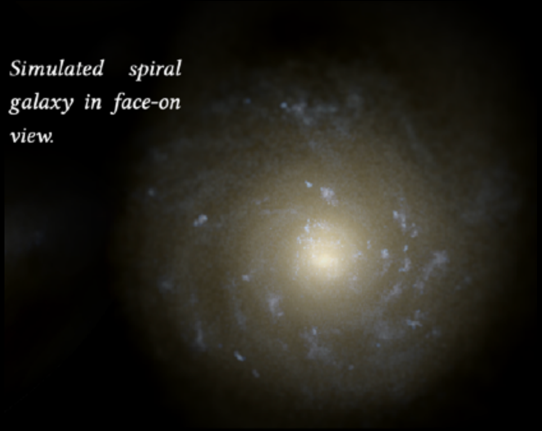
ASTR4004

COMPUTATIONAL ASTRONOMY

Week 8 https://github.com/svenbuder/astr4004_2025_week8

Spiral galaxy M74 in face-on view. ***Figure credit:*** *Gemini Observatory, GMOS Team*



Simulated spiral galaxy in face-on view.

Figure credit: *Tobias Buck*

Australian National University

My idea for the next 2 weeks:

Assignment 3
due on October 7

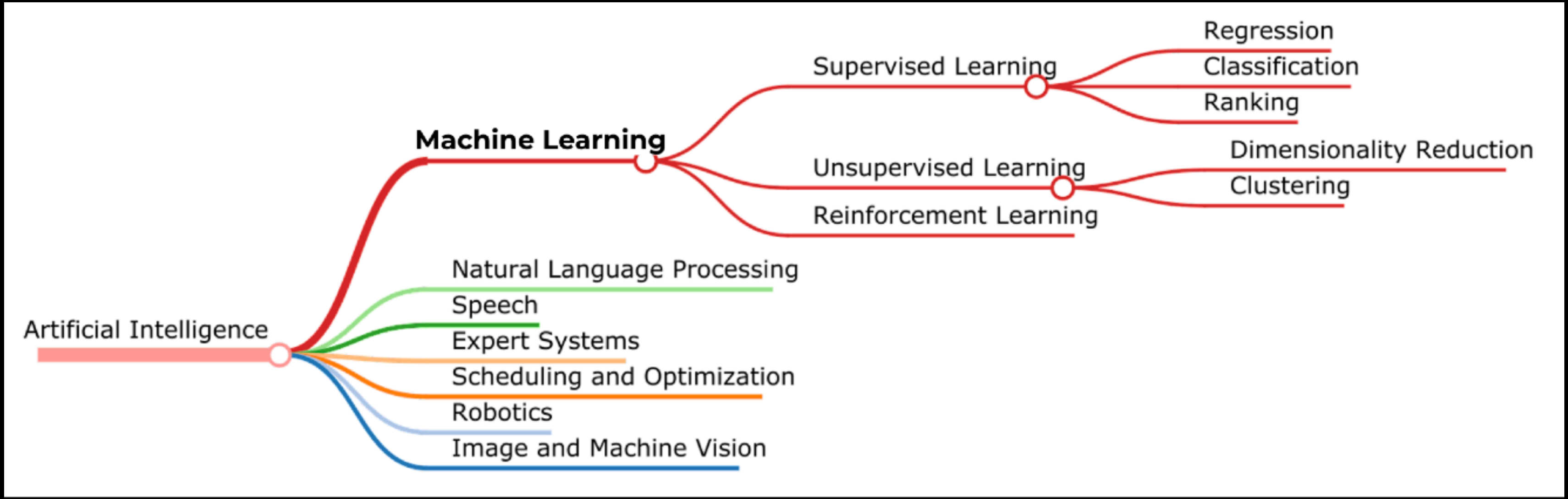
Week	Summary	What I actually plan to talk about
4	Data Processing	git, csv/FITS Files, ADQL/SQL, joining & cleaning catalogues, ...
	Statistics & Plots	Uncertainties & Plot Clinic
8	Regression	how to fit y = f(x), if y (and even x) have uncertainties, python fitting packages and when to apply them how to which function, ...
	Dimensionality Reduction	Principal Component Analysis (PCA), tSNE, ...
9	Clustering	k-means, HDBSCAN, Gaussian Mixture Models (GMM), ...
	Model Selection	AIC & BIC, train/test sets, ...
	Interdisciplinary Thinking	How to think abstract or creative and bridge barriers/gaps: How your expertise can help other researchers/industry, ...

