Intro to ML and Big Data

CS 4731 — Dr. Mihail
Department of Computer Science
Valdosta State University

September 4, 2019

Do not attempt a Mihail homework the night it's due.

1 Problem 1

Write **one** script that solves all the linear systems below, in order.

1 $$\begin{cases} 5x_1 - 3x_2 + 2x_3 = 3 \\ 2x_1 + 4x_2 - x_3 = 7 \\ x_1 - 11x_2 + 4x_3 = 3 \end{cases}$$

2 $$\begin{cases} x_1 + 4x_3 = 13 \\ 4x_1 - 2x_2 + x_3 = 7 \\ 2x_1 - 2x_2 - 7x_3 = -19 \end{cases}$$

3 $$\begin{cases} -2x_1 + x_2 = -3 \\ x_1 + x_2 = 3 \end{cases}$$

4 $$\begin{cases} 10x_1 - 7x_2 = 7 \\ -3x_1 + 2x_2 - 6x_3 = 4 \\ 5x_1 + x_2 + 5x_3 = -19 \end{cases}$$

5 $$\begin{cases} x_1 + 4x_2 - x_3 + x_4 = 2 \\ 2x_1 + 7x_2 + x_3 - 2x_4 = 16 \\ x_1 + 4x_2 - x_3 + 2x_4 = -15 \\ 3x_1 - 10x_2 - 2x_3 + 5x_4 = -15 \end{cases}$$

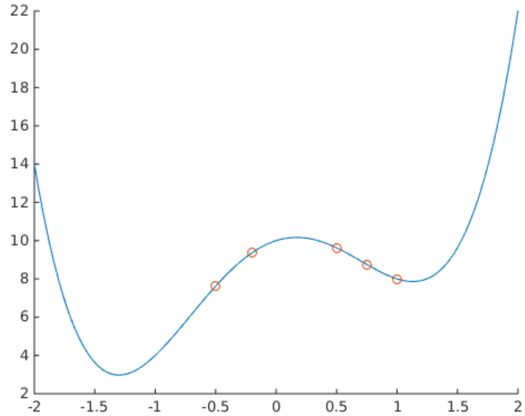


Figure 1: 4th degree polynomial

2 Problem 2 - curve fitting

You are given 5 data points sampled (without noise) from a fourth degree polynomial.

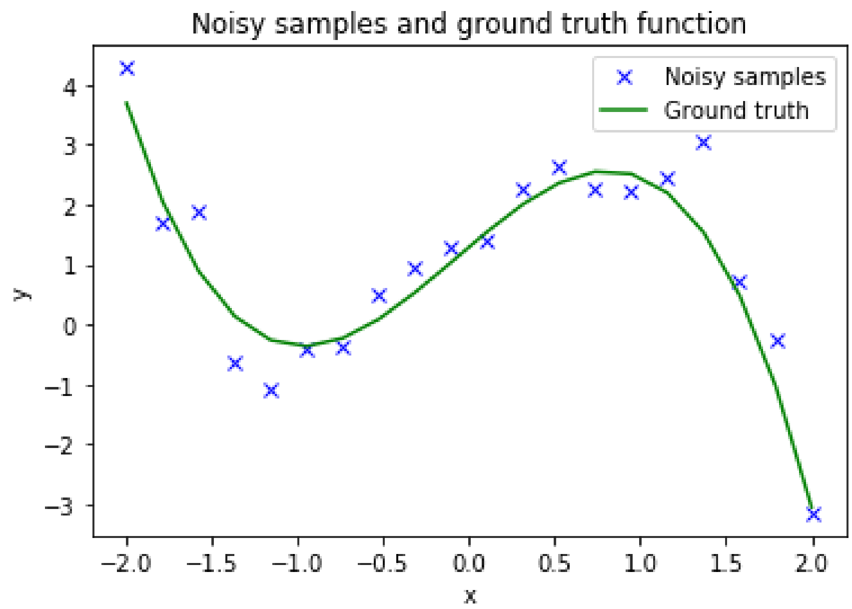
$$f(x) = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3 + a_4 x^4$$

This polynomial (and the data points) are shown in Figure 1.

