Complements

July 7, 2019

AI problems

- We will finish by discussing some issues regarding AI problems :
  - Optimization and gradient descent
  - Multivariate analysis
  - Overfitting

Overview

Example situation

- Let's say we have some data $(x_1, \ldots, x_n)$, for instance representing the heights of n a trees from some species.

Figure: Amandier

- For instance $x_1 = 5m$, $x_{10} = 10.5m$, $\ldots$.

Example situation

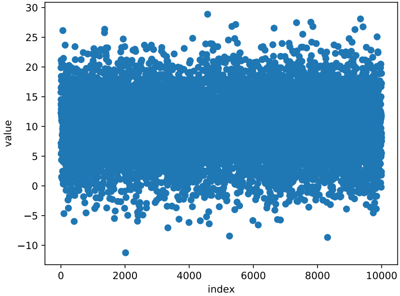
- Let's say we have some data $(x_1, \dots, x_n)$, for instance representing the heights of n a trees from some species.

Figure: Amandier

- For instance $x_1 = 5m$, $x_{10} = 10.5m$, $\dots$.
- We would like to analyze the distribution of the random variable x.

Tree heights

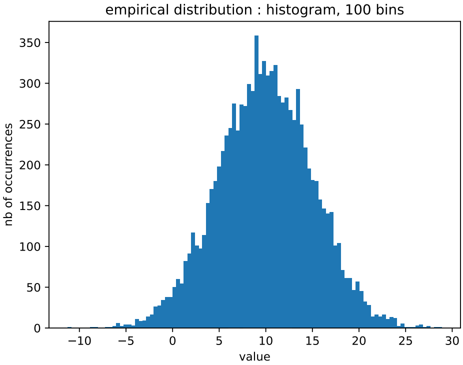
- Let's look at the data



- We have 10000 samples.

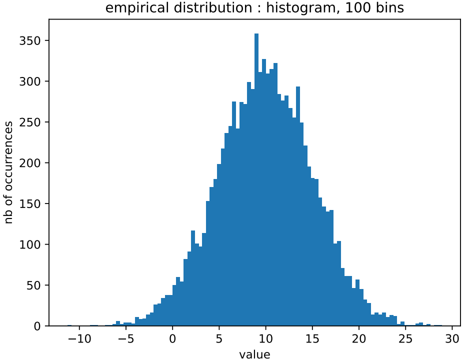
Tree heights

- A histogram is better :



Tree heights

- A histogram is better :



