Inferencia estadística

BIG DATA CON PYTHON

CURSOS DE VERANO UC

UNIVERSIDAD DE CANTABRIA

INTRODUCTION TO PYTHON FOR BIG DATA

Outline

1. Introducción a la optimización

2. Conceptos básicos
   - Variables, Función objetivo, Ligaduras

3. Resolver problemas
   - Solución analítica, grid, algoritmos

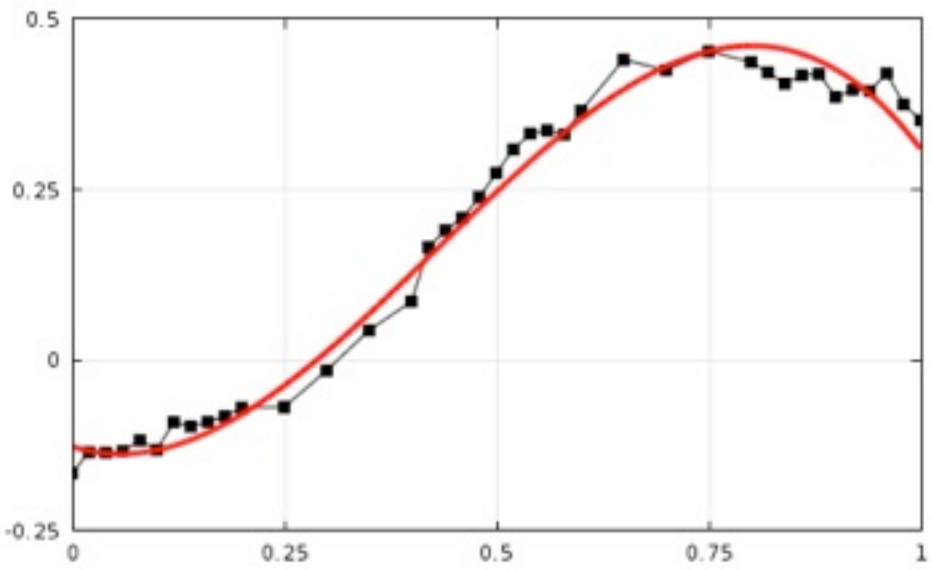
4. Tipos de problemas
   - Optimización lineal
   - Optimización no lineal

--- Ejemplos python

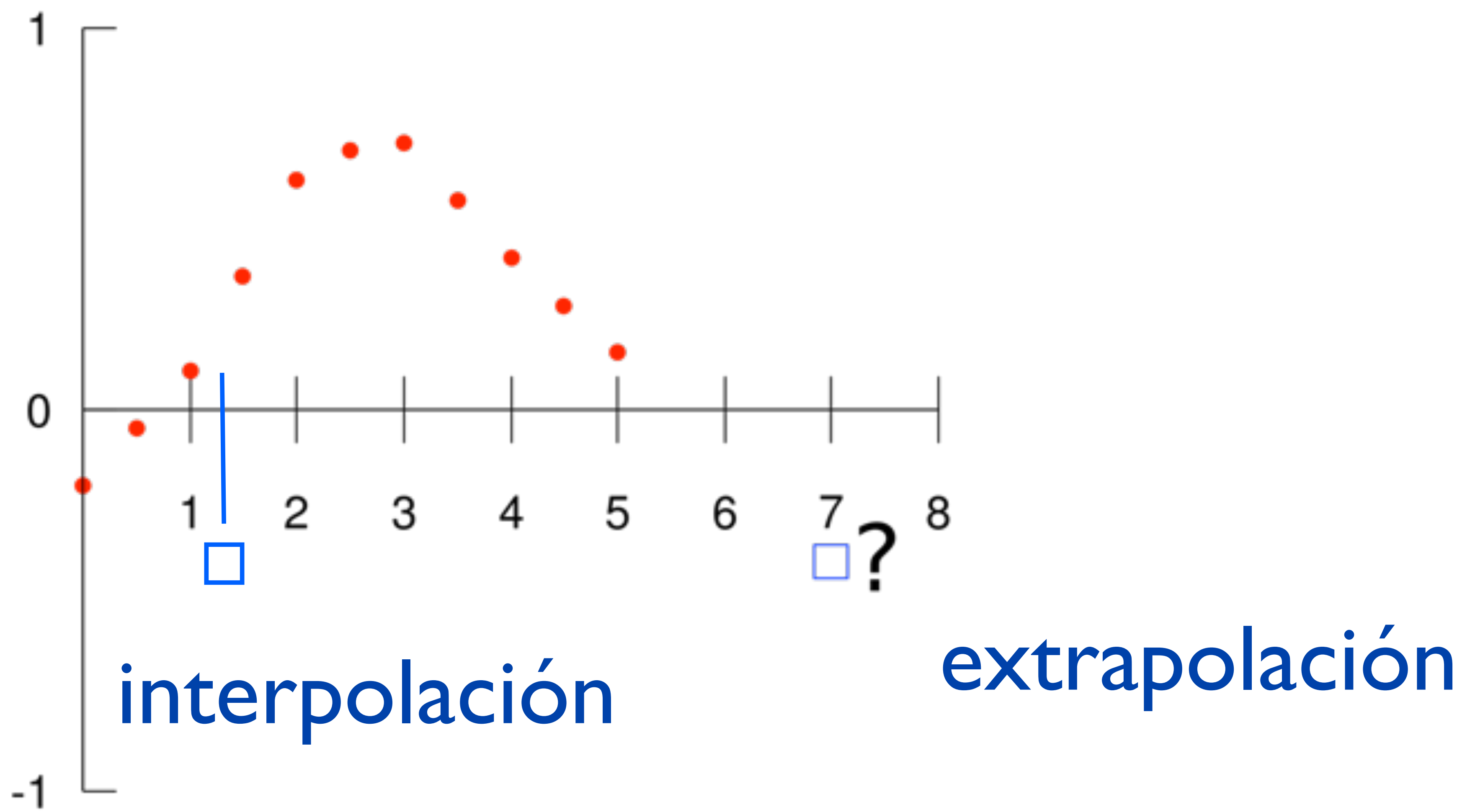
5. Inferencia estadística
   - interpolación y extrapolación
   - ajuste mínimos cuadrados

--- Ejemplos python

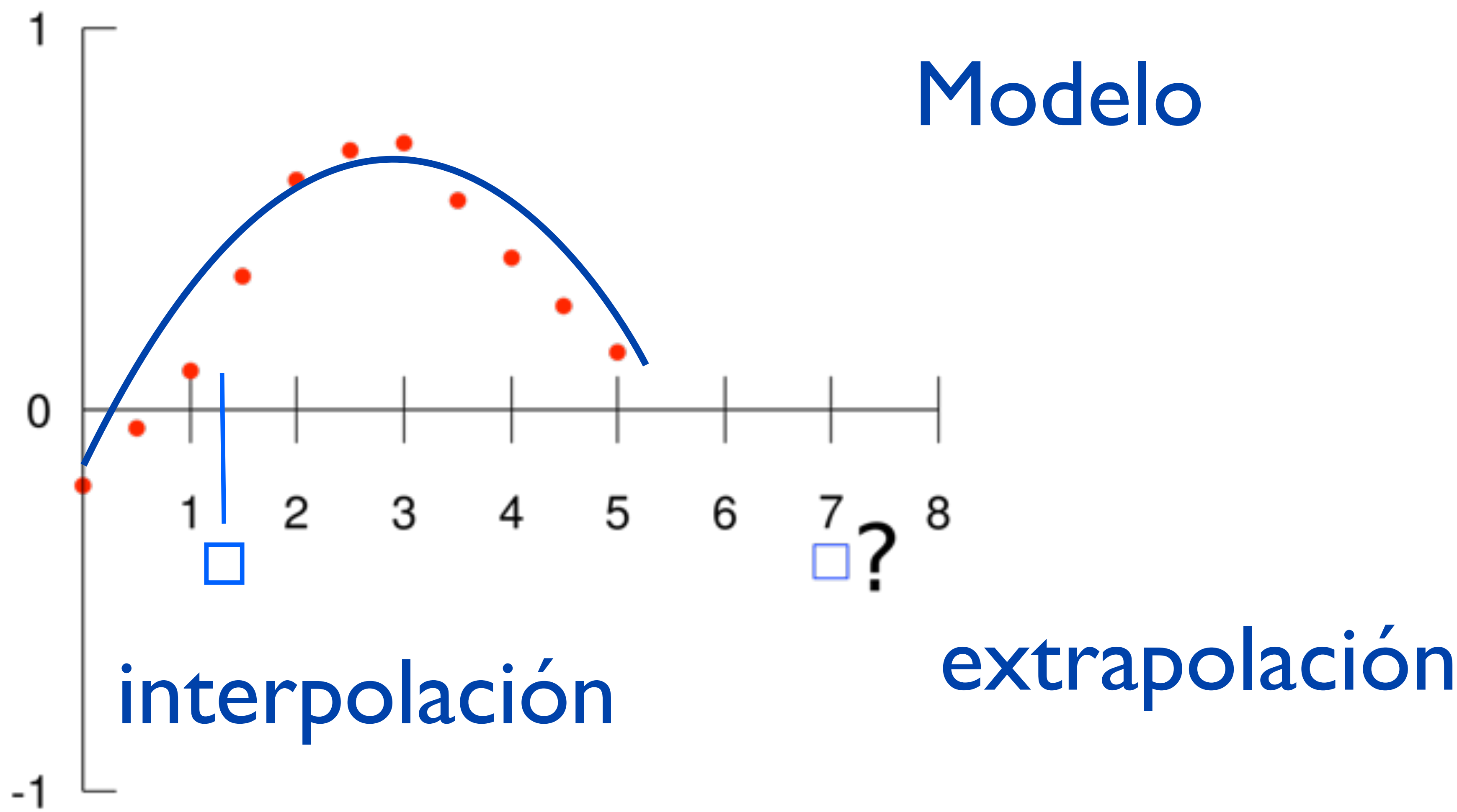
6. Lab



Interpolación, extrapolación y modelo



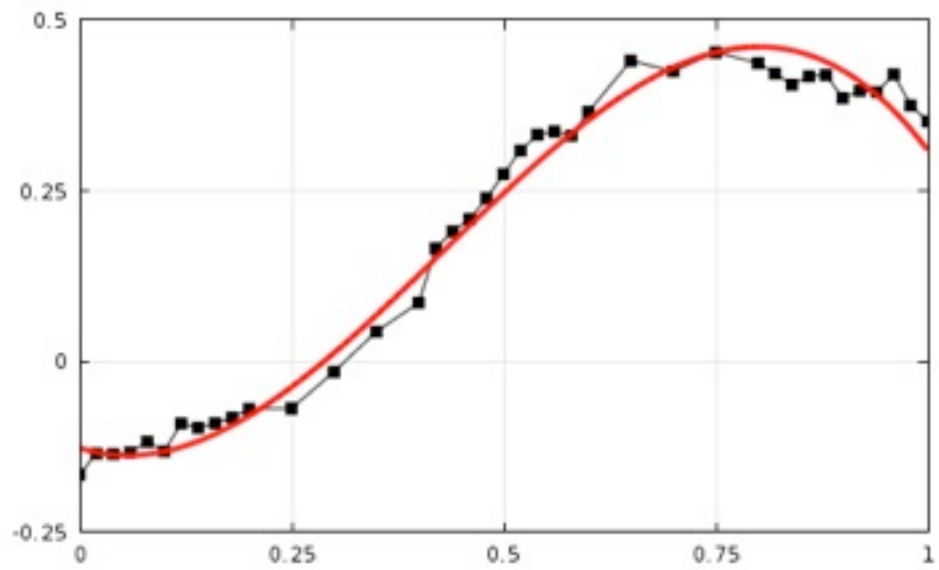
Interpolación, extrapolación y modelo



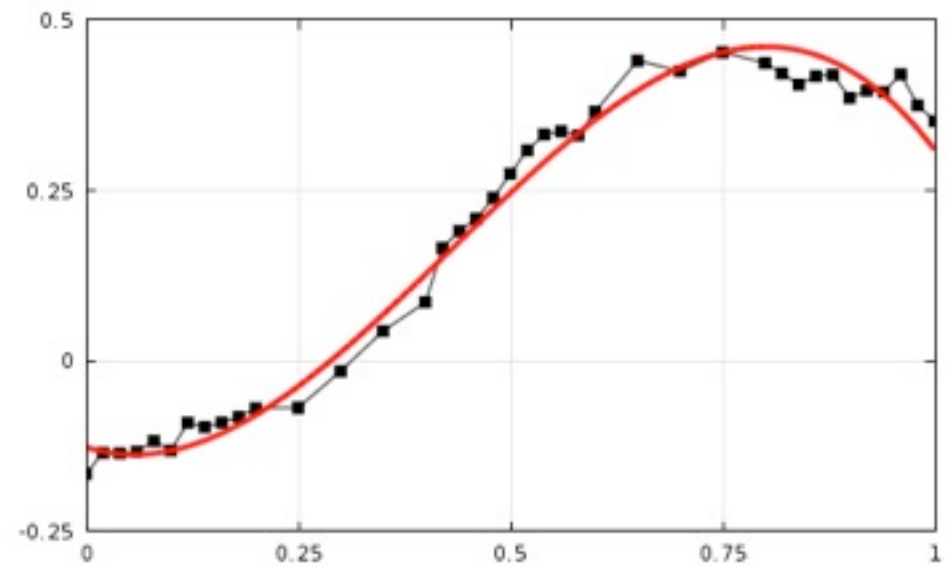
Modelizar: Dados unos datos encontrar una ecuación matemática que nos relacione dos o más variables.

Modelizar: Dados unos datos encontrar una ecuación matemática que nos relacione dos o más variables.

