Regression and Cloud Ready AI Infrastructure

Apertura: miércoles, 21 de enero de 2026, 00:00
Cierre: martes, 27 de enero de 2026, 23:59

Stellar luminosity

Linear and Polynomial Models for Regression

1. Introduction and Motivation

Astronomy is a data-driven science in which relationships between physical quantities are inferred and validated through observation. Classical examples include the relationships between stellar mass, temperature, radius, and luminosity. In this homework, you will implement linear regression and polynomial regression from first principles, without using machine-learning libraries.

Rather than calling pre-built fitting routines, you will explicitly define the hypothesis function, the loss function, and the optimization algorithm. The astronomical problem studied here is a simplified stellar luminosity modeling task, inspired by main-sequence behavior: luminosity grows rapidly with mass, and additional properties can introduce nonlinear and interaction effects.

2. Motivation for Cloud Execution and Enterprise Context

This homework is part of a four-week Machine Learning Bootcamp embedded in a course on Digital Transformation and Enterprise Architecture. In this context, machine learning is treated as a core architectural capability of modern enterprise systems.

Today, *intelligence* is increasingly considered a first-class quality attribute alongside scalability, availability, security, and performance. Intelligent behavior is no longer confined to offline analytics; it is embedded into platforms, decision-support services, and autonomous or semi-autonomous components.

As enterprise architects, it is not sufficient to understand what models do. We must also understand how they are built from first principles, executed and validated in controlled environments, and operated within cloud platforms.

3. General Rules and Delivery Requirements

1. Deliver all work in a single GitHub repository.
2. The repository must contain two Jupyter notebooks and one README.md.
3. All code must be written inside the notebooks.
4. All datasets must be defined directly in the notebooks (as hard-coded NumPy arrays).
5. Allowed libraries: Python, NumPy, Matplotlib (inline plots only).
6. Not allowed: scikit-learn, statsmodels, TensorFlow/PyTorch, or any high-level regression/optimization library.

4. Repository Structure

```
/
├── README.md
├── 01_part1_linreg_1feature.ipynb
└── 02_part2_polyreg.ipynb
```

5. Dataset and Notation

Use the following notation throughout:

- **M**: stellar mass (in units of solar mass, $M_{\odot}$)
- **T**: effective stellar temperature (Kelvin, K)
- **L**: stellar luminosity (in units of solar luminosity, $L_{\odot}$)

Part I dataset (one feature)

```
M = [0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4]
L = [0.15, 0.35, 1.00, 2.30, 4.10, 7.00, 11.2, 17.5, 25.0, 35.0]
```

Part II dataset (two features)

```
M = [0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4]
T = [3800, 4400, 5800, 6400, 6900, 7400, 7900, 8300, 8800, 9200]
L = [0.15, 0.35, 1.00, 2.30, 4.10, 7.00, 11.2, 17.5, 25.0, 35.0]
```

6. Notebook 1: 01_part1_linreg_1feature.ipynb

Objective

Model stellar luminosity as a function of stellar mass using linear regression with an explicit bias term: L_hat = w * M + b.

Required tasks

1. **Dataset visualization:** plot M vs L. Comment on linearity and plausibility.
2. **Model and loss:** implement prediction and mean squared error (MSE).
3. **Cost surface (mandatory):** evaluate J(w,b) on a grid of w and b. Plot a 3D surface or contour plot. Explain what the minimum represents.
4. **Gradients:** derive and implement dJ/dw and dJ/db.
5. **Gradient descent (non-vectorized):** compute gradients using an explicit loop over samples.
6. **Gradient descent (vectorized):** compute gradients using NumPy vectorization (no loop over samples).
7. **Convergence (mandatory):** plot loss vs iterations, and comment on convergence speed and stability.
8. **Experiments (mandatory):** run at least three learning rates; report final w, b, and loss.
9. **Final fit plot:** plot the regression line over the data and discuss systematic errors.
10. **Conceptual questions:**
    1. astrophysical meaning of w;
    2. Why is a linear model limited here?

7. Notebook 2: 02_part2_polyreg.ipynb

Objective

Capture nonlinear and interaction effects using polynomial feature engineering and an explicit bias term:

L_hat = X @ w + b.

Feature map

Construct the design matrix using the following features (do not include a constant column of ones):

```
X = [ M, T, M^2, M*T ]
```

Required tasks

1. **Dataset visualization:** plot L vs M and encode T (color or marker size).
2. **Feature engineering:** build X with NumPy vectorization.
3. **Loss and gradients (vectorized):** implement ME and gradients w.r.t. both w and b.

4. **Gradient descent + convergence:** train and plot loss vs iterations.
5. **Feature selection experiment (mandatory):** compare:
    - **M1:** X = [M, T]
    - **M2:** X = [M, T, M^2]
    - **M3:** X = [M, T, M^2, M*T]

   For each model: report final loss, learned parameters, and show predicted vs actual (inline plot).
6. **Cost vs interaction (mandatory):** for the full model (M3), vary the interaction coefficient w_MT across a reasonable range while keeping the other parameters fixed (e.g., at their trained values). Plot cost vs w_MT and explain what it indicates about interaction importance.
7. **Inference demo (mandatory):** predict luminosity for a new star (e.g., M=1.3, T=6600). Comment on reasonableness.

8. AWS SageMaker Requirement (Execution Only)

- Upload both notebooks to **AWS SageMaker** (Studio or Notebook Instances).
- Run all cells successfully (no errors).
- No model deployment, endpoints, or MLOps pipelines are required.

9. README.md Requirements (Mandatory Cloud Evidence)

Your README must include a section titled **AWS SageMaker Execution Evidence** containing:

- A brief description of how you uploaded the notebooks to SageMaker.
- Screenshots showing:
    - both notebooks visible/open in SageMaker,
    - successful execution (cells run and outputs visible),
    - at least one plot rendered in SageMaker.
- A brief note comparing local execution vs SageMaker execution (any differences observed).

10. Evaluation Criteria

- Correctness of implementation (loss, gradients, training loop)
- Proper use of vectorization (where required)
- Quality and completeness of plots (dataset, cost surface, interaction cost, convergence)
- Quality of explanations and interpretations
- Successful SageMaker execution with documented evidence in the README

Agregar entrega

Estado de la entrega

Estado de la entrega	Todavía no se han realizado envíos
Estado de la calificación	Sin calificar
Tiempo restante	2 horas 1 minutos restante
Última modificación	-

Comentarios de la entrega	Comentarios (0)

Enlaces de interés

Ministerio de Educación Nacional

Colombia Aprende

Red Latinoamericana de Portales Educativos

Red Universitarias Metropolitana de Bogotá

Biblioteca

Investigación e innovación

Enlace - Académico

Contacta

AK.45 No.205-59 (Autopista Norte).

Teléfono : +57(1) 668 3600

Correo electrónico : contactocc@escuelaing.edu.co

Escuela Colombiana de Ingeniería Julio Garavito