- What distribution would you use to **fit the data ?**

Fitting the data

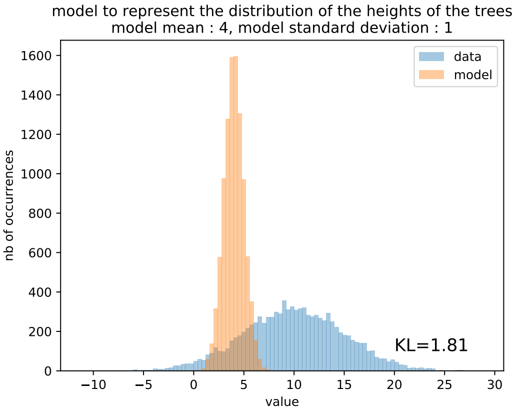


Figure: Bad model

Fitting the data

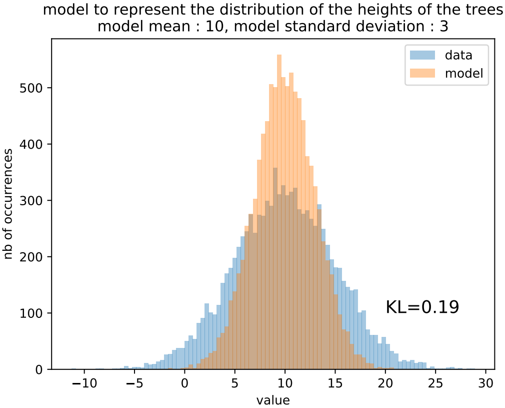


Figure: Better model

Fitting the data

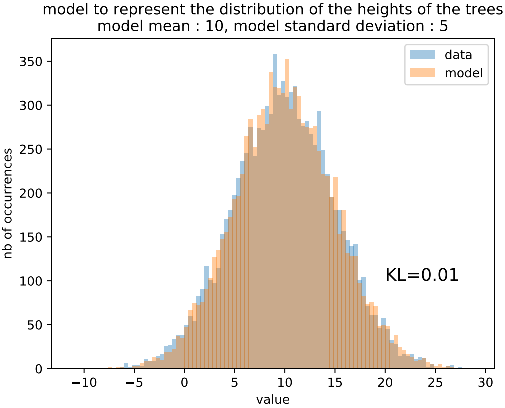


Figure: Good model

Summary

- In this situation, it was easy to know if our model was good or not by just sampling from our model and then looking at the histograms.
- In one dimension, we have **visual feedack**.

Real life problems.

- In this situation, it was easy to know if our model was good or not by just sampling from our model and then looking at the histograms.
- In one dimension, we have **visual feedack**.
- However, in most cases, it won't be that straightforward to fit a distribution.

Real life problems.

- In this situation, it was easy to know if our model was good or not by just sampling from our model and then looking at the histograms.
- In one dimension, we have **visual feedack**.
- However, in most cases, it won't be that straightforward to fit a distribution.
  - the **dimensionality** can be higher.
  - what distribution do we want to use ?
  - even if we know the right shape of the distribution, how to choose the parameters ?

Measuring the quality of a model

- We need tools to **quantify** the quality of our model.

Kullbach-Leibler

- The Kullbach-Leibler divergence is such a tool.
- It comes from information theory.

Kullbach-Leibler Divergence

- $$\mathcal{D}[p||q] = \mathbb{E}_{\sim p}[\log(\frac{p}{q})] \tag{1}$$
- For discrete variables
$$\mathcal{D}[p||q] = \sum_i p(i) \log \frac{p(i)}{q(i)} \tag{2}$$
- for continuous variables
$$\mathcal{D}[p||q] = \int_X p(x) \log \frac{p(x)}{q(x)} dx \tag{3}$$

Fitting the data

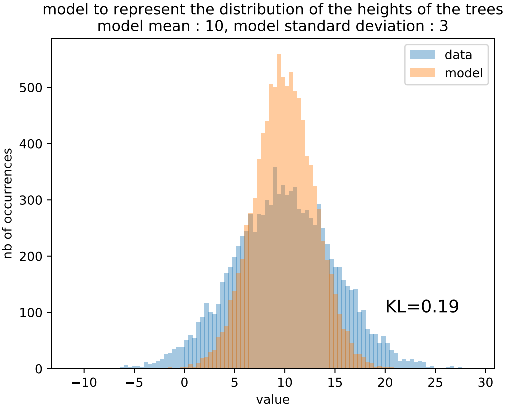


Figure: Better model : KL divergence of 0.19

Fitting the data

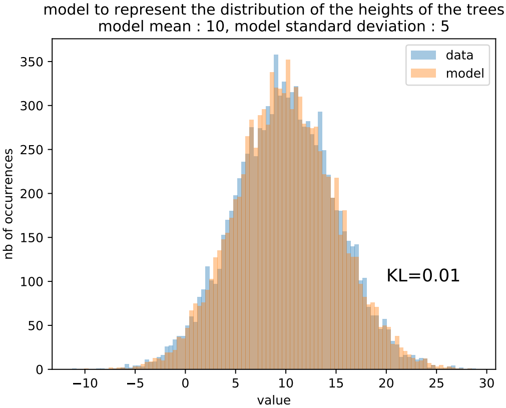


Figure: Good model : KL divergence of 0.01

Maximum Likelihood

The **Maximum Likelihood** method is a famous method used in Machine Learning.
Say you have a dataset $(x_1, ..., x_n)$.

Maximum Likelihood

The **Maximum Likelihood** method is one example method used in Machine Learning.
Say you have a dataset $(x_1, ..., x_n)$.
You first need to choose a **model** (which is the distribution) of your dataset, p.

Maximum Likelihood

The **Maximum Likelihood** method is the one used in Machine Learning.
Say you have a dataset $(x_1, ..., x_n)$.
You first need to choose a **model** (which is the distribution) of your dataset, p.
Then, you must optimize the **parameters of this model**, noted θ.