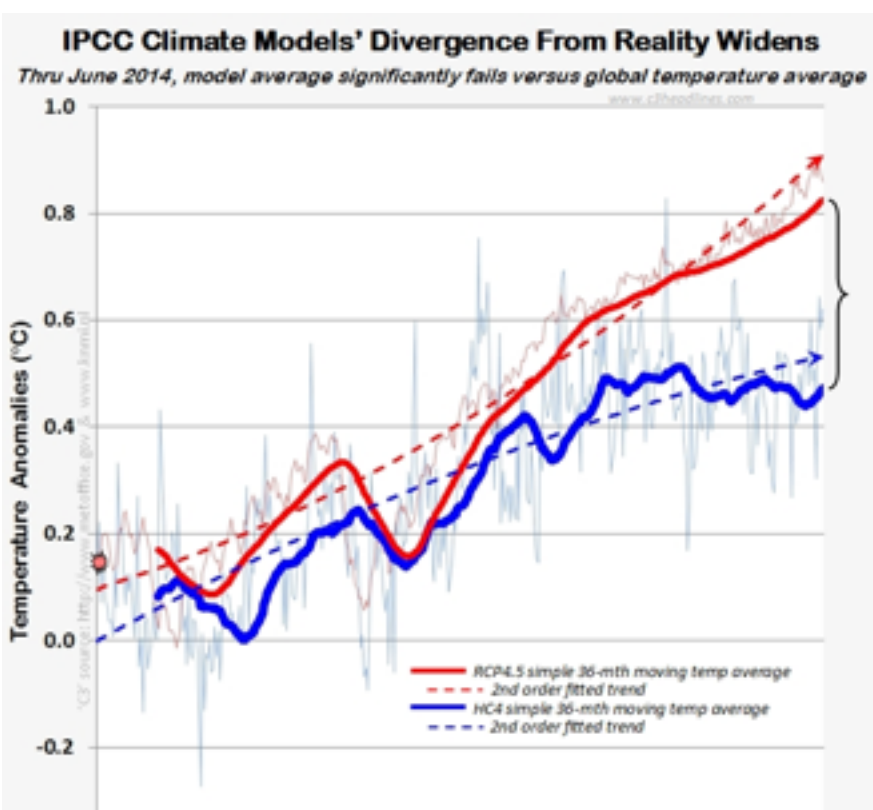
Modelo téorico: Tenemos una relación matemática que puede estar dado por leyes conocidas o por hipotésis con cierta **base teórica** detrás.



Modelizar: Dados unos datos encontrar una ecuación matemática que nos relacione dos o más variables.



Modelo téorico: Tenemos una relación matemática que puede estar dado por leyes conocidas o por hipotésis con cierta **base teórica** detrás.



Modelo empírico: A partir de los datos intentamos **encontrar una relación** entre variables que nos permita predecir resultados con datos nuevos. Ajuste a un polinomio, **redes neuronales**. (MAÑANA)

Interpolación y extrapolación

Ejemplo:Mareas

Entre las 0:00 y las 9:00 de la mañana medimos a cada hora (t) la distancia que ha subido el mar (d).

Queremos saber a que altura ha estado el agua a las 0:30h. **Interpolación**

Queremos saber a que altura estará el agua a las 10 horas. **Extrapolación**

Queremos saber la altura del agua a cada minuto durante todos los días. Necesitamos un **Modelo.**

Interpolación y extrapolación

Ejemplo:Mareas

Entre las 0:00 y las 9:00 de la mañana medimos a cada hora (t) la distancia que ha subido el mar (d).

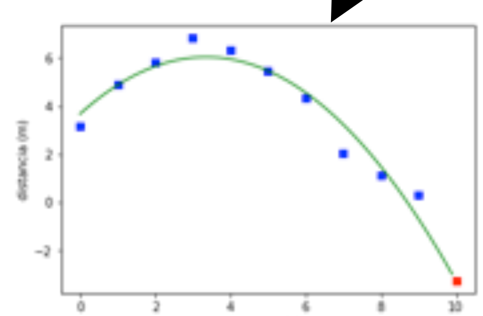
Queremos saber a que altura ha estado el agua a las 0:30h. **Interpolación**

Queremos saber a que altura estará el agua a las 10 horas. **Extrapolación**

Queremos saber la altura del agua a cada minuto durante todos los días. Necesitamos un **Modelo.**

$$f_L = 2\frac{GM_L}{r^2}\frac{R}{r}$$

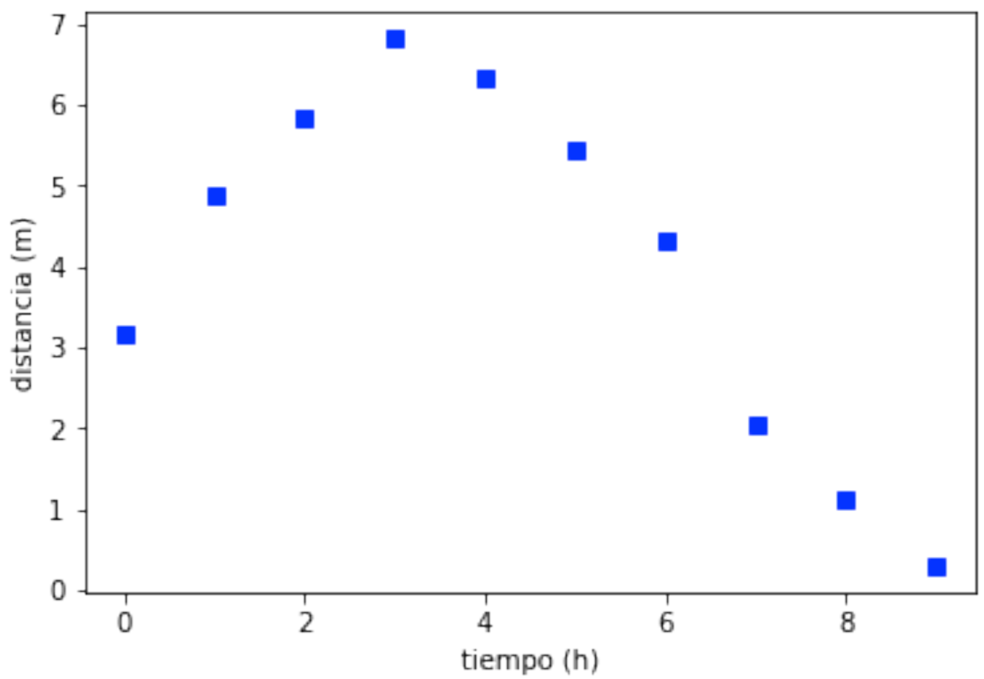
teórico



empírico

Interpolación y extrapolación.

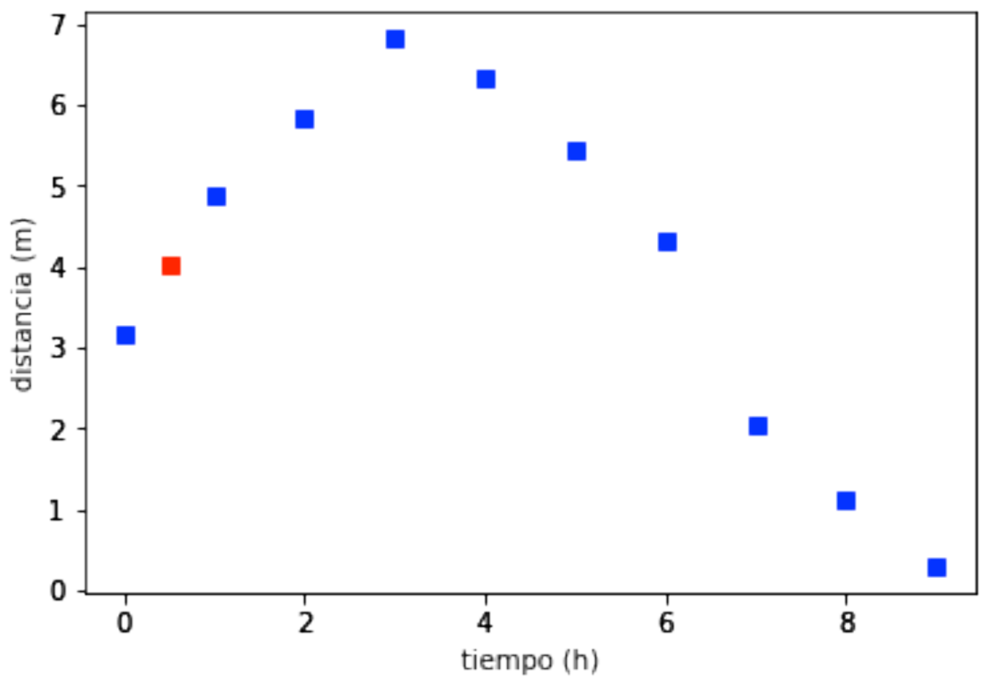
Interpolación



Con estos datos, podemos intentar inferir que es lo que pasaba a 0:30 h

Interpolación y extrapolación.

Interpolación



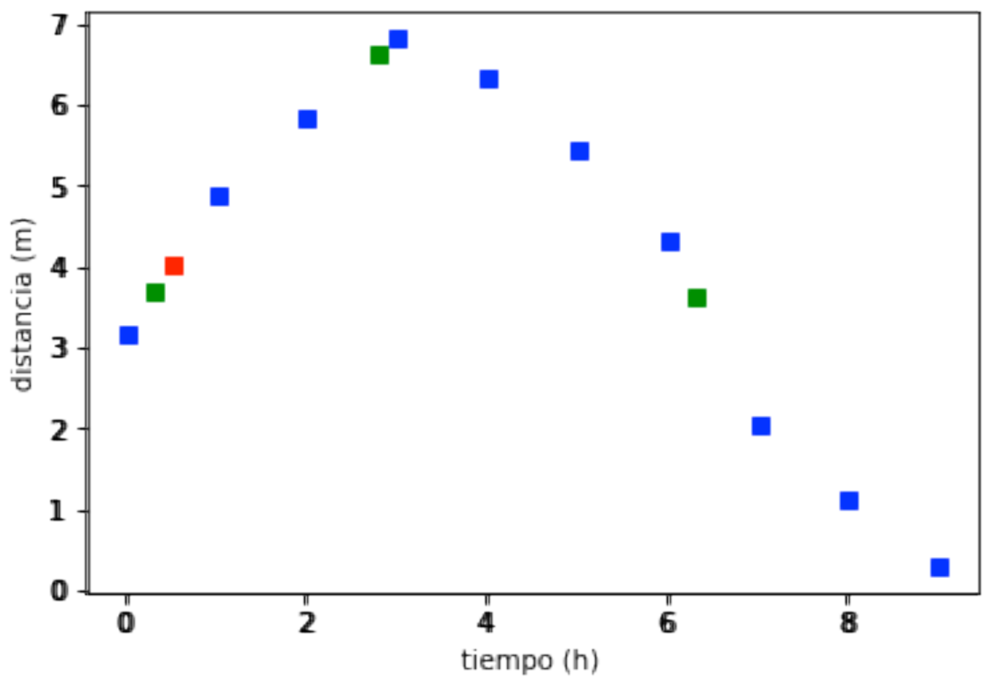
Con estos datos, podemos intentar inferir que es lo que pasaba a 0:30 h

por ejemplo: haciendo la media

t=0.5 --> $(d_0+d_1)/2$

Interpolación y extrapolación.

Interpolación



Con estos datos, podemos intentar inferir que es lo que pasaba a 0:30 h

por ejemplo: haciendo la media

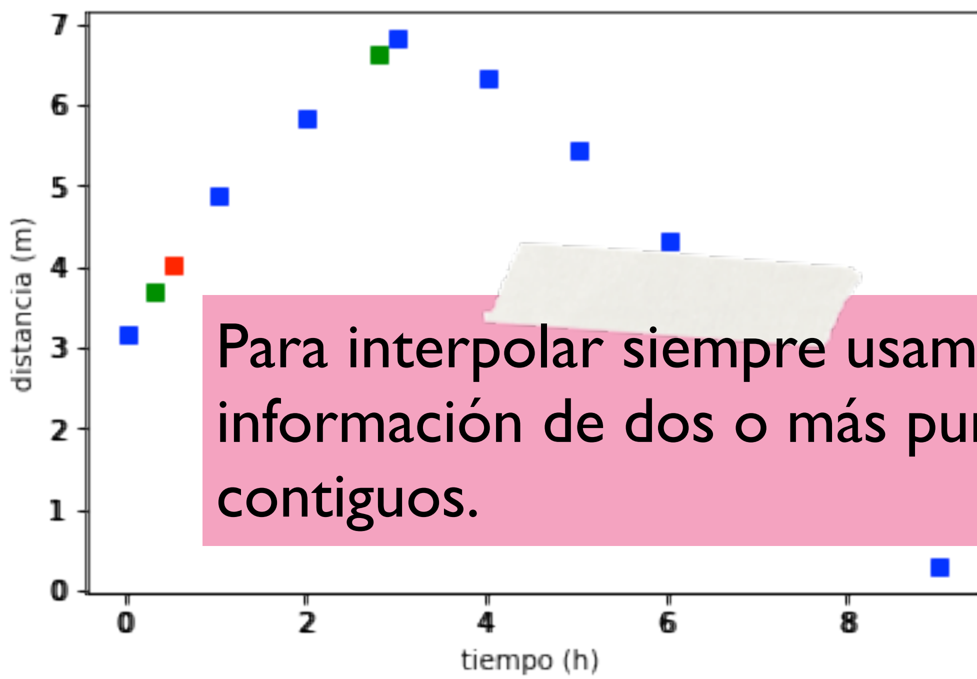
t=0.5 --> $(d_0+d_1)/2$

Para cualquier punto hay varios métodos de interpolación:

```
scipy.interpolate.interp1d(x, y, kind='linear', ...)
```

Interpolación y extrapolación.

Interpolación



Para interpolar siempre usamos la información de dos o más puntos contiguos.

