UNIVERSITAS GADJAH MADA

Multi-Feature Prediction with Gradient Descent on Regression Models

Louis Widi Anandaputra

22/492218/PA/21090

LOCALLY ROOTED,

GLOBALLY RESPECTED

Aim

- Produce a model that can take more than 1 feature to predict something
- Explore the use of hyperparameters on the model fitting process

Scope

- Comparing two predictive models which can be fitted through gradient descent
- Predicting continuous value data through numerical features

Dataset Description

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0

No	Column Name	Count	Data Type
1	Hours Studied	10000	Integer
2	Previous Scores	10000	Integer
3	Extracurricular Activities	10000	String
4	Sleep Hours	10000	Integer
5	Sample Question Papers Practiced	10000	Integer
6	Performance Index	10000	Float

There are a total 10.000 instances in the data set and no columns have any missing values.

This work would try to predict the target value of performance index. An attempt to know how well the students would perform based on several features.

Pre-Processing and Initial Analysis

Mapping on the *extracurricular activities* column:

Yes → 1

No → 0

Checking Correlation:

Using Pearson correlation to check the correlation between columns

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Correlation Matrix

From the correlation matrix, we will be using only two features having high correlation with the performance index:

- Hours Studied
- Previous Scores

UNIVERSITAS GADJAH MADA

Predictive Models

Linear (1) and Polynomial (2) Regression

$$\hat{y} = \beta + \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n \quad (1)$$

$$\hat{y} = \beta + \theta_0 X_0^{a_0} + \theta_1 X_1^{a_1} + \cdots + \theta_n X_n^{a_n} \quad (2)$$

n : number of features

x value of the specific feature

β bias

θ : weight

a : order

$\hat{y}$: predicted value

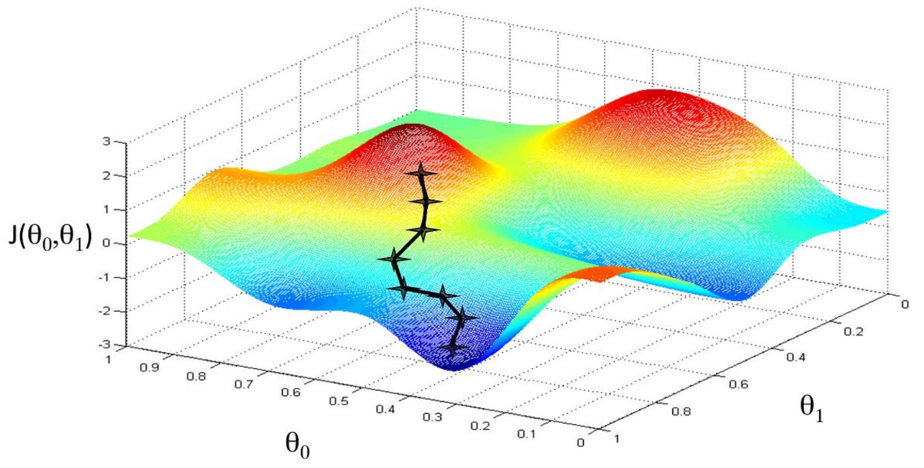
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

MSE would be used as the cost function for the model fitting process in gradient descent, but RMSE would be used on evaluating the testing results.

How to Find Beta, Theta, and a?

Gradient descent can be use to find the best parameters. It would work by changing the value of either *Beta, Theta*, and *a* by subtracting the original value with the rate of change (gradient) multiplied by the learning rate.



$$\theta_0' = \theta_0 - \frac{\partial E}{\partial \theta_0}\alpha$$

Changing theta 0 based on its rate of change

How to Find Beta, Theta, and a? (Detailed)

$$\hat{y} = \beta + \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n \quad (1)$$

$$\hat{y} = \beta + \theta_0 X_0^{a_0} + \theta_1 X_1^{a_1} + \cdots + \theta_n X_n^{a_n} \quad (2)$$

Finding Rate of Change for Gradient Descent

Rate of Change on Bias

$$\frac{\partial E}{\partial \beta} = \frac{2}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)$$

