Homework: Functional Programming using Racket

Due-date: Mar 1 at 11:59pm; Submit online on Canvas
(**Submit one Racket file named as hw3-⟨net-id⟩.rkt**)

Homework must be individual's original work. Collaborations and discussions of any form with any students or other faculty members are not allowed. If you have any questions and/or concerns, post them on Piazza and/or ask 342 instructor and TAs.

Learning Outcomes

- Application of knowledge of computing and mathematics
- Knowledge and application of Functional Programming
- Ability to follow requirement specification

Programming Rules

- Submit one Racket file named as hw3-⟨net-id⟩.rkt.
- ```
  #lang racket
  (provide (all-defined-out)) ;; for us to test
  (require racket/trace)      ;; in case you want to use tracing
  (require "PTS.rkt")          ;; to access the database of points
  ```
- You are **only allowed** to use functions, if-then-else, cond, list operations, operations on numbers. No imperative-style constructs, such as begin-end or explicitly variable assignments, such as get/set are allowed. If you do not follow the guidelines, your submission will not be graded. If you are in doubt that you are using some construct that may violate rules, please contact instructor/TA (post on Piazza).
- You are expected to test your code extensively. If your implementation fails any assessment test, then points will be deducted. Almost correct is equivalent to incorrect solution for which partial credits, if any, will depend only on the number of assessment tests it successfully passes.
- The focus is on correct implementation. In this assignment, we will not assess the efficiency; however, avoid re-computations and use tail-recursion, if possible.
- Please comment your test code (e.g., trace directives, test cases) in your submission file.
  ```

Questions

Given two variables, x (explanatory variable) and y (dependent variable), linear regression analysis aims to model the dependency of y on x (using a line, hence the name linear).

For the purpose of discussion of this problem-specification, consider the coordinates in a 2-D plan, where the x-coordinates of the points indicate the valuations of x-variable and the y-coordinates of the points indicates the corresponding valuations of y variable. For instance, the data contains n points in the plane, represented as $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. For linear regression of y on x, our objective is to obtain a line $\hat{y} = mx + c$, in particular the values of m (slope) and c (intercept), such that the points on this line "closely" reflect the data. That is, $\hat{y}_i = mx_i + c$ is a reflection of y_i when x is equal to x_i.

To realize the objective, we need to minimize the overall error E in predicting the value of y for a given value of x, i.e.,

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + c))^2 \tag{1}$$

1. The result is the least squares method for computing the values of m and c, which minimizes E.

   The method computes m and c as follows:

   $$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

   and

   $$c = \bar{y} - m\bar{x}$$

   In the above, $\bar{x}$ and $\bar{y}$ denote the mean of the x_is and y_is respectively.

   For example, if the points are $(0, 0), (1, 1), (2, 2)$, then $m = 1$ and $c = 0$. If the points are $(1, 0), (2, 1), (3, 2)$, then $m = 1$ and $c = -1$.

   You are given a set of points in a Racket file PTS.rkt:

   ```
   (define pts1 '((0 1) (2 3) (1 0) (3 3) (5 0)))
   (define pts2 '((0 0) (1 1) (2 2) (3 3)))
   (define pts3 '((1 0) (2 1) (3 2) (4 3)))
   ```

   Write a function `compute_mc` that takes as arguments the list of points and returns a list containing two elements: the first element is the valuation of m and the second that of c. `compute_mc` implements the least squares method. For example,

   ```
   > (compute_mc pts1)
   '(-0.02702702702702699 1.4594594594594592)
   > (compute_mc pts2)
   '(1.0 0.0)
   > (compute_mc pts3)
   '(1.0 -1.0)
   ```

2. We have applied least squares method for regression analysis in the previous question. However, this method may not be suitable for situation, where the number of dependent variables is greater than one. A general technique is based on iteratively refining m and c, starting from some approximate solutions (e.g., 0 for both m and c), with the objective of reducing the E.

   The iterative refinement takes into consideration three parameters that decide the termination condition: the learning rate L, the error-margin e and the maximum number of times the refinement can be performed cnt. Intuitively, L describes the degree of refinement in each step and e describes the closeness of the refined result to the expected result. The steps of the iterative refinement method is as follows:

   (a) Initialize m and c both to 0.0.

   (b) if E (as per Equation 1) is less than e (error margin) or the number of iteration already performed is greater than equal to cnt, then terminate. The current values of m and c are used to describe the regression line.

   (c) Otherwise,

      i. update the value of m to $m - L \times \frac{dE}{dm}$, where

$$\frac{dE}{dm} = \frac{-2}{n} \sum_{i=1}^{n} x_i(y_i - \hat{y_i})$$

      ii. update the value of c to $c - L \times \frac{dE}{dc}$, where

$$\frac{dE}{dc} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})$$

   (d) Repeat from step (b).

   This technique is called *gradient descent*.

   Write a function `gradient_mc` that takes as argument the list of points, the parameters corresponding to L and e, and the parameter cnt corresponding to the maximum number of times the iterative refinement can be performed, and returns a list containing three elements: the first element is the valuation of m, the second that of c, and the third element is E corresponding the valuations of m and c. The function implements gradient descent iterative technique as described above. For example for the same set of points `pts1, pts2, pts3`,

```
> (gradient_mc pts1 0.001 0.01 5000)
'(-0.015042330628280118 1.4188766008006697 1.8384651009912811)

> (gradient_mc pts2 0.001 0.01 5000)
'(0.9222504727270914 0.16598399730907806 0.009992616854817024)

> (gradient_mc pts3 0.001 0.01 5000)
'(0.908583804857839 -0.7312252383800322 0.012064947703199078)
```