Con estos datos, podemos intentar inferir que es lo que pasaba a 0:30 h

por ejemplo: haciendo la media

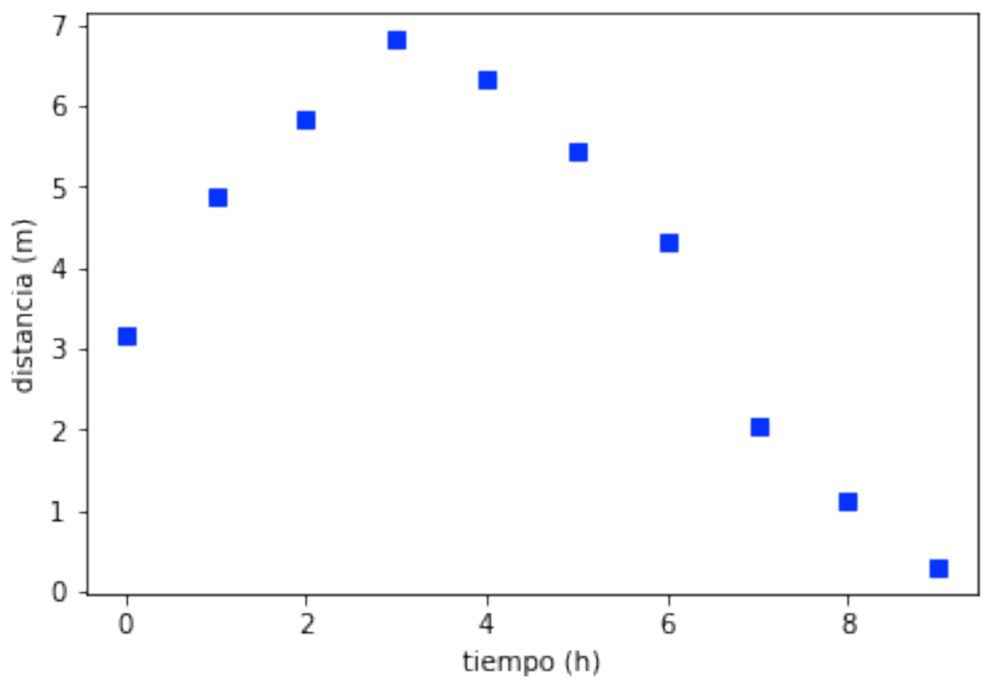
$t=0.5 \dashrightarrow (d_0+d_1)/2$

Para cualquier punto hay varios métodos de interpolación:

```
scipy.interpolate.interp1d(x, y, kind='linear', ...)
```

Interpolación y extrapolación.

Extrapolación

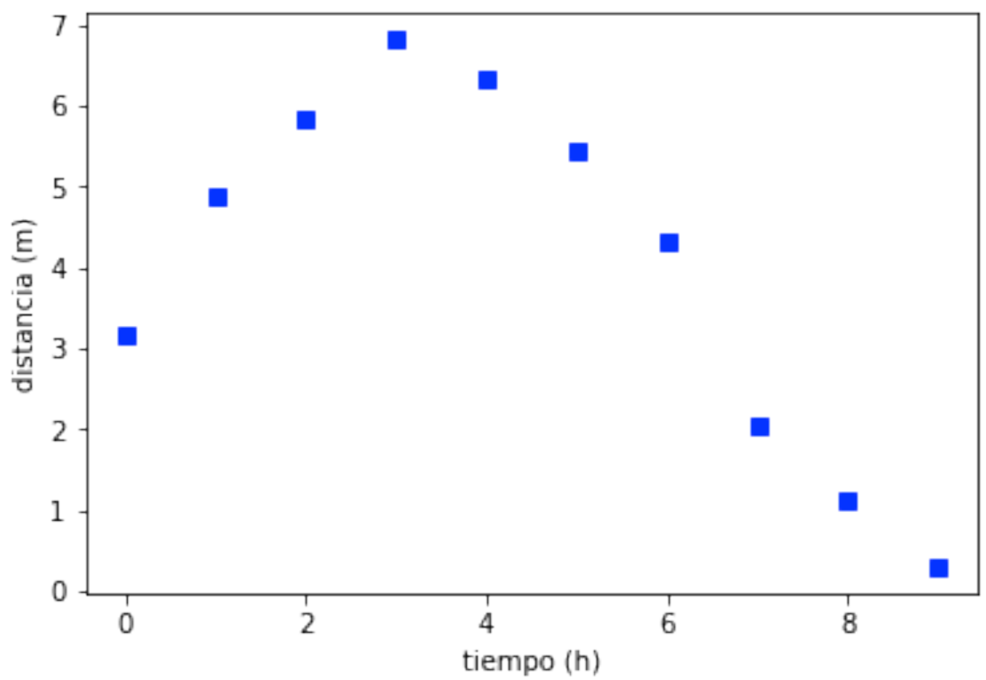


??

Para extrapolar solo podemos usar los puntos anteriores.

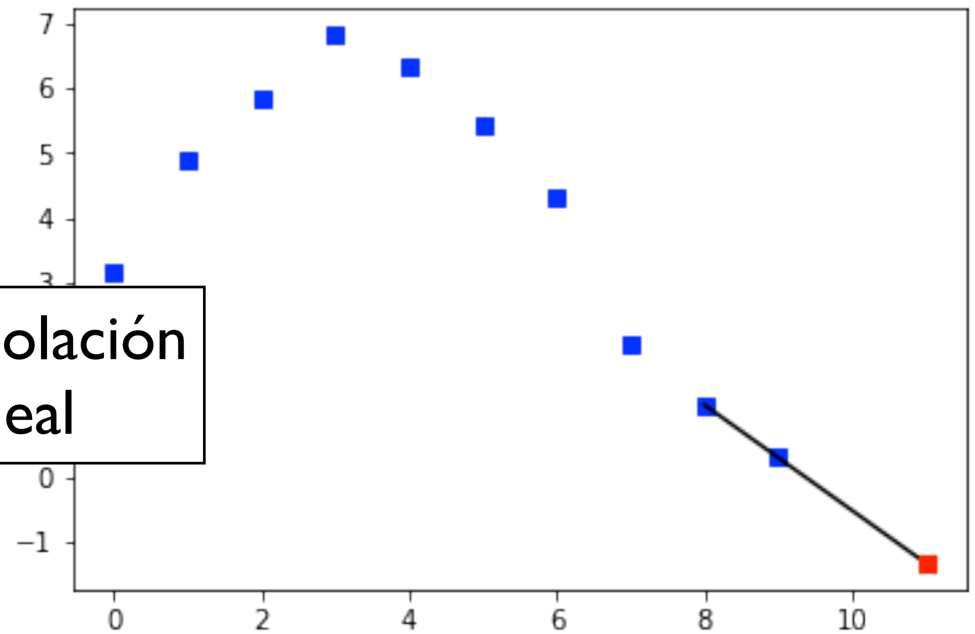
Interpolación y extrapolación.

Extrapolación



??

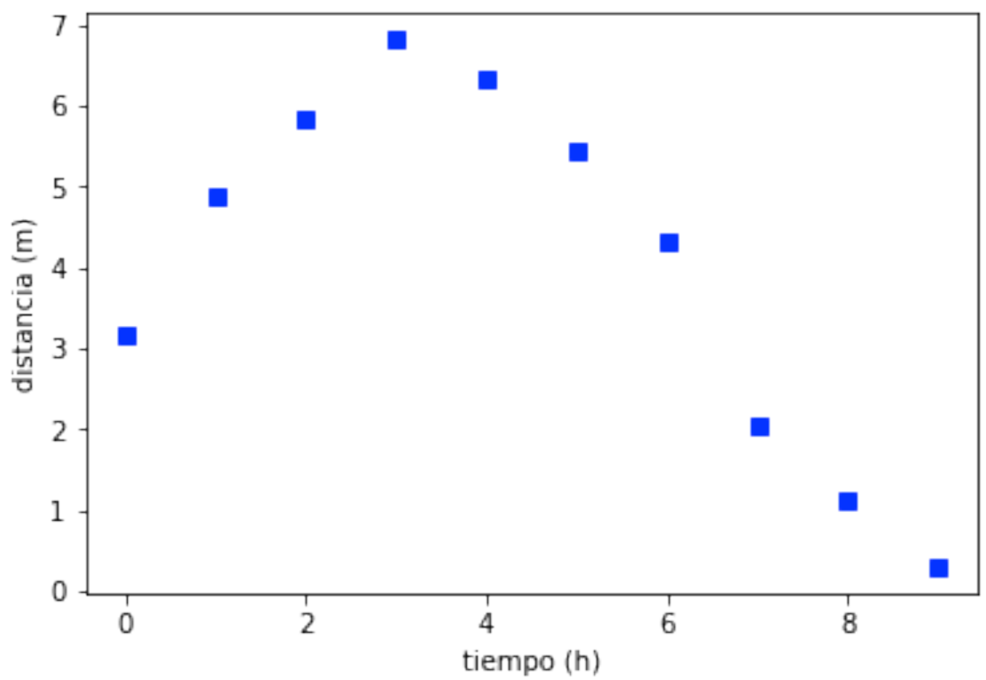
Para extrapolar solo podemos usar los puntos anteriores.



Extrapolación lineal

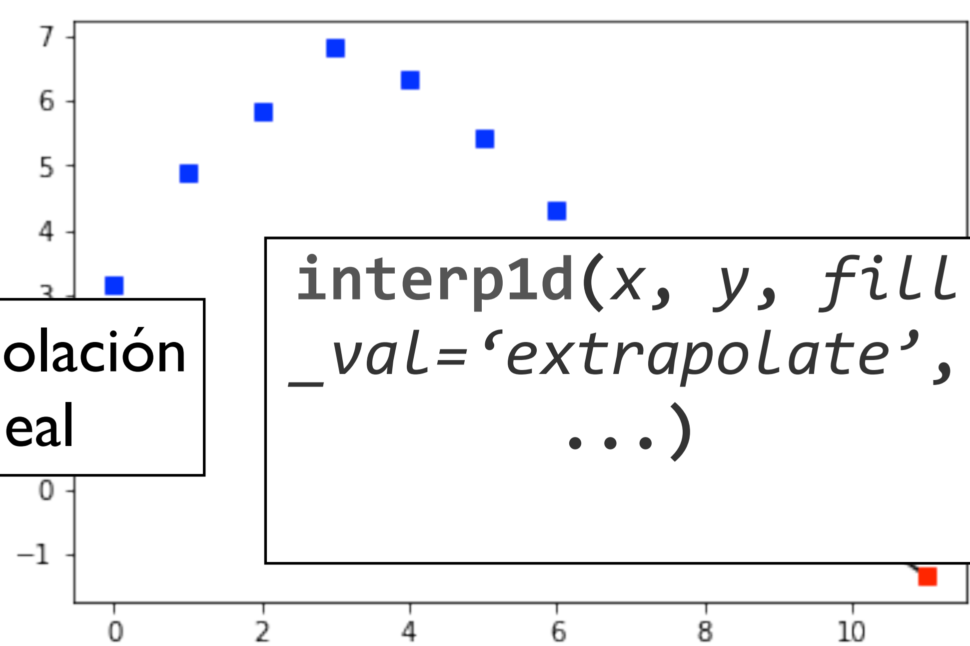
Interpolación y extrapolación.

Extrapolación



??

Para extrapolar solo podemos usar los puntos anteriores.

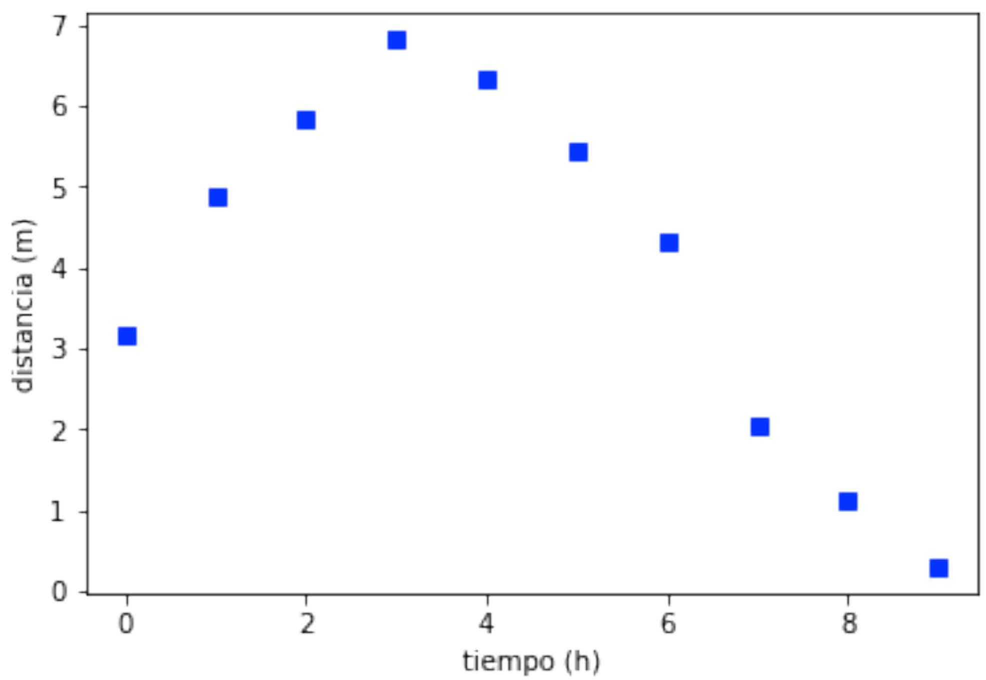


Extrapolación lineal

`interp1d(x, y, fill_val='extrapolate', ...)`

Interpolación y extrapolación.

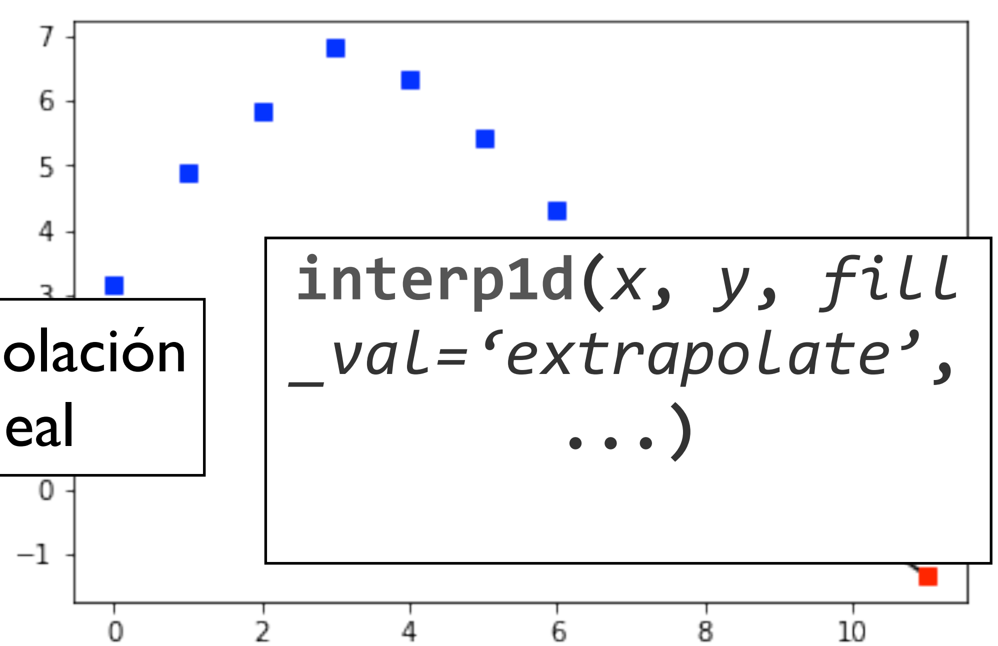
Extrapolación



Para extrapolar solo podemos usar los puntos anteriores.

