Finite Neuron Method
Programming Assignment

Jinchao Xu

KAUST and Penn State

xu@multigrid.org

CIRM, July 18th, 2023

CEMRACS 2023 Summer School

1. GD for nearly singluar systems
2. L^2-fitting: Training of shallow neural networks

Gradient Descent (GD) Method: $Au = g$

$$Au = g \Longleftrightarrow \arg\min \underbrace{\frac{1}{2}u^T Au - g^T u}_{f(u)}$$

- Gradient descent method:

$$u^{k+1} = u^k - \eta \nabla f(u^k) = u^k - \eta(Au^k - g)$$

- Scaled gradient descent method:

$$u^{k+1} = u^k - \eta[\mathrm{diag}(A)]^{-1}(Au^k - g)$$

GD for a nearly singular system

Consider: $A_\epsilon u = g$ $(A_\epsilon = A_0 + \epsilon I)$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply scaled gradient descent method with $\|A_\epsilon u^k - g\| \le 10^{-8}$:

ϵ	# of iter $= m$
1.	
10^{-1}	
10^{-2}	
10^{-3}	
10^{-4}	
0. [singular case]	

Ref for semi-definite case: Keller 1965; Lee, Wu, Xu and Zikatanov 2007

Remedy of GD: Over-parametrization

Write $u \in \mathbb{R}^3 = u_1 e_1 + u_2 e_2 + u_3 e_3$ as

$$u = \underset{\sim}{u}_1 e_1 + \underset{\sim}{u}_2 e_2 + \underset{\sim}{u}_3 e_3 + \underset{\sim}{u}_4 {\color{red}p} = P\underset{\sim}{u}$$

where

$$P = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in {\color{red}\ker(A_0)}.$$

Namely, we consider the coarse level with "lowest" frequency $p \in {\color{red}\ker(A_0)}$.

The equation $A_\epsilon u = g$ becomes

$$A_\epsilon P\underset{\sim}{u} = g \Longleftrightarrow ({\color{red}P^T A_\epsilon P})\underset{\sim}{u} = P^T g,$$

leading to a semi-definite system:

$$\color{red}{\begin{pmatrix} 1+\epsilon & -1 & 0 & \epsilon \\ -1 & 2+\epsilon & -1 & \epsilon \\ 0 & -1 & 1+\epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 3\epsilon \end{pmatrix}}\underset{\sim}{u} = \begin{pmatrix} -1 \\ -1 \\ 2 \\ 0 \end{pmatrix}.$$

# GD		
ϵ	original	scaled GD for expanded
1.		
10^{-1}		
10^{-2}		
10^{-3}		
10^{-4}		
10^{-5}		
10^{-9}		
10^{-10}		
0.		

Homework

1. Prove that the scaled gradient descent method converges for any symmetric, positive and definite A if $\eta \ll 1$.
2. Use Python to implement the gradient descent method for the example of the 3 by 3 system in the slides. Fill in the first table.
3. Use Python to implement the gradient descent (with different η) method for over-parametrization problem and check the convergence.
4. Use Python to implement the scaled gradient descent (with different η) method for over-parametrization problem and check the convergence. Fill in the second table.
5. Which method converges fastest?
6. Optional: Try to prove the faster convergence theoretically.

1. GD for nearly singluar systems
2. L^2-fitting: Training of shallow neural networks

L^2-fitting fitting using neural network

- Consider 1D L^2-fitting problem on $\Omega = [-\pi, \pi]$:

$$\min_{f_n \in \Sigma_n} \int_{-\pi}^{\pi} \frac{1}{2} |f(x) - f_n(x)|^2 dx. \tag{1}$$

- Σ_n is the spcae of ReLU shallow neural network with n neurons

$$\Sigma_n = \left\{ v(x) = \sum_{i=1}^{n} a_i \sigma(x + b_i) : a_i \in \mathbb{R}, \quad b_i \in \mathbb{R} \right\}, \quad \sigma(x) = \max(0, x).$$

- The resulting nonlinear, nonconvex optimization problem

$$\min_{a_i, b_i} \int_{-\pi}^{\pi} \frac{1}{2} |f(x) - \sum_{i=1}^{n} a_i \sigma(x + b_i)|^2 dx. \tag{2}$$

The above optimization problem (2) is usually solved by GD (or Adam).
Question

- Does the GD (or Adam) algorithm converge?
- Can we achieve the theoretical approximation rate, i.e., $\|f - f_n\|_{L^2} = O(n^{-2})$ in 1D?

Orthogonal greedy algorithm

- OGA
$$f_0 = 0, \quad g_n = \arg\max_{g \in \mathbb{D}} |\langle g, f - f_{n-1} \rangle|, \quad f_n = P_n(f), \tag{3}$$
where P_n is a projection onto $H_n = \text{span}\{g_1, g_2, ..., g_n\}$
- For 1D L^2-fitting for target $f(x)$ with $f(0) = 0$, the ReLU shallow neural network dictionary can be given by
$$\mathbb{D} = \{\sigma(x + b), b \in [-\pi, \pi]\} \tag{4}$$

Homework

Conaider a simple target function $f(x) = sin(x)$. You may also try a function of a higher frequency, say, $f(x) = sin(10x)$.

1. Solve the optimization problem using GD (or Adam). You may implement this with the help of PyTorch.
   - Record the L^2 errors for different number of neurons. Plot some numerical solutions for observation.
2. Use orthogonal greedy algorithm to train (build) a ReLU shallow neural network for fitting the target function.
   - Record the L^2 errors for different number of neurons.