Brief description

- Learning method: Linear Regression, using Gradient Descent
- Cost function: GD cost
- Error function: Root Mean Square Error.
- Implementation Language: MATLAB (R2017a – linux)
- Results:
    - t_part.m: partial (first 72%(3000/4177) of dataset for training, the rest for testing)
        - RMSE for test-set: **2.1276**
        - Exact Prediction Rate of test-set: **25.47%**
        - Exact Prediction Rate of all: **25.21%**
        - RMSE of all: **2.2174**
    - t_full.m
        - RMES(full-set): **2.2151**
        - Exact-prediction rate: **24.80%**
    - m_4vars.m: manually chosen columns of the dataset (top 4 correlations)
        - RMSE: **2.4826**
- "**PT**" is the partial test, the output of calculation 't_part.m'
- "Sex" field is converted to numerical-type: 30(M) 20(I) 10(F)

Questions

Regression choice

linear or logistic?

linear regression is suitable here; While linear is used to predict new data based on continuous response data, logistic regression is apt where response is enumerable or has a few possible values. Moreover, logistic is intended to predict a probability, which is not the case here.

Therefore I used the best linear regression method: Gradient Descent.

Attribute selecting

Since the correlations of different fields were close, I deemed the optimum solution would be to make the system find it automatically; I used Gradient Descent to achieve this: using all fields from training set, iterating until convergence (steps set manually), yielding the least square error (implemented as cost function of GD).

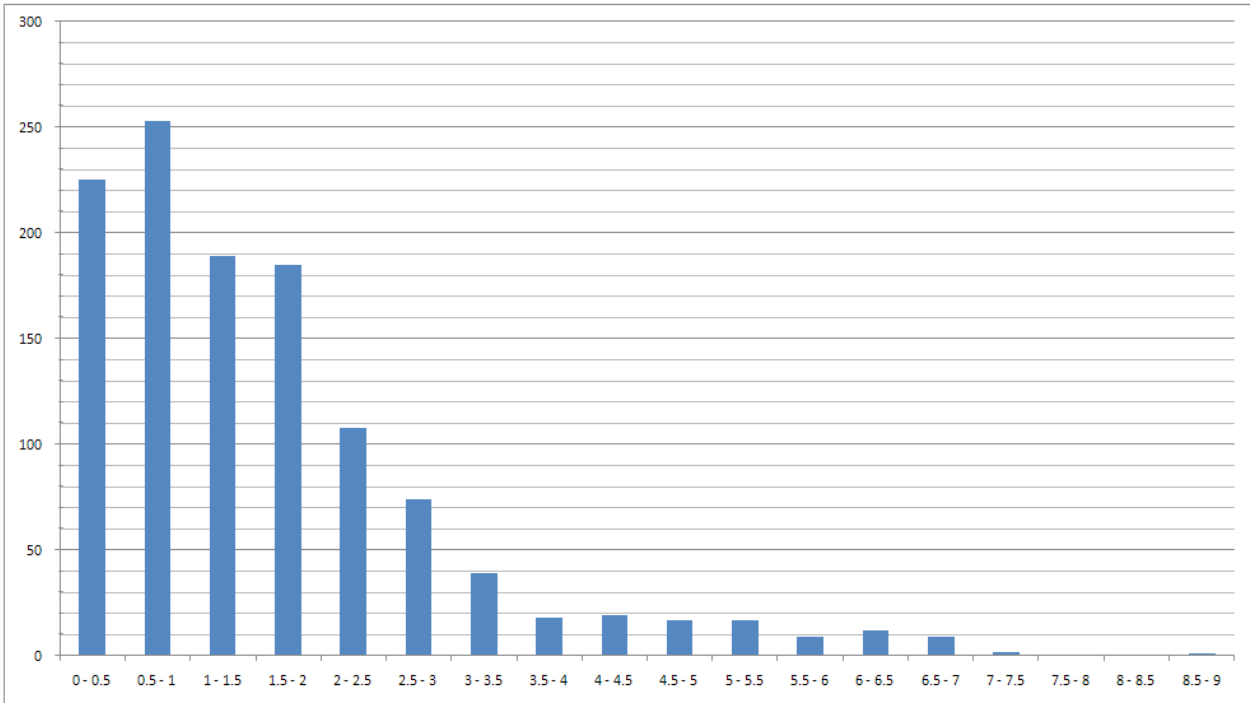
An intuitive explanation of Gradient Descent: for univariate functions, it is equal to derivation. For multivariate functions, say a 2-variable function,

f(x,y) -> z, Gradient Descent is basically trudging the curve that function f makes, in order to find the minimum. This way of finding the minimum could lead to erroneous results if it GD starts from a point near Local Minimum. To obviate this obstacle, I calculate multiple GDs, with different starting points (parameters of GD). (Conducted using m_part.m and m_full.m)

Is regression suitable?

