Advanced Python for Neuroscientists
Lecture 2: Regression

Summer 2021

Princeton Neuroscience Institute
Instructors: Yisi Zhang & Tyler Giallanza

July 8, 2021

Recap

Lecture 1

- Types of learning: supervised vs. unsupervised
- Supervised learning: regression vs. classification
- Bias-variance trade-off
- Training vs. testing

Outline

- Simple linear regression
- Multiple linear regression
- Variable selection
- Shrinkage

2.1 Simple linear regression

Suppose that we want to predict a quantitative response Y on the basis of a single predictor variable X and the assumption of an approximately linear relationship between X and Y, we can write this linear relationship as

$$Y = \beta_0 + \beta_1 X + \epsilon.$$

ϵ is a zero-mean random error.
To estimate the coefficients β_0 and β_1, we use data of n observation pairs $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$.

2.1 Simple linear regression

The problem can be written as matrix multiplication

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}. \tag{1}$$

$$Y = X\beta + \epsilon$$

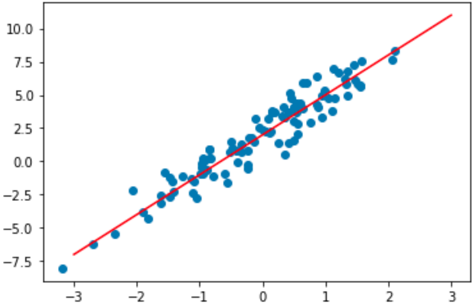
2.1 Simple linear regression

Exercise: Create random Xs and Ys

```
import numpy as np
n = 100
# create a random column vector as our Xs
x = np.random.normal(0,1,size=(n, 1))
# add ones to x
X = np.hstack((np.ones_like(x), x))
# Y = 2+3X+noise
# get Y using matrix multiplication
beta = np.array([[2.0],[3.0]])
y = X@beta + np.random.normal(0,1,size=(n,1))
```

2.1 Simple linear regression

You should get a plot of y against x like this



2.1 Simple linear regression

The Least Squares (LS) method estimates the model parameters β by minimizing the sum of the residual sum of squares (RSS) as

$$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

The LS estimator $\hat{\beta}_{LS}$ can be defined as

$$\hat{\beta}_{LS} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 = \arg\min_{\beta} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}).$$

Assuming $\boldsymbol{X}^T \boldsymbol{X}$ is invertible,

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

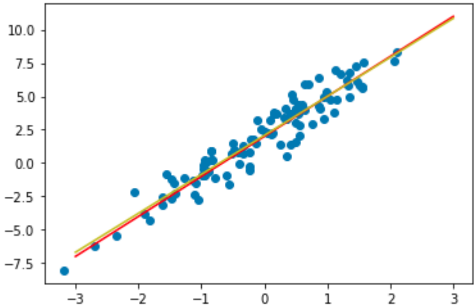
2.1 Simple linear regression

Exercise: Solve for βs in one step

```
from numpy.linalg import multi_dot
from numpy.linalg import pinv
# solve for beta using matrix inverse and multiplication
beta_ls = multi_dot([pinv(X.T@X), X.T, y])
```

2.1 Simple linear regression

The fitted model is very close to the truth



$\beta_0 = 2$ and $\beta_1 = 3$
$\hat{\beta}_{LS0} = 2.083$ and $\hat{\beta}_{LS1} = 2.924$

2.1 Simple linear regression

How confident are we with our estimation? The most important measure is the standard error (SE) of the coefficients. The distribution of $\hat{\beta}$ is approximated by

$$\hat{\beta}_{LS}|\boldsymbol{X} \sim N(\beta, \hat{\sigma}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}),$$

where $\hat{\sigma}^2$ is the estimator of σ^2 (remember that $\epsilon \sim N(0, \sigma^2)$). We can estimate σ^2 using residual standard error (RSE) given by the formula

$$RSE^2 = RSS/(n-2).$$

Then we can calculate SE of the coefficients using the diagonal elements of the matrix:

$$RSE^2(X^TX)^{-1}.$$

2.1 Simple linear regression

Exercise: Solve for $SE(\beta)$

```
# first calculate the covariance matrix of beta as RSS/(n-2)*inv(X.T X)
RSS = (y-(X@beta_ls)).T @ (y-(X@beta_ls))
RSS = RSS[0,0]
beta_ls_cov = RSS/(n-2) * pinv(X.T@X)
# take the diagonal of the covariance matrix and take the square root
beta_ls_se = np.diag(beta_ls_cov)**0.5
# calculate RSE, it should be close to 1
RSE = (RSS/(n-2))**0.5
```

