Polynomial Regression Impl 001

Gavin Simpson

December 2025

Contents

1 Resources and About

Date: *13/12/2025*
Revision: *3*

Github Repository: `github.com/GDSimpson3/Polynomial-regression-001-impl`
PDF URL: `github.com/GDSimpson3/Polynomial-regression-001-impl/README.pdf`

2 Dynamic Orders

2.1 Dynamic Exponents 3SLOTS 6PARAM DSQuadratic

Works on a simple Quadratic Dataset

Uses 6 Parameters:

Symbol	Meaning
Ca	Coefficient A
Ea	Exponent A
Cb	Coefficient B
Eb	Exponent B
Cc	Coefficient C
Ec	Exponent C

Table 1: Mapping of symbols to coefficient and exponent terms

Uses the Gradient Descent to find the best exponents and Coefficients
Currently was able to bring MSE down to **3** with **20,000** iterations
It currently uses 3 static slots in the form of

$$Cax^{Ea} + Cbx^{Eb} + Ccx^{Ec}$$

as the models Regression line formula

2.2 Loss Function

The loss function that we're applying the Gradient Descent to, which is the mean squared Error.

2.2.1 Mean Squared Error

Mean Squared Error:

$$\text{MSE(parameters)} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

MSE Of a Bivariate Linear Regression line

$$\text{MSE(m,b)} = \frac{1}{n}\sum_{i=1}^{n}(mx_i + b - y_i)^2$$

MSE Of a Multivariate Polynomial Regression Line

$$\text{MSE(a,b,c)} = \frac{1}{n}\sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^2$$

MSE Of a Multivariate Polynomial Regression Line

$$\text{MSE(Ca,Ea,Cb,Eb,Cc,Ec)} = \frac{1}{n}\sum_{i=1}^{n}(Cax_i^{Ea} + Cbx_i^{Eb} + Ccx_i^{Ec} - y_i)^2$$

2.2.2 Partial Derivatives

Partial Derivatives of a Simple linear regression line

$$\frac{\partial \text{MSE(m,b)}}{\partial \text{m}} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i - y_i)^1 x_i$$

$$\frac{\partial \text{MSE(m,b)}}{\partial \text{b}} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i)^1$$

Partial Derivatives of a Simple Quadratic regression line

$$\frac{\partial \text{MSE(a,b,c)}}{\partial \text{a}} = \frac{2}{n}\sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^1 x_i^2$$

$$\frac{\partial \text{MSE(a,b,c)}}{\partial \text{b}} = \frac{2}{n}\sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^1 x_i$$

$$\frac{\partial \text{MSE(a,b,c)}}{\partial \text{c}} = \frac{2}{n}\sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^1$$

Generic Partial Derivatives of a Polynomial regression line
with respect to Coefficient N

$$\frac{\partial \text{MSE(Cn,En)}}{\partial \text{Cn}} = \frac{2}{n}\sum_{i=1}^{n}(Cnx_i^{En}... - y_i)^1 x_i^{En}$$

with respect to Exponent N

$$\frac{\partial \text{MSE(Cn, En,......)}}{\partial \text{En}} = \frac{2}{n}\sum_{i=1}^{n}(Cnx_i^{En}... - y_i)^1 Cn[(x_i^{En})\ln(x_i)]$$

2.3 NTerms

Aim: Making it dynamic so that it can compute upto N terms

2.3.1 Normalisation

Now this here is simply squashing all of our values into **0 and 1**. EG: The whole Sin X function that goes from 0 to 10 (X), is now squashed into 0 and 1 VIA a normalisation function

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$$

Why should we Normalise? Our Partial derivative for the Exponents, include a ln(x), If the Values of X are big, we'll get massive gradients which can shoot up the Exponents. Which can result in overflow errors (values too big for variables)

2.3.2 Exponent Clamping

Here we're going to Clamp our exponents within a range (between -5 and 8 for now) this is another safeguard against exponent explosions. For now we're using the **Numpy.clip** function

```
RegressionTerms[TermIndex][1] =
np.clip(RegressionTerms[TermIndex][1], min_exp,
max_exp)
```

