BY James 'Xiaozhong' Ding
james.ding.illinois@gmail.com
September 14, 2017

Abstract

The following discussion attempts to demonstrate the equivalence between numerical methods for PDE's and carefully constructed convolutional network.

It tries to provide theoretical evidence to support two empirical observations in practice:
(1) Why convolutional networks tend to work better than other A.I. models (Analogy to PDE's)
(2) Why deep neural networks work better than shallow ones in spite of universal approximation theorem (Analogy to PDE discretization scheme and unit grid size)

Chapter 1. Convolutional Neural Network

1.1 Fundamental Definition

Convolutional Neural Network (CNN) is defined as:

$$f(t) = \int x(a)w(j-a)da \coloneqq (x * w)(t)$$

Or in discrete terms:

$$f(t) = \sum_{-\infty}^{\infty} x(a)w(t-a)$$

1.2 Model Design

Let us consider a special case that:

$$\begin{cases} w(t-a) = 0; \ |a| > 1 \\ w(t-a) \geq 0; \ |a| \leq 1 \end{cases}$$

So we define such a network with (N+1)-layer and M inputs:

Input Layer:

$$\vec{x} \coloneqq \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_j \\ \vdots \\ x_M \end{bmatrix}; x_j \in \mathbb{R}$$

Where $\vec{x}$ are interpolations of discrete samplings from continuous real number space $\mathbb{R}$, with:

$$x_{j+1} - x_j = \Delta x; \ x_{j,j+1} \in \mathbb{R}$$

Perception Layer:

Using Rectified Linear Unit (ReLU), we define:

$$\overrightarrow{f^N} = \begin{bmatrix} f_0^N \\ f_1^N \\ \vdots \\ f_j^N \\ \vdots \\ f_M^N \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_j) \\ \vdots \\ f(x_M) \end{bmatrix} = \left(\vec{x} + \vec{k}\right)^+$$

Where $\vec{k}$ is a vector of constants which provides level for each input x_j. In Financial Engineering problems, this is nothing more than the payoff function, with k being the strike prices.

Hidden Layer Neuron Function:

$$\overrightarrow{\phi^i} \coloneqq \begin{bmatrix} \phi_0^i \\ \phi_1^i \\ \vdots \\ \phi_j^i \\ \vdots \\ \phi_M^i \end{bmatrix} = W^i \cdot \overrightarrow{f^i} \coloneqq \begin{bmatrix} b_0^i & c_0^i & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & a_1^i & b_1^i & c_1^i & 0 & \dots & \dots & 0 \\ 0 & 0 & a_2^i & b_2^i & c_2^i & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & a_j^i & b_j^i & c_j^i & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & a_{M-1}^i & b_{M-1}^i & c_{M-1}^i \\ 0 & \dots & \dots & \dots & \dots & 0 & a_M^i & b_M^i \end{bmatrix} \cdot \begin{bmatrix} f_0^i \\ f_1^i \\ \vdots \\ f_j^i \\ \vdots \\ f_M^i \end{bmatrix}$$

Where,

$$j \in [0, M-1], i \in [0, N-1]$$

Or in scalar form:

$$\phi_j^i = a_j^i f_{j-1}^i + b_j^i f_j^i + c_j^i f_{j+1}^i$$

Neuron Signal Strengths (Kernel):

$$W^{i+1} := \begin{bmatrix} b_0^{i+1} & c_0^{i+1} & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & a_1^{i+1} & b_1^{i+1} & c_1^{i+1} & 0 & \dots & \dots & 0 \\ 0 & 0 & a_2^{i+1} & b_2^{i+1} & c_2^{i+1} & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & a_j^{i+1} & b_j^{i+1} & c_j^{i+1} & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & a_{M-1}^{i+1} & b_{M-1}^{i+1} & c_{M-1}^{i+1} \\ 0 & \dots & \dots & \dots & \dots & 0 & a_M^{i+1} & b_M^{i+1} \end{bmatrix}$$

Hence, we conclude the **Feature Maps** at each layer:

$$W^{i+1} \cdot \overrightarrow{f^{i+1}} = \overrightarrow{\phi^i} \qquad (1.2.1a)$$

Or, in scalar form:

$$a_j^{i+1} f_{j-1}^{i+1} + b_j^{i+1} f_j^{i+1} + c_j^{i+1} f_{j+1}^{i+1} = \phi_j^i = a_j^i f_{j-1}^i + b_j^i f_j^i + c_j^i f_{j+1}^i \qquad (1.2.1b)$$

1.3 Cost Function

Optimal configurations of $W^{i,i+1}$ can be obtained by solving:

$$\min_{W^{i,i+1},\vec{k}} \frac{1}{M} \left[\overrightarrow{f^0} - \overrightarrow{f^{observed}}\right]^T \left[\overrightarrow{f^0} - \overrightarrow{f^{observed}}\right] \qquad (1.2.2)$$