2.1 Simple linear regression

We can then compute the confidence intervals (CI) of our $\hat{\beta}$ using $SE(\hat{\beta})$. For example, we can construct a 95% CI of $\hat{\beta}$ approximately as

$$[\hat{\beta}_l - 2SE(\hat{\beta}_l),\ \hat{\beta}_l + 2SE(\hat{\beta}_l)],\ l = 0, 1.$$

2.1 Simple linear regression

We can also perform hypothesis tests on the coefficients. Most commonly, we test the *null hypothesis* of

$$H_0 : \text{there is no relationship between } X \text{ and } Y\text{, i.e., } \beta_1 = 0$$

versus the *alternative hypothesis*

$$H_a : \text{there is no relationship between } X \text{ and } Y\text{, i.e., } \beta_1 \neq 0.$$

In practice, we compute a t-statistic, given by

$$t = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)},$$

with $n - 2$ degrees of freedom and calculate the corresponding *p-value*.

2.1 Simple linear regression

To assess the accuracy of the model, we estimate *the proportion of variability in Y that can be explained using X* using R^2 statistic, given by

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS},$$

where $TSS = \sum(y_i - \bar{y})^2$ is the total sum of squares.

2.1 Simple linear regression

Exercise: Test hypothesis

```
from scipy.stats import t
# approx. 95% CI
ci = np.array([beta_ls[:,0] - 2*beta_ls_se,
beta_ls[:,0]+2*beta_ls_se]).T
# t-statistic
ts = beta_ls[:,0]/beta_ls_se
# p-value (two-sided)
pv = (1 - t.cdf(abs(ts), n-2)) * 2
# R2
TSS = (y-y.mean()).T @ (y-y.mean())
TSS = TSS[0, 0]
R2 = 1 - RSS/TSS
```

```
import statsmodels.api as sm
ols = sm.OLS(y, X)
ols_result = ols.fit()
ols_result.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.908
Model:	OLS	**Adj. R-squared:**	0.907
Method:	Least Squares	**F-statistic:**	970.0
Date:	Thu, 10 Jun 2021	**Prob (F-statistic):**	1.25e-52
Time:	09:28:22	**Log-Likelihood:**	-141.56
No. Observations:	100	**AIC:**	287.1
Df Residuals:	98	**BIC:**	292.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.0825	0.101	20.658	0.000	1.882	2.283
x1	2.9236	0.094	31.145	0.000	2.737	3.110

Omnibus:	1.542	**Durbin-Watson:**	2.101
Prob(Omnibus):	0.462	**Jarque-Bera (JB):**	1.550
Skew:	-0.226	**Prob(JB):**	0.461
Kurtosis:	2.591	**Cond. No.**	1.09

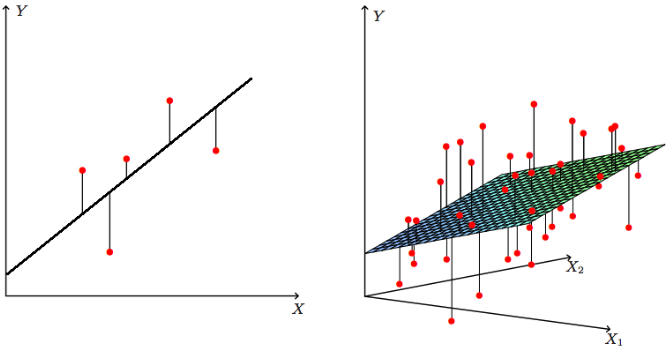
2.2 Multiple linear regression

The multiple linear regression takes the form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon,$$

where X_j represents tlhe jth predictor and β_j quantifies the association between that variable and the response.
The strategy adopted to estimate β is the exactly the same as the simple linear regression case (which is...?).

2.2 Multiple linear regression



2.2 Multiple linear regression

What do we care about multiple regression?
Think that the predictors $X_1, X_2, ...X_p$ are the activity of p neurons that we measured using multi-channel electrodes and we want to use them to predict a behavioral outcome Y, e.g. moving direction, hand position, vocal output,... We may ask

- 1. Is at least one of the neurons useful in predicting the behavior Y?
- 2. Do all the neurons help to explain Y, or is only a subset of them useful?

2.2 Multiple linear regression

Q1. We can answer this question by testing the null hypothesis,

$$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0$$

versus tlhe alternative

$$H_a : \text{at least one } \beta_j \text{ is non-zero.}$$

This hypothesis test is performed by computing the F-statistic,

$$F = \frac{(TSS - RSS)/p}{RSS/(n - p - 1)}.$$

2.2 Multiple linear regression

Exercise: Test hypothesis

