Exercises Statistical Learning - Week 1

Julian D. Karch

February 9, 2023

1 Inference vs. Prediction

Do Exercise 2 (Page 52) from the book.

2 Bias can be beneficial

For the remaining exercises, first download the exercises folder `01-exercises-package.zip` from Brightspace.

Convince yourself that the R function `make_ex_1` available within the `functions.R` file available in the exercise folder implements the following algorithm.

1. Generate $n = 50$ training observations with a single predictor variable $x \sim \mathcal{U}(-3, 3)$
2. Generate the response $y = .1x + \epsilon$, with $\epsilon \sim \mathcal{N}(0, 1)$
3. Generate 10,000 test observations from the same distribution.
4. Estimate $\hat{\beta}_{OLS}$ on the training observations; exclude the intercept from the model formula;
5. Generate a vector of shrinkage coefficients: `s_vector <- seq(0, 1, by = 0.1)`.
6. For each shrinkage coefficient s within `s_vector` , generate predictions for the test observations by multiplying x and $\hat{\beta}_{OLS}$ and applying shrinkage: $\hat{y} = x \cdot s \cdot \hat{\beta}_{OLS}$.
7. For each value s, compute the test MSE: $\frac{1}{N} \sum (y - \hat{y})^2$.
8. Plot the test MSE values as a function of shrinkage s.

To eliminate effects due to chance fluctuations, repeat the above experiment 100 times, and plot the test MSEs (averaged over the 100 replications) as a function of shrinkage s. For each repetition, also save the obtained beta value. For each shrinkage coefficient, make a boxplot. Since in this case mean = median and inter-quartile range proportional to variance, the boxplot visualizes bias and variance for each shrinkage coefficient.

A similar function is provided for Python users within the `functions.py` file. However, contrary to the R code we did not check the Python code. So proceed with caution. This applies to all Python material we provide as the main language for the course is R. Should you find any errors in Python code, please correct them and send the corrected file to j.d.karch@fsw.leidenuniv.nl

Assignment: Execute the code. What is the best amount of shrinkage according to MSE? Explain why this is the case considering the plot visualizing bias and variance. Repeat the experiment but now with a larger training sample size, say $n = 100$. Again, what is the best amount of shrinkage? Again, explain why. Repeat but now increase the effect of X to $Y = .2X + \epsilon$.

3 Under- and overfitting with polynomial regression

Generate a training and test set (each of size 50) of data consisting of a single predictor $X \sim \mathcal{U}(-5, 5)$ and

$$Y = X + 8\sin(X/2) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, 1)$. Fit polynomial regression models to the training data of degree 1 to 15, make predictions on the test set and compute the test MSE for each degree. Plot the test MSE as a function of the degree of the polynomial. Create a plot which shows the training observations, and fitted curves for the degree 1, 2, 3 and 15 polynomials.

- Hints both: You can reuse parts of the code provide for Exercise 2.
- Hints R: Use `poly` to fit the model, use `predict` to generate predictions, and a `for` loop. Because you use the predict function, it is smartest to provide the generated variables with the same names in both training and test data and assign them to separate `data.frames`. E.g.: `x <- ...`
  ```
  y <- ...
  train <- data.frame(x = x, y = y)
  ```
- Hints Python: Use `LinearRegression` class again (see Exercise 1). Polynomial features are available via `from sklearn.preprocessing import PolynomialFeatures`. The `LinearRegression` class also has a predict function.

4 Curse of Dimensionality

Generate a dataset: $p = 10,000$ and $n = 100$. Further, $X \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; that is, the predictors follow a multivariate normal distribution with means 0, variances

1 and covariances 0. Create a histogram of the pairwise Euclidian distances between all points in the dataset, but first only use the first column of X to compute the distances (i.e., $p = 1$). Create another histogram, but now use the first two columns (i.e., $p = 2$). Repeat for $p \in \{10, 100, 1000, 10000\}$. Are the nearest neighbours near in 1-dimensional space? In 2-dimensional space? In 10-, 100-, 1000-, 10000-dimensional space?

- Hints R: use functions `dist` to compute pairwise distances, use function `hist` to create a histogram. Specify argument `xlim` for each histogram, to make sure the value of 0 is included on the x-axis.

- Hints Python: distance are avaible via `numpy.linalg.norm`, histograms via `matplotlib.pyplot`.

5 Classifying digits

This exercise is on recognizing handwritten digits. Observations are images of handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. Images have been deslanted and normalized, yielding 16x16 grayscale images. The predictor variables reflect the grayscale value for each pixel.

Compare the classification performance of linear regression and k-nearest neighbour classification on the `zipcode` data (`read_data.R` and `read_data.py` in exercise package). Specifically:

1. The zip-code training and test data and the R/Python code to load it are included in the exercise package.

2. Fit a GLM with `binomial family` using function `glm`. Regress the `V1` variable on all remaining variables in the dataset.

3. Use function `knn` from library **`class`** to apply kNN. Apply the function for $k = 1$, 3, 5, 7 and 15. Note: the `knn` function requires you to specify both training and test data when fitting the model and it has no `predict` method. So to compute training and test error, you have to apply the function to the training dataset twice.

4. Compute the misclassification error rates for training and test datasets, for each value of k.

5. Make a plot, where you show both the training and test error (y-axis for each choice of k (x-axis). Also indicate the performance of the GLM in the plot (e.g., using a horizontal line).

Classes for python: `LogisticRegression` within `sklearn.linear_model`, `from sklearn.neighbors import KNeighborsClassifier`