46-927, Fall 2017: Homework #1

Due 5:20PM, Monday, January 22, 2017.

Homework should be submitted electronically on canvas before the deadline. Please submit your homework as a single file (pdf or html). We recommend using Rmarkdown or jupyter notebooks to prepare your homework.

Note: For this homework, both R and python solutions are acceptable. I encourage you to try to use python, as we will be moving in that direction as we progress into more ML methods. However, your backgrounds in python will vary, this assignment is not intended to take too much time, and we will more time to get to know python as the course progresses.

Please post any questions on the Piazza discussion board. I recognize that having a homework due in the first class is a little undesirable, but it will let us get off to a quick start and avoid piling up homeworks in the second week.

Some general python hints

- New python users: python is more heavily class based so try to be open to the structural differences between it and R.
- If you are using python, matplotlib.plot is a nice graphing tool and in jupyter notebooks use the magic `%matplotlib inline` to get inline graphics without having to do `plt.show()`.
- Similar to R, `function.name?` is one way to ipython to give you the docstring information (ex: `scipy.stats.multivariate_normal?`).
- Like learning any new language, expect to use Google to figure out how to do new things. For example: fitting regression for this homework.

1. Get python, ipython, jupyter set up on your machine. Install `scipy`, `numpy`, `sklearn`. These will be useful for this homework if you do it in python (which I recommend trying), but will also be important going forward in class. *No need to submit anything for this problem. It's just a good thing to do.*

2. *Impact of correlated variables on parameter estimation and on prediction.* Let's build a very simple regression model with two variables and an outcome. Construct variables X_1, X_2 to be bivariate normal with standard deviation 1, mean 0, correlation ρ. Let $Y = X_1 + X_2 + \varepsilon$ where $\varepsilon \sim N(0,1)$. For all of this example, use $n = 500$ observations for fitting the models.

   (a) Vary ρ from 0 to 0.9 in steps of 0.1. For each value of ρ, generate 500 data points, fit a linear model, and find the variance of $\widehat{\beta}_1$. Plot these variances versus ρ. Is there a strong relationship between ρ and the coefficient variance?

   (b) For the same range of ρ, measure the average squared prediction error for predicting Y using the linear model you fit. Evaluate these errors using predictions on a new data set of size $n = 500$ (for each ρ). (That is, fit your model on one set of 500 points and evaluate it on a new set of 500 points.) Plot the prediction error versus ρ. Is there a strong relationship between ρ and the prediction error?

Hints:

- Everyone: Read both (a) and (b) before you code. You might find it helpful to create small helper functions.
- Python Users:
  - Recommended packages for new users: `scipy.stats`, `statsmodels`, `numpy`, [and `pandas` if desired],
  - to save some time - `sklearn` linear regression doesn't produce a lot of what you want (that I why I'm recommending `statsmodels` for this problem. (You'll probably still want to remember that $\mathrm{Cov}(\widehat{\beta}) = \sigma^2(XX^T)^{-1}$).

3. *What happens to linear regression in high dimensions?* Simulate the following setting: To generate each data point, generate independent variables $X_1, ..X_p \sim N(0,1)$, and let $Y = 4X_1 + \varepsilon$, where $\varepsilon \sim N(0,1)$. This means that the linear model assumption is *true* for this data, though there are many useless additional variables when p is large.

   Holding n fixed at 100 observations, vary the number of variables p from 1 to 80. For each setting, run 100 simulations of:

   (i) Draw $n = 100$ data points

   (ii) Fit a linear regression of Y on $X_1, \ldots, X_p$ using these data points. Call the resulting coefficient vector $\widehat{\beta}$.

(iii) Draw a test set of $m = 1000$ data points, compute predictions at these points using your estimated $\widehat{\beta}$, and compute mean squared prediction error of these points:
$\frac{1}{m}\sum_{i=1}^{m}(y_i - x_i^T\widehat{\beta})^2$

Plot average (over simulations) mean squared prediction error vs. p. What do you see?

> **Hint: Python users**
>
> If you have the time switch to `sklearn` for this problem, see `sklearn.linear_model` and `sklearn.metrics.mean_squared_error`.

4. *Error measures and prediction.* We're going to consider a very, very simple prediction problem.

   Suppose that you know I'm going to draw data points Y from an Exponential(1) distribution. You're going to try to predict the data points that I draw, using two possible estimators:

   I. The mean of an Exponential(1) (which happens to be 1)

   II. The median of an Exponential(1) (which happens to be log(2)).

   (a) Simulate 1000 points from an Exponential(1). What is the mean squared prediction error of estimators I and II? That is: $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{y})^2$, where $\widehat{y} \in \{1, \log(2)\}$. Which estimator does better for this error measure?

   (b) Simulate 1000 points from an Exponential(1). What is the median squared prediction error of estimators I and II? That is: $\frac{1}{n}\sum_{i=1}^{n}|Y_i - \widehat{y}|$, where $\widehat{y} \in \{1, \log(2)\}$. Which estimator does better for this error measure?