Machine Learning Methods for Numerical Solutions of Partial Differential Equations

Julia Costacurta, Cameron Martin, Hongyuan Zhang
Supervisors: Adam Stinchcombe and Mihai Nica

August 28, 2019

Fields Undergraduate Summer Research Program

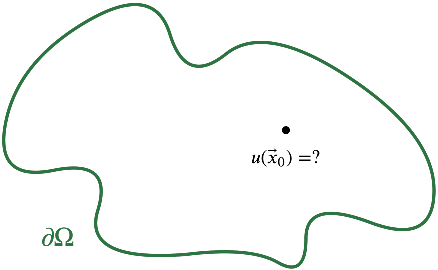
Introduction to Problem

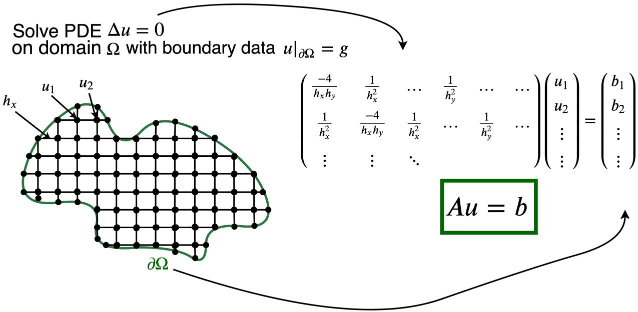
Goal: Find numerical solution to (possibly high dimensional) PDE problem with irregular boundary.

Example PDE Problem:

$$\Delta u = 0 \text{ on } \Omega$$

with $u|_{\partial\Omega} = g$ (Dirichlet boundary conditions).



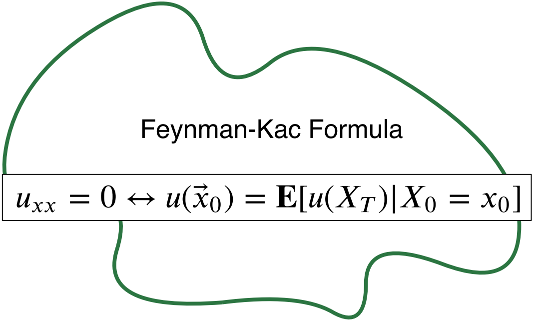


- $\Delta u = 0, u|_{\partial\Omega} = g \leftrightarrow Av = b$
- Challenging to grid domain - we want a **grid-free method**

How do we make a grid-free method?

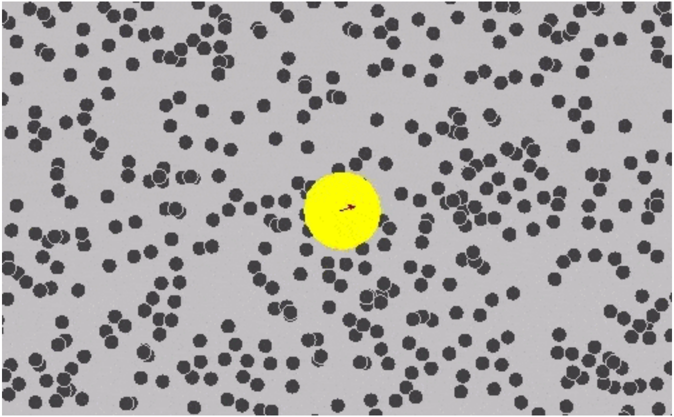
Connection between PDE and Brownian Motion

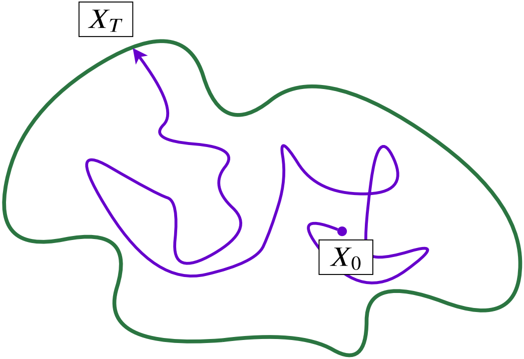
How do we make a grid-free method?



By exploiting a relationship between PDEs and Brownian motions, we can derive an expression for the solution to our PDE that depends on some Brownian motion X_t.

What is a Brownian motion?





$$\boxed{u_{xx} = 0 \leftrightarrow u(\vec{x}_0) = \mathbf{E}[u(X_T)|X_0 = x_0]}$$

Monte Carlo method – average value of $u(X_T)$ for a large number of walkers

Machine Learning in Three Slides

Machine Learning (ML) in Three Slides

- Problem: Given data $(x_i, y_i)_{1 \leq i \leq N}$ find a function $u(x)$ such that $u(x_i) = y_i$ for all i.

