The two best-known techniques of shrinking methods are ridge regression and lasso.

Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
  - Ridge regression
  - Lasso regression
  - ElasticNet regression
- Grid Search

Ridge Regression

- Recall that the least squares fitting procedure estimates β using the values that minimize

$$\sum_{i=1}^{n}(y^{(i)} - \beta_0 - \sum_{j=1}^{k}\beta_j x_j^{(i)})^2$$

- The ridge regression coefficient estimates β trying to minimize

$$\sum_{i=1}^{n}(y^{(i)} - \beta_0 - \sum_{j=1}^{k}\beta_j x_j^{(i)})^2 + \lambda\sum_{j=1}^{k}\beta_j^2$$

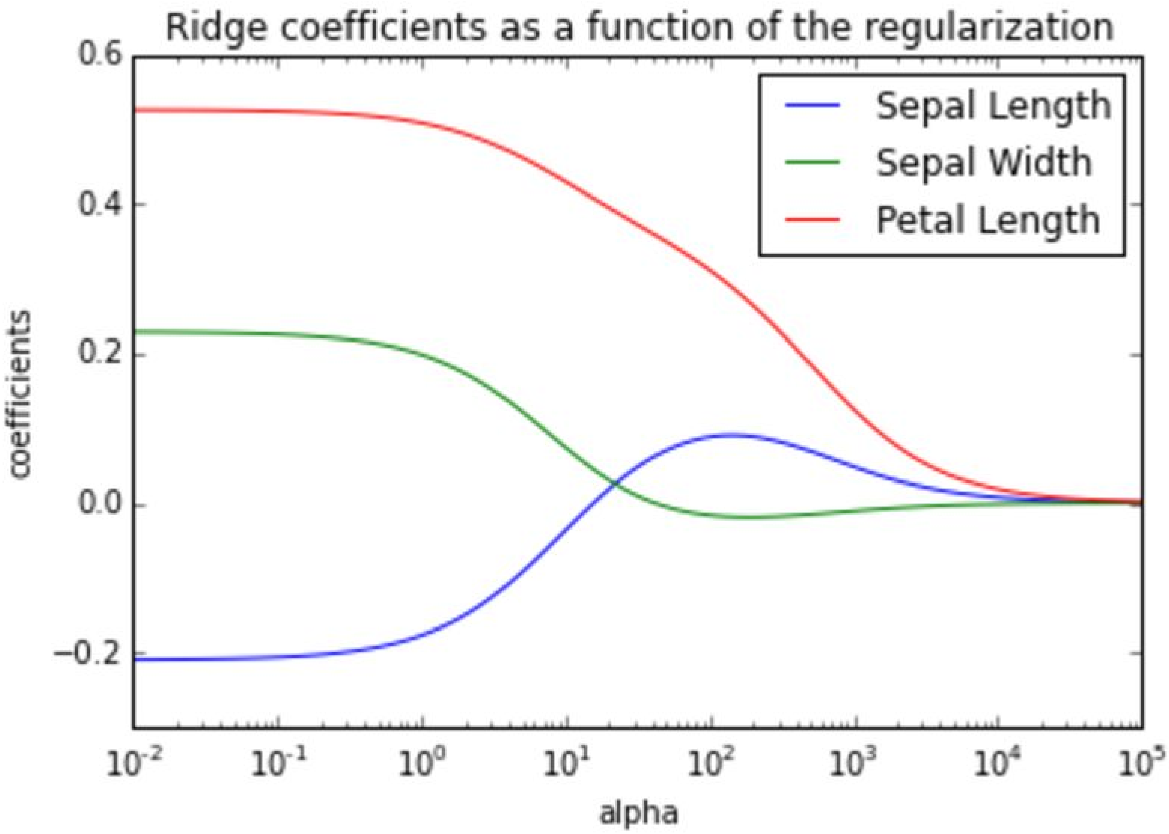
where λ is a tuning parameter to be determined.

Ridge Regression

- By adding the term $\lambda \sum_{j=1}^{k} \beta_j^2$ in the loss function, the estimation will try to shrink the parameters toward 0; hence this term is also called the shrinkage penalty. This is also called *L2 penalty*.
- The tuning parameter λ is used to control the relative impact of the complexity on the regression coefficient estimates.
  - When λ=0, the ridge estimate is the same as the least square estimate.
  - When λ→∞, all the parameters except β_0 will be 0, because we do not penalize β_0 in the loss function.

Ridge Regression:

- The curves are the plot of coefficients as functions of α.



Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
  - Ridge regression
  - **Lasso regression**
  - ElasticNet regression
- Grid Search

Lasso Regression

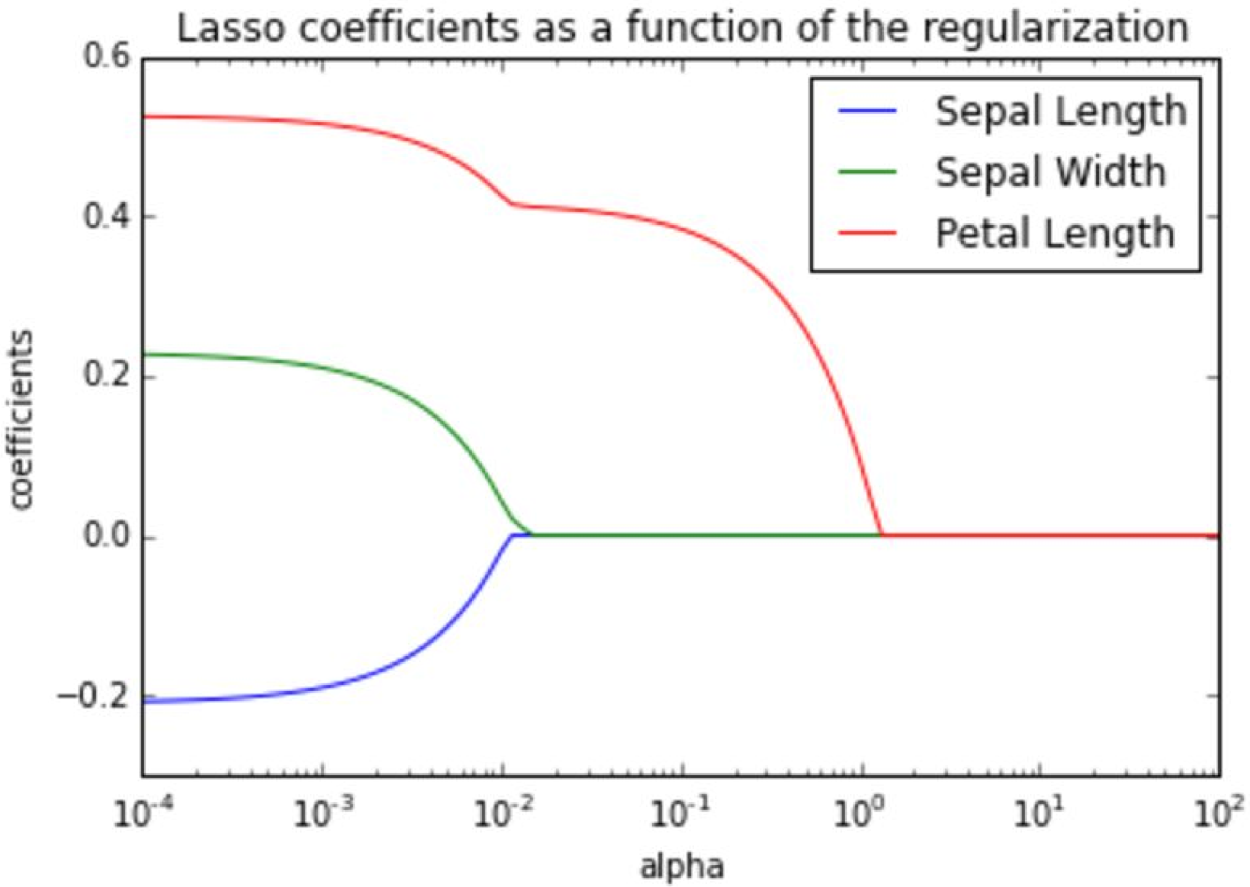
- Similar to ridge regression, Lasso adds an extra penalty term in the loss function.
- But the penalty term is slightly different from the one of ridge regression. The loss function of Lasso is :

$$\sum_{i=1}^{n}(y^{(i)} - \beta_0 - \sum_{j=1}^{k}\beta_j x_j^{(i)})^2 + \lambda\sum_{j=1}^{k}|\beta_j|$$

- The penalty term in lasso is sum of the **absolute values** of the coefficients (often called **L1 penalty**), instead of the sum of the squares of the coefficients (often called **L2 penalty**) as in ridge regression.

Lasso Regression:

