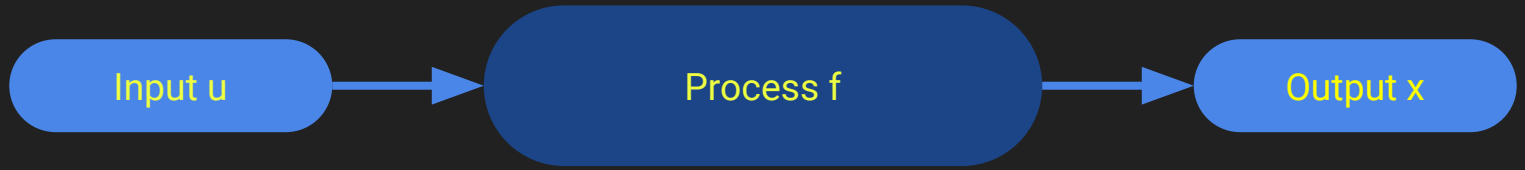
Deep Learning for Inverse Problems

Seminar 11-12-2022

Tran Thu Le

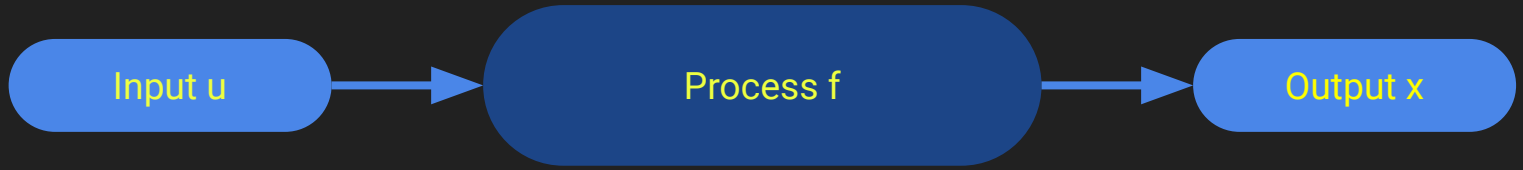
Forward Problems



Forward problem: have u, find x

$$f(u) = x$$

Inverse Problems



Inverse problem: have x, find u

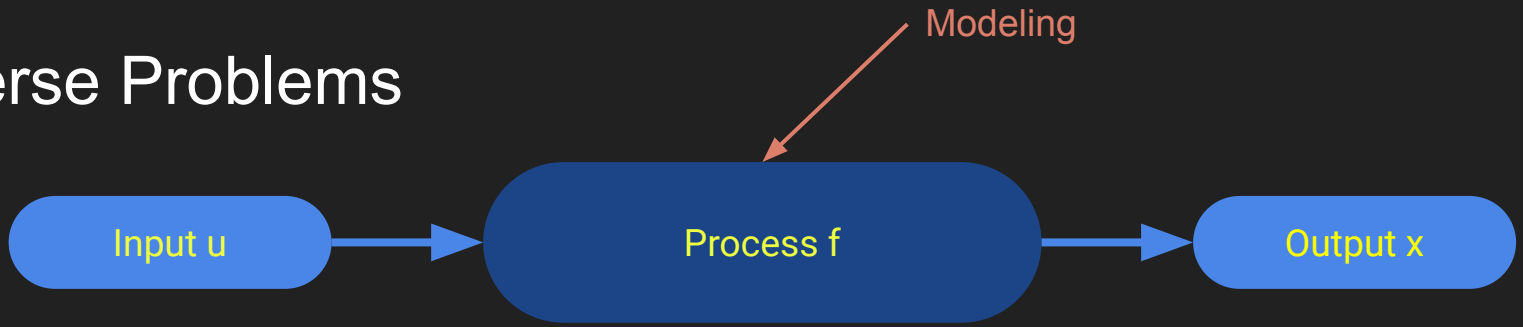
$$u = argmin_{u'} F(u') = ||f(u) - x||$$

