Ensimag – Grenoble INP – UGA — Year 2024-2025
Introduction to Statistical Learning and Applications
Pedro L. C. Rodrigues pedro.rodrigues@inria.fr
Alexandre Wendling alexandre.wendling@univ-grenoble-alpes.fr

TD 2: Some questions from previous exams

▶ Exercise 1 (credits to EPFL CS-433)

Assume we are doing linear regression with mean-squared loss and L2-regularization on four one-dimensional data points. Our prediction model can be written as $f(x) = ax + b$ and the optimization problem can be written as

$$a^\star, b^\star = \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{4} \Big(y_i - f(x_i)\Big)^2 + \lambda a^2$$

Assume that our data points (x_i, y_i) are $\{(-2, 1), (-1, 3), (0, 2), (3, 4)\}$. What is the optimal value for the bias, $b^\star$?

(A) Depends on the value of λ.

(B) 3

(C) 2.5

(D) None of the above answers.

The objective function can be rewritten as

$$\mathcal{L}(a, b) = \sum_{i=1}^{4} \Big(y_i - (ax_i + b)\Big)^2 + \lambda a^2$$

so the partial derivative with respect to b is

$$\frac{\partial \mathcal{L}}{\partial b} = -2 \sum_i \Big(y_i - (ax_i + b)\Big)$$

which is zero if, and only if,

$$\sum_i y_i = a \sum_i x_i + 4b$$

But since $\sum_i x_i = 0$ then $b^\star = \dfrac{\sum_i y_i}{4} = 2.5$

▶ Exercise 2 (credits to Berkeley CS-189)

In the following statements, the word "bias" is referring to the bias-variance decomposition. Which one of them is true?

(A) A model trained with N training points is likely to have lower variance than a model trained with $2N$ training points.

(B) If my model is underfitting, it is more likely to have high bias than high variance.

(C) Increasing the number of parameters (weights) in a model usually improves the test set accuracy.

(D) None of the above.

▶ Exercise 3 (credits to EPFL CS-433)

Consider a regression model where data (x, y) is generated by input $x \in \mathbf{R}$ uniformly sampled between $[0, 1]$ and $y = x + \varepsilon$, where ε is random noise with mean 0 and variance 1. Two models are carried out for regression: model $\mathcal{A}$ is a trained quadratic function $g_{\mathcal{A}}(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2$ and model $\mathcal{B}$ is a constant function $g_{\mathcal{B}}(x) = \frac{1}{2}$. Compared to model $\mathcal{B}$, model $\mathcal{A}$ has

(A) Higher bias, higher variance.

(B) Lower bias, higher variance.

(C) Higher bias, lower variance.

(D) Lower bias, lower variance.

▶ Exercise 4

Consider the following `python` script:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
np.random.seed(0)
# number of variables
pt = 201
# number of predictors
p = pt - 1
# sample size
n = 30 * p
# generate data
D = np.random.randn(n, pt)
df = pd.DataFrame(data=D)
df = df.rename(columns={0:'Y'})
# do multiple linear regression
df['intercept'] = 1
model = sm.OLS(df['Y'], df.drop(columns='Y'))
results = model.fit()
print(results.summary())
```

(a) What does the script do? Run it on your computer.

The script first generates a dataset with $N = 6000$ data points, where each $y_i = \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0, 1)$ and all predictors are independent with each other with $x_{ij} \sim \mathcal{N}(0, 1)$. The script then estimates a linear regression model relating the y_i with the x_i, despite the fact that they are certainly not related, as we can see from the way we generated them.

(b) What is the true distribution of the random variable Y given the first 200 columns of matrix D, which we shall call $X_1, \ldots, X_{200}$?

$$Y \mid X_1, \ldots, X_{200} \sim \mathcal{N}(0, 1)$$

(c) Write an equation defining the model estimated by `model.fit()`. What is the difference between this model and the one defined above?

The model that `python` estimates looks like the following: for each data point i,

$$y_i = \hat{\beta}_0 + \sum_{i=1}^{200} \hat{\beta}_i x_{ij}$$

(d) Using `results.pvalues` count how many estimated parameters have a p-value under 0.05. What is going on?