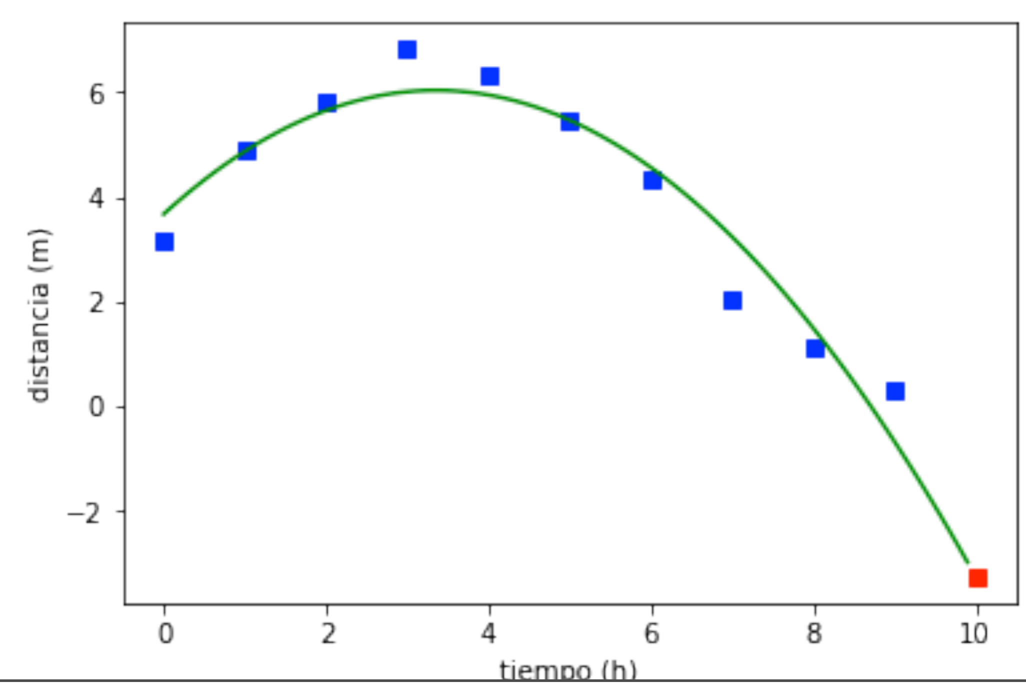
Y si usamos todos los puntos?

??



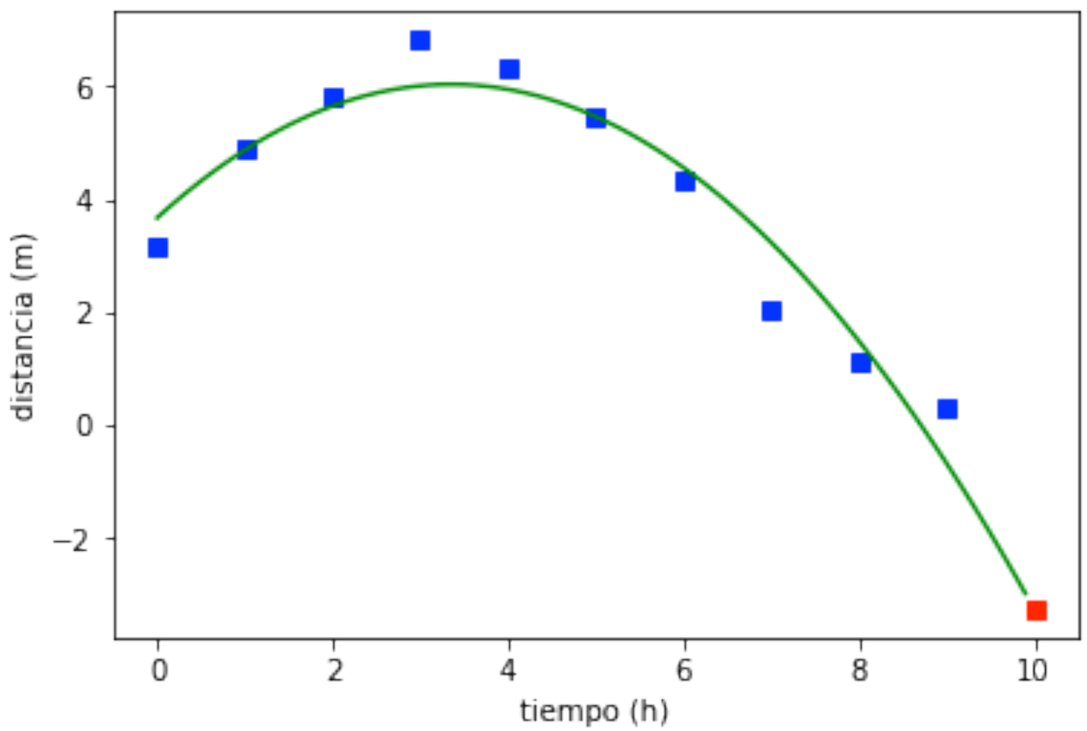
Extrapolación lineal

```
interp1d(x, y, fill_val='extrapolate', ...)
```



Interpolación y extrapolación.

Modelo



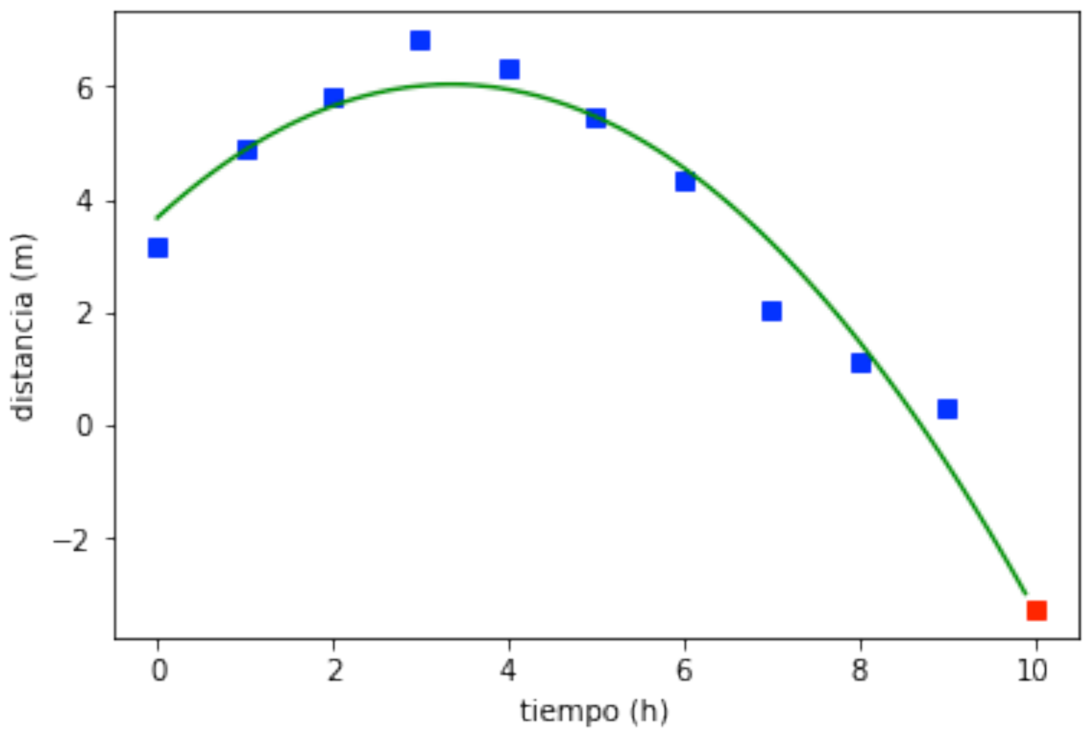
Hemos hecho un **ajuste** a un polinomio de orden 2

$$d = at^2 + bt + c$$

y hacemos un ajuste con python:
`numpy.polyfit(t,d,2)`

Interpolación y extrapolación.

Modelo



Hemos hecho un **ajuste** a un polinomio de orden 2

$$d = at^2 + bt + c$$

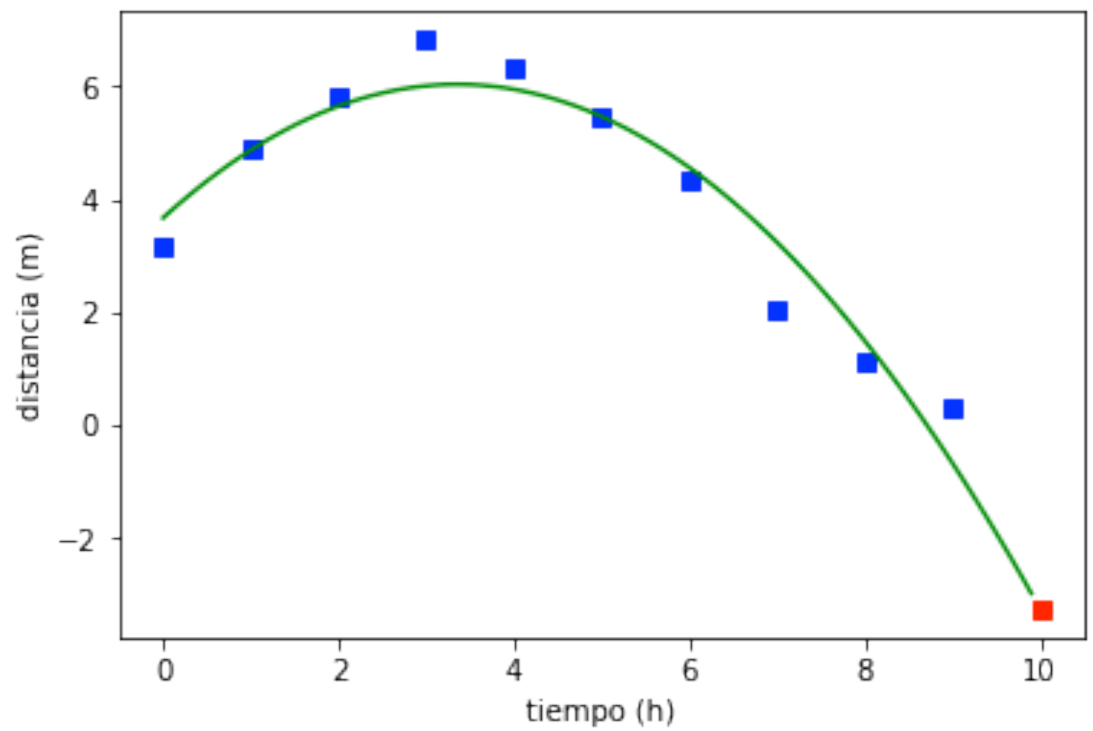
y hacemos un ajuste con python:
`numpy.polyfit(t,d,2)`

np.polyfit me da los **coeficientes**

$$d = -0.2t^2 + 1.4t + 3.7$$

Interpolación y extrapolación.

Modelo



Hemos hecho un **ajuste** a un polinomio de orden 2

$$d = at^2$$

y hacemos

python:

`numpy.po`

¡Ya está!

d=f(t)

A cada hora podemos saber la distancia que recorre el mar!

¿Ya podemos predecir la distancia a las 12 horas?

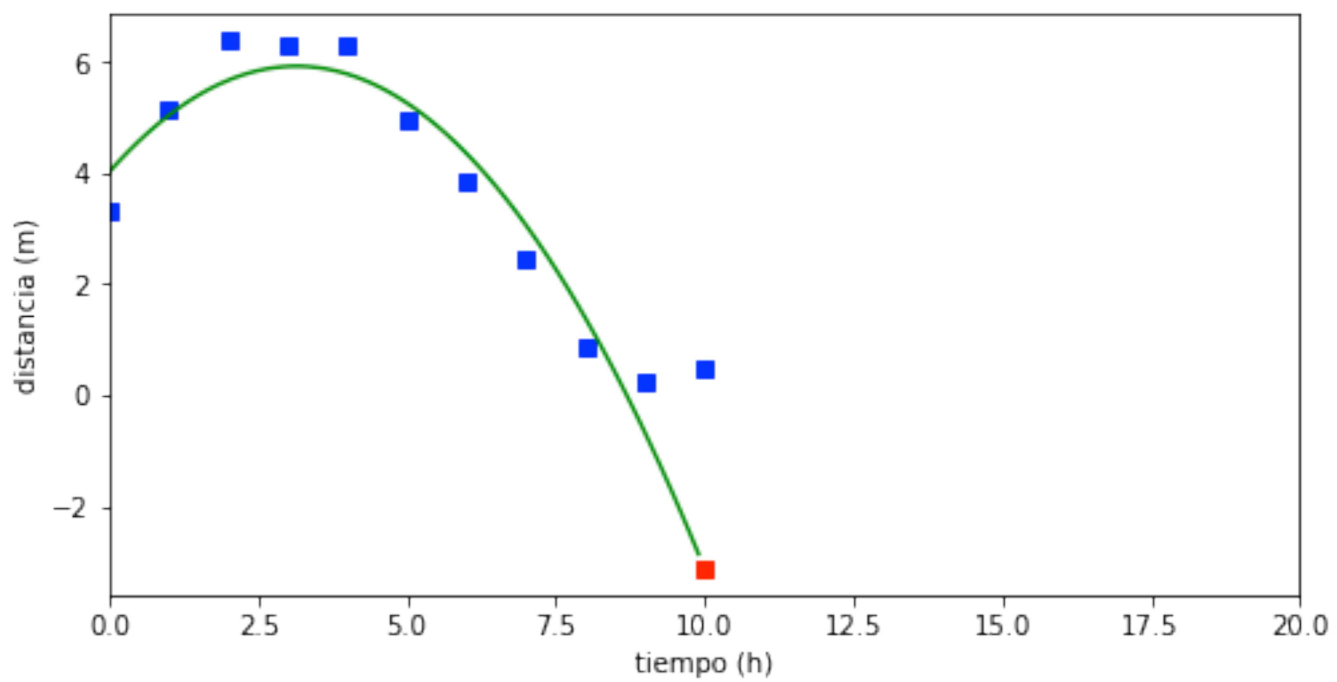
np.polyfit me da los **coeficientes**

$$d = -0.2t^2 + 1.4t + 3.7$$

Modelizar.

