Function Interpolation and Polynomial Interpolation (I)

Monday, October 7, 2024 10:36 AM

Class 18: October 7, 2024

Recall: Last time, we wrapped up our section on algorithms for non-linear systems of equations / rootfinding problems. In our last session, we introduced one more algorithm: **Steepest Descent** (also known as **Gradient Descent**). We also introduced a core idea which should be very useful if you ever need to learn numerical methods for optimization: that given an optimization problem, we can find an equivalent root finding problem:

Optimization problem: **Find a local minima r of q(x), for q:R^n -> R smooth (C^2)**
Rootfinding problem: **Find a specific kind of root r of Grad_q(x) = 0 (one where q is local minima)**

And the relationship goes the other way: if you want to find the root r such that F(r)=0, then you can relate that to

Find a local minima of q(x) = (1/2) (F_1(x)^2 + F_2(x)^2 + … + F_n(x)^2) , where q(r) = 0. ; Grad_q(x) = J_F(x)^T F(x)

The Steepest Descent (SD) method is an algorithm designed to solve the local minima problem, but it can (as we saw in the exa mples) be used for rootfinding by defining q(x) from F(x) as shown above. The step in this method is of the form:

x_{k+1} = x_k - alpha_k Grad_q(x_k)

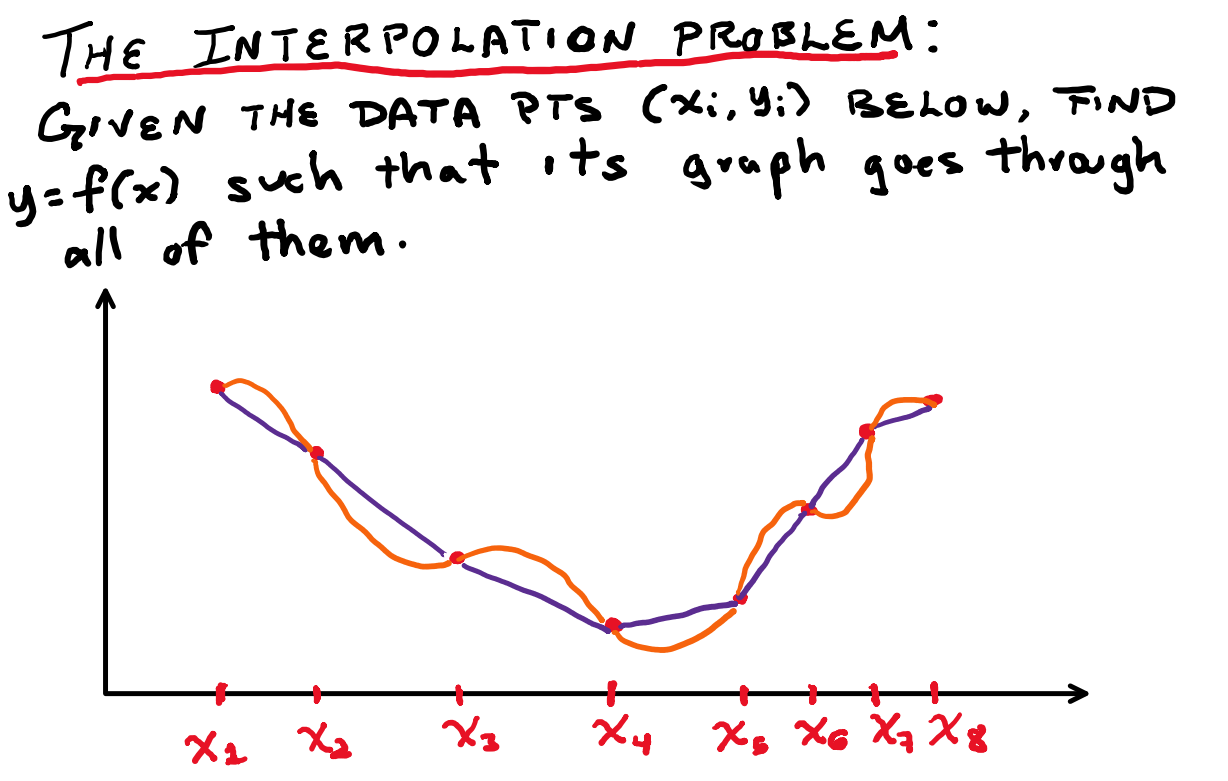
In general, you have to be careful how you choose **alpha_k** to ensure **sufficient descent** (that is, q(x_{k+1}) should be smaller than q(x_k) by some criteria); if you pick alpha_k = 1, the SD method *will likely not converge at all*. We discussed one way to do this (**a backtracking linesearch method**), but there are others (**trust region**, **interpolating with a quadratic**, etc).

If we do implement a good strategy to find alpha_k, then the SD method can be shown to converge linearly, often with trajecto ries that "zig-zag". Finally, I mentioned FYI that one can implement Newton and QuasiNewton algorithms for optimization (!!). The steps for these methods look as follows (H_q(x) being the n x n matrix of 2nd derivatives of q):

Newton step:
x_{k+1} = x_k - alpha_k H_q(x_k)^{-1} Grad_q(x_k)

QuasiNewton step:
x_{k+1} = x_k - alpha_k B_k^{-1} Grad_q(x_k)

Today we switch gears and tackle a different mathematical problem: that of interpolating data (using polynomials or some othe r set of functions.