Supervised Learning

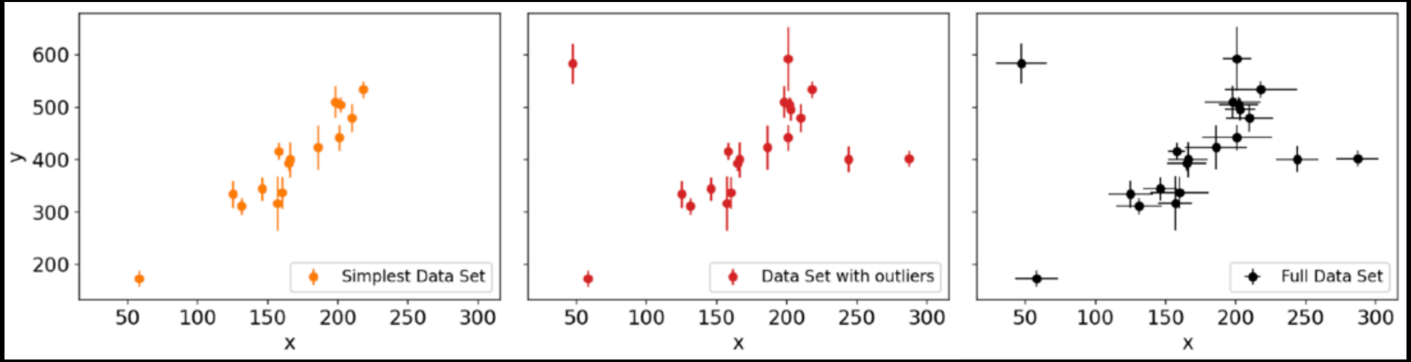
https://scikit-learn.org/stable/supervised_learning.html

Unsupervised Learning

https://scikit-learn.org/stable/unsupervised_learning.html



Today’s tasks: Fitting a line & get distances from parallaxes properly!

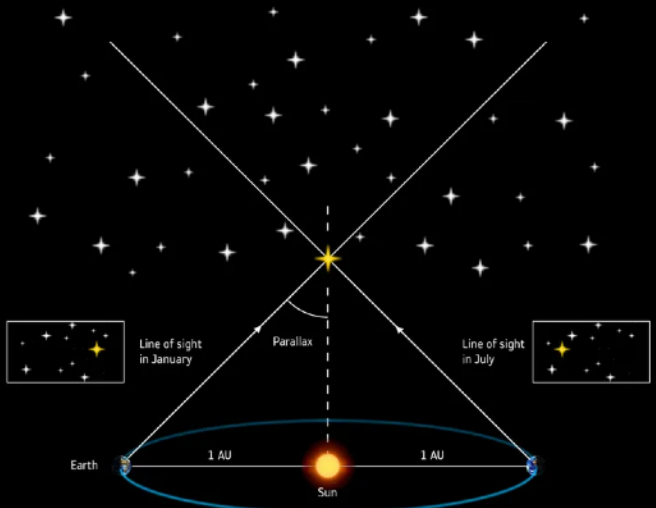


David Hogg, Jo Bovy, Dustin Lang (2010): arxiv.org/abs/1008.4686

Abstract

We go through the many considerations involved in fitting a model to data, using as an example the fit of a straight line to a set of points in a two-dimensional plane. ...

Today's tasks: Fitting a line & get distances from parallaxes properly!



$$D_{\varpi} = distance = \frac{1}{parallax} \cdot \frac{1\text{ pc}}{1\text{ arcsec}} = \frac{1}{\varpi} \cdot \frac{1\text{ pc}}{1\text{ arcsec}}$$

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w}) \times p(\mathbf{w})$$ or $$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

What if we know that all $D_{\varpi} \geq 0$ pc?!

