Computing for Data Sciences – 2017

PGDBA, First Year, First Semester, Indian Statistical Institute

Assignment 2

Posted on 31 August 2017 | Clarify by 8 September 2017 | Submit by 18 September 2017

Problem 1 [30 points]

Background: Given an n-sample-p-feature training set $\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, y^{(i)}\}$ for $i = 1, \ldots, n$, one may assume an initial linear relation between the features $\{x_1, x_2, \ldots, x_p\}$ and y as follows:

$$y \approx h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p,$$

and try to learn the optimal values of the relational coefficients $\theta = \{\theta_0, \theta_1, \ldots, \theta_p\}$ using an *ordinary least squares* model for linear regression, which minimizes the following cost function:

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2n}\sum_{i=1}^{n}\left(\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_p x_p^{(i)} - y^{(i)}\right)^2.$$

Batch Gradient Descent is a well-known representative for a broad class of iterative algorithms for minimizing $J(\theta)$, based on the least mean squares (LMS) update rule, also known as the Widrow-Hoff learning rule, with the parameter α controlling the learning rate, as follows:

$$\theta_j \leftarrow \theta_j - \alpha\frac{\partial J}{\partial \theta_j} = \theta_j - \alpha \cdot \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} \qquad \text{for each } j = 0, 1, \ldots, p.$$

Task : Write a Python function – `BatchGradientDescent()` – as discussed above, which takes into consideration *all* n training samples $\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, y^{(i)}\}$ at every iteration of the update rule for *each* parameter θ_j (denoted by the summation). Your function should take as input the training dataset $\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, y^{(i)}\}$, the learning rate α, the initial value $\theta = \{\theta_0, \theta_1, \ldots, \theta_p\}$, and an accuracy margin ϵ that controls the termination of the algorithm. The function should output the final value of the coefficients $\theta = \{\theta_0, \theta_1, \ldots, \theta_p\}$ once the accuracy margin ϵ is met.

Problem 2 [20 points]

Background : Notice that in the batch gradient descent algorithm, each update of θ requires you to scan through the entire set of n training samples (note the summation), making it slow for large n. One may modify the algorithm for gradient descent so that it starts updating θ_j (for every j) by scanning a single training sample at a time, and continue this way for the complete training set. This modified version of the algorithm is also known as *stochastic gradient descent.*

Task : Write a Python function – `StochasticGradientDescent()` – as above, which takes into account *one* of the n training samples $\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, y^{(i)}\}$ at every iteration of the update rule for *each* parameter θ_j (omitting the summation). Your function should take as input the training dataset $\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, y^{(i)}\}$, the learning rate α, the initial value $\theta = \{\theta_0, \theta_1, \ldots, \theta_p\}$, and an accuracy margin ϵ that controls the termination of the algorithm. The function should output the final value of the coefficients $\theta = \{\theta_0, \theta_1, \ldots, \theta_p\}$ once the accuracy margin ϵ is met.

Submission : Submit a single Python file – `groupXXassignment2.py` – containing both the functions – `BatchGradientDescent()` and `StochasticGradientDescent()` – as mentioned above in Problems 1 and 2, as well as any other associated piece of code, where `XX` is your group number.

Problem 3 [50 points]

Background : Consider the training dataset `wineTrain.csv` (attached), which has 12 variables:

```
"FixedAcidity"      "VolatileAcidity"     "CitricAcid"           "ResidualSugar"
"Chlorides"         "FreeSulphurDioxide"  "TotalSulphurDioxide"  "Density"
"pH"                "Sulphates"           "Alcohol"              "Quality"
```

The target is to fit an *optimal* linear regression model to predict the `"Quality"` of the wine.

Train Dataset : `http://www.souravsengupta.com/cds2017/evaluation/wineTrain.csv`

Task : Construct the *best model*, in your opinion, to predict the response variable, that is, the `"Quality"` of the wine, in case of the given dataset `wineTrain.csv`. Justify each of the choices you make in the process of building the *best model*, using comments within the `R` file itself.

Submission : Submit a single R file – `groupXXassignment2.R` – containing all the steps and comments for building the model mentioned above in Problem 3, where `XX` is your group number.

Your submission should be emailed to `sg.sourav@gmail.com` by midnight of 18 September 2017.

Properly acknowledge every source of information that you referred to, including discussions with other groups. Verbatim copy from any source is strongly discouraged, and plagiarism will be heavily penalized. It is strongly recommended that you write the codes completely on your own.