Linear Regression 1

MAE301 Applied Experimental Statistics

Max Yi Ren

Department of Mechanical Engineering
Arizona State University

October 22, 2015

1. introduction
2. ordinary least square
3. information criterion
4. summary

Motivation

Regression (or “Curve fitting”) is prevalent in engineering applications.

- Control: Use linear regression to fit a dynamics model for online control
- Design: Use response surface to predict responses from complex systems during optimization

→ statistic.

Regression analysis often has two purposes: **Inference** and **prediction**. → machine learning

Example of inference: What factors will influence class attendance? (focus on model interpretation)

Example of prediction: How many cars can we sell next year? (focus on prediction accuracy)

Linear model

covariate

Consider a sample $\underline{\mathbf{x} = [x_1, x_2, \ldots]^T}$ and its $\underline{\text{response } y}$. A linear model is *linear in its coefficients*:

$$\boxed{y} = a_1 \underline{f_1(\mathbf{x})} + a_2 \underline{f_2(\mathbf{x})} + \ldots + a_p \underline{f_p(\mathbf{x})}. \tag{1}$$

Here p is the degree of freedom (complexity) of a linear model, f_p are known functions of $\mathbf{x}$. The following models are all linear:

$$\frac{dy}{da} = \begin{bmatrix} x_1^2 \\ \sin x_2 \end{bmatrix}$$

$$\begin{aligned} y &= a_1 x_1 + a_2 x_2 + \ldots + a_p x_p, \\ y &= \underline{a_1} x_1^2 + \underline{a_2} \sin(x_2), \\ y &= \exp(a_1 x_1 + a_2 x_2). \end{aligned} \tag{2}$$

$$y = [\underline{x_1^2}, \underline{\sin(x_2)}] \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

$$\Rightarrow \underline{\log y} = \underline{a_1 x_1 + a_2 x_2}$$

In the following discussion, without loss of generality, we will focus on the form $y = a_1 x_1 + a_2 x_2 + \ldots + a_p x_p$.
Give an example of a nonlinear model.

Ordinary least square regression for linear models

Consider training data $(\mathbf{X}, \mathbf{y})$, with each row of $\mathbf{X}$ being a data point and that of $\mathbf{y}$ the corresponding response. A linear model assumes:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3}$$

where $\boldsymbol{\varepsilon}$ are random errors following i.i.d. normal, i.e., $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$. The goal of OLS is to estimate the model parameters $\boldsymbol{\beta}$ so that the estimations $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$ are close to the observed $\mathbf{y}$. This can be formulated as follows:

$$\boldsymbol{\beta}^* = arg \min_{\boldsymbol{\beta}} \ ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2, \tag{4}$$

which has an analytical solution:

$$\boldsymbol{\beta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{5}$$

What can go wrong with this solution $\boldsymbol{\beta}^*$?

Covariance matrix

The matrix $\mathbf{X}^T\mathbf{X}$ is called the covariance matrix[1] of a linear model.

Possible reasons for a singular covariance matrix:

Cause 1 - “Small n large p”:

1. Understand RNA string (10^5 in size) functionalities from a few cases (e.g. tumor) and controls;
2. Associate brain functionalities (e.g. speech) with brain cells (size depend on resolution) using a few fMRI scanning;
3. Identify your face with a few photos uploaded on Facebook;
4. Detect moving objects (time sensitive); $\emptyset$
5. Estimate covariance of stocks (time sensitive); $\emptyset$ X^TX

Cause 2 - Linear dependency: e.g., $x_1 = 2x_2$

How will redundant observation affect your model?

[1] Rigorously, this is an estimator of the asymptotic covariance matrix

OLS for nonlinear models

For a nonlinear model $y = f(\mathbf{x}, \boldsymbol{\beta})$, the least square problem is:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \boldsymbol{\beta}))^2 \tag{6}$$

How do we solve this problem?

R^2 measure

$R^2=1$, when fit perfectly. $\hat{y} = X\beta^*$, where $\beta^* = (X^TX)^{-1}X^Ty$.