Modelo

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos

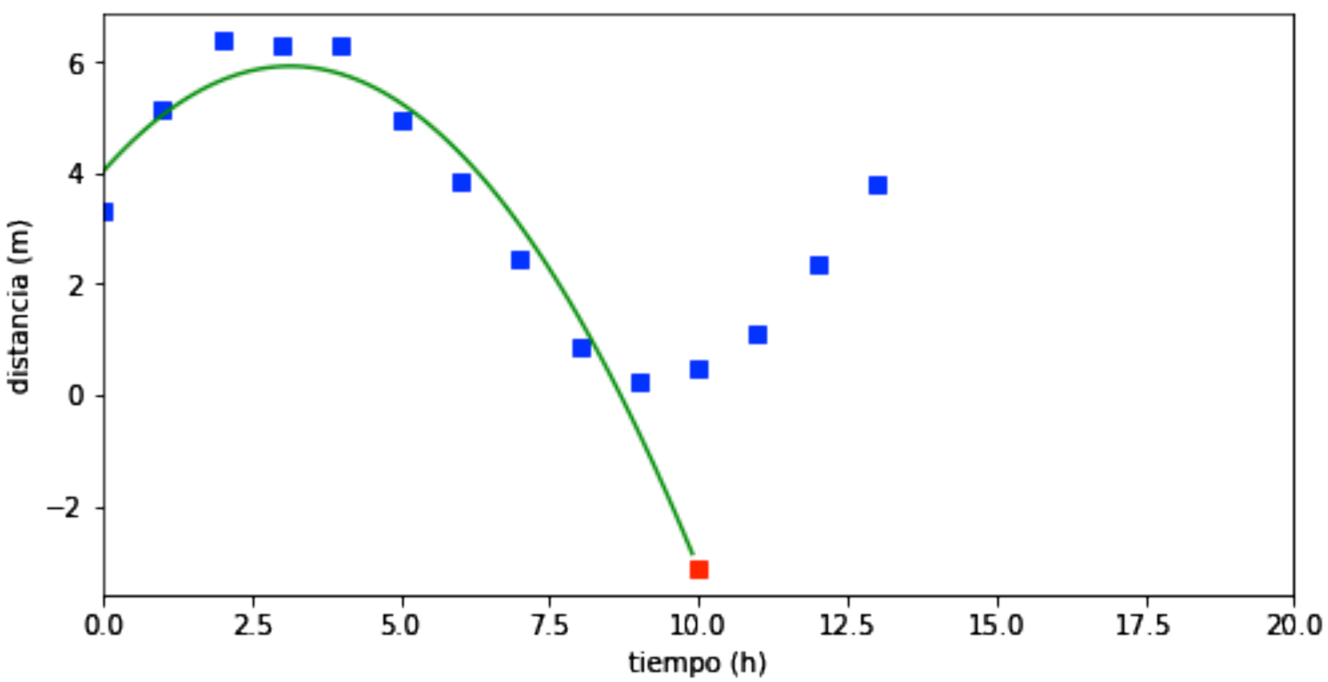


Tomamos más datos.

Modelizar.

Modelo

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos

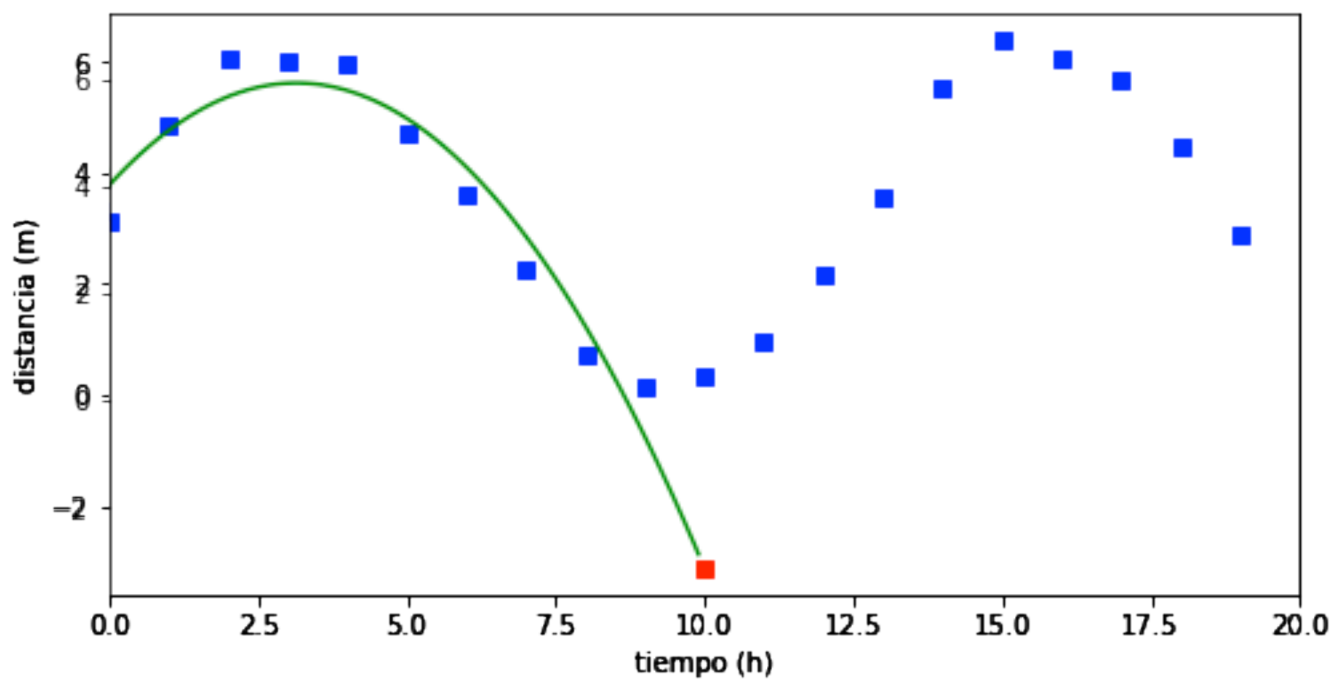


Tomamos más datos.

Modelizar.

Modelo

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos

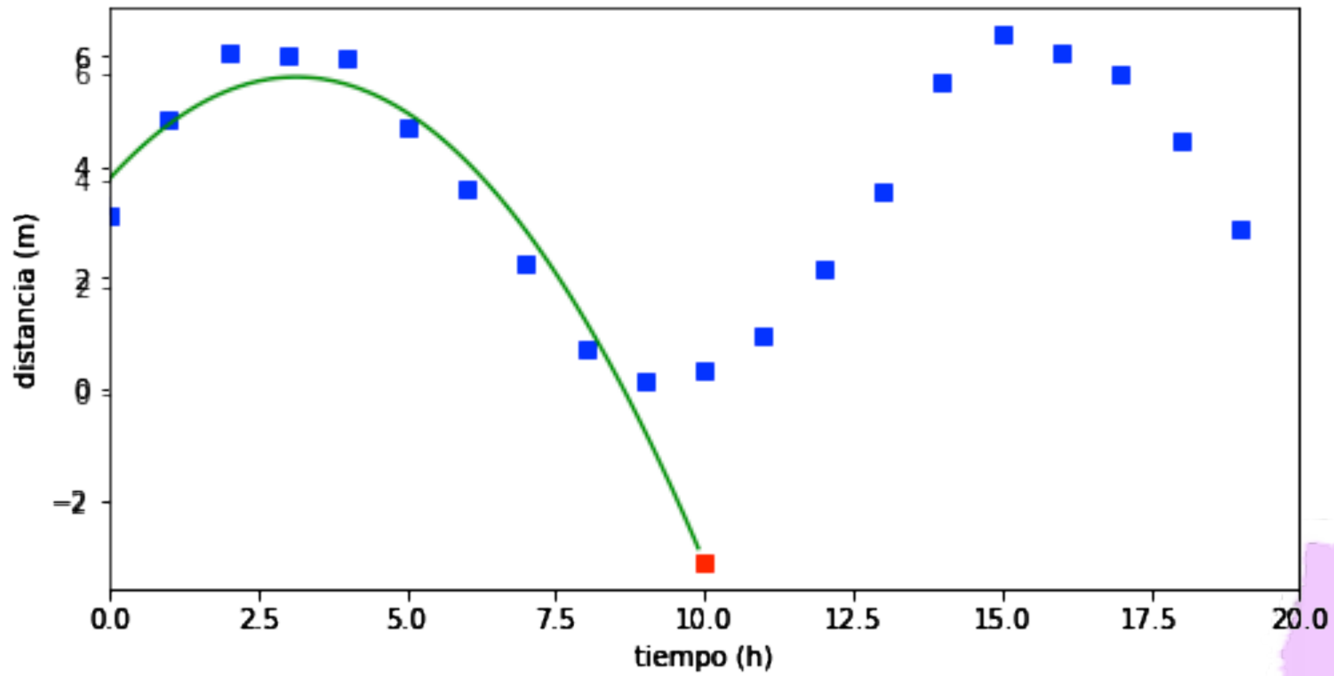


Tomamos más datos.

Modelizar.

Modelo

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos



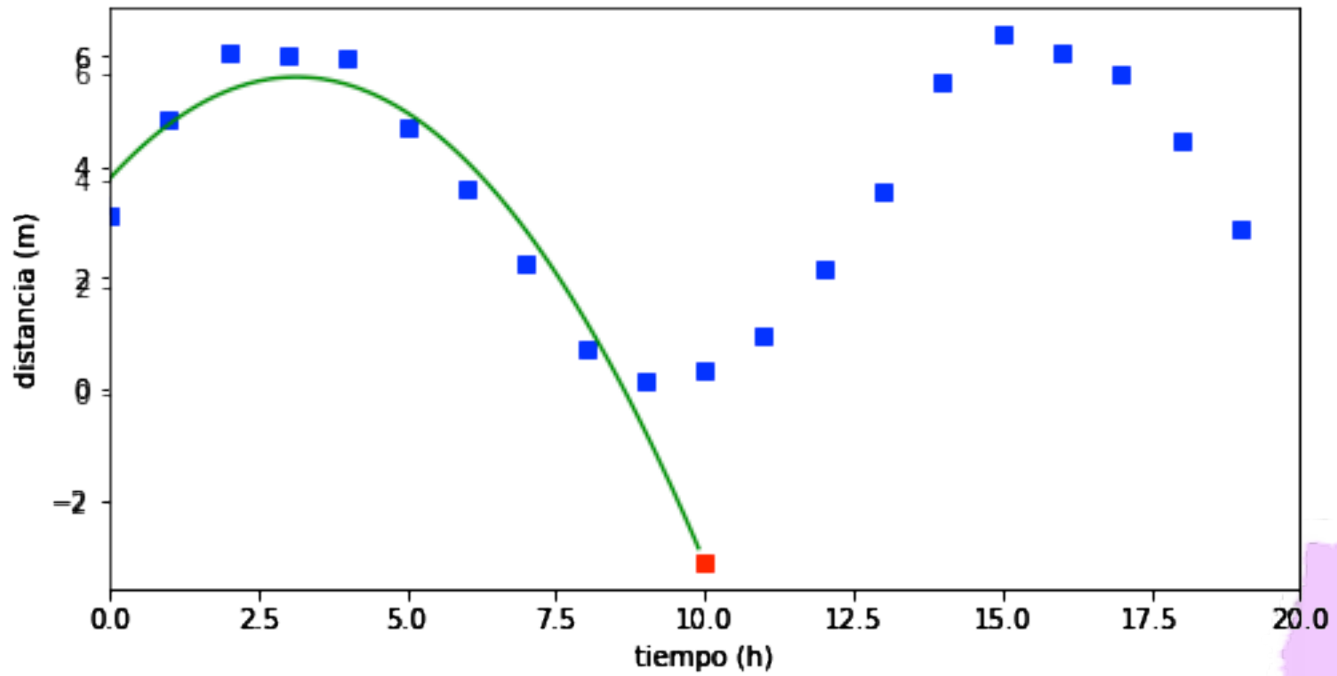
Tomamos más datos.

Definitivamente el modelo que hemos construido NO sirve

Modelizar.

Modelo

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos



Tomamos más datos.