Maximum Likelihood

The Likelihood of your model is

$$L(\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{4}$$

Maximum Likelihood

The Likelihood of your model is

$$L(\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{5}$$

This is the function that you want to **maximise**.

Maximum Likelihood

Most of the time it's written this way : "minimise $-logL(\theta)$"
Why ?

Maximum Likelihood

Most of the time it's written this way : "minimise $-logL(\theta)$"
Because the log **transforms the product into a sum**, which is easier to **derivate**.

Maximum Likelihood

$$-logL(\theta) = -\sum_{i=1}^{n} \log(p(x_i|\theta)) \tag{6}$$

Max Likelihood

So how can we minimise $-logL(\theta)$? In the case of very large datasets, and large numbers of parameters (tens, hundredths, more), most of the time an **analytic solution** is not available. So people use **gradient descent**.

The gradient descent

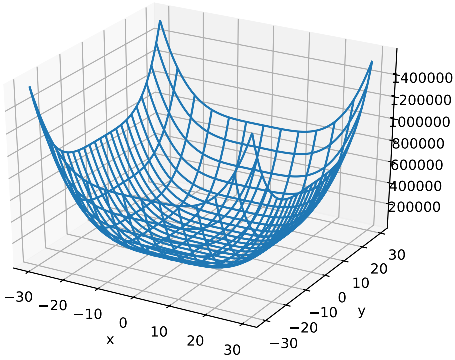
We want x to **minimise** f. We perform, until some criteria is satisfied :

$$x \leftarrow x - \alpha \nabla_f(x) \tag{7}$$

Use the file "gradient_algo.py" and implement the gradient algorithm on a simple example.
I inserted two errors in the code.

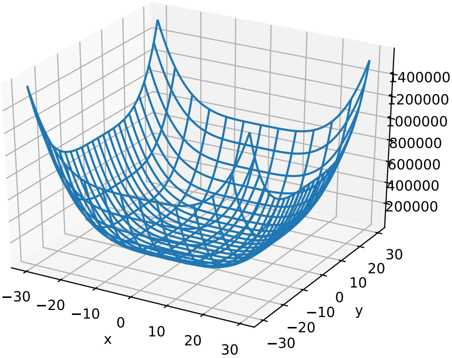
The gradient descent

$$x \leftarrow x - \alpha \nabla_f(x) \tag{8}$$



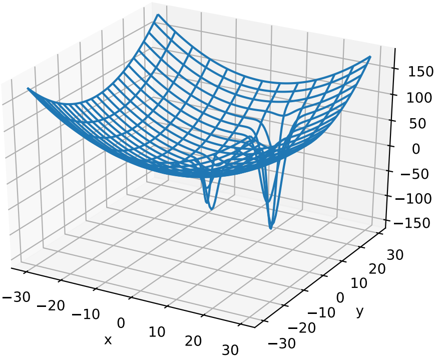
The gradient descent

Experiment with it, try to change all the parameters and to break it again. Is it stable ?



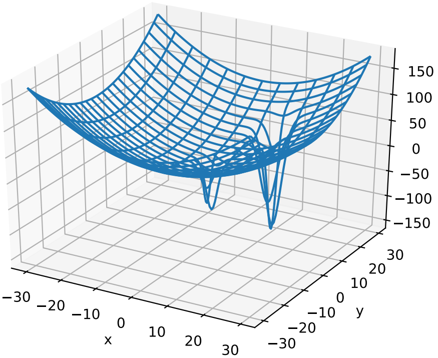
The gradient descent

Let's try with a different function



The gradient descent

Let's try with a different function that has **several local minima**.



Multidimensional vectors

We can consider data that live in higher dimensional spaces than 2.

Multidimensional vectors

We can consider data that live in higher dimensional spaces than 2. Examples ?

Multidimensional vectors

We can consider data that live in higher dimensional spaces than 2. Examples ?

- images
- sensor that receives **multimodal information**

Correlation

Sometimes the components of a multidimensonial vector $(x_1, \ldots, x_n)$ are not independent.

Correlation

Sometimes the components of a multidimensonial vector $(x_1, \ldots, x_n)$ are not independent.
To study this, we can use the **covariance** of the two components, or the **correlation** which is actually clearer.

Correlation, expected value

- Let us introduce these two important quantities (backboard).

