practical_exercise_7 , Methods 3, 2021, autumn semester

[FILL IN YOUR NAME]

[FILL IN THE DATE]

Exercises and objectives

1) Estimate bias and variance based on a true underlying function

2) Fitting training data and applying it to test sets with and without regularization

For each question and sub-question, please indicate one of the three following answers:
i. I understood what was required of me
ii. I understood what was required of me, but I did not know how to fulfil the requirement
iii. I did not understand what was required of me

EXERCISE 1 - Estimate bias and variance based on a true underlying function

We can express regression as $y = f(x) + \epsilon$ with $E[\epsilon] = 0$ and $var(\epsilon) = \sigma^2$ (E means expected value)

For a given point: x_0, we can decompose the expected prediction error , $E[(y_0 - \hat{f}(x_0))^2]$ into three parts - **bias**, **variance** and **irreducible error** (the first two together are the **reducible error**):

The expected prediction error is, which we also call the **Mean Squared Error**:
$E[(y_0 - \hat{f}(x_0))^2] = bias(\hat{f}(x_0))^2 + var(\hat{f}(x_0)) + \sigma^2$

where **bias** is;

$bias(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$

1) Create a function, $f(x)$ that squares its input. This is our **true** function
   i. generate data, y, based on an input range of [0, 6] with a spacing of 0.1. Call this x
   ii. add normally distributed noise to y with $\sigma = 5$ (set a seed to 7 `np.random.seed(7)`) to y and call it y_{noisy}
   iii. plot the true function and the generated points

2) Fit a linear regression using `LinearRegression` from `sklearn.linear_model` based on y_{noisy} and x (see code chunk below associated with Exercise 1.2)
   i. plot the fitted line (see the `.intercept_` and `.coef_` attributes of the `regressor` object) on top of the plot (from 1.1.iii)
   ii. now run the code chunk below associated with Exercise 1.2.ii - what does X_quadratic amount to?
   iii. do a quadratic and a fifth order fit as well and plot them (on top of the plot from 1.2.i)
3) Simulate 100 samples, each with sample size `len(x)` with $\sigma = 5$ normally distributed noise added on top of the true function
   i. do linear, quadratic and fifth-order fits for each of the 100 samples

   ii. create a **new** figure, `plt.figure`, and plot the linear and the quadratic fits (colour them appropriately); highlight the true value for $x_0 = 3$. From the graphics alone, judge which fit has the highest bias and which has the highest variance for x_0

iii. create a **new** figure, `plt.figure`, and plot the quadratic and the fifth-order fits (colour them appropriately); highlight the true value for $x_0 = 3$. From the graphics alone, judge which fit has the highest bias and which has the highest variance for x_0

iv. estimate the **bias** and **variance** at x_0 for the linear, the quadratic and the fifth-order fits (the expected value $E[\hat{f}(x_0)] - f(x_0)$ is found by taking the mean of all the simulated, $\hat{f}(x_0)$, differences)

v. show how the **squared bias** and the **variance** is related to the complexity of the fitted models

vi. simulate **epsilon**: `epsilon = np.random.normal(scale=5, size=100)`. Based on your simulated values of **bias, variance and epsilon**, what is the **Mean Squared Error** for each of the three fits? Which fit is better according to this measure?

```
# Exercise 1.2
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit() ## what goes in here?
```

```
# Exercise 1.2.ii
from sklearn.preprocessing import PolynomialFeatures
quadratic = PolynomialFeatures(degree=2)
X_quadratic = quadratic.fit_transform(x.reshape(-1, 1))
regressor = LinearRegression()
regressor.fit() # what goes in here?
y_quadratic_hat # calculate this
```

EXERCISE 2: Fitting training data and applying it to test sets with and without regularization

All references to pages are made to this book: Raschka, S., 2015. Python Machine Learning. Packt Publishing Ltd.

1) Import the housing dataset using the upper chunk of code from p. 280
   i. and define the correlation matrix `cm` as done on p. 284

   ii. based on this matrix, do you expect collinearity can be an issue if we run multiple linear regression by fitting MEDV on LSTAT, INDUS, NOX and RM?

2) Fit MEDV on LSTAT, INDUS, NOX and RM (standardize all five variables by using `StandardScaler.fit_transform`, (`from sklearn.preprocessing import StandardScaler`) by doing multiple linear regression using `LinearRegressionGD` as defined on pp. 285-286
   i. how much does the solution improve in terms of the cost function if you go through 40 iterations instead of the default of 20 iterations?

   ii. how does the residual sum of squares based on the analytic solution (Ordinary Least Squares) compare to the cost after 40 iterations?
   iii. Bonus question: how many iterations do you need before the Ordinary Least Squares and the Gradient Descent solutions result in numerically identical residual sums of squares?

3) Build your own cross-validator function. This function should randomly split the data into k equally sized folds (see figure p. 176) (see the code chunk associated with exercise 2.3). It should also return the Mean Squared Error for each of the folds

   i. Cross-validate the fits of your model from Exercise 2.2. Run 11 folds and run 500 iterations for each fit

   ii. What is the mean of the mean squared errors over all 11 folds?

4) Now, we will do a Ridge Regression. Use `Ridge` (see code chunk associated with Exercise 2.4) to find the optimal `alpha` parameter (λ)
   i. Find the *MSE* (the mean of the *MSE's* associated with each fold) associated with a reasonable range of `alpha` values (you need to find the lambda that results in the minimum *MSE*)

   ii. Plot the *MSE* as a function of `alpha` (λ). Make sure to include an *MSE* for `alpha=0` as well

   iii. Find the *MSE* for the optimal `alpha`, compare its *MSE* to that of the OLS regression
   iv. Do the same steps for Lasso Regression `Lasso` (2.4.i.-2.4.iii.)
   v. Describe the differences between these three models, (the optimal Lasso, the optimal Ridge and the OLS)

```
# Exercise 2.3
def cross_validate(estimator, X, y, k): # estimator is the object created by initialising LinearRegress
    mses = list() # we want to return k mean squared errors
    fold_size = y.shape[0] // k # we do integer division to get a whole number of samples
    for fold in range(k): # loop through each of the folds

        X_train = ?
        y_train = ?
        X_test = ?
        y_test = ?

        # fit training data
        # predict on test data
        # calculate MSE

    return mses
```

```
# Exercise 2.4
from sklearn.linear_model import Ridge, Lasso
RR = Ridge(alpha=?)
LassoR = Lasso(alpha)
```