Rate of Change on Weights

$$\begin{bmatrix} \frac{\partial E}{\partial \theta_1} \\ \frac{\partial E}{\partial \theta_2} \\ \vdots \\ \frac{\partial E}{\partial \theta_n} \end{bmatrix} = \frac{2}{m} \begin{bmatrix} \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{1i}) \\ \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{2i}) \\ \vdots \\ \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{ni}) \end{bmatrix}$$

Rate of Change on Orders

$$\begin{bmatrix} \frac{\partial E}{\partial a_1} \\ \frac{\partial E}{\partial a_2} \\ \vdots \\ \frac{\partial E}{\partial a_n} \end{bmatrix} = \frac{2}{m} \begin{bmatrix} \theta_1 \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{1i}^{a_1}) ln(x_{1i}) \\ \theta_2 \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{2i}^{a_2}) ln(x_{2i}) \\ \vdots \\ \theta_n \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{ni}^{a_n}) ln(x_{ni}) \end{bmatrix}$$

Updating the Values

Updating the Bias

$$\beta' = \beta - \frac{\partial E}{\partial \beta} \alpha$$

Updating the Weights

$$\begin{bmatrix} \theta'_1 \\ \theta'_2 \\ \vdots \\ \theta'_n \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} - \begin{bmatrix} \frac{\partial E}{\partial \theta_1} \\ \frac{\partial E}{\partial \theta_2} \\ \vdots \\ \frac{\partial E}{\partial \theta_n} \end{bmatrix} \alpha$$

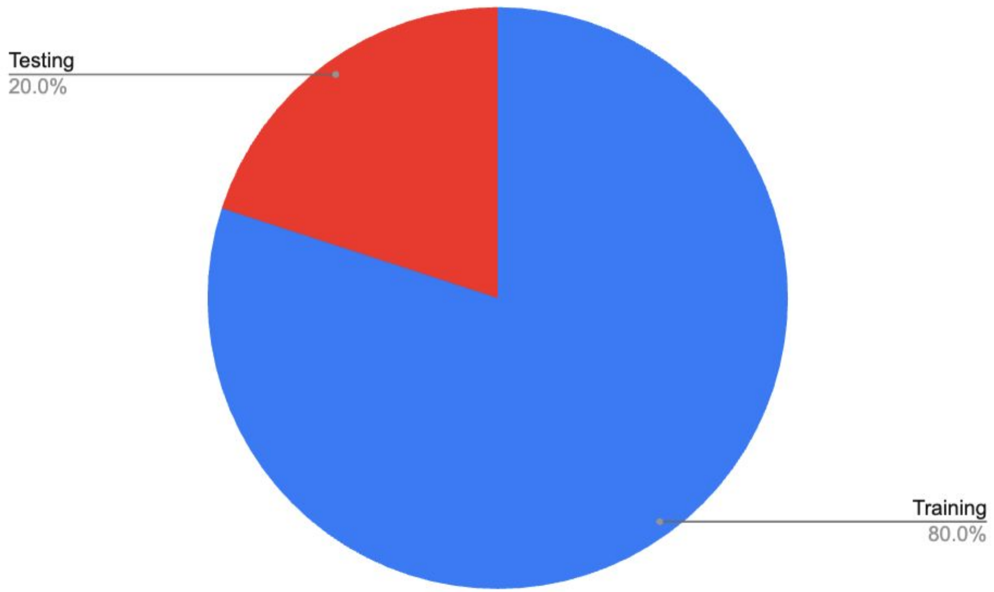
Updating the Orders

$$\begin{bmatrix} a'_1 \\ a'_2 \\ \vdots \\ a'_n \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} - \begin{bmatrix} \frac{\partial E}{\partial a_1} \\ \frac{\partial E}{\partial a_2} \\ \vdots \\ \frac{\partial E}{\partial a_n} \end{bmatrix} \alpha$$

Testing Conditions

Method	Iterations	Learning Rate
Linear Regression	2000	1.00E-03
		1.00E-04
		1.00E-05
		1.00E-06
		1.00E-07
		1.00E-08
		1.00E-09
		1.00E-10
Polynomial Regression		1.00E-03
		1.00E-04
		1.00E-05
		1.00E-06
		1.00E-07
		1.00E-08
		1.00E-09
		1.00E-10

Data Splitting



A random seed is used for reproducibility on future testing, ensuring that even though random, the choice of which subset of data goes into the training or testing set is always the same.

