04 – Regression

Data Science and Management

Corso di Laurea Magistrale in Ingegneria Gestionale

Marco Mamei, Natalia Hadjidimitriou, Fabio D’Andreagiovanni, Matteo Martinelli

{marco.mamei, selini, fabio.dandreagiovanni, matteo.martinelli}@unimore.it

- Regression
- Forecasting

Regression

We speak of **regression** when the target variable to be predicted/forecast is a **real-valued variable**

Examples:

- Predict salary as a function of employee features
- Forecast stock price given historical data

Regression

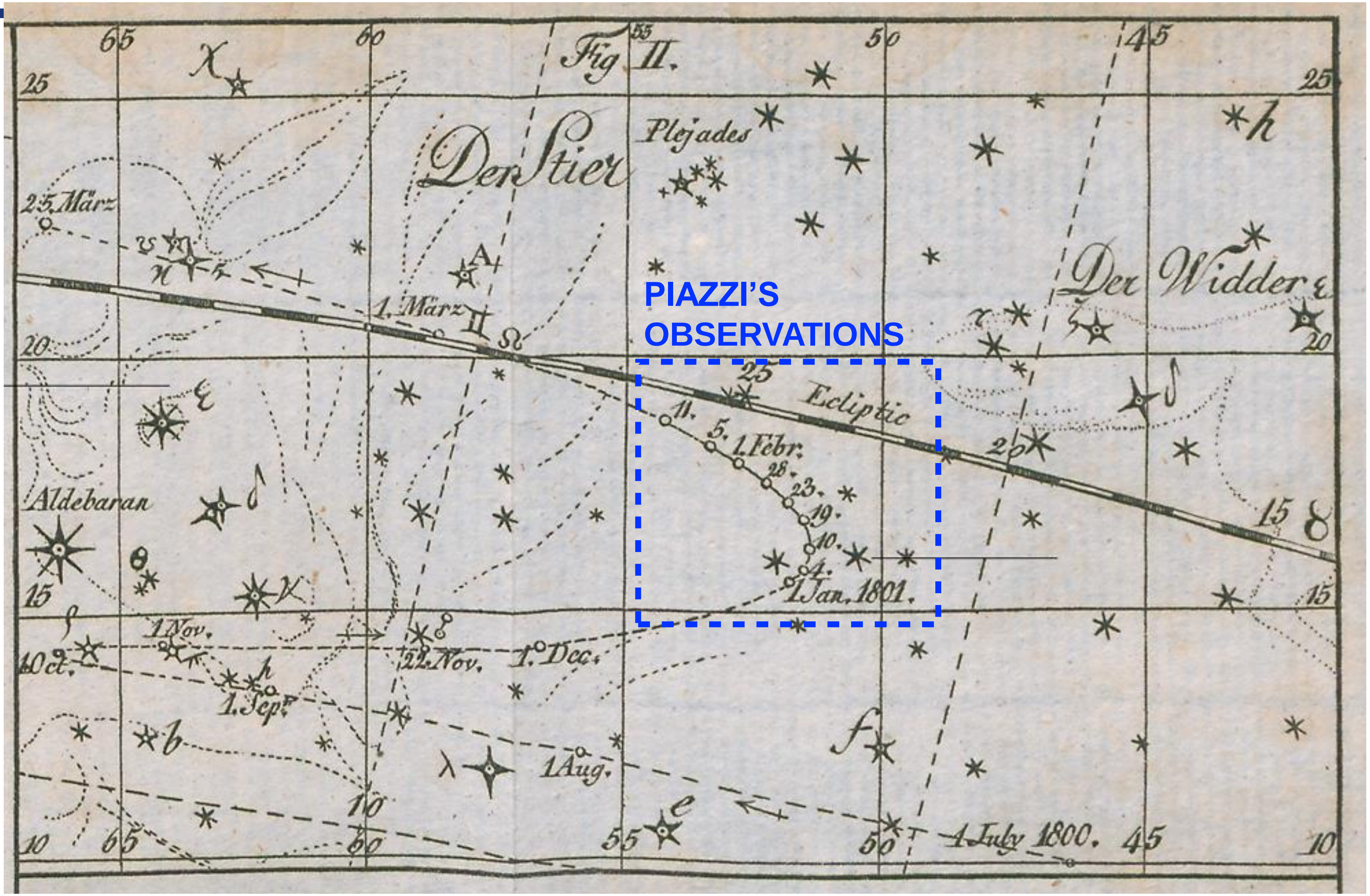
The astronomer **Giuseppe Piazzi** in 1801 discovered what he thought to be a new planet.