Machine Learning (ML) in Three Slides

- Problem: Given data $(x_i, y_i)_{1\leq i\leq N}$ find a function $u(x)$ such that $u(x_i) = y_i$ for all i.
- Example: $x_i \in \mathbb{R}^{784}$ - pixels in an image, $y_i \in \{0,1\}$ - is it a cat? Find $u : \mathbb{R}^{784} \longrightarrow \{0,1\}$ such that

- Problem: Given data $(x_i, y_i)_{1 \leq i \leq N}$ find a function $u(x)$ such that $u(x_i) = y_i$ for all i.
- Example: $x_i \in \mathbb{R}^{784}$ - pixels in an image, $y_i \in \{0, 1\}$ - is it a cat? Find $u : \mathbb{R}^{784} \longrightarrow \{0, 1\}$ such that

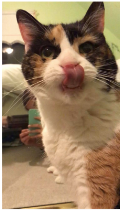
$u\Bigg($  $\Bigg) = 1$

Figure 1: A cat

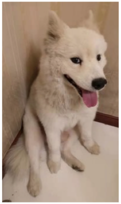
$u\Bigg($  $\Bigg) = 0$

Figure 2: Not a cat

Machine Learning (ML) in Three Slides

- Problem: Given data $(x_i, y_i)_{1 \le i \le N}$ find a function $u(x)$ such that $u(x_i) = y_i$ for all i.
- Example: $x_i \in \mathbb{R}^{784}$ - pixels in an image, $y_i \in \{0, 1\}$ - is it a cat? Find $u : \mathbb{R}^{784} \longrightarrow \{0, 1\}$ such that

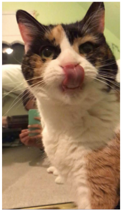
$u\Bigg($  $\Bigg) = 1$

Figure 1: A cat

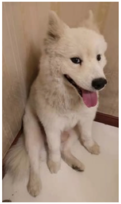
$u\Bigg($  $\Bigg) = 0$

Figure 2: Not a cat

Machine Learning (ML) in Three Slides

- Problem: Given data $(x_i, y_i)_{1\leq i\leq N}$ find a function $u(x)$ such that $u(x_i) = y_i$ for all i.
- Example: $x_i \in \mathbb{R}^{784}$ - pixels in an image, $y_i \in \{0, 1\}$ - is it a cat? Find $u : \mathbb{R}^{784} \longrightarrow \{0, 1\}$ such that
$$u(x_i) = y_i = \begin{cases} 0, & \text{image } x_i \text{ is not a cat} \\ 1, & \text{image } x_i \text{ is a cat} \end{cases}$$

Machine Learning (ML) in Three Slides

- Problem: Given data $(x_i, y_i)_{1\leq i\leq N}$ find a function $u(x)$ such that $u(x_i) = y_i$ for all i.
- Example: $x_i \in \mathbb{R}^{784}$ - pixels in an image, $y_i \in \{0,1\}$ - is it a cat? Find $u : \mathbb{R}^{784} \longrightarrow \{0,1\}$ such that

$$u(x_i) = y_i = \begin{cases} 0, & \text{image } x_i \text{ is not a cat} \\ 1, & \text{image } x_i \text{ is a cat} \end{cases}$$

- Strategy:
  1. Have $u = u_\theta$ depend on parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ (e.g. $u_\theta(x) = \theta_1 x + \theta_0$)
  2. Define a “loss function” of these parameters and minimize it.

ML in Three Slides: Minimizing Loss with Gradient Descent

- A **loss function** is a function of the parameters $\theta_1, \ldots, \theta_n$ that tells you “how bad” the function approximation is on the data set, e.g.

ML in Three Slides: Minimizing Loss with Gradient Descent

- A **loss function** is a function of the parameters $\theta_1, \ldots, \theta_n$ that tells you “how bad” the function approximation is on the data set, e.g.

$$L(\theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{N} \sum_{i=1}^{N} (y_i - u_\theta(x_i))^2$$

ML in Three Slides: Minimizing Loss with Gradient Descent

- A **loss function** is a function of the parameters $\theta_1, \ldots, \theta_n$ that tells you "how bad" the function approximation is on the data set, e.g.

$$L(\theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{N} \sum_{i=1}^{N} (y_i - u_\theta(x_i))^2$$

- How do we minimize L? Gradient descent!