Example

Look at the data contained in **mysterious_distro_3.csv**
They contain a random variable with 5 dimensions. Some of these dimensions are correlated.
Think for instance to physics : temperature and pressure, etc. If you have measurements of temperature and pressure, the two would probably be **correlated**.

Covariance matrix

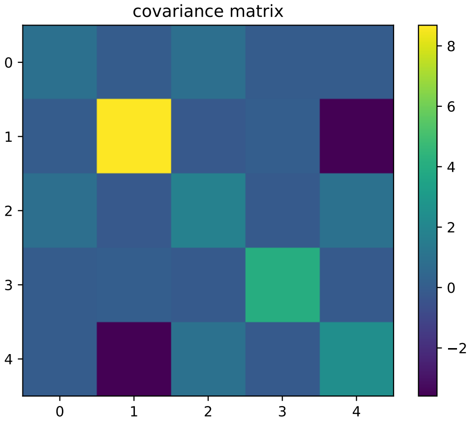


Figure: Covariance matrix for the distribution

Correlation matrix

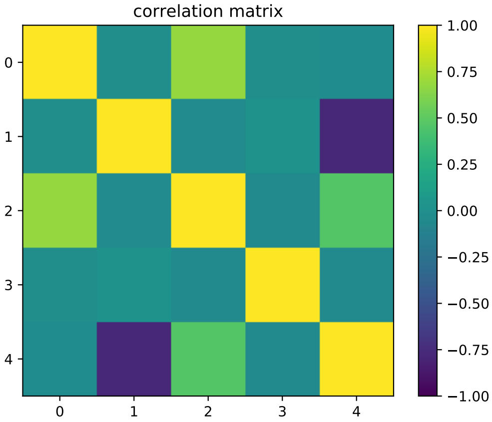


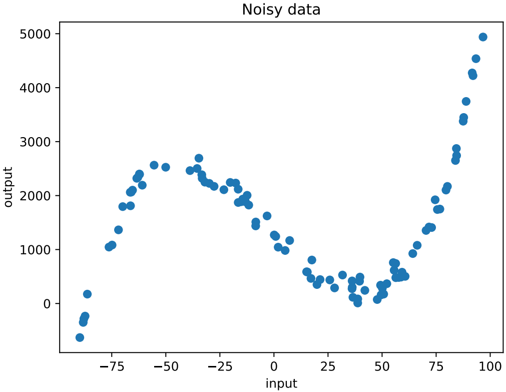
Figure: Correlation matrix for the distribution

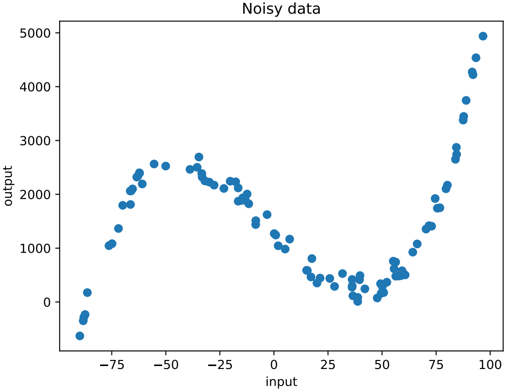
Generation of the data

- Generation of the data

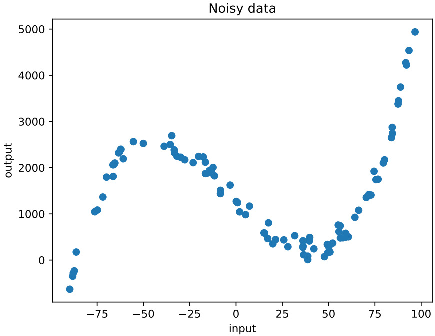
Overfitting

We will learn a **model** of the following data, in a **supervised learning** context.





Our **model** should allow us to predict the **output** for new **inputs**.
For instance what should be predicted fot an input of -48 ?



We need to choose:

- A **class** of model.
- A relevant **complexity** once the class is chosen.

Overfitting

- What could be the drawbacks of using a very simple model (very few parameters) ?