He observed the new celestial body for over a month, then it was **lost** in the glare of the Sun.

Gauss computed the orbit of the planet with the **least-squares method** (also developed by Legendre)

That “planet” is now known as the asteroid **Ceres**.

Regression


PIAZZI'S
OBSERVATIONS
Fig. II.
Der Stier
Plejades
Der Widder
Ecliptic
Aldebaran
1. Febr.
1 Jan. 1801.
1. März
25. März
1 Nov.
22. Nov.
1. Dec.
1. Sep.
1 Aug.
1 July 1800.

Regression

Classic approaches

- **KNN-regression**
- **Least-squares linear regression**
- Neural networks
- Regression trees
- …

Non-Parametric Regression

Parametric Regression

- Choosing **in advance** the shape of the function
- E.g., Linear regression

Non-Parametric Regression

- **No prior choice** on the shape of the function
- E.g., KNN-regression

Regression

KNN-regression

- An extension of KNN algorithm to **regression** case
- Simply compute target value as (weighted) **combination** of those of the K nearest neighbors

$$\hat{y}_j = \frac{\sum_{k \in \mathcal{N}_j} w_k y_k}{\sum_{k \in \mathcal{N}_j} w_k}$$

$$w_k = \frac{1}{d(x_j, x_k)}$$

Linear Regression

Linear regression with **least-squares** approach

Minimize the **Residual Sum of Squares** (RSS)

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

where $\hat{y}_i$ is computed as

$$\hat{y}_i = \beta^T x = \sum_{i=1}^{K} \beta_i x_i$$

Linear Regression

In matrix form we are given N examples with K features: $Y \in \mathbb{R}^N, \beta \in \mathbb{R}^K, X \in \mathbb{R}^{N \times K}$

$$\min_\beta \|Y - \beta^T X\|^2$$

Linear Regression

The problem is **over-dimensioned**: we have more equations than variables (N >> K)…

