Forecasting Forces in Porosity Rock with Deep Neural Network

Aleksandr Katrutsa

Skolkovo Institute of Science and Technology
Moscow Institute of Physics and Techology

Moscow
2015

Plan

1. Problem Statement
2. Data Generation
3. PDE solving
4. Deep Learning Application

Problem Statement

Let A be the covariance from the following family:

$$A(x, y) = \exp\left(-\frac{\|x - y\|^2}{b^2}\right).$$

The random field for the given covariance $k_A(x, y)$.
Consider PDE:

$$\mathrm{div}(k_A(x, y)\mathrm{grad}(u(x, y))) = 1$$

If solution $u(x, y)$ is known, one can compute the integral:

$$f = \int\limits_{\Omega} u(x, y) dS$$

The main goal is to find function g, which maps random field $k_A(x, y)$ to the f without direct solution of the PDE:

$$g : k_A(x, y) \to f$$

Data Generation

The goal of this stage is to generate random fields with given covariance A fast.
The procedure:

- Compute decomposition $A = RR^\top$
- Generate two random vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ from $\mathcal{N}(0, I)$ and compose $\mathbf{v} = \mathbf{v}_1 + i\mathbf{v}_2$
- Multiply $R\mathbf{v} = \mathbf{f}$, real and imaginary parts of $\mathbf{f}$ are independent random fields with given covariance A

How to compute $A = RR^\top$ efficiently?

- The covariance $A(x, y)$ is a symmetric Block Toeplitz with Toeplitz Block (BTTB) matrix $\Rightarrow$ to store it one needs only first column
- Such matrix could be embedded to Block Circulant with Circulant Blocks (BCCB) matrix C
- BCCB matrix is diagonalized by 2D Fourier transform:

$$C = F^* \Lambda F$$

- $R = F^* \Lambda^{1/2} F$
- Now matvec is really fast!

Example

Random fields with different parameters b^2.

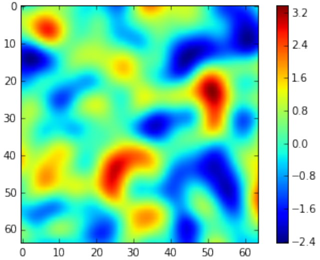

Figure : $b^2 = 0.01$

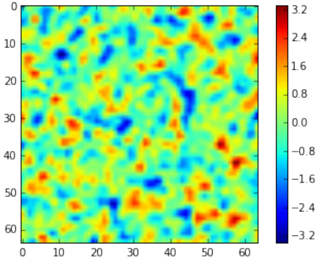

Figure : $b^2 = 0.001$

Partial Differential Equation

Consider partial differential equation:

$$\mathrm{div}(k_A(x, y)\mathrm{grad}(u(x, y))) = 1,$$

where $k_A(x, y)$ is the generated random field and $u(x, y)$ is unknown function.
Discretize domain with uniform mesh and convert continuous PDE to system of linear equation.

Parameter $b^2 = 0.01$.

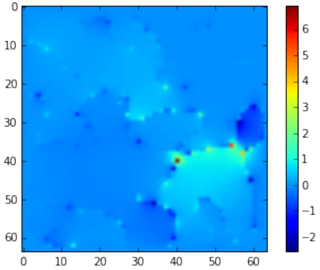

Figure : Solution

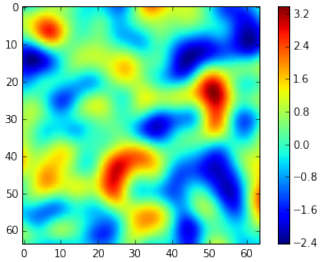

Figure : Random field $k_A(x, y)$

Solution visualization

Parameter $b^2 = 0.001$.

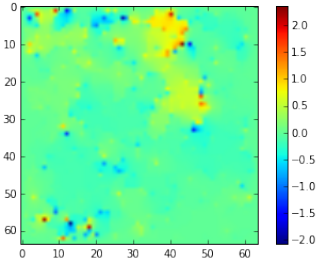

Figure : Solution

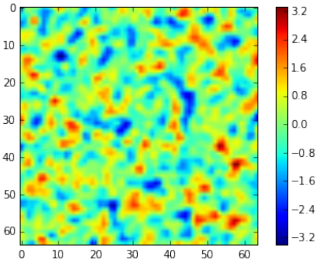

Figure : Random field $k_A(x, y)$

Why Deep Neural Network?

We want to find function g:

$$g : k_A(x, y) \to f.$$

Input: image of the random field

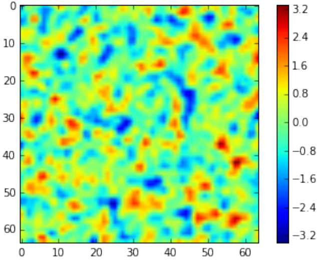

Figure : Random field $k_A(x, y)$

Output: real number f

Network Configuration

$$\text{Conv64-3} \rightarrow \text{Pool2} \rightarrow \text{FC100} \rightarrow$$
$$\rightarrow \text{ReLU} \rightarrow \text{FC150} \rightarrow \text{ReLU} \rightarrow \text{Output}$$

Parameters:

- number of epochs - 20
- learning rate $= 0.01$, momentum $= 0.9$

Results

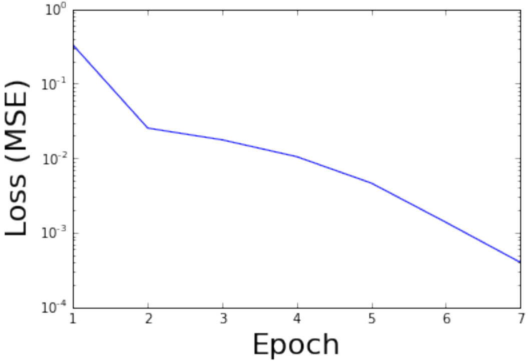

Figure : Convergence

The End