ML in Three Slides: Minimizing Loss with Gradient Descent

- A **loss function** is a function of the parameters $\theta_1, \ldots, \theta_n$ that tells you "how bad" the function approximation is on the data set, e.g.

$$L(\theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{N} \sum_{i=1}^{N} (y_i - u_\theta(x_i))^2$$

- How do we minimize L? Gradient descent!
- Compute gradient of loss function:

$$\nabla_\theta L = \left(\tfrac{\partial L}{\partial \theta_1}, \quad \tfrac{\partial L}{\partial \theta_2}, \quad \ldots, \quad \tfrac{\partial L}{\partial \theta_n} \right)^T.$$

ML in Three Slides: Minimizing Loss with Gradient Descent

- A **loss function** is a function of the parameters $\theta_1, \ldots, \theta_n$ that tells you "how bad" the function approximation is on the data set, e.g.

$$L(\theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{N} \sum_{i=1}^{N} (y_i - u_\theta(x_i))^2$$

- How do we minimize L? Gradient descent!
- Compute gradient of loss function:

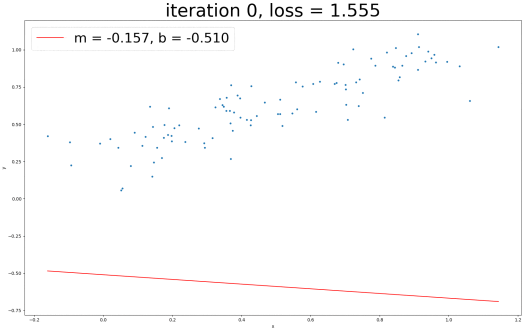
$$\nabla_\theta L = \left(\frac{\partial L}{\partial \theta_1}, \quad \frac{\partial L}{\partial \theta_2}, \quad \ldots, \quad \frac{\partial L}{\partial \theta_n} \right)^T.$$

- Update parameters and iterate

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

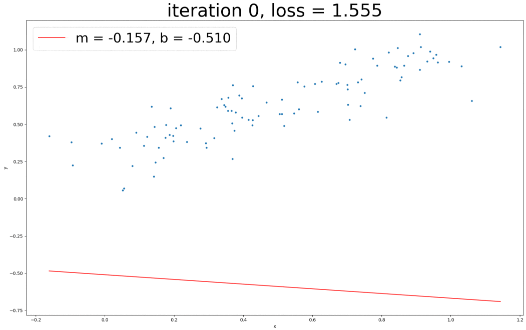
- α is called the **learning rate**

ML in Three Slides: An Elementary Example



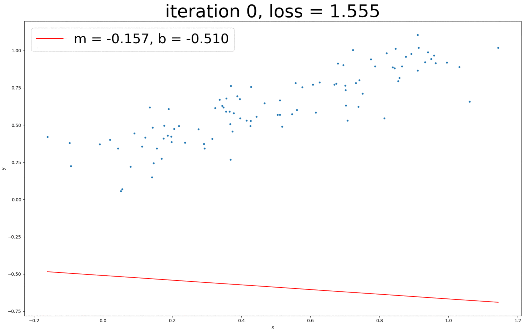
ML in Three Slides: An Elementary Example

- Perform linear regression on a set of data points (x_i, y_i)



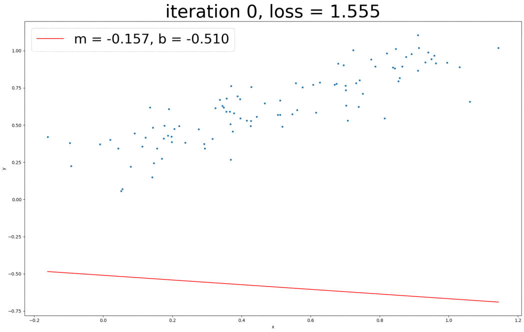
ML in Three Slides: An Elementary Example

- Perform linear regression on a set of data points (x_i, y_i)
- Choose function form $u_\theta(x) = mx + b$



ML in Three Slides: An Elementary Example

- Perform linear regression on a set of data points (x_i, y_i)
- Choose function form $u_\theta(x) = mx + b$
- Start with random guesses for m, b



Putting the Pieces Together

A New Loss Function

- We want to use $L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(u(x_i) - u_\theta(x_i)\right)^2$, but we don't know u

A New Loss Function

- We want to use $L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(u(x_i) - u_\theta(x_i)\right)^2$, but we don't know u
- Solution: use Feynman-Kac formula
  $u(x) = \mathbf{E}\left[u(X_T) | X_0 = x\right]$