$R^2=0$, $\hat{y}_i = \bar{y}$

$$R^2 = 1 - \frac{\sum_i (\overset{\text{predicted response}}{\hat{y}_i} - \overset{\text{observed}}{y_i})^2}{\sum_i (y_i - \underset{\text{sample mean of } y_i}{\bar{y}})^2},$$

where $\mathbf{y}$ are from the test data, $\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ are estimates of $\mathbf{y}$. $\bar{y}$ is the average of $\mathbf{y}$. We can use R^2 to evaluate the test performance of the model. Higher R^2 indicates better performance.

crossvalidation for model selection

For a given modeling technique, e.g., OLS, one would still like to choose among models, e.g., the number of covariates or degree of polynomials. One way is to perform crossvalidation on each model and pick the one with the lowest average validation error.

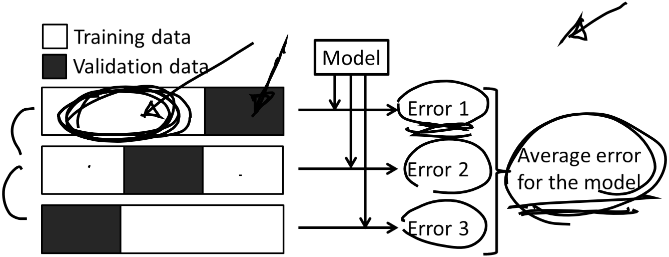

Figure: Three-fold crossvalidation

When validation size is one, it's called leave-one-out crossvalidation.

Akaike information criterion AIC.

Occam's razor: Among competing hypotheses, the hypothesis with the fewest assumptions should be selected.
AIC: A measure of goodness of fit and model complexity, for a given set of data. Provides a means for model selection.

$$AIC = 2p - 2\ln(L), \tag{7}$$

where p is the number of parameters and L is the maximum likelihood.
In OLS

$$L = \exp\left(-\frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right), \tag{8}$$

where n is the sample size.
AIC and leave-one-out crossvalidation are asymptotically ($n \to \infty$) equivalent.

Akaike information criterion

AICc: AIC with a correction:

$$AICc = \underline{AIC + \frac{2p(p+1)}{n-p-1}}. \tag{9}$$

Use AICc instead of AIC when $n/p < 40$.

Bayesian information criterion

BIC.

AIC 2p.

BIC:

p ln(n)

$$BIC = -2\ln(L) + p\ln(n), \tag{10}$$

Compared with AIC, BIC penalizes the number of covariates more strongly.

CV, AIC, BIC on model selection

Information-criterion based model selection (AIC, BIC) is computationally less expensive than CV, but it relies on a proper estimation of model complexity (degree of freedom).
For OLS with large data size (n) and relatively smaller dimensions (p), information criterion is more recommended than CV.
On the other hand, CV is applicable across various modeling techniques.

summary of the class

- Linear regression models and model parameter estimates
- Covariance matrix
- Information criteria

Python code for demos in the class

```
#modified from: http://scikit-learn.org/stable/auto_examples/model_selection/...
#plot_underfitting_overfitting.html

import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation

np.random.seed(0)

n_samples = 30
degrees = [1, 3, 15]
error = np.zeros(len(degrees))

true_fun = lambda X: np.cos(1.5 * np.pi * X) + np.sin(0.5 * np.pi * X)
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

plt.figure(figsize=(14, 7))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i + 1)
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()

    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)
    error[i] = ((pipeline.predict(X[:, np.newaxis]) - y)**2).sum()

# continue on the next page...
```

Python code for demos in the class

```
# continue from the last page...
    # Evaluate the models using crossvalidation
    scores = cross_validation.cross_val_score(pipeline,
        X[:, np.newaxis], y, scoring="mean_squared_error", cv=10)

    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")

    plt.title("Degree {}\nCV_MSE = {:.2e}(+/- {:.2e})\nAIC = {:.2e}\nBIC = {:.2e}".format(
        degrees[i], -scores.mean(), scores.std(),
        2*degrees[i]+error[i], np.log(n_samples)*degrees[i]+error[i]))
plt.show()
```