
Homework 9

Due: 7:00pm, Friday via Gradescope

Reading: Notes: Chapter 7, **Videos:** 7.1 - 7.3; Chapter 8, 8.1, 8.2; Chapter 9, 9.1-9.4

Gradescope Uploading Instructions: As always, please be sure to assign problems to PDF pages when you are uploading to Gradescope. We will no longer grade submissions that fail to do so. Detailed instructions are available in the Homework folder on Blackboard.

Problem 9.1 (**Video 8.1, 8.2**, **Lecture Problem**) A bank teller has a service time per customer that is a continuous random variable uniformly distributed between 1 and 5 minutes. Assume that the service times of customers: $T_1, T_2, \ldots$ are all independent. Let T be the total time to service 100 customers. That is, $T = \sum_{i=1}^{100} T_i$

(a) Compute $\mathbb{E}[T]$.

(b) Compute $\mathsf{Var}[T]$.

(c) Estimate the probability that $T > 330$ minutes in terms of the normal Gaussian error function $Q(\cdot)$ or the standard Gaussian CDF $\Phi(\cdot)$.

Problem 9.2 (**Video 8.1, 8.2**)

Each time Caitlin completes a hole at Finkbine Golf Course, she requires 1, 2, or 3 putts, each with equal probability. A good day on the course is to have 60 or fewer putts on 36 holes.

An answer in this question may use the standard Gaussian CDF $\Phi(\cdot)$.

(a) Using an approximation based on the central limit theorem, write an expression for the probability that Caitlin has a good day.

(b) If Caitlin plays n holes, where n is a large number, does her number of putts become very likely to be exactly $2n$? Precisely, if Y_n is her number of putts on n holes, does

$$\lim_{n\to\infty} \mathbb{P}[Y_n = 2n] = 1$$

hold? Explain why or why not.

Problem 9.3 (**Video 7.1, 7.2, 8.1, 8.2**, **Quick Calculations**) For each of the scenarios below, determine the requested quantities. (You should be able to do this without any long calculations or integration.)

(a) Assume that X is Uniform$(1, 3)$, V is Gaussian$(1, 1)$, X and V are independent, and $Y = X + V$. Determine the LLSE estimator of X given $Y = y$ and the corresponding mean-squared error.

(b) Let $X_1, X_2, \ldots, X_{400}$ be a collection of continuous, independent, identically distributed continuous random variables, each of which is uniformly distributed over $[-5, 5]$. Let $Y = \frac{1}{400}\sum_{i=1}^{400}(X_i)^3$. Compute $\mathbb{E}[Y]$ and $\mathsf{Var}[Y]$.

(c) Let $X_1, X_2, \ldots, X_{100}$ be a collection of continuous, independent, Exponential($\frac{1}{2}$ random variables. Let $Y = \frac{1}{100}\sum_{i=1}^{100} X_i$. Use the Central Limit Theorem to estimate $\mathbb{P}[Y \leq 1.9]$.

Problem 9.4 (**Video 8.1, 8.2, 9.1, 9.2**, **Lecture Problem**)

Let $X_1, \ldots, X_n$ be i.i.d. random variables with PMF

$$P_X(x) = \begin{cases} 1/2 & x = 2 \\ 1/2 & x = 4 \\ 0 & \text{otherwise.} \end{cases}$$

Let $S_{300} = X_1 + \cdots + X_{300}$.

(a) Determine the mean of and variance of S_{300}.

(b) Use the Central Limit Theorem approximation to estimate the probability $\mathbb{P}\big[\big|S_{300} - E[S_{300}]\big| \geq 40\big]$. You can leave your answer in terms of the standard normal CDF $\Phi(z)$.

(c) Suppose you did not know the PMF $P_X(x)$, but you knew that the standard deviation of the random variables X_k was one. You measure $X_1, \ldots, X_{300}$ and compute the sample mean $M_{300} = \frac{1}{300}\sum_{k=1}^{300} X_k$. Find a symmetric confidence interval for the true mean around the observed value M_{300} with confidence level 0.95.

Use the following assumptions: $Q(1.28) = 1 - \Phi(1.28) = 0.1$; $Q(1.645) = 1 - \Phi(1.645) = 0.05$; $Q(1.96) = 1 - \Phi(1.96) = 0.025$.

(d) Suppose you also did not know the standard deviation, but you were able to compute the sample variance of the 300 samples X_k as $V_{300} = 1.21$. Find a symmetric confidence interval for the true mean around the observed value M_{300} with confidence level 0.95.

Use the following assumptions: if W has a t-distribution with 299 degrees of freedom, its CDF satisfies: $F_W(-1.9679) = 0.025; F_W(-1.65) = 0.05; F_W(-1.2844) = 0.05$.