2.3.3 Ensure X is never zero

Because of our Ln in the exponent function, if we have values of X in our dataset, it'll go straight to negative infinity

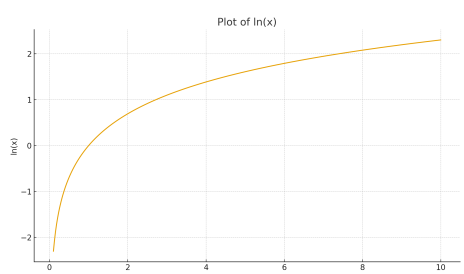

Figure 1: $\lim_{x\to 0} \ln x = -\infty$ Diagram

Hence we need to get rid of all X values in our dataset that Are 0.

We can do this By clamping X to a Safe Minimum (anything less than $1 * 10^{-6}$ will become $1 * 10^{-6}$. Including Negatives

```
X = np.clip(X, 1e-6, None)
```

NOTE: THIS IS X, Not Y, X

2.3.4 Alpha Configuration

```
AlphaC = 0.01
AlphaE = 1e-6
```

Make steps of Exponent almost nano. We dont want the Exponent to be changing too much.

2.4 Dynamic Number of Terms

Currently its fixed, there's only N Cnx_i^{En} in the regression line. How can we make it dynamic? We can Start with 20/50 terms (much more than needed), and eventually we can prune them out VIA L1 Regularization

2.5 L1 Regularization

Basically, this is all about starting with a high number of Cnx_i^{En} terms in our regression equation and just eliminating the ones where the co-efficients become 0.

Additionally we prune all duplicated terms (similar out of bound exponents that have similar co-efficients)

2.5.1 L1

It adds $\pm\lambda$ to the partial derivatives depending on the sign of the **coefficient**. Pushes unwanted terms close to 0.

Think of Choosing λ (lambda) as "how aggressively to delete terms".

Value	Effect
0.0001	barely deletes anything
0.001	light pruning
0.01	strong pruning
0.1	brutal pruning — will kill most terms

Table 2: Choosing λ (Lambda) Values

The formula of it is

$$\text{Total Gradient for } Cn_i = \left(\frac{\partial MSE}{\partial Cn_i}\right) + \lambda_{L1} \cdot \text{sign}(x_i)$$

2.5.2 Pruning

Every **PruneFrequency** times, we clear all the terms that have their co-efficients close to the **prunethreshold** which is about **1e-5**.

Additionally we get rid of Repeated terms that accumulate at the end. This is typically a symptom of them hitting the upper exponent bound as part of the Exponent Clamping. We have a tolerance between how similar the coefficient **and** exponent must be, in order to eliminate one of them.

3 Parameters

3.1 Parameter Definitions

Parameter	Meaning
AlphaC	Learning rate of the coefficients
AlphaE	Learning rate of the exponents
MaxIterations	Maximum number of iterations
MaxDegree	Number of terms the regression equation initializes with
minExp	Exponent clamping – lower bound
maxExp	Exponent clamping – upper bound
WarmupPeriod	Iteration at which pruning begins
lambdaL1	Pruning rate – how aggressively terms are pushed to zero
PruneFrequency	How often pruning occurs
pruneThreshold	Coefficient magnitude below which terms are pruned
ExponentDuplication Tolerance	Threshold for removing duplicated terms by exponent
CoefficientDuplication Tolerance	Threshold for removing duplicated terms by coefficient

Table 3: Parameter Documentation

3.2 Parameter Usage (Stage 3.2)

Parameter	Meaning
AlphaC	0.01
AlphaE	2e-5
MaxIterations	100000
MaxDegree	20
minExp	-5
maxExp	20
WarmupPeriod	int(0.2 * MaxIterations)
lambdaL1	0.01
PruneFrequency	100
pruneThreshold	1e-5
ExponentDuplication Tolerance	1e-3
CoefficientDuplication Tolerance	1e-2

Table 4: Parameter Values (Stage 3.2)