1.4 Numerical Methods

Option (1): LU Decomposition

$$W^{i+1} = LU \xrightarrow{yields} \overrightarrow{f^{i+1}} = U^{-1}\left[L^{-1}\left(W^i \overrightarrow{f^i}\right)\right]$$

Option (2): Projected Successive Over Relaxation (Gauss-Siedel)

For path dependent problems such as American options in Financial Engineering, or in case of Rectified Linear Unit defined for each layer.

at iteration (+), do:

$$f_j^{i;+} = \frac{\begin{bmatrix} a_j^{i+1} & b_j^{i+1} & c_j^{i+1} \end{bmatrix} \cdot \begin{bmatrix} f_{j-1}^{i+1} \\ f_j^{i+1} \\ f_{j+1}^{i+1} \end{bmatrix} - \begin{bmatrix} a_j^i & c_j^i \end{bmatrix} \begin{bmatrix} f_{j-1}^{i;-} \\ f_{j+1}^{i;+} \end{bmatrix} - s_j^i}{b_j^i}$$

$$f_j^{i;+} = f_j^{i;-} + \omega\left(f_j^{i;+} - f_j^{i;-}\right);\ error_j^i = \left|f_j^{i;+} - f_j^{i;-}\right|$$

$$f_j^{i;+} = ReLU\left(f_j^{i;+}\right) = \left(f_j^{i;+} + k\right)^+$$

while: $maximum\left(error_j\right)^i > precision \,\&\&\, iteration < max\_iterations$

where s_j^i is bias at network layer i and input level j, ω is the hyper-parameter for PSOR

Chapter 2. 1 Dimensional Parabolic Partial Differential Equation

2.1 Feynman-Kac Theorem

Suppose we have a stochastic process x_t:

$$dx_t = \mu(x_t, t)dt + \sigma(x_t, t)dW_t^{\mathbb{Q}} \qquad (2.1.1)$$

Where $W_t^{\mathbb{Q}}$ is Brownian motion under probability measure $\mathbb{Q}$

Then, a stochastic system $f(x_t, t)$ which is a differentiable function of x_t must satisfy:

$$\frac{\partial f}{\partial t} + \mu(x_t, t)\frac{\partial f}{\partial x} + \frac{1}{2}\sigma(x_t, t)^2\frac{\partial^2 f}{\partial x^2} = 0 \qquad (2.1.2)$$

With boundary condition $f(x_T, T)$, and solution:

$$f(x_t, t) = E^{\mathbb{Q}}[f(x_T, T)|\mathcal{F}_t]$$

Conversely, the relationship also holds if we use the PDE to infer the process which x_t follows.

2.2 Crank-Nicolson Discretization Scheme:

$$\frac{\partial f}{\partial t} \approx \frac{f_j^{i+1} - f_j^i}{\Delta t}$$

$$\frac{\partial f}{\partial x} \approx \frac{f_{j+1}^i - f_{j-1}^i + f_{j+1}^{i+1} - f_{j-1}^{i-1}}{4\Delta x}$$

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f_{j+1}^i - 2f_j^i + f_{j-1}^i + f_{j+1}^{i+1} - 2f_j^{i+1} + f_{j-1}^{i-1}}{2\Delta x^2}$$

Also,

$$t_i = t_0 + i\Delta t$$
$$x_j = x_0 + j\Delta x$$

Let's substitute them into 2.1.2 and rearrange the equation, we have:

$$f_{j-1}^{i+1}\left[-\frac{\sigma(x_j, t_i)^2}{4\Delta x^2} + \frac{\mu(x_j, t_i)}{4\Delta x}\right] + f_j^{i+1}\left[\frac{\sigma(x_j, t_i)^2}{2\Delta x^2} - \frac{1}{\Delta t}\right] + f_{j+1}^{i+1}\left[-\frac{\sigma(x_j, t_i)^2}{4\Delta x^2} - \frac{\mu(x_j, t_i)}{4\Delta x}\right]$$
$$= f_{j-1}^{i}\left[\frac{\sigma(x_j, t_i)^2}{4\Delta x^2} - \frac{\mu(x_j, t_i)}{4\Delta x}\right] + f_j^{i}\left[-\frac{\sigma(x_j, t_i)^2}{2\Delta x^2} - \frac{1}{\Delta t}\right] + f_{j+1}^{i}\left[\frac{\sigma(x_j, t_i)^2}{4\Delta x^2} + \frac{\mu(x_j, t_i)}{4\Delta x}\right] \qquad (2.2.1)$$

Therefore, we can clearly see, it is of the same form as equation $(1.2.1b)$, where:

$$a_j^{i+1} = \left[-\frac{\sigma(x_j, t_i)^2}{4\Delta x^2} + \frac{\mu(x_j, t_i)}{4\Delta x}\right];\ b_j^{i+1} = \left[\frac{\sigma(x_j, t_i)^2}{2\Delta x^2} - \frac{1}{\Delta t}\right];\ c_j^{i+1} = \left[-\frac{\sigma(x_j, t_i)^2}{4\Delta x^2} - \frac{\mu(x_j, t_i)}{4\Delta x}\right] \qquad (2.2.2a)$$

$$a_j^i = -a_j^{i+1};\ b_j^i = -b_j^{i+1} - \frac{2}{\Delta t};\ c_j^i = -c_j^{i+1} \qquad (2.2.2b)$$

Moreover, we can easily see in similar fashion that an explicit discretization scheme yields a much more straight forward CNN without the need to define neuron functions $\vec{\phi}^i$ at each layer i.