Overfitting

- What could be the drawbacks of using a very simple model (very few parameters)
    - Weak expressive power

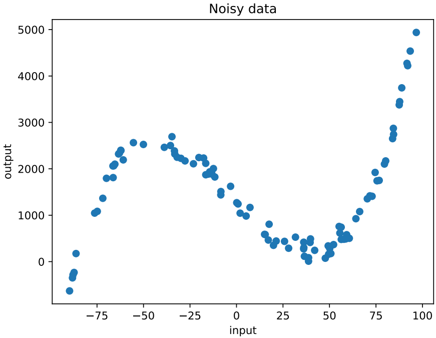
Overfitting

- What could be the drawbacks of using a very simple model (very few parameters)
  - Weak expressive power
- What could be the drawbacks of having a very complex model (that contains a very large number of parameters, e.g. millions as in a very deep neural network) ie a very high expressive power ?

Overfitting

- What could be the drawbacks of using a very simple model (very few parameters)
  - Weak expressive power
- What could be the drawbacks of having a very complex model (that contains a very large number of parameters, e.g. millions as in a very deep neural network) ie a very high expressive power ?
  - Harder to optimize
  - Harder to interpret
  - Can **overfit**

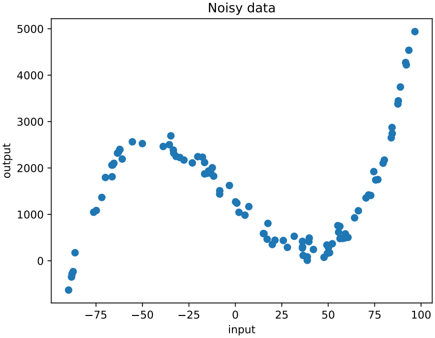
Exercise 3 : fitting



We want to perform supervised learning in order to be able to predict the output y for a new sample x.

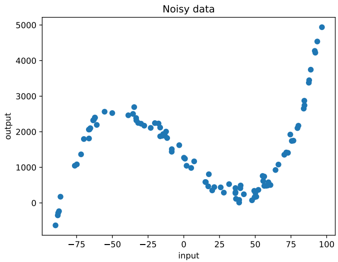
Exercise 3 : fitting

- We want to perform supervised learning in order to be able to predict the output y for a new sample x.



- To illustrate the problem of overfitting, we will use **polynoms** as models.

Exercise 3 : fitting



We will divide the dataset into two subsets :

- a **training set** : used to learn the most relevant polynom once the degree is chosen
- a **test set** : used to evaluate overfitting

Exercise 3 : fitting

- **cd overfitting**. Use the dataset contained in **linear_noisy_data.csv**, load it from **fit_data.py** in order to assess the impact of the **degree** of the polynom on overfitting.
- You need to edit the loop at the end of the file.

Exercise 3 : fitting

- The higher the degree of the polynom, the more parameters it has and the better it can fit the training points :

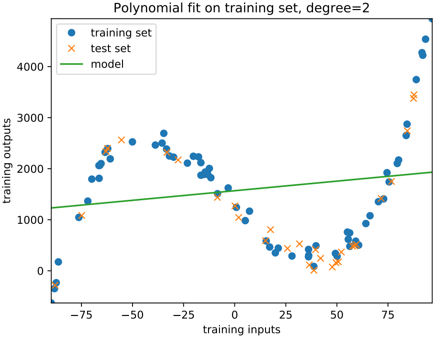


Figure: degree 2

Exercise 3 : fitting

- The higher the degree of the polynom, the more parameters it has and the better it can fit the training points :

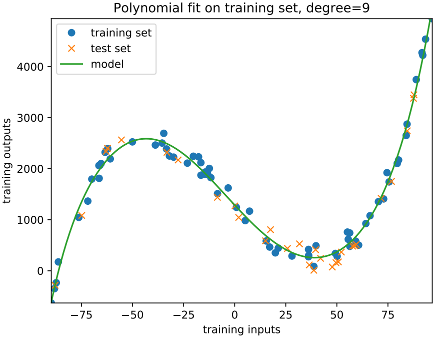


Figure: degree 9

Exercise 3 : fitting

- The higher the degree of the polynom, the more parameters it has and the better it can fit the training points :