```
print(np.sum(results.pvalues < 0.05))
```

```
## 8
```

This is a clear demonstration of what is commonly called the "multiple comparison problem" in the statistics literature. Since each individual statistical test has a 0.05 significance level, the fact of doing 200 of them will, in average, reject $200 \times 0.05 = 10$ times out of simple randomness.

▶ Exercise 5

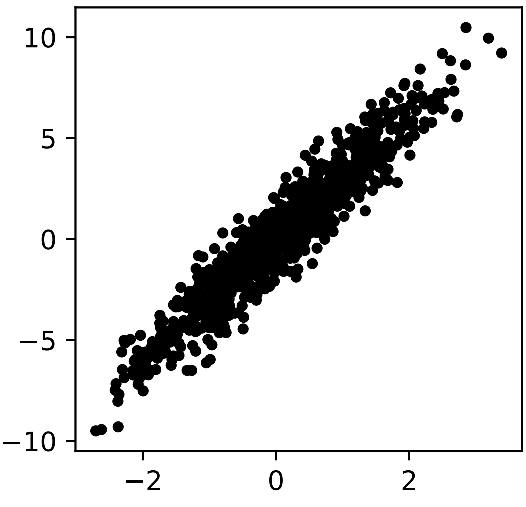
In this exercise, you will perform multiple linear regression on simulated data under different conditions. To ensure reproducibility on your results, set the seed with `numpy.random.seed(0)` at the beginning of your script.

(a) Simulate a dataset of size $N = 1000$ of the following generating model:

$$\begin{aligned} X_{1,i} &= \varepsilon_{1,i} \\ X_{2,i} &= 3X_{1,i} + \varepsilon_{2,i} \\ Y_i &= X_{2,i} + X_{1,i} + 2 + \varepsilon_{3,i} \end{aligned}$$

where $i \in \{1, \ldots, N\}$ and the ε_{ij} are independent $\mathcal{N}(0,1)$ random variables. For a given i, what is the distribution of $(X_{1,i}, X_{2,i})$? Plot the clouds of points of the simulated values of $(X_{1,i}, X_{2,i})_{i=1,\ldots,n}$. What is its shape? Can you write an analytical formula for it?

```
import numpy as np
import matplotlib.pyplot as plt
N = 1000
X = np.zeros((N, 2))
X[:, 0] = np.random.randn(N)
X[:, 1] = 3 * X[:, 0] + np.random.randn(N)
Y = X[:, 1] + X[:, 0] + 2 + np.random.randn(N)
fig, ax = plt.subplots(figsize=(3, 3))
ax.scatter(X[:, 0], X[:, 1], c='k', s=10)
ax.set_xlabel('X1')
ax.set_ylabel('X2')
fig.show()
```



First notice that $X_1 \sim \mathcal{N}(0,1)$ and $X_2 = 3X_1 + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0,1)$ so writing all this in matrix notation we have:

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

so that we get

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 3 & 10 \end{bmatrix} \right)$$

(b) Let us consider the following two regression models:

$$\begin{aligned} \text{Model A:} \quad Y_i &= \alpha_1 X_{1,i} + \alpha_0 + \tilde{\varepsilon}_{A,i} \\ \text{Model B:} \quad Y_i &= \beta_2 X_{2,i} + \beta_0 + \tilde{\varepsilon}_{B,i} \end{aligned}$$

where $\tilde{\varepsilon}_{A,i} \sim \mathcal{N}(0, \sigma_A^2)$ and $\tilde{\varepsilon}_{B,i} \sim \mathcal{N}(0, \sigma_B^2)$. What should be the values of $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\sigma}_A^2, \hat{\beta}_0, \hat{\beta}_2, \hat{\sigma}_B^2$ when $N \to \infty$? Consider $N = 1000$ and check whether the estimates of the parameters are close to the true values that you've calculated. Now do `np.random.seed(3)` and simulate again a dataset $X_{1,i}, X_{2,i}, Y_i$ for $n = 10$. Estimate the parameters. What happens?

Recall that the true model of the observations is written as:

$$Y = 2 + X_1 + X_2 + \varepsilon$$

Exploring the way that the predictors are generated, we can rewrite the true model so to appear only the variable X_1:

$$Y = 2 + X_1 + X_2 + \varepsilon \iff Y = 2 + X_1 + (3X_1 + \varepsilon_1) + \varepsilon$$

which indicates that when $N \to \infty$ the model $\mathcal{A}$ will converge in such a way that:

$$\alpha_1 = \lim_{N\to\infty} \hat{\alpha}_1 = 4 \quad \text{and} \quad \alpha_0 = \lim_{N\to\infty} \hat{\alpha}_0 = 2 \quad \text{and} \quad \sigma_A^2 = \lim_{N\to\infty} \hat{\sigma}_A^2 = 2$$

Similarly, we can rewrite things so to get a model depending only of X_2 as per

$$Y = 2 + \left(\frac{X_2}{3} - \frac{\varepsilon_2}{3} \right) + \varepsilon = 2 + \frac{4}{3} X_2 + \left(\varepsilon - \frac{\varepsilon_2}{3} \right)$$

which indicates that when $N \to \infty$ the model $\mathcal{A}$ will converge in such a way that:

$$\beta_2 = \lim_{N\to\infty} \hat{\beta}_2 = \frac{4}{3} \quad \text{and} \quad \beta_0 = \lim_{N\to\infty} \hat{\beta}_0 = 2 \quad \text{and} \quad \sigma_B^2 = \lim_{N\to\infty} \hat{\sigma}_B^2 = \frac{10}{9}$$

Running the following script we get the estimates for $N = 10^4$