- Vector space of polynomials $p(x)$ of degree $\leq n$

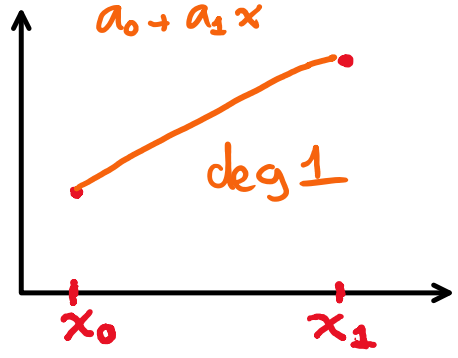
↳ $\mathcal{P}_n \longrightarrow \boxed{p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n}$

Basis $\{1, x, x^2, \ldots, x^n\}$ ($n+1$ LI functions)

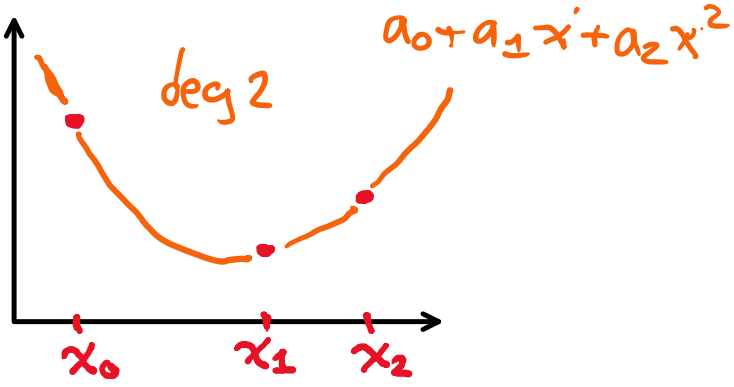
Coefficients $(a_0, a_1, \ldots, a_n) \in \mathbb{R}^{n+1}$

INTERPOLATION PROBLEM:

•) $n=1$ (2 points)



•) $n=2$ (3 pts)



Conjecture: $n+1$ points $x_0, x_1, \ldots, x_n$ then asking for $p(x)$ of deg $\leq n$ → unique sol.*

(*) x_i have to be distinct.

---

2 points: (x_0, y_0), (x_1, y_1) → Find a_0, a_1 such that:

$$\begin{cases} a_0 + a_1 x_0 = y_0 \\ a_0 + a_1 x_1 = y_1 \end{cases}$$

$$\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

---

3 points: (x_0, y_0), (x_1, y_1), (x_2, y_2)

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 = y_0 \end{cases}$$

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 = y_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 = y_2 \end{cases}$$

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

V, $\vec{a} = \vec{y}$

V Vandermonde matrix.

$$\det(V) = \prod_{i>j} (x_i - x_j) \neq 0 \text{ iff } x_i \neq x_j \text{ for } i > j.$$

INPUTS $\vec{x}, \vec{y}$

FORM V

SOLVE $\vec{a} = np.linalg.solve(V, \vec{y})$ EXPENSIVE $O(n^3)$

return $\vec{a}$.

Condition number $\underline{\underline{K(V)}}$ as a func of n?

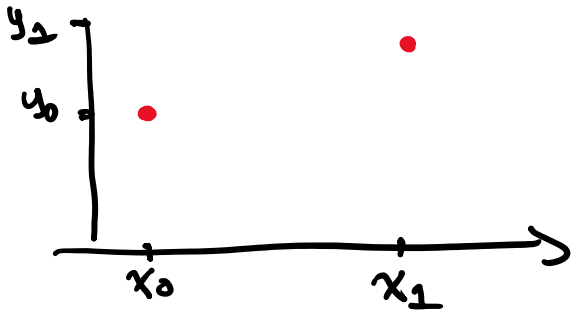
↳ exponential in n.

(not stable!)

Lagrange Interpolation.

↳ Find a better basis.

Two linear polynomials



Two linear polynomials $L_0(x)$, $L_1(x)$ such that.

$$p(x) = \beta_0 L_0(x) + \beta_1 L_1(x)$$

Equations

$$\begin{cases} \beta_0 L_0(x_0) + \beta_1 L_1(x_0) = y_0 \\ \beta_0 L_0(x_1) + \beta_1 L_1(x_0) = y_1 \end{cases}$$

$L_0(x_0) = 1 \quad L_0(x_1) = 0$

$L_1(x_0) = 0 \quad L_1(x_1) = 1$

$$\begin{bmatrix} L_0(x_0) & L_1(x_0) \\ L_0(x_1) & L_1(x_1) \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

$$\Rightarrow \boxed{p(x) = y_0 L_0(x) + y_1 L_1(x)}$$

$L_0(x) = c\,(x - x_1)$, $L_0(x_0) = c\,(x_0 - x_1) = 1$

$$\boxed{L_0(x) = \frac{x - x_1}{x_0 - x_1}}$$

$$\boxed{L_1(x) = \frac{x - x_0}{x_1 - x_0}}$$

$$p(x) = y_0 \underbrace{\left(\frac{x - x_1}{x_0 - x_1}\right)}_{L_0} + y_1 \underbrace{\left(\frac{x - x_0}{x_1 - x_0}\right)}_{L_1}$$

---

3 points $\rightarrow$ $L_0(x)$, $L_1(x)$, $L_2(x)$

$L_0(x_0) = 1$, $L_0(x_1) = L_0(x_2) = 0$