Gradient method

$$u = u - \alpha \nabla F(u)$$

Challenge: find gradient → use backpropagation method → use Pytorch package

Inverse Problems



Inverse problem: have x, find u

$$u = argmin_{u'} F(u') = ||f(u) - x||$$

Solving method: Gradient descent

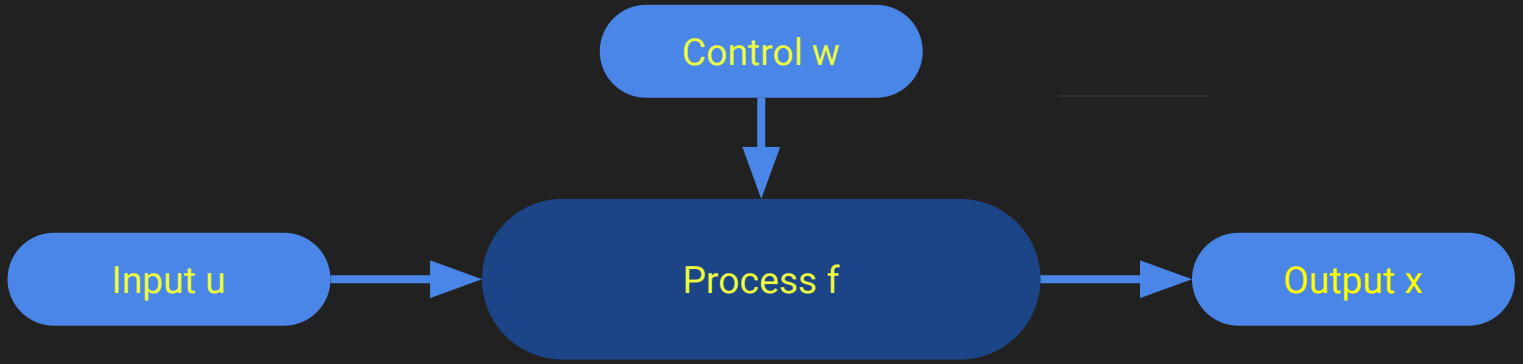
Solving

$$u = u - \alpha \nabla F(u)$$

Programming

Challenge: find gradient → use backpropagation method → use Pytorch package

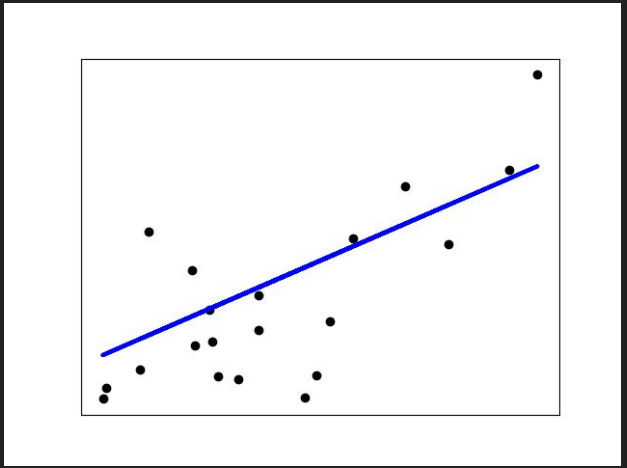
Inverse Parametric Problems



Problem: have x and u, find w such that $f(u, w) = x$

$$w = argmin_{w'} F(w') = \frac{1}{N} \sum_{i=1,n} ||f(u^{(i)}, w') - x^{(i)}||$$

Inverse Parametric Problems



Example 1. Linear Regression

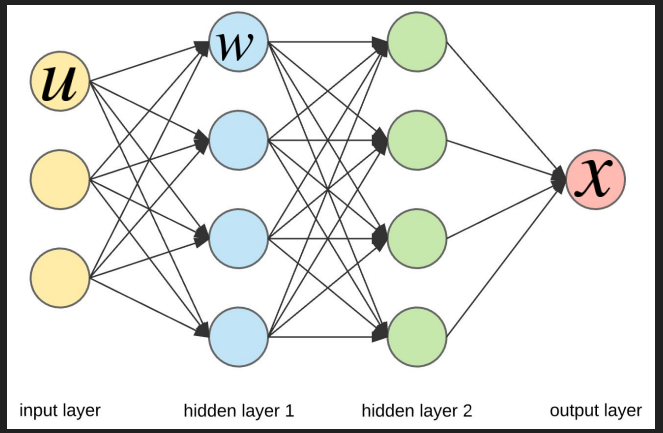
In linear regression, the process f is linear

$$x \approx f(u, w) = w_1 u + w_0$$

We find the line by least squares method

$$w = argmin_{w'} F(w') = \frac{1}{N} \sum_{i=1,n} (f(u^{(i)}, w') - x^{(i)})^2$$