1 2023AJ....166..269B 2023/12 cited: 7
Estimating Distances from Parallaxes. VI. A Method for Inferring Distances and Transverse Velocities from Parallaxes and Proper Motions Demonstrated on Gaia Data Release 3
Bailer-Jones, C. A. L.

2 2021AJ....161..147B 2021/03 cited: 1260
Estimating Distances from Parallaxes. V. Geometric and Photogeometric Distances to 1.47 Billion Stars in Gaia Early Data Release 3
Bailer-Jones, C. A. L.; Rybizki, J.; Fouesneau, M.
and 2 more

3 2018AJ....156...58B 2018/08 cited: 1623
Estimating Distance from Parallaxes. IV. Distances to 1.33 Billion Stars in Gaia Data Release 2
Bailer-Jones, C. A. L.; Rybizki, J.; Fouesneau, M.
and 2 more

4 2016ApJ...833..119A 2016/12 cited: 128
Estimating Distances from Parallaxes. III. Distances of Two Million Stars in the Gaia DR1 Catalogue
Astraatmadja, Tri L.;
Bailer-Jones, Coryn A. L.

5 2016ApJ...832..137A 2016/12 cited: 131
Estimating Distances from Parallaxes. II. Performance of Bayesian Distance Estimators on a Gaia-like Catalogue
Astraatmadja, Tri L.;
Bailer-Jones, Coryn A. L.

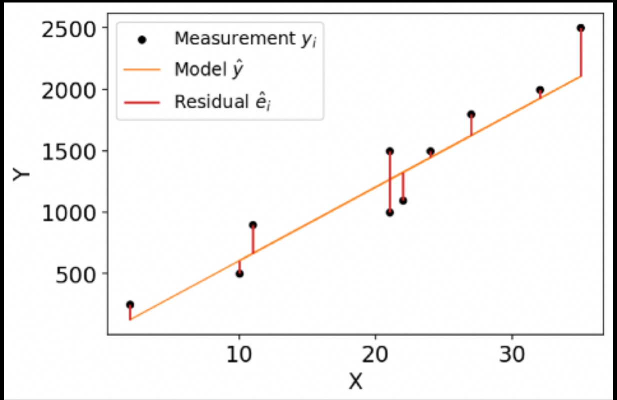
6 2015PASP..127..994B 2015/10 cited: 432
Estimating Distances from Parallaxes
Bailer-Jones, C. A. L.

HOW TO:
FIT A LINE

Australian National University

What we will cover:

1. Building intuition
2. Only uncertainties for y:
    2.1. Linear algebra
    2.2. Numerical solutions
        2.2.1. np.polyfit
        2.2.2. scipy.optimize.curve_fit
        2.2.3. statsmodel.api
3. Uncertainties for x and y:
    3.1. scipy.optimize.minimize
    3.2. scipy.odr



Fitting a linear model to data: Complexity Level 0

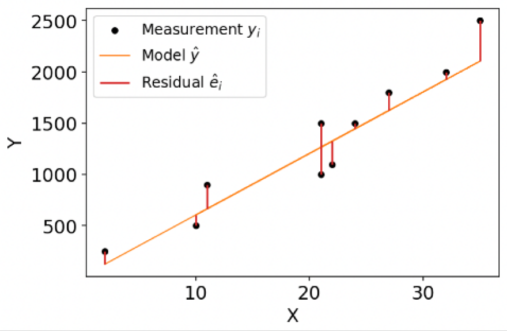
What we have n data points (x_i, y_i) without uncertainties

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

We want to fit a linear function with intercept c_0 and slope c_1

$$y_i = c_0 + c_1 x_i + \epsilon_i$$

Matrix form: $\mathbf{y} = \mathbf{X}\mathbf{c} + \epsilon$



$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Data Vector

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

Design matrix

$$\mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$$

Coefficient Vector

Fitting a linear model to data

Matrix form: $\mathbf{y} = \mathbf{X}\mathbf{c} + \epsilon$

To find the coefficients c that minimise the sum of squared residuals, we use the normal question (multiply above with transpose matrix X^T):

$$\mathbf{X}^T\mathbf{X}\mathbf{c} = \mathbf{X}^T\mathbf{y}$$

multiply each side with $(\mathbf{X}^T\mathbf{X})^{-1}$

$$\mathbf{c} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathrm{Cov}(\mathbf{c}) = (\mathbf{X}^T\mathbf{X})^{-1}$$

