Line H. Clemmensen
Section of Statistics and Data Analysis
DTU Compute

Exercises 02582
Module 1
Spring 2019

February 6, 2019

Topics: OLS - Ridge - Bias - Variance

Exercises (programming hints at end of exercises):

1 Solve the Ordinary Least Squares (OLS) computationally (for the diabetes data set). Note, that data have been mean centered and normalized.

(a) What is the difference between inverting the matrix and calculating the OLS solution directly and a numerically 'smarter' implementation which solves a linear system of equations? Compute the ordinary least squares solution to the diabetes data set for both options and look at the relative difference.

(i) In Matlab: Use for example `inv` and Matlab's backslash operator (for info type `help mldivide`).

(ii) In R: Use for example `inv` to invert the matrix and `solve` to solve the linear system of equation.

(iii) In Python: Use for example `np.linalg.inv` to invert the matrix and `lng.lstsq` to solve the linear system of equation.

(b) How could you include an intercept term in Matlab/R/Python in the matrix/vector formulation? This should be equivalent to using the model $\mathbf{y} = \beta_0 + \mathbf{x}_1\beta_1 + ... + \mathbf{x}_p\beta_p + \epsilon$ rather than $\mathbf{y} = \mathbf{x}_1\beta_1 + ... + \mathbf{x}_p\beta_p + \epsilon$.

(c) Evaluate the mean squared error $MSE = 1/n \sum_{i=1}^{n}(y_i - \mathbf{x}_i\boldsymbol{\beta})^2$.

(d) Evaluate the residual sum of squares $RSS = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$ and the total sum of squares $TSS = \|\mathbf{y} - \bar{y}\|_2^2$, where $\bar{y}$ is the estimated mean of $\mathbf{y}$. Report on the R^2 measure, that is, the proportion of variance in the sample set explained by the model: $R^2 = 1 - \frac{RSS}{TSS}$.

2 Examine Bias and Variance for the OLS:

(a) Investigate the unbiasedness of OLS using simulated data as follows.

(i) Create a random matrix $\mathbf{X}$ consisting of three random variables which are NID, with $X \sim N(\mathbf{0}, \mathbf{I})$ and sample 10 observations from each.

(ii) Create the true regression parameters $\boldsymbol{\beta}_{true} = [1, 2, 3]^T$.

(iii) Create the response $\mathbf{y}$ by the linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Make sure the errors $\boldsymbol{\epsilon}$ are homoscedastic and zero-mean $\epsilon \sim N(0, \sigma^2)$, where σ^2 denotes the noise level in the simulated data (you can start by setting $\sigma^2 = 0.1$).

(iv) Estimate the regression coefficients $\boldsymbol{\beta}$ from this data

(v) Repeat steps (iii)-(iv) 100 times. Note, that we can resample data as we have simualted data, this is not always possible for real examples.

(vi) Use meaningful plots to investigate bias and variance for the model. What happens when you change the noise level σ^2?

3 Solve the Ridge regression problem and examine Bias and Variance for Ridge:

a) Derive (using pen and paper) the ridge regression solution by, as you would when minimizing any differentiable analytical function, differentiating $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\| + \|\boldsymbol{\beta}\|_2$ with respect to $\boldsymbol{\beta}$, setting to zero and solving for $\boldsymbol{\beta}$. That is, solve $\frac{\partial}{\partial \beta} \left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2\right] = 0$ for $\boldsymbol{\beta}$.

b) Compute ridge-regression solutions for the diabetes data set for 100 values of the regularization parameter in the range 10^{-4} to 10^3. Plot the solutions as a function of this parameter. In the next lecture you will learn how to choose a single parameter value which suits the problem at hand.

c) Change the experiment in exercise 2 to investigate bias and variance for ridge regression instead of OLS. Can you lower the variance without introducing too much bias?

4 Implement and solve KNN regression:

(a) Implement a KNN regression algorithm:

(i) Find the K nearest neighbours using a suitable distance metric (e.g. Euclidean).

(ii) Optional: Compute weights for the neighbours as the proportion of its distance to the total distance for the K nearest neighbours.

(iii) Compute the predicted response as the (weighted) mean of the K neighbours.

(b) Find a solution to the diabetes data using KNN regression. Try different options for K - what happens?

Line H. Clemmensen
Section of Statistics and Data Analysis
DTU Compute

Exercises 02582
Module 1
Spring 2019

February 6, 2019

Topics: OLS - Ridge - Bias - Variance

Resources for this exercise:

Listing 1: Resources in Matlab

```
readtable('DiabetesDataNormalized.txt') % load data in the .txt file
        % (from CampusNet)
y = T{:,11}; % response
X = T{:,1:10}; % matrix with variables
whos % lists variables in memory - check what you have loaded (X and y)
rand(n, p); % simulate random uniform data
randn(n, 1);  % simulate random gaussian data
logspace(-4, 3, 100) % 100 values on the logarithmic scale
        % from 1e-4 to 1e3
knnsearch(X, X(i,:), 'K', K+1) % find the K nearest neighbours.
```

Listing 2: Resources in R

```r
library('lars') # load lars library (includes diabetes data)
data(diabetes) # load data
str(diabetes$x) # list structure of x
str(diabetes$y) # list structure of y
runif(n*p, min = 0, max = 1) # simulate random uniform data
rnorm(n, mean=0, sd = 1) # simulate random normal data
library('pracma') # load pracma library for 'logspace'
logspace(-4, 3, 100) # 100 values on the logarithmic scale from 1e-4 to 1e
get.knnx(X, query, k=K+1, ...) # find the K nearest neighbors
```

Listing 3: Resources in Python

```python
import numpy as np # numpy library
import scipy.linalg as lng # linear algebra from scipy library
import matplotlib.pyplot as plt # library for plots
from scipy.spatial import distance # load distance function
from sklearn import preprocessing as preproc # load preproc function

T = np.loadtxt('DiabetesDataNormalized.txt', delimiter = ' ',
skiprows = 1) # load data in the .txt file (from CampusNet)
y = T[:, 10] # response
X = T[:,:10] # variables
np.random.randn(n, p) # simulate random gaussian data
np.random.rand(n, 1) # simulate random uniform data
np.logspace(-4, 3, k) # 100 values on the logarithmic scale
        # from 1e-4 to 1e3
preproc.scale(X) # normalize to zero mean and unit variance
distance.euclidean(X[i,:], X[j, :]) # compute distance between two points
```

End of exercise