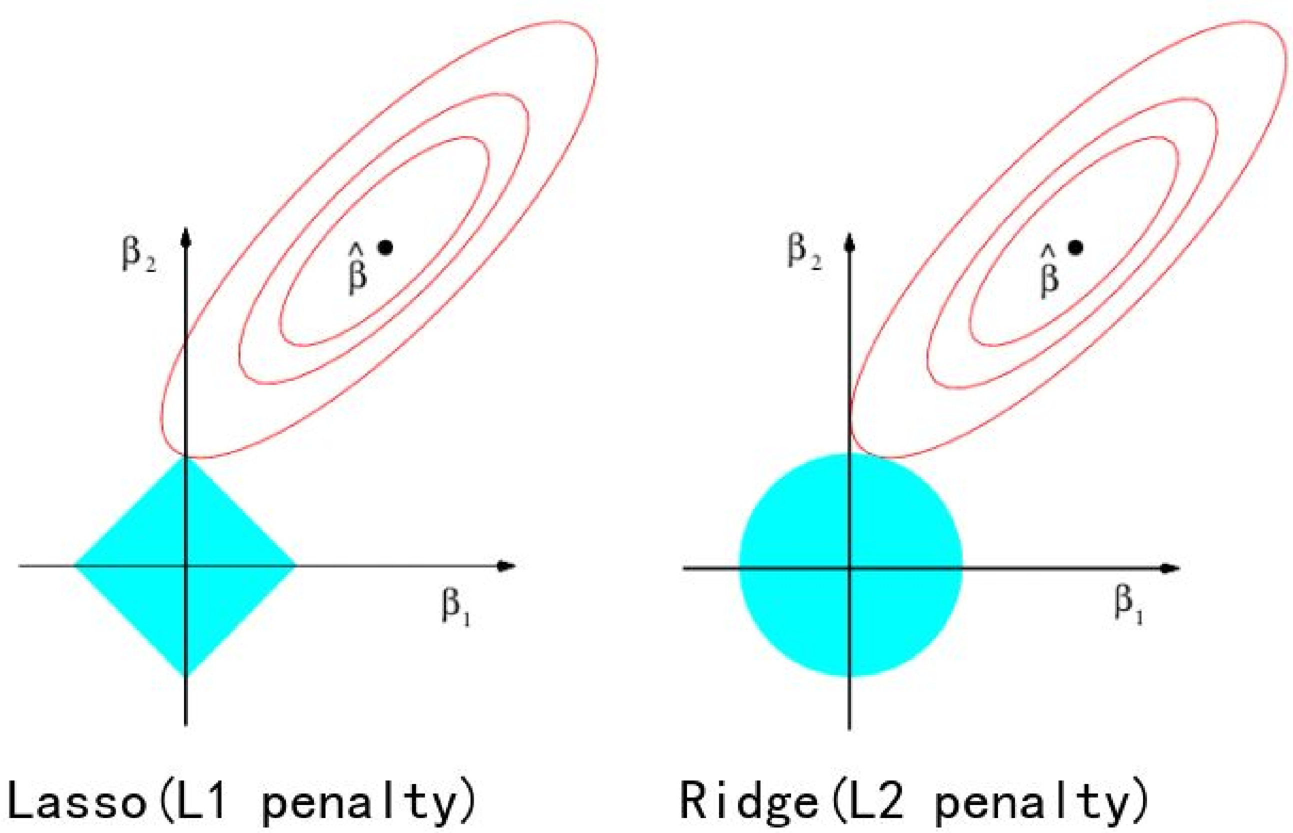
- The curves are the plot of coefficients as functions of α.



Ridge v.s. Lasso:

- We see from the plots of the previous slide that:
  - In ridge regression, all the coefficients tend to reach 0 at the same time.
  - In lasso, the coefficients tend toward 0 one by one.
  - Lasso selects features with the appropriate penalty.

Ridge v.s. Lasso:



This picture comes from An Introduction to Statistical Learning

Ridge v.s. Lasso: which to use?

- Neither ridge regression nor lasso universally dominates the other.
- In general, one might expect lasso to perform better when the response is a function of only a relatively small number of predictors. However, the number of predictors that is related to the response is never known.
- Techniques such as cross-validation can be used to determine which approach is better on a particular data set.

Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
    - Ridge regression
    - Lasso regression
    - **ElasticNet regression**
- Grid Search

ElasticNet

- ElasticNet is a linear regression model that combines L1 with L2 regularization.

- This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge.

- ElasticNet is useful when there are multiple features correlated with one another.

ElasticNet

- Here is the objective function of ElasticNet:

$$\min_{\theta} \frac{1}{2n} \parallel X\theta - y \parallel_2^2 + \alpha\rho|\theta|_1 + \frac{\alpha(1-\rho)}{2} \parallel \theta \parallel_2^2$$

where n refers to the number of observations, and $0 \leq \rho \leq 1$.

  - α controls the effects of regularization.
  - ρ control the convex combination of L1 and L2 norms.
    - ρ=0 degenerates back to ridge regression.
    - ρ=1 degenerates back to lasso.

Hands-on Session

- Please go to the lecture code for:
  - **"Ridge Regression in Scikit-Learn"**
  - **"Lasso Regression in Scikit-Learn"**
  - **"ElasticNet in Scikit-Learn"**

Outline

- Cross-Validation
- Bootstrap
- Feature Selection
- Regularization
    - Ridge regression
    - Lasso regression
    - ElasticNet regression
- **Grid Search**

Grid Search

- There are two types of parameters in machine learning models:
  - Those learned from the training data
    - Example: beta coefficients in linear regression
  - Parameters that are optimized separately
    - Example: α in ridge, lasso and elastic net regressions
- The second class of parameters are sometimes known as
  - Tuning parameters
  - Hyperparameters
- Grid search is the primary technique for finding optimal tuning parameters

Grid Search

- The grid search provided by GridSearchCV exhaustively generates candidates from a grid of parameter values specified.
  - The GridSearchCV instance implements the usual estimator API: when "fitting" it on a dataset all the specified combinations of parameter values are evaluated and the best combination is retained.
  - The evaluation of all combinations is computationally expensive.
- RandomizedSearchCV is an alternative to GridSearchCV
  - It draws random parameter combinations from sampling distributions with a specified budget

Hands-on Session

- Please go to **"Grid Search"** in the lecture code.