```
N = 10_000
X = np.zeros((N, 2))
X[:, 0] = np.random.randn(N)
X[:, 1] = 3 * X[:, 0] + np.random.randn(N)
Y = X[:, 0] + X[:, 1] + 2 + np.random.randn(N)

df = pd.DataFrame()
df['Y'] = Y
df['intercept'] = np.ones(N)
df['X1'] = X[:, 0]
model_A = sm.OLS(df['Y'], df[['intercept', 'X1']])
results = model_A.fit()
print('model A')
```

```
## model A
```

```
print(results.params)
```

```
## intercept    2.017743
## X1           4.025612
## dtype: float64
```

```
print('sigma2_A = ', results.scale)
```

```
## sigma2_A =  1.96628446603217
```

```
print('')
```

```
df = pd.DataFrame()
df['Y'] = Y
df['intercept'] = np.ones(N)
df['X2'] = X[:, 1]
model_B = sm.OLS(df['Y'], df[['intercept', 'X2']])
results = model_B.fit()
print('model B')
```

```
## model B
```

```
print(results.params)
```

```
## intercept    1.993373
## X2           1.304322
## dtype: float64
```

```
print('sigma2_B = ', results.scale)
```

```
## sigma2_B =  1.092014440032445
```

(c) Let us now consider the full model

$$Y_i = \gamma_2 X_{2,i} + \gamma_1 X_{1,i} + \gamma_0 + \varepsilon_i$$

where $i \in \{1, \dots, n\}$ and the ε_i are independent $\mathcal{N}(0, \sigma^2)$ random variables. For the previously simulated data with $n = 10$, estimate $\hat{\gamma}_0, \hat{\gamma}_1, \hat{\gamma}_2, \hat{\sigma}^2$ and compare them with the parameters obtained in item (b). What can you say about the effects of X_1 and X_2 on Y? And about their correlation?

Running the estimation with the full model but only $N = 10$:

```
N = 10
X = np.zeros((N, 2))
X[:, 0] = np.random.randn(N)
X[:, 1] = 3 * X[:, 0] + np.random.randn(N)
Y = X[:, 0] + X[:, 1] + 2 + np.random.randn(N)

df = pd.DataFrame()
df['Y'] = Y
df['intercept'] = np.ones(N)
df['X1'] = X[:, 0]
df['X2'] = X[:, 1]
model = sm.OLS(df['Y'], df[['intercept', 'X1', 'X2']])
results = model.fit()
print('full model')
```

```
## full model
```

```python
print(results.params)
```

```
## intercept    1.825560
## X1           2.611279
## X2           0.590033
## dtype: float64
```

```python
print('sigma2 = ', results.scale)
```

```
## sigma2 =  0.6198383421868624
```

We see that the estimates are quite far from the true values. We can also inspect the summary to see the standard errors of the estimates:

```python
print(results.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      Y   R-squared:                       0.963
## Model:                            OLS   Adj. R-squared:                  0.953
## Method:                 Least Squares   F-statistic:                     91.31
## Date:                Sun, 16 Feb 2025   Prob (F-statistic):           9.67e-06
## Time:                        15:55:18   Log-Likelihood:                -10.015
## No. Observations:                  10   AIC:                             26.03
## Df Residuals:                       7   BIC:                             26.94
## Df Model:                           2
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## intercept      1.8256      0.255      7.173      0.000       1.224       2.427
## X1             2.6113      0.835      3.129      0.017       0.638       4.585
## X2             0.5900      0.294      2.004      0.085      -0.106       1.286
## ==============================================================================
## Omnibus:                        1.431   Durbin-Watson:                   2.175
## Prob(Omnibus):                  0.489   Jarque-Bera (JB):                0.288
## Skew:                          -0.415   Prob(JB):                        0.866
## Kurtosis:                       3.069   Cond. No.                         8.44
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
##
## /opt/homebrew/Caskroom/miniforge/base/envs/isla2025/lib/python3.11/site-packages/scipy/stats/_
##   warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```