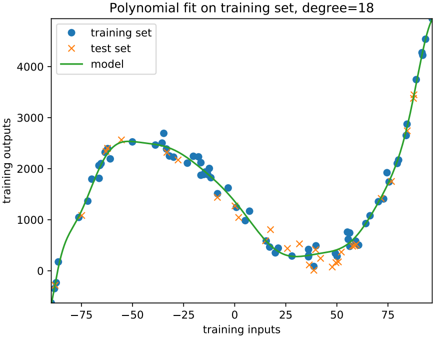


Figure: degree 19

Exercise 3 : fitting

- However, the error on the test set increases and the model looses **signification**

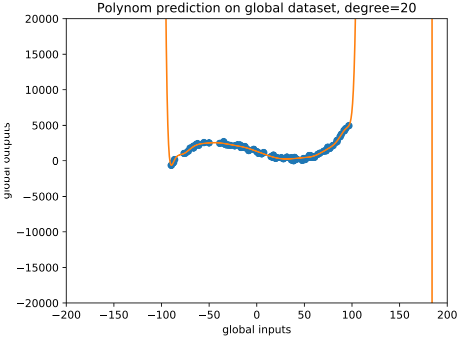


Figure: Useless solution

Exercise 3 : fitting

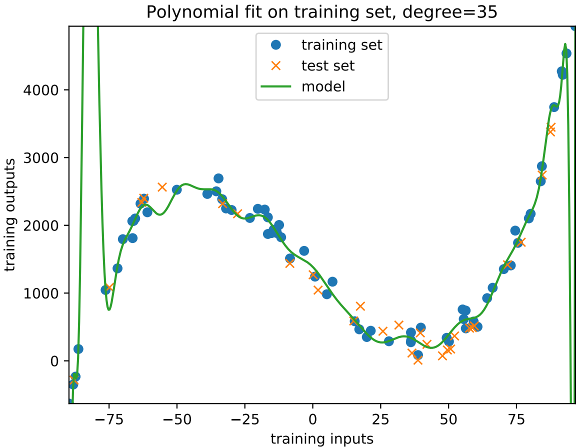


Figure: When the degree is too high.

Exercise 3 : fitting

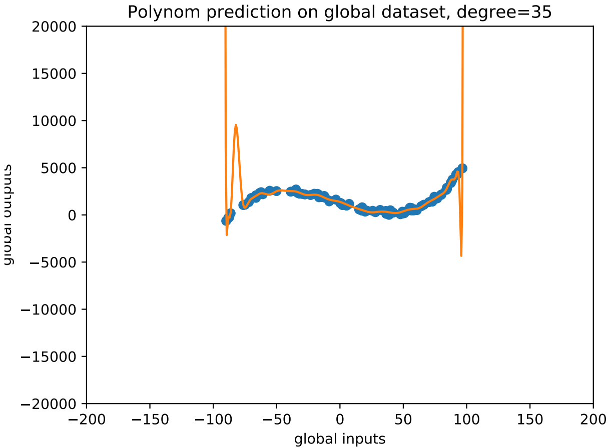
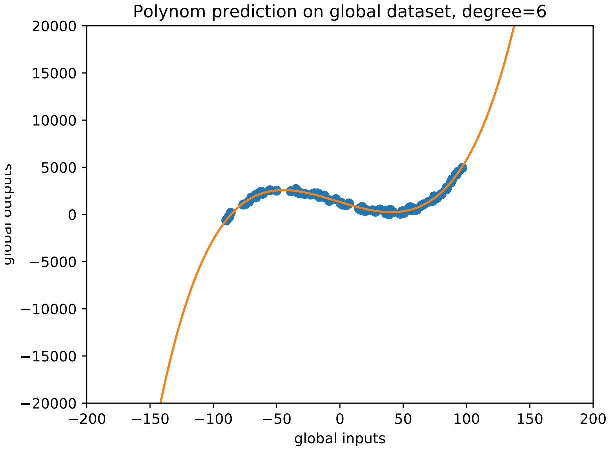


Figure: When the degree is too high.

Exercise 3 : fitting

In that situation, what degree should we use ?



Trying to prevent overfit

- The problem of overfitting is linked to that of **generalisation** : to what extent are we allowed to extrapolate the knowledge obtained on the training samples to new samples ?
- To improve generalisation, one can use :
  - a **validation set**
  - **regularization**

Regularization methods

- Penalize the magnitude of the weight in a neural network
- Remove neurons in a neural network (pruning)
- use smooth functions (continuous)