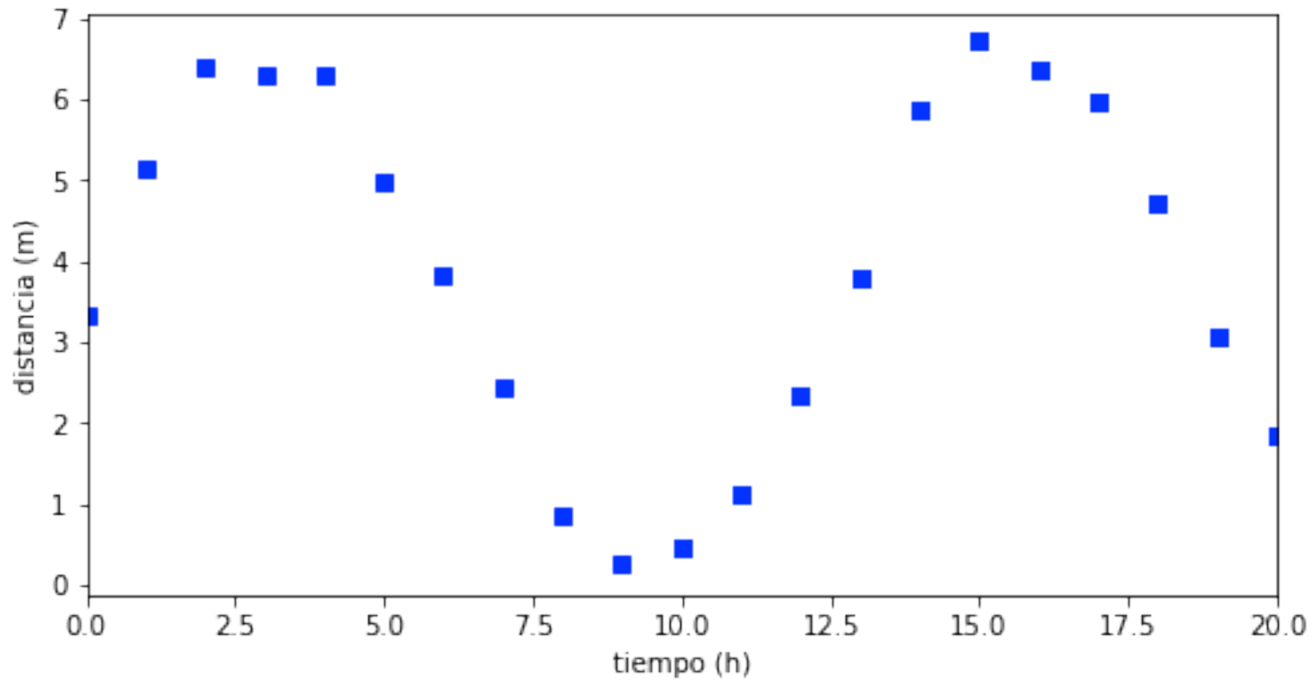
podríamos volver a buscar un polinomio con los nuevos datos, pero en general es solo **válido para el rango del ajuste**.

Definitivamente el modelo que hemos construido NO sirve

Modelizar.

Modelo

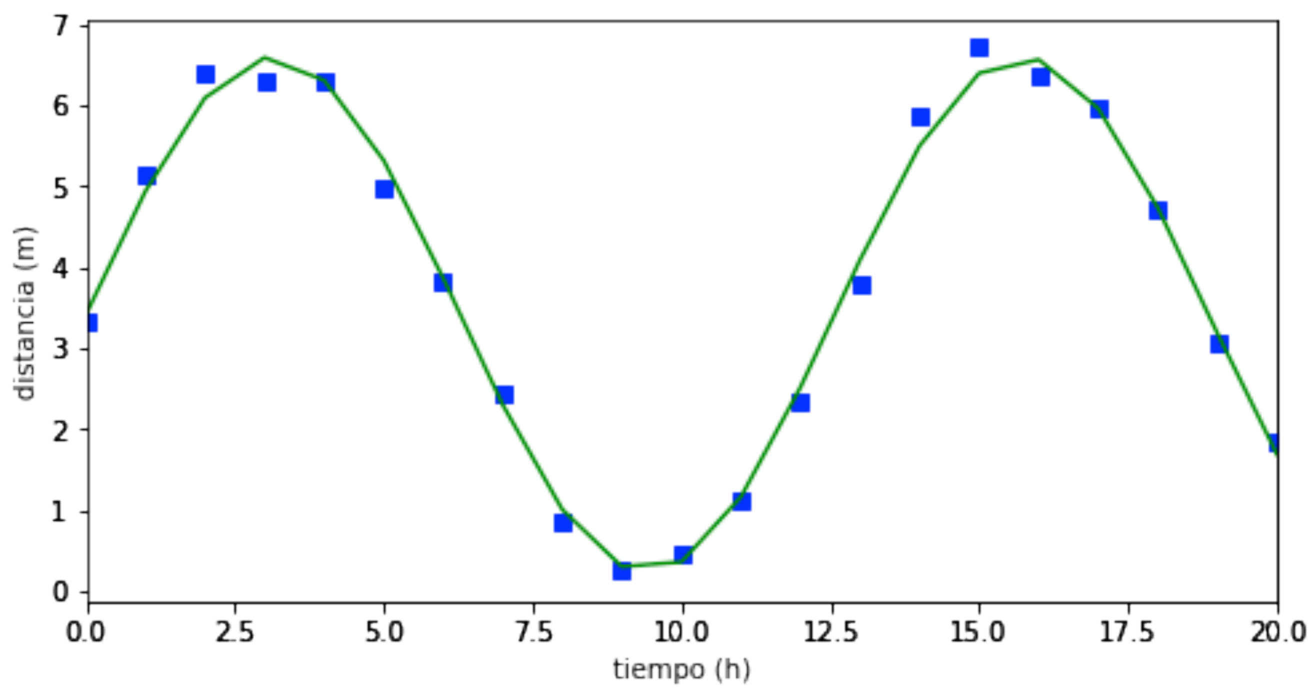
Probamos otra función: $$d = a\sin(bt) + c$$



Modelizar.

Modelo

Probamos otra función: $d = a\sin(bt) + c$



yo le he dicho la función que quiero probar y python nos da los parametros:

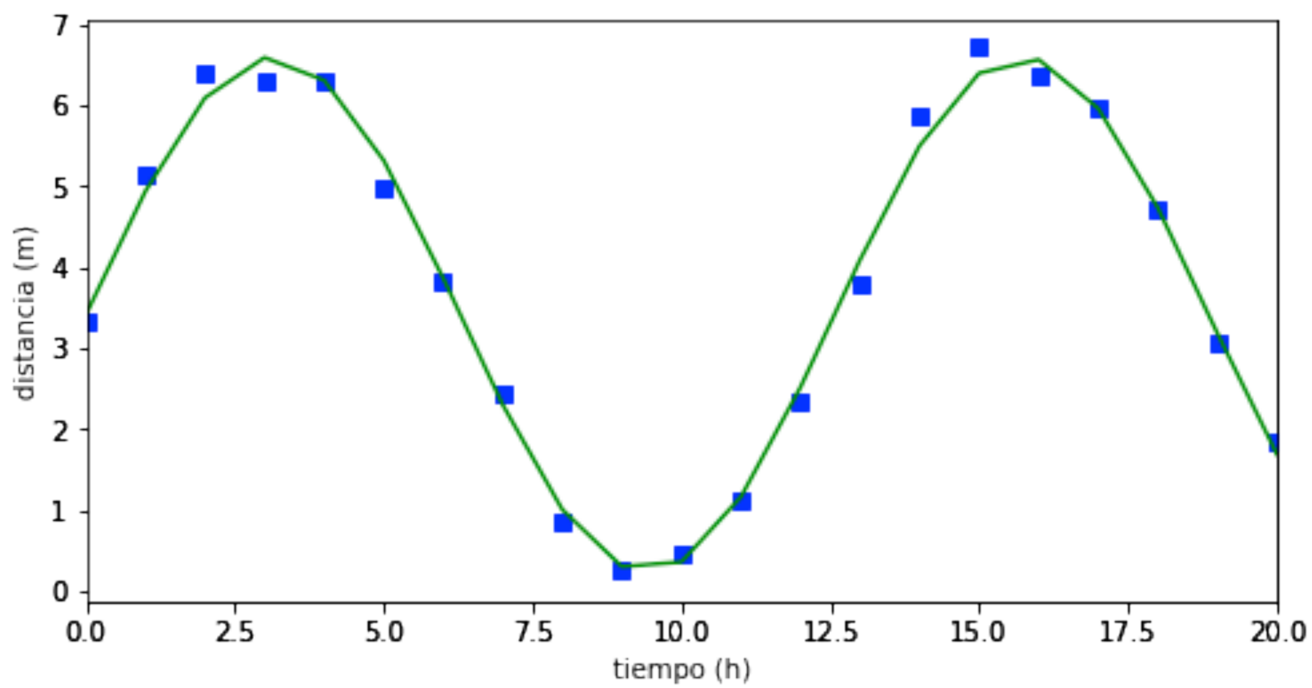
```
scipy.optimize.curve_fit(
func, t, d)
```

$$d = 3.1\sin(0.5t) + 3.4$$

Modelizar.

Modelo

Probamos otra función: $$d = a\sin(bt) + c$$



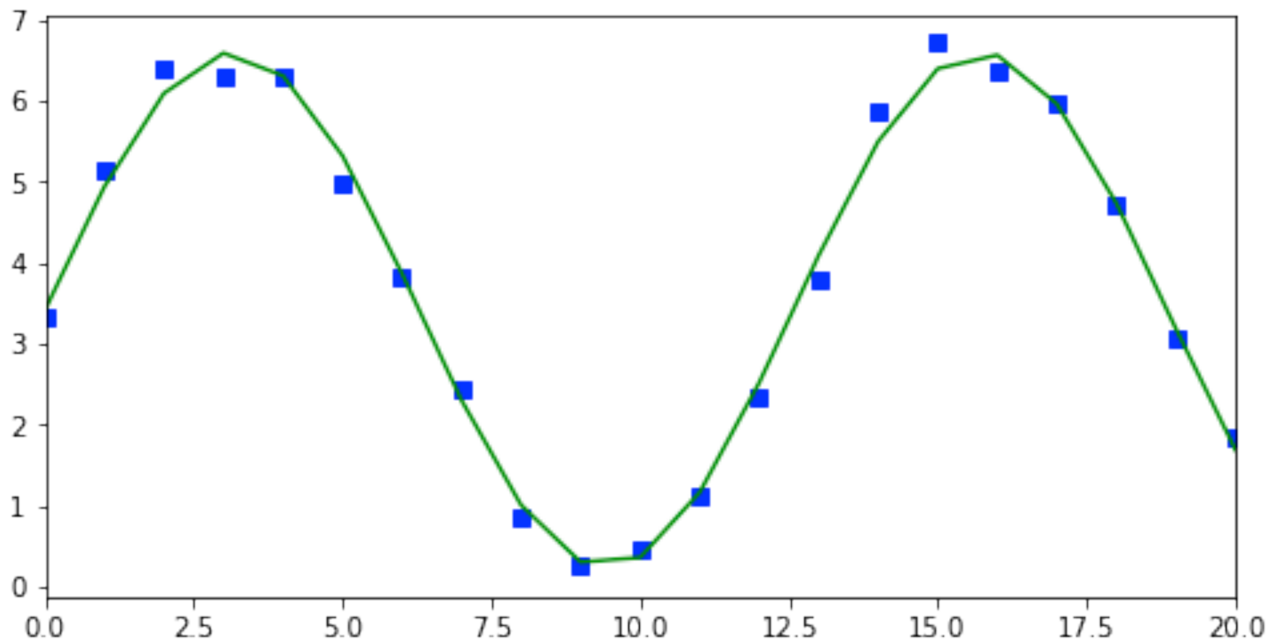
yo le he dicho la función que quiero probar y python nos da los parametros: `scipy.optimize.curve_fit(func, t, d)`

$$d = 3.1\sin(0.5t) + 3.4$$

¡Este modelo parece que ajusta mejor a los datos!

Modelizar.

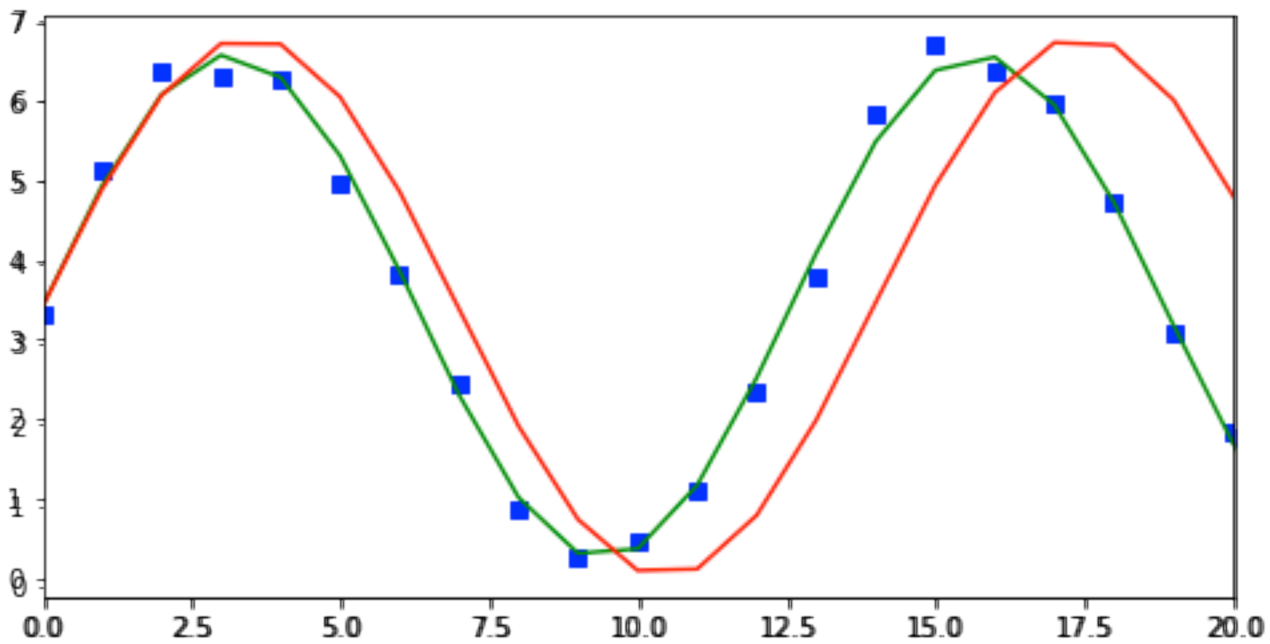
Modelo



$$d = 3.1\sin(0.5t) + 3.4$$

Modelizar.