Covariance matrix of c

$$\sigma_{c_0} = \sqrt{\mathrm{Cov}_{00}}$$

$$\sigma_{c_1} = \sqrt{\mathrm{Cov}_{11}}$$

Example

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

$$(1,2), (2,2.8), (3,3.6)$$

$$\mathbf{y} = \begin{pmatrix} 2 \\ 2.8 \\ 3.6 \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ 6 & 14 \end{pmatrix}$$

$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 2 \\ 2.8 \\ 3.6 \end{pmatrix} = \begin{pmatrix} 8.4 \\ 18.8 \end{pmatrix}$$

$$\widehat{\mathrm{Cov}}(\hat{c}) = \hat{\sigma}^2 (X^\top X)^{-1}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{3 \cdot 14 - 6 \cdot 6} \begin{pmatrix} 14 & -6 \\ -6 & 3 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 14 & -6 \\ -6 & 3 \end{pmatrix} = \begin{pmatrix} \frac{7}{3} & -1 \\ -1 & \frac{1}{2} \end{pmatrix}$$

det(X)

$$\mathbf{c} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{c} = \begin{pmatrix} \frac{7}{3} & -1 \\ -1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 8.4 \\ 18.4 \end{pmatrix} = \begin{pmatrix} 1.2 \\ 0.8 \end{pmatrix}$$

$$y = 1.2 + 0.8 \cdot x$$

```python
import numpy as np

# Step 1: Define the data
x = x_data_simple
y = y_data_simple

# Step 2: Create the design matrix X
# We add a column of ones for the bias term (c_0)
X = np.column_stack((np.ones(x.shape[0]), x))

# Step 3: Compute the normal equation components
# X^T X and X^T y
XT_X = np.dot(X.T, X)  # X.T is the transpose of X
XT_y = np.dot(X.T, y)

# Step 4: Compute the covariance matrix of the coefficients
cov_matrix = np.linalg.inv(XT_X)

# Step 5: Solve for the coefficients (c_0, c_1)
coefficients = cov_matrix.dot(XT_y)

# Step 5: Output the coefficients
c_0, c_1 = coefficients

# Step 6: Exctract coefficient sigma
diagonal_entries_sigma = np.sqrt(np.diag(cov_matrix))
c_0_sigma = diagonal_entries_sigma[0]
c_1_sigma = diagonal_entries_sigma[1]

# Step 7: Use the coefficients to predict y values
y_pred = X.dot(coefficients)
```

$$\sigma_{c_0} = \sqrt{\mathrm{Cov}_{00}} \quad \sigma_{c_1} = \sqrt{\mathrm{Cov}_{11}}$$

Complexity Level 1: uncertainties on y_i

$$x = [1, 2, 3]$$
$$y = [2, 2.8, 3.6]$$
$$\sigma_y = [0.1, 0.2, 0.3]$$

$$y_i = c_0 + c_1 x_i + \epsilon_i$$

$$\mathbf{y} = \mathbf{X}\mathbf{c} + \epsilon$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

Design matrix

$$\mathbf{W} = \mathrm{diag}\left(\frac{1}{\sigma_{y_1}^2}, \frac{1}{\sigma_{y_2}^2}, \ldots, \frac{1}{\sigma_{y_n}^2}\right)$$

Weights matrix

Weighted normal equation:

Multiply with $X^T W$:

$$\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{c} = \mathbf{X}^T\mathbf{W}\mathbf{y}$$