Inverse Parametric Problems



Example 2. Deep Learning

In deep learning the function f contains many sub-functions (nodes). Given u and x, the goal is to change (learn) parameters w of each node so that:

$$f(u, w) = x$$

Inverse Parametric Linear Programming (PLP) [1]

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}(\mathbf{u},\mathbf{w})'\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}(\mathbf{u},\mathbf{w})\mathbf{x} \leq \mathbf{b}(\mathbf{u},\mathbf{w}), \end{aligned}$$

Model: Let x be the optimal solution of the process f of solving PLP with parameters u, w: f(u, w)=x

Problem: Assume that w be fixed and unknown. Given (ui, xi) such that f(ui, w)=xi. Recover w !

[1] Tan, Yingcong, Andrew Delong, and Daria Terekhov. "Deep inverse optimization."

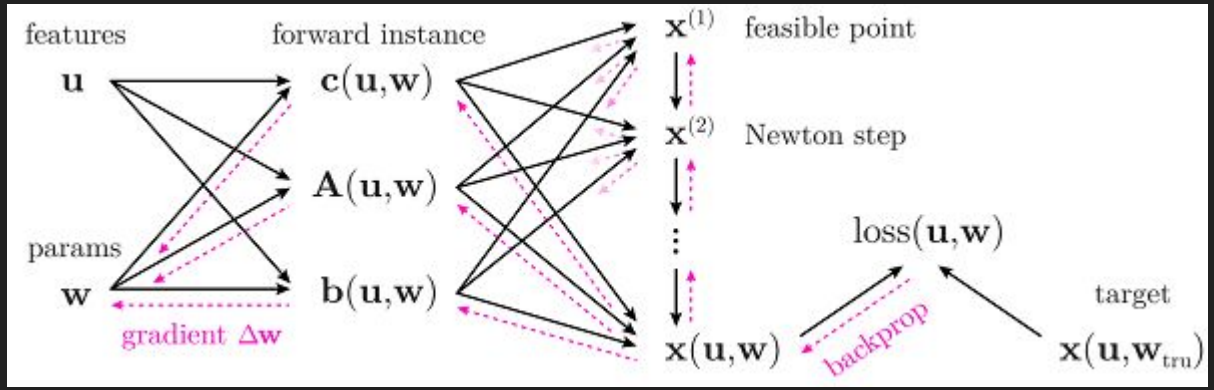
Inverse Parametric Linear Programming

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}(\mathbf{u},\mathbf{w})'\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}(\mathbf{u},\mathbf{w})\mathbf{x} \leq \mathbf{b}(\mathbf{u},\mathbf{w}), \end{aligned}$$

Algorithm 1 Deep inverse optimization framework.

Input: $\mathbf{w}_{\text{ini}}$; $(\mathbf{u}^n, \mathbf{x}^n_{\text{tru}})$ for $n = 1, \ldots N$,
Output: $\mathbf{w}_{\text{lrn}}$

$\mathbf{w} \leftarrow \mathbf{w}_{\text{ini}}$
for s in 1 .. `max_steps` **do**
  $\Delta\mathbf{w} \leftarrow \mathbf{0}$
  for n in 1 .. N **do**
    $\mathbf{x} \leftarrow \mathbf{FO}(\mathbf{u}^n, \mathbf{w})$ ▷ Solve forward problem
    $\ell \leftarrow \mathcal{L}(\mathbf{x}, \mathbf{x}^n_{\text{tru}})$ ▷ Compute loss
    $\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \frac{\partial \ell}{\partial \mathbf{w}}$ ▷ Accumulate gradient by backprop
  end for
  $\beta \leftarrow$ `line_search`$(\mathbf{w}, \boldsymbol{\alpha} \cdot \frac{\Delta\mathbf{w}}{N})$ ▷ Find safe step size
  $\mathbf{w} \leftarrow \mathbf{w} - \beta\boldsymbol{\alpha} \cdot \frac{\Delta\mathbf{w}}{N}$ ▷ Update weights
end for
Return $\mathbf{w}$



[1] Tan, Yingcong, Andrew Delong, and Daria Terekhov. "Deep inverse optimization."

Python Code

https://github.com/Tran-Thu-Le/share/tree/main/deep_inv_opt