Problem 9.5 (**Video 8.1, 8.2, 9.1, 9.2**, **Lecture Problem**)

You are working with an immortal cell line that is used in labs across the world and known to have a mean radius of $5.02\mu m$ with a standard deviation of $0.50\mu m$. You collect 100 samples, and assume that these are well-modeled as i.i.d. Gaussian random variables.

You may find one or more of the following values useful: $Q(1.28) = 1-\Phi(1.28) = 0.1$; $Q(1.645) = 1 - \Phi(1.645) = 0.05$; $Q(1.96) = 1 - \Phi(1.96) = 0.025$.

(a) What is the variance of the sample mean $\mathsf{Var}[M_{100}]$?

(b) If your sample mean is $5.10\mu m$ is your sample significantly different from the baseline model at a significance level of 0.05? Justify your approach and support your answer numerically.

(c) Say you do not know the true mean (but you still know the true standard deviation is $0.50\mu m$). You observe a sample mean of $5.10\mu m$. Construct a confidence interval centered around the sample mean with confidence level 0.9 for the true mean.

Problem 9.6 (**Video 7.3**, **Exploring Data**)

In this problem, we will apply the LLSE estimation framework to predict the compressive strength of concrete. Specifically, we will use the Concrete Compressive Strength Data Set created by I.-C. Yeh (initially for the paper [1]) and hosted by the UCI Machine Learning Repository [2].

The dataset consists of 1030 concrete samples (each occupying one row of the data matrix), the following 9 features (each occupying the corresponding column of the data matrix)

1. Cement (kg/m^3),
2. Fly ash (kg/m^3),
3. Blast furnace slag (kg/m^3),
4. Water (kg/m^3),
5. Superplasticizer (kg/m^3),
6. Coarse aggregate (kg/m^3),
7. Fine aggregate (kg/m^3),
8. Age at test time (days),
9. Compressive strength (in MPa).

We are going to develop an estimator of the compressive strength (the data in column 9) based on the values in the first 8 features.

In the Homework 9 folder, you will find the comma-separated value (CSV) file `concretedata.csv`. The first row of the file contains the title of the feature in each column, and the next 1030 lines contain the data for samples of concrete. You will also find a script (`hw9_P6.m` or `hw9_P6.ipynb` for MATLAB or Python). The script will load the data into Python environment, create the 1030×1 column vector `X` consisting of the compressive strength values we wish to predict, and the 1030×8 matrix `Y` consisting of the observations we will use as inputs. Note that the first row of the file contains the names of the columns, and is not used as data.

Your first task will be to design and implement an LLSE function that takes an input set of observation samples `Y` and true value samples `X`. The function will compute the LLSE estimate of X given $\underline{Y}$ for each sample value of Y, and also compute the mean squared error in the approximation. The incomplete function is included in the scripts provided. Recall, from Video 7.3, that the vector LLSE estimator is $\hat{x}_{\text{LLSE}}(\underline{y}) = \mu_X + \mathbf{\Sigma}_{X,\underline{Y}}\mathbf{\Sigma}_{\underline{Y}}^{-1}(\underline{y} - \underline{\mu}_{\underline{Y}})$ where μ_X is the mean of X, $\underline{\mu}_{\underline{Y}}$ is the mean vector of $\underline{Y}$, $\mathbf{\Sigma}_{X,\underline{Y}}$ is the cross-covariance matrix of X and $\underline{Y}$, and $\mathbf{\Sigma}_{\underline{Y}}$ is the covariance matrix of $\underline{Y}$.

(a) Your first task will be to design and implement an LLSE function that takes an input set of observation samples `Y` and true value samples `X`. The function will compute the LLSE

[1] Yeh, I-Cheng, "Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research, Vol. 28, No. 12, pp. 1797-1808 (1998).
[2] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [`http://archive.ics.uci.edu/ml`]. Irvine, CA: University of California, School of Information and Computer Science.

estimate of X given $\underline{Y}$ for each sample value of Y, and also compute the mean squared error in the approximation. The incomplete function is included in the scripts provided. Recall, from Video 7.3, that the vector LLSE estimator is $\hat{x}_{\text{LLSE}}(\underline{y}) = \mu_X + \mathbf{\Sigma}_{X,\underline{Y}}\mathbf{\Sigma}_{\underline{Y}}^{-1}(\underline{y} - \underline{\mu_Y})$ where μ_X is the mean of X, $\underline{\mu_Y}$ is the mean vector of $\underline{Y}$, $\mathbf{\Sigma}_{X,\underline{Y}}$ is the cross-covariance matrix of X and $\underline{Y}$, and $\mathbf{\Sigma}_{\underline{Y}}$ is the covariance matrix of $\underline{Y}$.