Modelo



$$d = 3.1\sin(0.5t) + 3.4$$

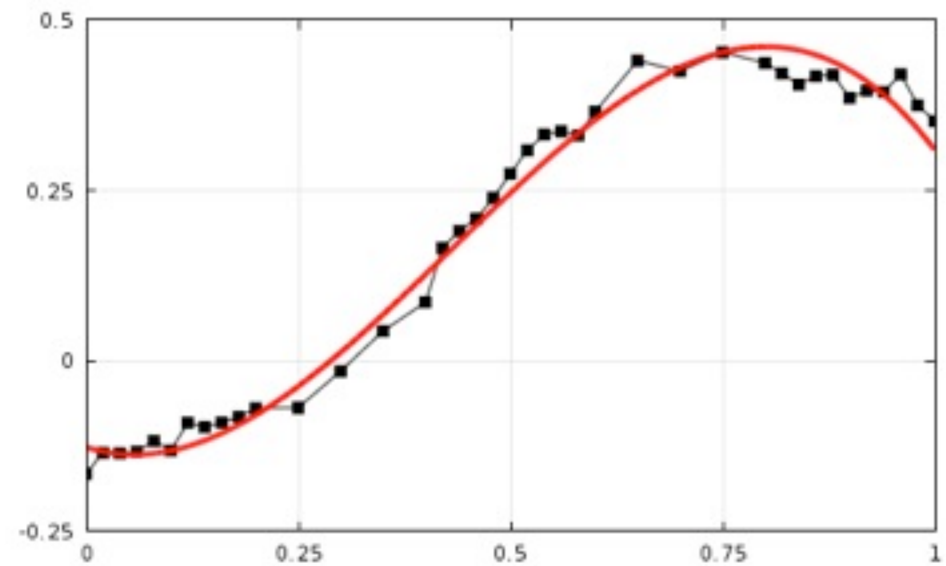
$$d = 3\sin(0.3t) + 3$$

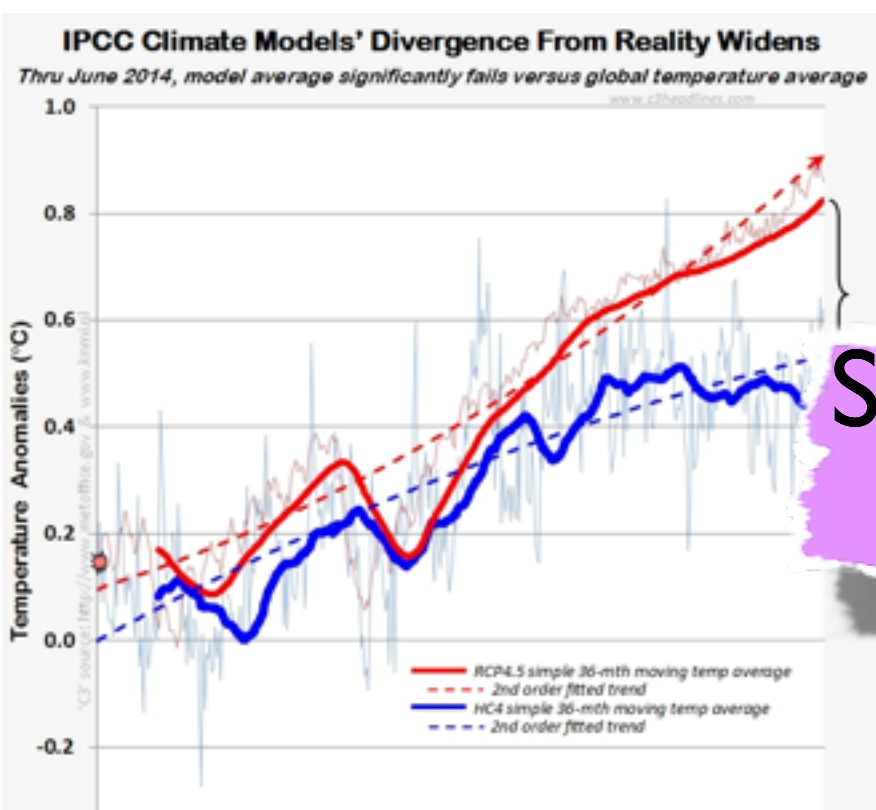
Modelo== Curva ⟶ función + parametros. **Cambiamos parámetros, cambiamos el modelo !!!**

Modelizar: Dados unos datos encontrar un modelo que nos permita hacer predicciones.

Modelo téorico: Tenemos una relación matemática que puede estar dado por leyes conocidas o por hipo... ica detrás.

Generalización





Modelo empírico: A partir de los datos intentamos **encontrar una relación** entre variables que nos ...

Sólo seguro en el **rango** de los datos.

Interpolación y extrapolación

¿Cómo hemos obtenido esa curva?

Modelizar

Ajuste por mínimos cuadrados

- Minimizamos la distancia promedio entre los datos y la función matemática en el mismo punto

$$E = \sum_i \frac{(d - f(t))^2}{N}$$

- donde

$$f(t) = a \sin(bt) + c$$

Modelizar.

Ajuste. Mínimos cuadrados

PROBLEMA DE OPTIMIZACIÓN:

Función objetivo:

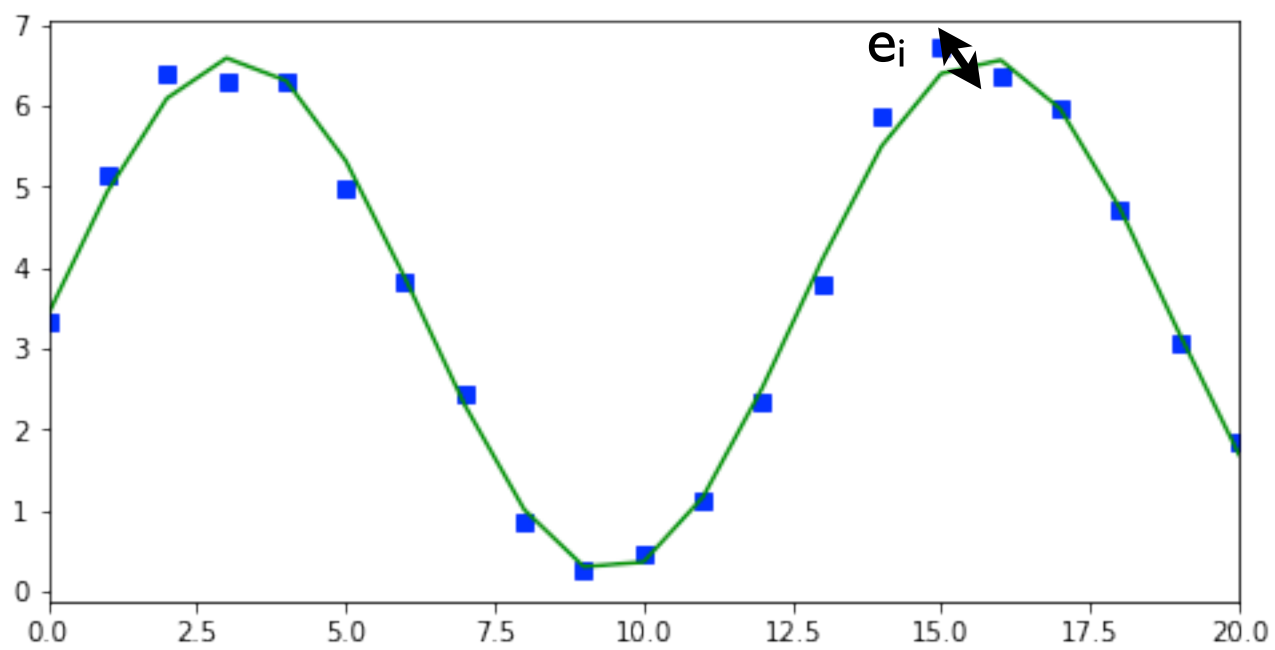
$$E = \sum_i \frac{(d_i - a\sin(bt_i) - c)^2}{N}$$

Variables:

a, b, c

* Buscamos la **curva más cercana a los datos**. La que minimiza la distancia promedio con los datos.

$$E = \sum_i e_i^2$$

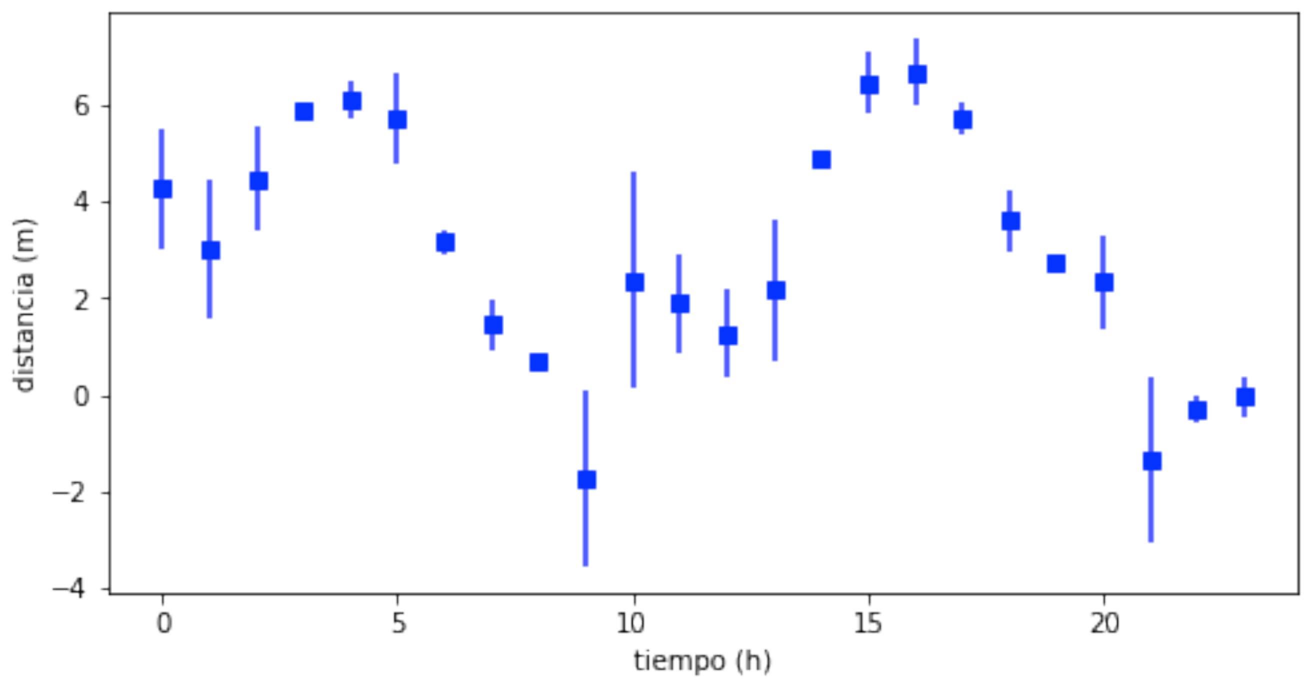


* `optimize.curve_fit()` y `numpy.polyfit()`

les das f(t) y los datos y calculan E y buscan el mínimo

Modelizar.

Ajuste. Mínimos cuadrados pesados

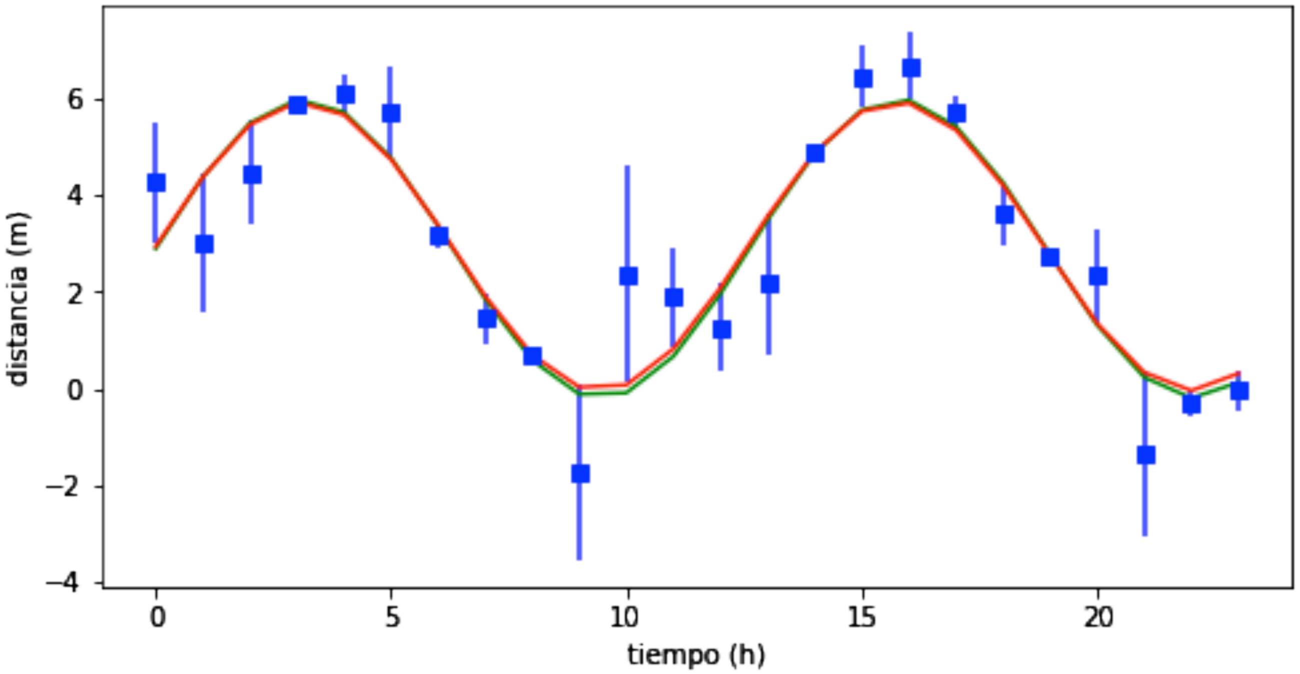


* Weighted least squares.

* Los datos suelen ser ruidosos, (errores al medir, ...)

Modelizar.

Ajuste. Mínimos cuadrados pesados



* Weighted least squares.