```
import numpy as np
from sklearn import datasets
from sklearn import linear_model
# there are 10 variables predict diabetes
X, y = datasets.load_diabetes(return_X_y=True)
regr = linear_model.LinearRegression()
n,p = X.shape
TSS = np.sum((y - y.mean())**2)
y_hat = regr.predict(X)
RSS = np.sum((y - y_hat)**2)
F = (TSS-RSS)/p/RSS*(n-p-1)
```

2.2 Multiple linear regression

Compare your results with the built-in statistics function

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.518
Model:                            OLS   Adj. R-squared:                  0.507
Method:                 Least Squares   F-statistic:                     46.27
Date:                Thu, 10 Jun 2021   Prob (F-statistic):           3.83e-62
Time:                        09:58:01   Log-Likelihood:                -2386.0
No. Observations:                 442   AIC:                             4794.
Df Residuals:                     431   BIC:                             4839.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        152.1335      2.576     59.061      0.000     147.071     157.196
x1           -10.0122     59.749     -0.168      0.867    -127.448     107.424
x2          -239.8191     61.222     -3.917      0.000    -360.151    -119.488
x3           519.8398     66.534      7.813      0.000     389.069     650.610
x4           324.3904     65.422      4.958      0.000     195.805     452.976
x5          -792.1842    416.684     -1.901      0.058   -1611.169      26.801
x6           476.7458    339.035      1.406      0.160    -189.621    1143.113
x7           101.0446    212.533      0.475      0.635    -316.685     518.774
x8           177.0642    161.476      1.097      0.273    -140.313     494.442
x9           751.2793    171.902      4.370      0.000     413.409    1089.150
x10           67.6254     65.984      1.025      0.306     -62.065     197.316
==============================================================================
Omnibus:                        1.506   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.471   Jarque-Bera (JB):                1.404
Skew:                           0.017   Prob(JB):                        0.496
Kurtosis:                       2.726   Cond. No.                         227.
==============================================================================
```

2.2 Multiple linear regression

Q2. Now we conclude that there should be at least one variable predicting Y, we should wonder which ones they are.

We can take a look at the p-values to have some idea, but for large p-values there could be some false discoveries.

Potentially, there are 2^p possible selections of variable combinations which could be a huge number if p is large. What should we do?

This is a problem of variable selection.

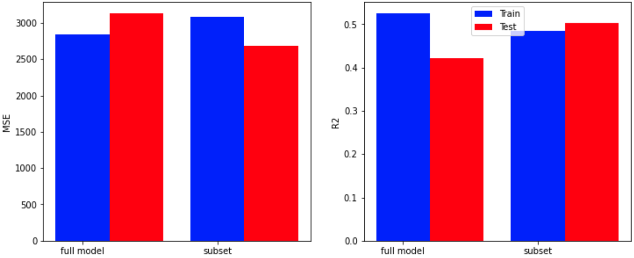
2.3 Variable selection

Also referred to as feature selection or a special case of model selection, excluding irrelevant variables from a multiple regression model can lead to a model that is more easily interpreted.