The function will use the samples `Y` and true value samples `X` to estimate the following data: $\mu_X, \underline{\mu_Y}, \mathbf{\Sigma}_{X,\underline{Y}}, \mathbf{\Sigma}_{\underline{Y}}, and \mathbf{\Sigma}_X$. The scripts contain directions for doing this. The function will also compute the predicted MMSE error, using the LLSE formula $\mathbf{\Sigma}_e = \mathbf{\Sigma}_{\underline{X}} - \mathbf{\Sigma}_{X,\underline{Y}}\mathbf{\Sigma}_{\underline{Y}}^{-1}\mathbf{\Sigma}_{X,\underline{Y}}^T$.

Using these matrices, the function will then predict the estimate $Xhat(i)$ for each sample data vector $\underline{Y}(i)$, in a do loop, using

$$Xhat(i) = \mu_X + \mathbf{\Sigma}_{X,\underline{Y}}\mathbf{\Sigma}_{\underline{Y}}^{-1}(\underline{Y}(i) - \underline{\mu_Y})$$

In our notation, recognize that vectors are assumed to be column vectors. However, the data $\underline{Y}(i)$ is in the i-th row of `Y`, and the sample average $\underline{\mu_Y}$ is also a row vector (in MATLAB, mostly; Python avoids this issue). Thus, be careful to transpose the data vector and the estimated mean vector so that they are column vectors in the function.

The LLSE function should return the vector of estimates Xhat and the theoretical MSE error $\mathbf{\Sigma}_e$.

To Submit: Your completed LLSE function.

(b) The empirical performance of the LLSE estimator can be measured using the mean-squared error (MSE), which we will estimate from data as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2$$

where n is the number of rows in lstinlineX.

In addition, it can be useful to report the performance in terms of a "scale-free" metric. One popular choice is the *coefficient of determination* R^2, which is the fraction of the variance of X explained by our estimate $\hat{X}$. We will estimate this from data as

$$R^2 = 1 - \frac{\text{MSE}}{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_X)^2} \ .$$

(This turns out to be equal to the square of the correlation coefficient between X and $\hat{X}$, $R^2 = \rho^2_{X,\hat{X}}$. Work this out on your own if you are interested.)

In the scripts, we provide a function `ComputePerformance` that takes as inputs X, $Xhat$ and returns MSE and R^2. Fill in the missing lines of code to estimate MSE and R^2.

To Submit: Your completed ComputePerformance function.

(c) With the completed `LLSE(X,Y)` function and `ComputePerformance(X,Xhat)` function, create the vector LLSE estimate of `X` using all 8 features in `Y`. Note the values of SigmaE, MSE and R^2. Create a scatter plot of your estimate `Xhat` versus `X`. Label the plot "Vector LLSE Prediction", the vertical axis as Predictive Compressive Strength (MPa), and the horizontal axis as True Compressive Strength (MPa).

To Submit: The scatter plot, the values of SigmaE, MSE and R^2.

(d) Do we need all 8 features from `Y` to predict `X`? Let's try to estimate `X` using only the first three features. Using your completed `LLSE(X,Y)` function, estimate `X` using data only from the first three columns of `Y`, and compute the resulting theoretical MSE value, empirical MSE value, R^2 value, estimate `Xhat`. Create a scatter plot of the resulting estimate `Xhat` versus `X` as before.

To Submit: The scatter plot, the values of theoretical and empirical MSE and R^2

(e) We can think of our previous work as finding the best linear function (plus offset)

$$\hat{x}_{\text{LLSE}}(\underline{Y}) = a_0 + \sum_{i=1}^{8} a_i Y_i$$

to minimize the squared-error. However, we knew going in that the compressive strength is best modeled as a non-linear function of the features. Is there an easy way to quickly fit a non-linear function? For example, consider the following quadratic function:

$$\hat{x}_{\text{LLSE}}(\underline{Y}) = a_0 + \sum_{i=1}^{8} a_i Y_i + \sum_{i=1}^{8} b_i Y_i^2$$

We can use our `LLSE(X,Y)` function to fit this quadratic function by augmenting our feature data with the quadratic of the feature data. In Python, we do this as Yaug = `np.column_stack((Y,Y**2))`; in MATLAB, we do it by Yaug = `[Y,Y.^2]`. We then use `Yaug` as our input to LLSE in place of `Y`.[1] Note the resulting values of theoretical and empirical MSE and R^2. Create a scatter plot of your estimate `Xhat` versus `X` as before. Comment on whether this is a better fit to the data than your linear fit from part (d).

To Submit: Your comments, the scatter plot, the values of theoretical and empirical MSE and R^2.

[1] Conceptually, we are first creating "new" random variables $Y_9 = Y_1^2$, $Y_{10} = Y_2^2$, ..., $Y_{16} = Y_8^2$ and then using these 16 random variables to form a linear estimate.