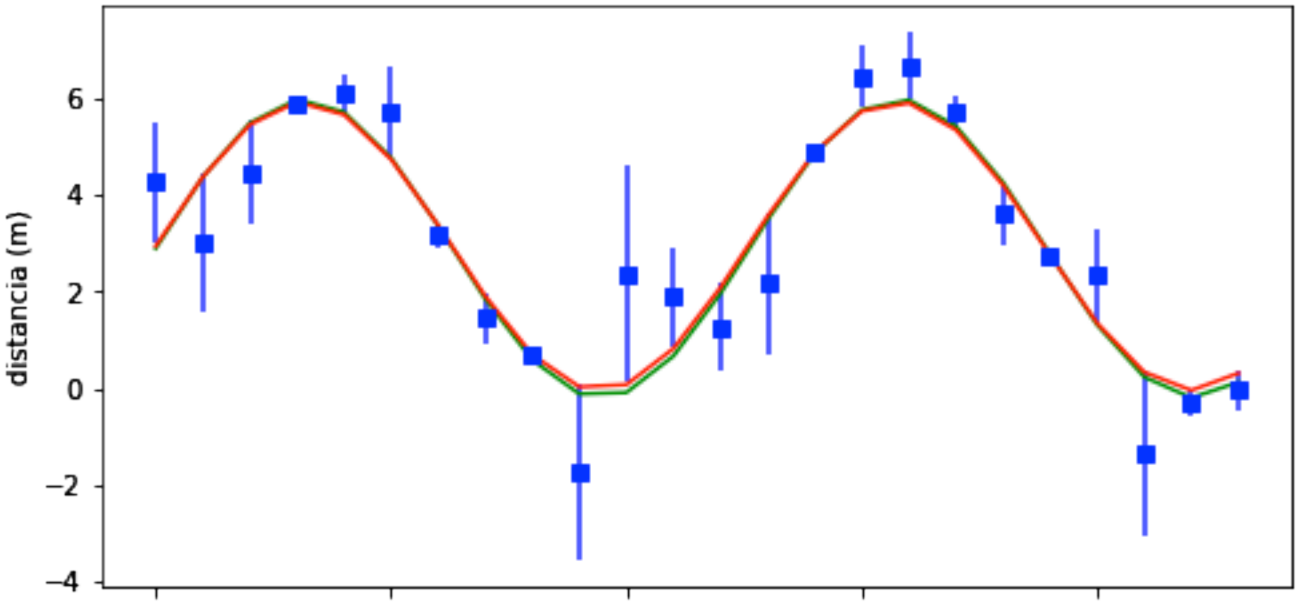
* Los datos suelen ser ruidosos, (errores al medir, ...)

Weighted least squares method

$$\chi^2 = \sum_i^N \frac{(y_i - f(x_i))^2}{\sigma^2}$$

Modelizar.

Ajuste. Mínimos cuadrados pesados



* Weighted least squares.

* Los datos suelen ser ruidosos, (errores al

```
In [376]: print 'results     ',np.round(popt,2)
          print 'results Err',np.round(poptErr,2)
          print 'True val    ',[3.0,0.5,3.0]

results      [3.09 0.5   2.89]
results Err  [2.99 0.5   2.94]
True val     [3.0, 0.5, 3.0]
```

```
Data2=np.loadtxt('Data2_modelling_noisy.txt')
x=Data2[:,0];y=Data2[:,1];yerr=Data2[:,2]
plt.figure(0,figsize=(8,4))


popt, pcov = opt.curve_fit(func, x, y,p0=(0.5,0.5,0.5))
poptErr, pcov = opt.curve_fit(func, x, y,p0=(0.5,0.5,0.5),sigma=yerr)
# esto nos devolverá los parametros a,b y c

plt.errorbar(x, y, yerr=yerr,  marker='s',color='b',label='data',linestyle='')
#popt, pcov = opt.curve_fit(func, x, y,method='lm')
a=popt[0];b=popt[1];c=popt[2]
#model=func(x,3,0.5,3)
model=func(x,a,b,c)
modelErr=func(x,*poptErr)
```

$$= \sum_{i}^{N} \frac{(y_i - f(x_i))^2}{\sigma^2}$$

Modelizar

Como sabemos si un ajuste está bien hecho?
Goodness of fit. Hay varios tests que nos pueden decir cuan bueno es un ajuste. Dependiendo del método o estadísitco que estemos usando. Ejemplo:

R-squared

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

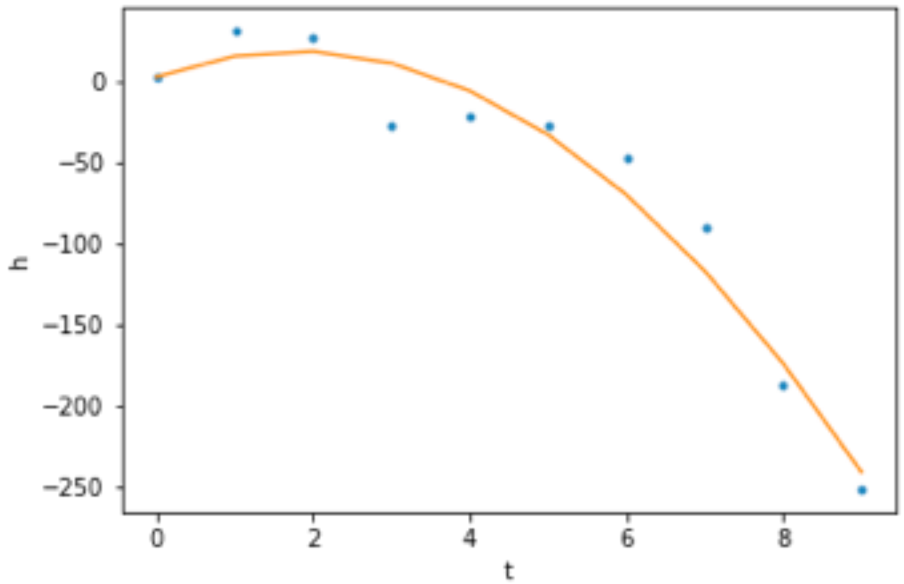
R-squared puede ir entre 0 y 1 . (Los errores en la estimación deberían ser menores a la varianza de la muestra)

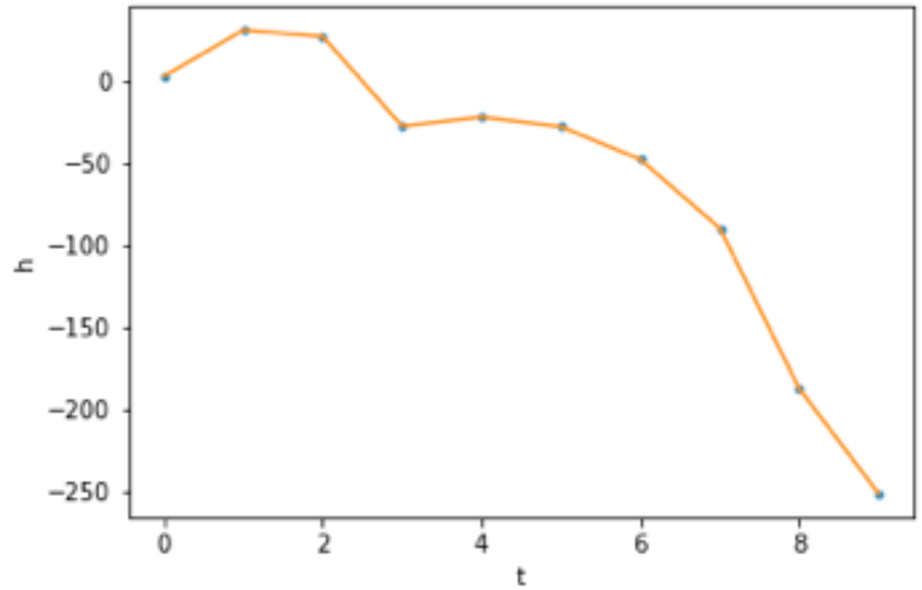
Modelizar

Sobreajuste

PERO. Que un ajuste esté bien hecho no implica que el modelo sea el correcto. El modelo tiene que estar bien motivado!!

OVERFITTING!!





Si el modelo no es del todo conocido es necesario tener muchos datos para hacer el ajuste con una muestra y luego probarlo en la otra. Si tenemos muy buen fit en una y en la otra falla es que hemos sobreajustado (más en *machine learning*)

El ajuste de la derecha nos dará mejor R-squared y mejor chi-squared

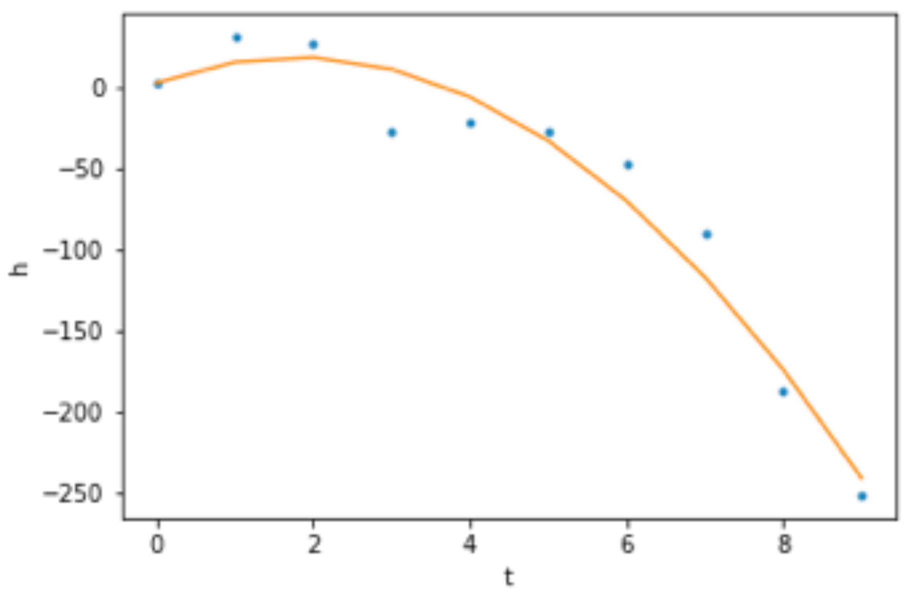
Modelizar

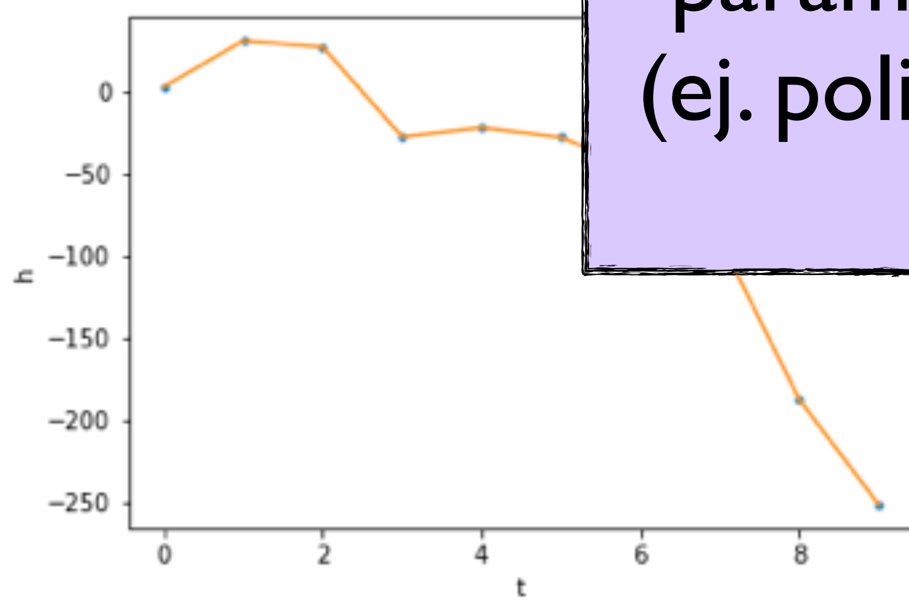
Sobreajuste

PERO. Que un ajuste esté bien hecho no implica que el modelo sea el correcto. El modelo tiene que estar bien motivado!!

Modelo con menos parámetros preferido (ej. polinomio de grado inferior)

OVERFITTING!!





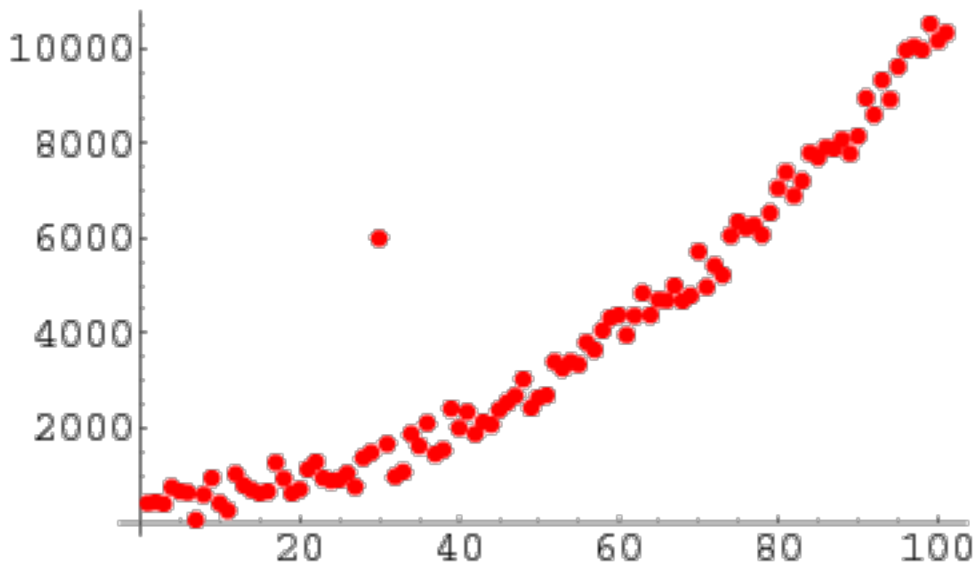
Si el modelo no es del todo conocido es necesario tener muchos datos para hacer el ajuste con una muestra y luego probarlo en la otra. Si tenemos muy buen fit en una y en la otra falla es que hemos sobreajustado (más en *machine learning*)

El ajuste de la derecha nos dará mejor R-squared y mejor chi-squared

Modelizar

Outliers

Vigilar con los Outliers



¡Mirar los datos antes de nada!

Valores muy extremos pueden pesar mucho en un ajuste. Hay que mirar los datos primero y descartar valores extremos, o al menos mirarlos.

Ajustes con Python

(vemos ejemplo con notebook)

interp1d(x,y)	interpolación
numpy.polyfit(x,y,d)	ajuste con polinomio de grado d mediante minimos cuadrados
curve_fit(func,x,y)	ajuste con función (func) mediante minimos cuadrados
minimize(ObjFunc,..)	ajuste mediante cualquier función