In addition, including all predictors may not be beneficial for future predictions.

2.3 Variable selection

Training vs. testing revisited.



$$MSE_{test} \neq MSE_{train}$$

$$R^2_{test} \neq R^2_{train}$$

2.3 Variable selection

Model selection based on *adjusted* criteria.

$$AIC = n\ln(\hat{\sigma}^2) + 2d,$$
$$BIC = n\ln(\hat{\sigma}^2) + d\ln(n),$$
$$\text{Adjusted } R^2 = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)},$$

where $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ and d is the number of predictors.
We select the model with the lowest AIC or BIC, or the highest Adjusted R^2.

2.3 Variable selection

Alternatively, we can directly evaluate test error and R^2 using cross-validation.

We will cover this in next class.

2.3 Variable selection

Using these criteria, what is your strategy to select the *best model* from among the 2^p possibilities?

Forward selection and backward selection are two greedy but efficient algorithms that select variables.

2.3 Variable selection

Forward selection

- 1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
- 2. For $k = 0, ..., p - 1$:
  (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 $\mathcal{M}_0, ..., \mathcal{M}_p$ using cross-validation prediction error, AIC, BIC, or adjusted R^2.

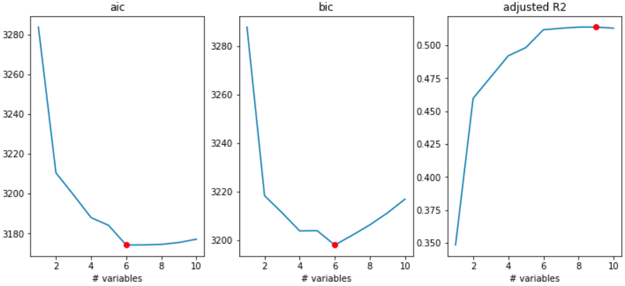
2.3 Variable selection

Exercise: Forward selection

```
import numpy as np
from sklearn import linear_model
def forward_select(X,y,c):
    n,p = X.shape
    mdl = [] # initiate M0
    regr = linear_model.LinearRegression()
    for k in range(p):
        m = p - k
        remain_inds = np.array(list(set(range(p)) - set(mdl)),
dtype=int)
        cri = np.zeros(m)
        for i,j in enumerate(remain_inds):
            mdl_temp = np.append(mdl, [j])    ...
```

2.3 Variable selection

Compare different criteria.



2.4 Shrinkage

An alternative method for variable selection is using shrinkage. Recall that so far we have adopted the least squares method to estimate βs:

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

Minimizing RSS should give us the best result for training accuracy. Now if our goal is shifted to not only minimizing errors but also the number of predictors, we can modify our objective function to minimize:

2.4 Shrinkage

Ridge regression minimizes

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2.$$

Lasso regression minimizes

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|.$$

The optimal tuning parameter $\lambda > 0$ can be determined using methods for model selection (AIC, BIC, cross-validation).

2.4 Shrinkage

Exercise: Ridge and Lasso

```
# apply ridge (or lasso) using sklearn.linear_model
regr = linear_model.Ridge() #regr = linear_model.Lasso()
alphas = np.logspace(-4, -1, 6)
scores = [regr.set_params(alpha=alpha).fit(X_train,
y_train).score(X_test, y_test) for alpha in alphas]
best_alpha = alphas[scores.index(max(scores))]
regr.alpha = best_alpha
regr.fit(X_train, y_train)
```

Homework

- Make sure you understand all the exercises above
- Run through the codes here that should replicate all the figures
  https://github.com/yisiszhang/AdvancedPython/blob/main/colab/Lecture2.ipynb
- Compare variable selections obtained from AIC/BIC/adjusted R2 and shrinkage.
- What is the difference between the coefficients estimated with ridge and lasso? Are they both performing variable selection?