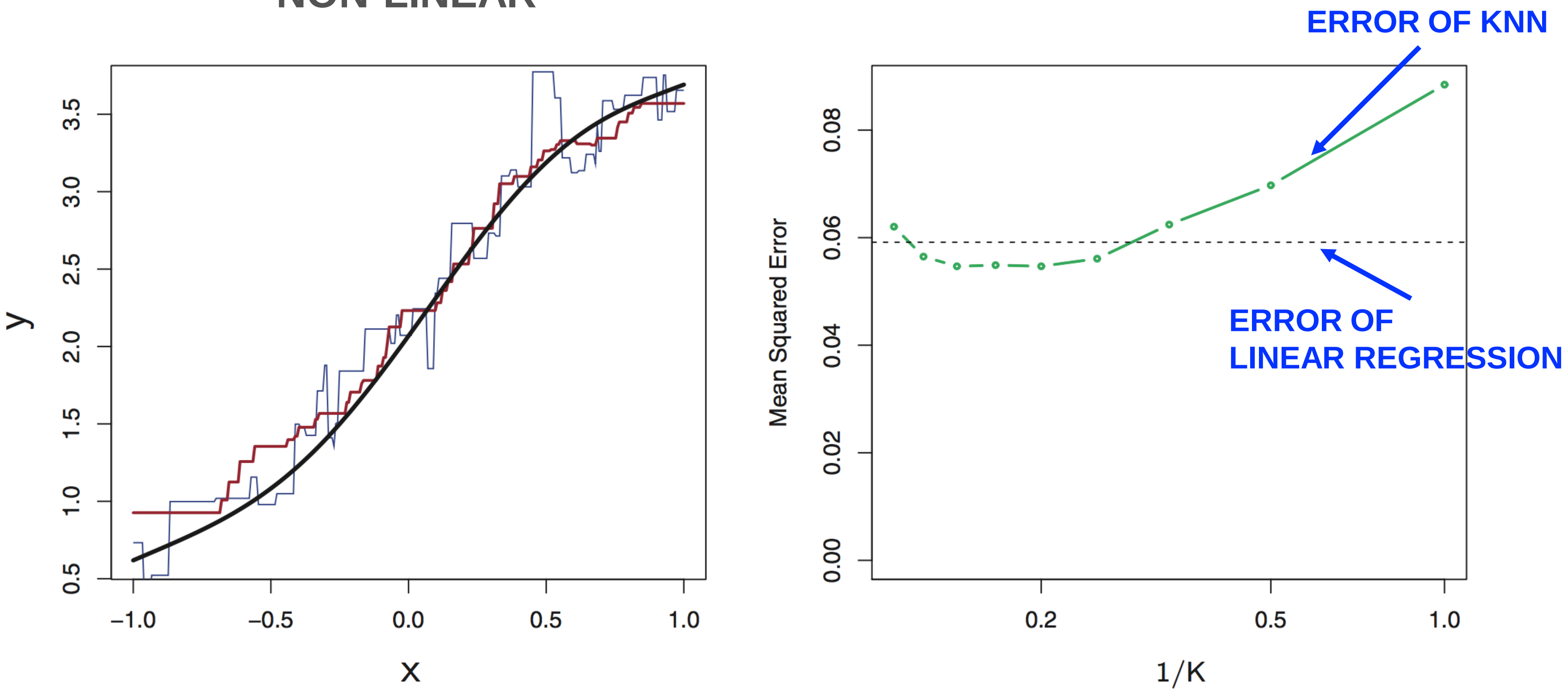
There exist a closed-form solution

$$N >> K$$

$$\beta^* = \underbrace{\boxed{(X^T X)^{-1} X^T}}_{\text{MOORE-PENROSE PSEUDO-INVERSE}} Y$$