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{E}\left[u(X_{\Delta t}) | X_0 = x_i\right] - u_\theta(x_i)\right)^2$$

- approximate this expectation value with a random sample to get new loss

A New Loss Function

- We want to use $L(\theta) = \frac{1}{N}\sum\limits_{i=1}^{N}(u(x_i) - u_\theta(x_i))^2$, but we don't know u
- Solution: use Feynman-Kac formula
$u(x) = \mathbf{E}\left[u(X_T) | X_0 = x\right]$

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{E}\left[u(X_{\Delta t}) | X_0 = x_i\right] - u_\theta(x_i)\right)^2$$

- approximate this expectation value with a random sample to get new loss

$$L_{\text{new}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(X_0)\right)^2$$

A New Loss Function

$$L_{\text{new}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(X_0) \Big)^2$$

A New Loss Function

$$L_{\text{new}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(X_0) \Big)^2$$

- We replaced the expectation value $\mathbf{E}\left[u(X_{\Delta t}) | X_0 = x_i\right]$ with $u_\theta(X_{\Delta t}^{(i)})$

A New Loss Function

$$L_{\text{new}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(X_0) \Big)^2$$

- We replaced the expectation value $\mathbf{E}\left[u(X_{\Delta t}) | X_0 = x_i\right]$ with $u_\theta(X_{\Delta t}^{(i)})$
- $u_\theta(X_{\Delta t}^{(i)})$ is our data (analogous to y_i from intro to ML), generated by the random process X_t.

A New Loss Function

$$L_{\text{new}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(X_0)\Big)^2$$

- We replaced the expectation value $\mathbf{E}\left[u(X_{\Delta t})|X_0 = x_i\right]$ with $u_\theta(X_{\Delta t}^{(i)})$
- $u_\theta(X_{\Delta t}^{(i)})$ is our data (analogous to y_i from intro to ML), generated by the random process X_t.
- $\nabla_\theta L \approx \nabla_\theta L_{\text{new}}$, so gradient descent still works!

Loss Function Example

Applying the New Loss Function

- **Problem:**
  $u'' + u' - u = 2\pi \cos(2\pi x) - \left(1 + (2\pi)^2\right) \sin(2\pi x)$,
  $x \in [0,1], u(0) = u(1) = 0$

Applying the New Loss Function

- **Problem:**
  $u'' + u' - u = 2\pi \cos(2\pi x) - \left(1 + (2\pi)^2\right) \sin(2\pi x)$,
  $x \in [0, 1], u(0) = u(1) = 0$
- **Analytic solution:** $u(x) = \sin(2\pi x)$

Applying the New Loss Function

- **Problem:**
  $u'' + u' - u = 2\pi \cos(2\pi x) - \left(1 + (2\pi)^2\right) \sin(2\pi x)$,
  $x \in [0, 1], u(0) = u(1) = 0$
- **Analytic solution:** $u(x) = \sin(2\pi x)$
- Can write function approximation as finite linear combination of polynomials

$$u_\theta(x) = \sum_{k=0}^{6} \theta_k x^k.$$

Applying the New Loss Function

- **Problem:**
  $u'' + u' - u = 2\pi \cos(2\pi x) - \left(1 + (2\pi)^2\right) \sin(2\pi x)$,
  $x \in [0, 1], u(0) = u(1) = 0$
- **Analytic solution:** $u(x) = \sin(2\pi x)$
- Can write function approximation as finite linear combination of polynomials

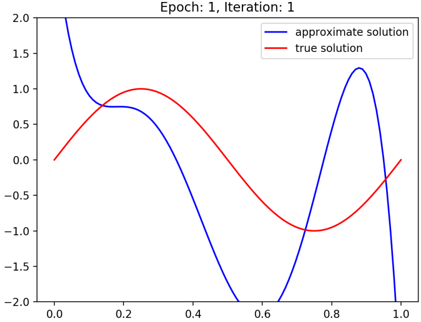
$$u_\theta(x) = \sum_{k=0}^{6} \theta_k x^k.$$

- Using the modified loss function,

$$\frac{\partial L}{\partial \theta_k} = \frac{1}{N} \sum_{i=1}^{N} -2x_i^k \left(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(x_i)\right)$$

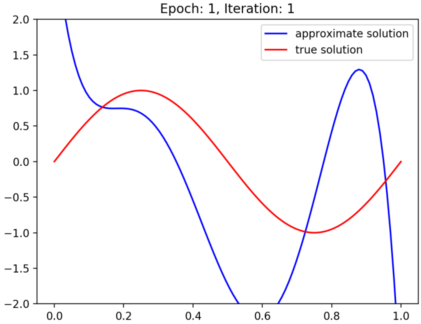
An Example - 1D Value Function Approximation

- Update rule: $\theta_k \leftarrow \theta_k + \alpha \frac{1}{N} \sum_{i=1}^{N} 2x_i^k \left(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(x_i) \right)$



An Example - 1D Value Function Approximation

- Update rule: $\theta_k \leftarrow \theta_k + \alpha \frac{1}{N} \sum_{i=1}^{N} 2x_i^k \left(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(x_i) \right)$



Key point: loss function makes no reference to solution! We approximate solution using only randomly generated data.