Training Results

	Evaluation Result			Gradient Descent Result			
Method	RMSE Train	RMSE Test	CPU Time (s)	Learning Rate	Weights	Orders	Bias
Linear Regression	NaN	NaN	2.23	1.00E-03	NaN	-	NaN
	6.796740627	6.846577806	2.34	1.00E-04	[1.70809367 0.69933541]	-	-0.4869035663
	7.651835097	7.767531076	2.67	1.00E-05	[0.5994583 0.76852519]	-	-0.02089330316
	8.222203741	8.345789931	2.44	1.00E-06	[0.1997553 0.7952542]	-	0.01569374495
	10.49219509	10.44251047	2.36	1.00E-07	[0.14680064 0.70901746]	-	0.01789375072
	41.71523799	41.65846637	2.13	1.00E-08	[0.10937308 0.22978923]	-	0.01172428718
	49.8835971	49.86021334	2.2	1.00E-09	[0.10102128 0.11421157]	-	0.0101891575
	50.78935555	50.7697765	2.54	1.00E-10	[0.10010303 0.10143433]	-	0.01001909458
Polynomial Regression	780064.9968	826361.102	8.71	1.00E-03	[-2461138.89914758 3367565.81928505]	[-5.05183722e+06 -2.42368073e+15]	206264.9177
	23.58131749	23.79	9.34	1.00E-04	[9.65632208 479.41764933]	[0.90620718 -32.12133829]	5.128227794
	2.952672056	2.954937071	9.46	1.00E-05	[0.5272984 0.08493531]	[1.662774 1.4826695]	-0.0002403392061
	7.31751491	7.36879711	9.2	1.00E-06	[0.17966348 0.26691264]	[1.02842 1.25234507]	0.01146827931
	7.494475089	7.546776613	10.1	1.00E-07	[0.11893783 0.27957005]	[1.00429843 1.24302046]	0.01231775052
	33.30683044	33.20227994	12.7	1.00E-08	[0.10880372 0.22127837]	[1.00167722 1.10562838]	0.01161063271
	49.6482174	49.62352338	10.3	1.00E-09	[0.10101914 0.11417976]	[1.00018577 1.00669291]	0.01018873153
	50.76980688	50.75011937	11.7	1.00E-10	[0.10010301 0.10143405]	[1.00001868 1.00062827]	0.01001909083

CPU time is the time measured for training process and testing results sequentially*

Average Computing Time	
Model	Time
Linear Regression	2.36375
Polynomial Regression	10.18875

Final Models

UNIVERSITAS GADJAH MADA

From the training process we have...

Best model for linear regression for this dataset:

$$\hat{y} = -0.48690356626347364 \; + \; 1.70809367 \, Hours\ Studied \; + \; 0.69933541 \, Previous\ Scores$$

Best model for polynomial regression for this dataset:

$$\hat{y} = -0.00024033920610194357 \; + \; 0.5272984 \, Hours\ Studied^{1.662774} \; + \; 0.08493531 \, Previous\ Scores^{1.4826695}$$

Key Points Findings and Recommendations

UNIVERSITAS GADJAH MADA

Key Findings

- Learning rates that are too big may cause divergence on finding the best parameters
- Learning rates that are too small also may not reach the optimal value
- Polynomial regression managed to become a more optimal representation model of the data than the linear regression, but cost almost 5 times the computational complexity compared to linear regression.
- This data set exhibits good representation of information as the training RMSE is really close to the testing RMSE

Recommendations

- Use cross validation on future training process
- Try to implement regularization to avoid overfitting
- Try to implement RMSE on the cost function of the gradient descent process
- Try to construct a model that can capture time-series properties (AR, MA, ARIMA, RNN, LSTM)

Full Code

Feel free to review!

https://github.com/louiswids/Numerical-Method-Approximation
Find the ipnyb file for full code*

UNIVERSITAS
GADJAH MADA

Thank you for the attention