No; predictions yield an error-rate (RMSE) of 2.12 to 2.2, which is high. Moreover both in test-dataset and full-dataset, results show a low rate of exact-predictions of 25%, Which is low, compared to other methods(like Neural Networks, KNN, etc.).

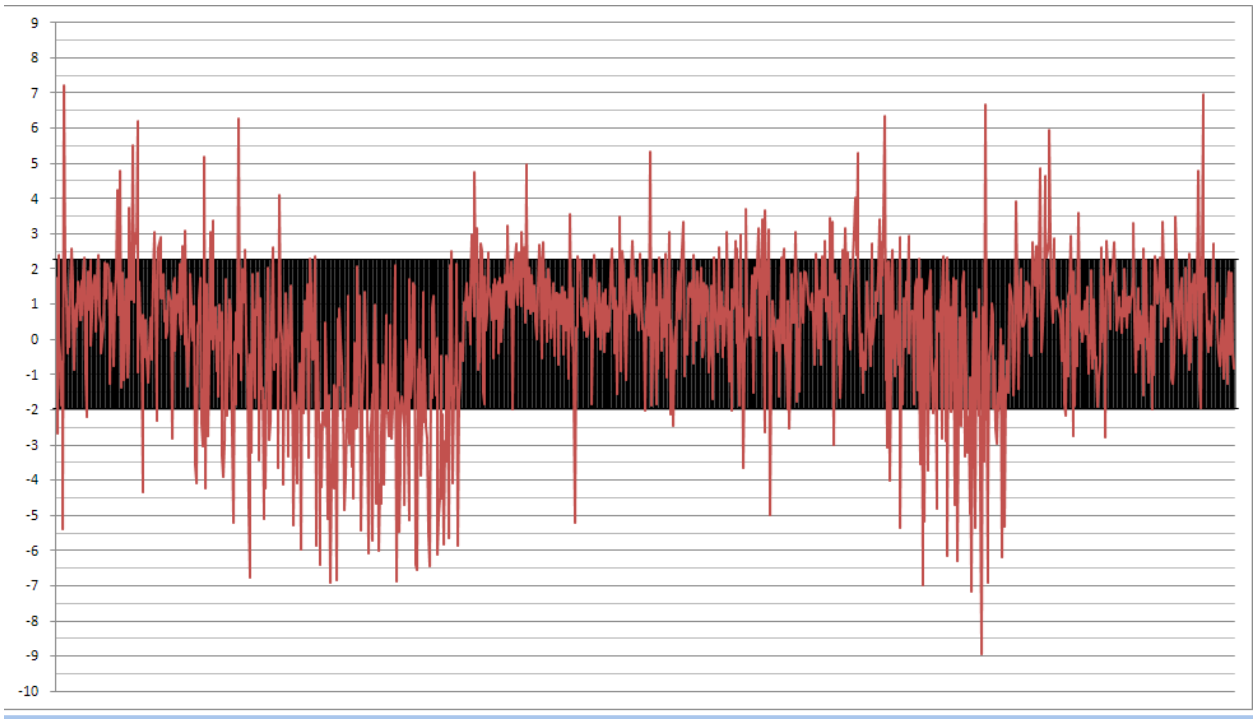
Observe the **PT** (3001-4177) ***error-distribution*** chart:



Prediction errors are high.

the (Prediction minus Response), for the **PT** on the following chart:

(the black area is Standard Deviation, which is equivalent to error rate)



Note

I used other methods, pre-built templates of MATLAB, to see they could achieve a better score(lower RMSE).

Here are RMSE's of them:

- Linear regression: 2.24
- Quadratic linear regression: 2.62
- SVM linear: 2.28
- SVM medium-Gaussian: **2.15**

My GD-Linear method: **2.13 (2.1276)**

Conclusion

Linear regression is not an optimal method for this dataset.

It could be used as a minimum tool, but not much useful.