Given the following data points:

x	$f(x)$
-0.5	7.625
-0.2	9.3632
0.5	9.625
0.75	8.7578
1	8

solve for the polynomial's coefficients (a_0, a_1, a_2, a_3, a_4). You have to formulate this curve fitting problem as the solution to a linear system of equations, with 5 equations and 5 unknowns. Each equation corresponds to one data point, and can be thought of as a constraint on the possible fourth degree polynomials. Write a Python script to generate the system and solve it using the method shown in class, matrix inverse. Set up the coefficients matrix A, the column bias vector b, then the solution to the system will be your coefficients, that you compute as follows: $x = A^{-1}b$. Write a script that produces the plot in Figure 1.



3 Problem 2 - curve fitting

In this problem, you are given some code that samples data points from a known cubic (generated at random):

```
import numpy as np
import matplotlib.pyplot as plt

# this defines the family of cubics
f = lambda w, x: w[0] + w[1]*x + w[2]*(x**2) + w[3]*(x**3)

# sample 20 equally spaced values between -2 and 2
dom = np.linspace(-2, 2, 20)

# generate noisy data points, as values of a particular cubic + noise
val = f(w, dom) + np.random.randn(20)/2

# generate ground truth samples for visualization only
val_truth = f(w, dom)

# plot
fig, ax = plt.subplots(1, 1)
ax.plot(dom, val, 'bx')
ax.plot(dom, val_truth, '-g')
ax.legend(('Noisy samples', 'Ground truth'))
ax.set_xlabel('x')
ax.set_ylabel('y')
```

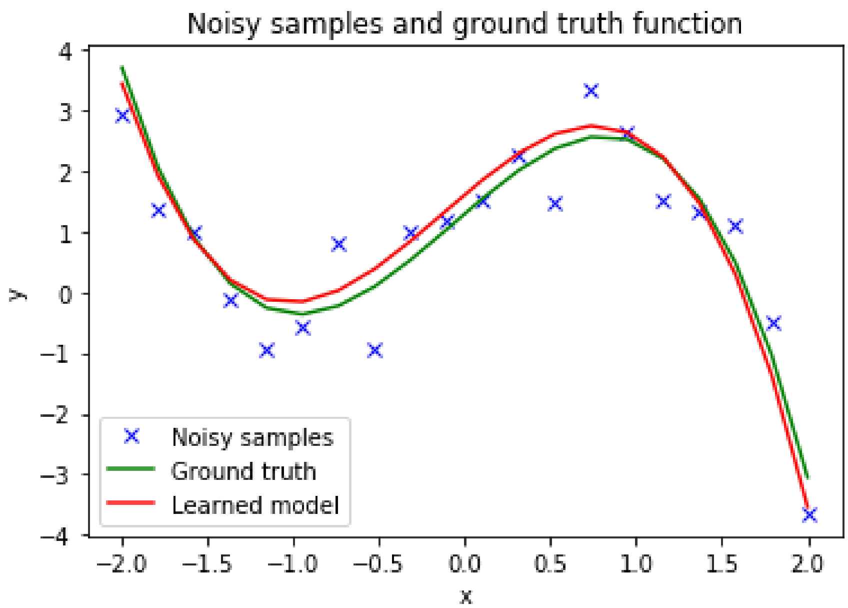


Figure 2: Final output

```
ax.set_title('Noisy samples and ground truth function')
```

Using a similar technnique as in Problem 2, formulate an **overdetermined** system of linear equations (linear with respect to model coefficients). This system will have more equations than unknowns, hence an exact solution **does not exist**. However, an approximate solution exists and can be found by least squares. You will use a numpy implementation of such a function, from the linear algebra package, called: *lstsq*. A complete reference to this function can be found here: `https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.lstsq.html`. Your script will produce an output similar to Figure 2.

Due dates

Problems due before midnight, Thursday September 12th. Submit via electronic assignment submission web page as assignment 1.

Useful links

- Systems of equations with the same number of equations and unknowns: `https://en.wikipedia.org/wiki/System_of_linear_equations`
- Underdetermined systems of equations: `https://en.wikipedia.org/wiki/Overdetermined_system`
- Least squares approximation: `https://en.wikipedia.org/wiki/Least_squares`

- Tiling vectors and matrices: `http://lagrange.univ-lyon1.fr/docs/numpy/1.11.0/reference/generated/numpy.tile.html`