Linear vs. Non-Linear

NON-LINEAR



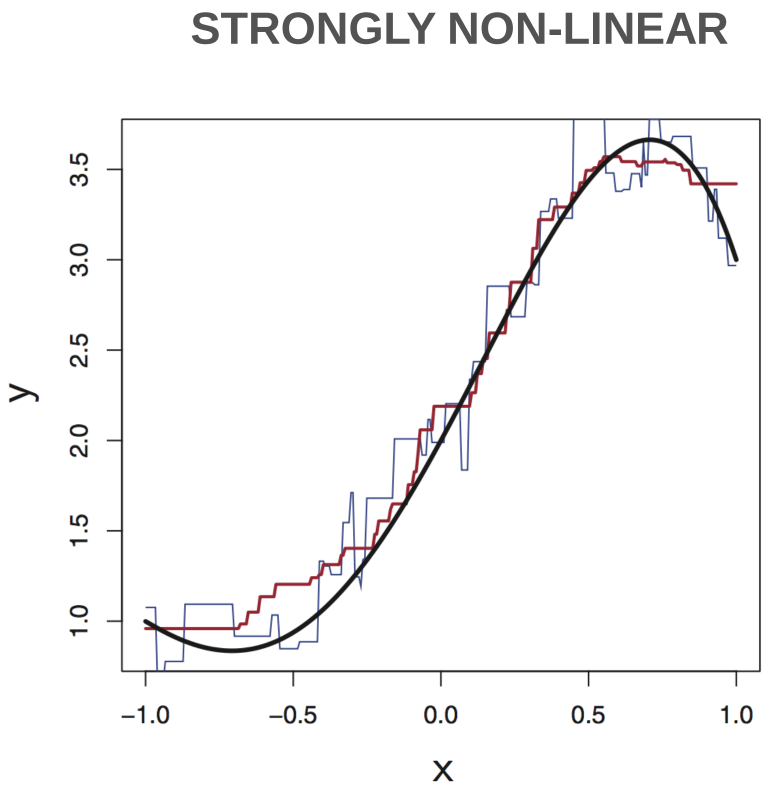
BLACK
True relationship

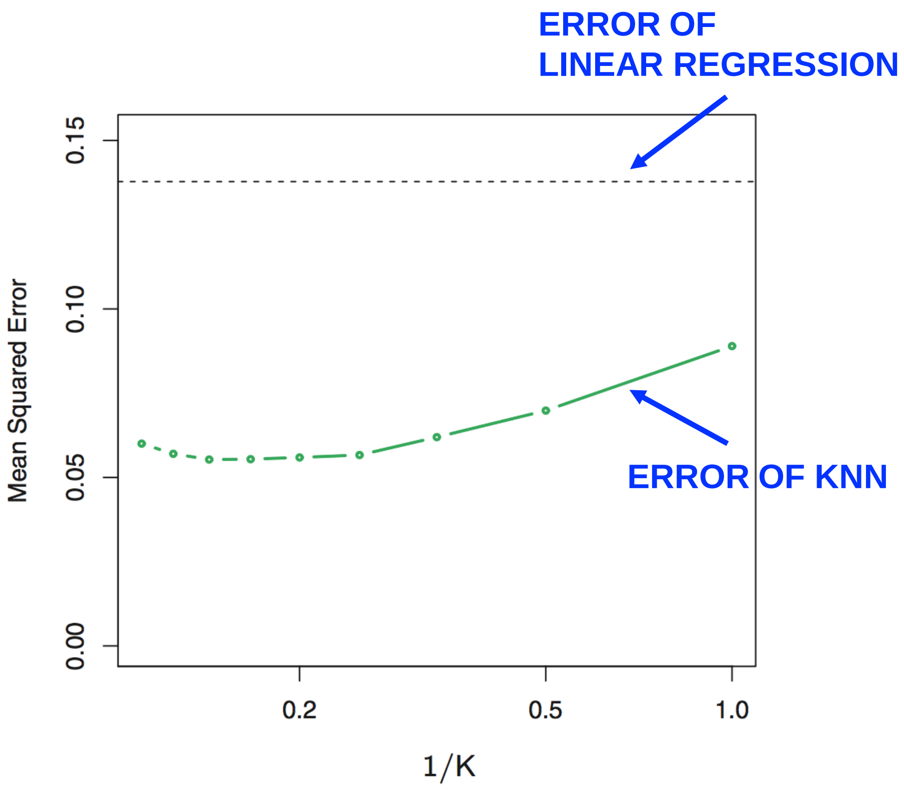
BLUE
KNN with K=1

RED
KNN with K=9

Figure from [James et al.]

Linear vs. Non-Linear





BLACK
True relationship

BLUE
KNN with K=1

RED
KNN with K=9

Figure from [James et al.]

More on Regression

Other regression models:

- Regression trees
- Neural networks
- …

2024 – Marco Mamei, Full Professor (marco.mamei@unimore.it)

Regression Trees

An **extension** of decision trees to regression

- Use the **improvement** in Mean Squared Error to evaluate and select attributes during training
- In a leaf node, compute the **average** of the target variable across the training examples in the leaf
- Use **ensemble** techniques (Random Forests and Gradient Boosting) as for classification

Neural networks

Any neural architecture can be used for regression

- For Multi-Layer Perceptron, just use **one neuron** in the output layer, with linear activation function
- Use a **loss function** that evaluates regression tasks, such as Mean Squared (or Absolute) Error
- Specific architectures are dedicated to time-series analysis: i.e., **recurrent neural networks** such as Long Short-Term Memory (LSTM) networks

Example on Time-Series

Parameters to control when forecasting time-series?

Example with Scikit-Learn

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

df = pd.read_csv("yahoo_stock.csv", sep=",")

data = df["Close"].values.astype('float32')
X = []
y = []
W = 10
for i in range(W, data.shape[0]):
    X.append(data[i-W:i])
    y.append(data[i])
X = np.array(X)
y = np.array(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

UNIVARIATE TIME-SERIES

WINDOW = NUMBER OF FEATURES

Example with Scikit-Learn

```
V0 V1 V2 V3 . . .

X0 X1 X2 X3 -> Y = X5
X1 X2 X3 X4 -> Y = X6
X2 X3 X4 X5 -> Y = X7
. . .
. . .
```

Example with Scikit-Learn

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor

print("*** Random Forest ***")
clf = RandomForestRegressor()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(mean_absolute_error(y_test, y_pred))

print("*** Linear Regression ***")
clf = LinearRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(mean_absolute_error(y_test, y_pred))

print("*** KNN Regression ***")
clf = KNeighborsRegressor(n_neighbors=3)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(mean_absolute_error(y_test, y_pred))
```

Example with Keras

```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping

model = Sequential()
model.add(Dense(50, input_shape=(X_train.shape[1],)))
model.add(Dense(20))
model.add(Dense(1, activation='linear'))

es = EarlyStopping(monitor='val_loss', patience=10)

model.compile(loss='mean_squared_error', optimizer='adam',
metrics=['mean_squared_error'])

model.fit(X_train, y_train, epochs=1000, batch_size=16,
validation_split=0.2, callbacks=[es])
```

OUTPUT NEURON

REGRESSION LOSS

- https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424