Multiply with $(X^T W X)^{-1}$

$$\mathbf{c} = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{y}$$

$$\mathrm{Cov}(\mathbf{c}) = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}$$

```python
import numpy as np

# Step 1: Define the data and uncertainties
x = x_data_simple
y = y_data_simple
y_sigma = y_sigma_simple

# Step 2: Create the design matrix X
# We add a column of ones for the bias term (c_0)
X = np.column_stack((np.ones(x.shape[0]), x))

# Step 3: Create the weights matrix W
W = np.diag(1 / y_sigma**2)  # Diagonal matrix of 1/y_sigma^2

# Step 4: Compute the weighted normal equation components
# X^T W X and X^T W y
XT_W_X = np.dot(X.T, np.dot(W, X))
XT_W_y = np.dot(X.T, np.dot(W, y))

# Step 5: Compute the covariance matrix of the coefficients
cov_matrix = np.linalg.inv(XT_W_X)

# Step 5: Solve for the coefficients (c_0, c_1)
coefficients = cov_matrix.dot(XT_W_y)

# Step 6: Output the coefficients
c_0, c_1 = coefficients

# Step 7: Exctract coefficient sigma
diagonal_entries_sigma = np.sqrt(np.diag(cov_matrix))
c_0_sigma = diagonal_entries_sigma[0]
c_1_sigma = diagonal_entries_sigma[1]

# Step 8: Use the coefficients to predict y values
y_pred = X.dot(coefficients)
```

Complexity Level 2: outliers
Complexity Level 3: quadratic function

$$y = c_0 + c_1 x + c_2 x^2$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}$$

$$\mathbf{X}^T \mathbf{W} \mathbf{X} \mathbf{c} = \mathbf{X}^T \mathbf{W} \mathbf{y}$$

$$\mathbf{X}^T \mathbf{X} \mathbf{c} = \mathbf{X}^T \mathbf{y}$$

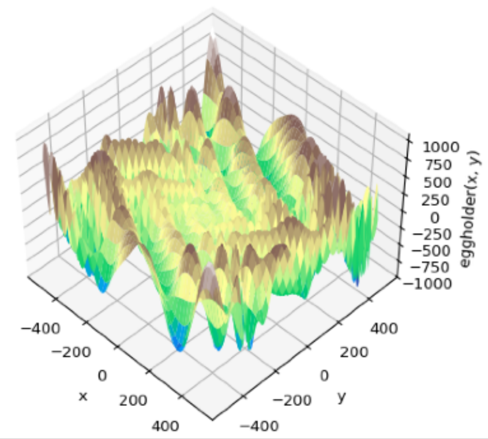
$$\mathrm{Cov}(\mathbf{c}) = \left(\mathbf{X}^T \mathbf{X}\right)^{-1}$$

The maths stays the same!

IN PRACTICE:

FUNCTIONS CAN BE MORE COMPLICATED.

WHEN TO USE PACKAGES FOR FITTING?



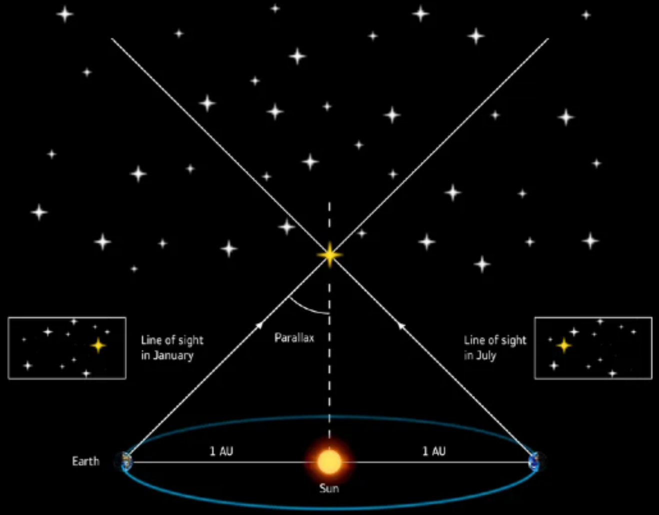
Australian National University

BREAK UNTIL 2PM

Australian National University

HOW TO:

ESTIMATE DISTANCES FROM PARALLAXES



Australian National University

PLOT CLINIC