Machine Learning with Artificial Neural Networks

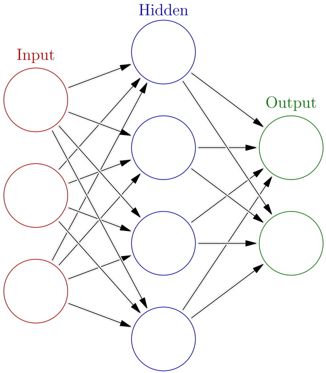
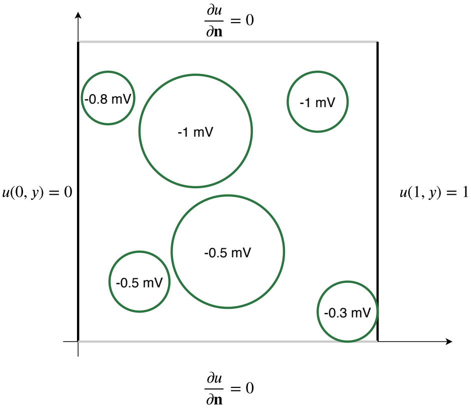


Figure 3: ANN Example

A neural network is a compositional function that depends on parameters (weights and biases).

An Example: Cell Battery



PDE: $\Delta u = 0$ on a unit square (representing a battery).

$u(x, y)$ is the voltage, which jumps when entering/exiting a cell due to negative resting potential within the cell.

Method and Implementation

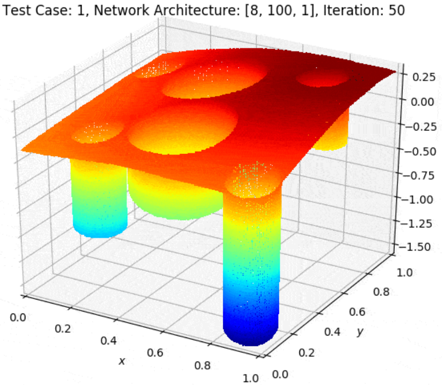
Loss function:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big(u_\theta(X_{\Delta t}^{(i)}) - u_\theta(X_0^{(i)}) - VoltChange(X^{(i)})\Big)^2$$
$$+ \frac{1}{M} BdryWeight \sum_{i=1}^{M}\Big(g(P^{(i)}) - u_\theta(P^{(i)})\Big)^2$$

where P are points on boundary and $BdryWeight$ is boundary weight, a constant.

To approximate real solution, minimize loss by updating θ (weights and biases) via gradient descent.

Numerical Results



Summary

Summary

	Finite Difference	Monte Carlo	ML w/ Basis Fns	ML w/ ANN
Solves for solution at…	Grid of points	One point	Every point	Every point
Gains information…	All at once	After all walkers finish	At every step of walkers	At every step of walkers
Requires basis?	No basis functions	No basis functions	Need to choose basis functions	No basis functions
Best for…	Simple boundaries, low dimensions	Complex boundaries, sol. at single point	Complex boundaries, high dimensions, know good basis	Complex boundaries, high dimensions, don't know basis

Thank you!

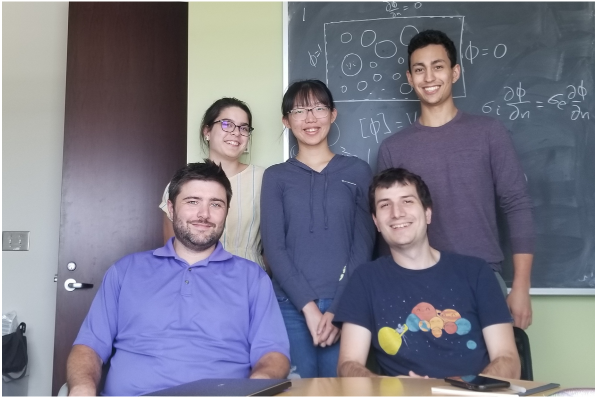


Figure 4: Our Team