2.3 Dimensions Reduction

At this point, we can already solve the problem specified by 1.2.2. However, due to semi-shared weights, the high number of optimization dimensions made it computationally still expensive. That being said, we can generalize to reduce number of optimization dimensions significantly by making some assumptions:

2.3.1 Linear Scheme:

$$\mu(x_j, t_i) = \hat{\mu}x_j + m = \hat{\mu}(x_0 + j\Delta x) + m = \hat{\mu}j\Delta x + (\hat{\mu}x_0 + m);\ \hat{\mu}, m \in \mathbb{R} \quad (2.3.1a)$$

$$\sigma(x_j, t_i) = \hat{\sigma}x_j + s = \hat{\sigma}(x_0 + j\Delta x) + s = \hat{\sigma}j\Delta x + (\hat{\sigma}x_0 + s);\ \hat{\sigma}, s \in \mathbb{R} \quad (2.3.1b)$$

$$\vec{k} = \hat{k}$$

Since we really don't care much about the level provided by x_0 and m, we can bundle them together and write:

$$\hat{m} = \hat{\mu}x_0 + m$$

$$\hat{s} = \hat{\sigma}x_0 + s$$

Subsequently we can reduce the expressions into:

$$\mu(x_j, t_i) = \hat{\mu}j\Delta x + \hat{m}$$

$$\sigma(x_j, t_i) = \hat{\sigma}j\Delta x + \hat{s}$$

So, we arrive at:

$$\frac{\mu(x_t, t)}{4\Delta x} = \frac{\hat{\mu}j\Delta x + \hat{m}}{4\Delta x} = \frac{1}{4}\left(\hat{\mu}j + \frac{\hat{m}}{\Delta x}\right)$$

$$\frac{\sigma(x_t, t)^2}{\Delta x^2} = \frac{\hat{\sigma}^2 j^2\Delta x^2 + \hat{s}^2 + 2\hat{\sigma}\hat{s}j\Delta x}{\Delta x^2} = \hat{\sigma}^2 j^2 + \frac{2\hat{\sigma}\hat{s}j}{\Delta x} + \frac{\hat{s}^2}{\Delta x^2}$$

Applying results to 2.2.2a/b:

$$a_j^{i+1} = \frac{1}{4}\left[-\left(\hat{\sigma}^2 j^2 + \frac{2\hat{\sigma}\hat{s}j}{\Delta x} + \frac{\hat{s}^2}{\Delta x^2}\right) + \left(\hat{\mu}j + \frac{\hat{m}}{\Delta x}\right)\right] = \frac{1}{4}\left[\hat{\mu}j - \hat{\sigma}^2 j^2 + \frac{\hat{m} - 2\hat{\sigma}\hat{s}j}{\Delta x} - \frac{\hat{s}^2}{\Delta x^2}\right]$$

$$b_j^{i+1} = \left[\frac{1}{2}\left(\hat{\sigma}^2 j^2 + \frac{2\hat{\sigma}\hat{s}j}{\Delta x} + \frac{\hat{s}^2}{\Delta x^2}\right) - \frac{1}{\Delta t}\right]$$

$$c_j^{i+1} = \frac{1}{4}\left[-\left(\hat{\sigma}^2 j^2 + \frac{2\hat{\sigma}\hat{s}j}{\Delta x} + \frac{\hat{s}^2}{\Delta x^2}\right) - \left(\hat{\mu}j + \frac{\hat{m}}{\Delta x}\right)\right] = -\frac{1}{4}\left[\hat{\mu}j + \hat{\sigma}^2 j^2 + \frac{\hat{m} + 2\hat{\sigma}\hat{s}j}{\Delta x} + \frac{\hat{s}^2}{\Delta x^2}\right]$$

$$a_j^i = -a_j^{i+1};\ b_j^i = -b_j^{i+1} - \frac{2}{\Delta t};\ c_j^i = -c_j^{i+1} \quad (2.3.1c)$$

Thus, the problem described in 1.1.2 becomes:

$$\min_{\hat{\mu},\hat{m},\hat{\sigma},\hat{s},\hat{k}} \frac{1}{M}\left[\overrightarrow{f^0} - \overrightarrow{f^{observed}}\right]^T \left[\overrightarrow{f^0} - \overrightarrow{f^{observed}}\right] \quad (2.3.1d)$$

2.3.2 Alpha, Beta, Gamma Scheme:

$$dx_t = \mu(x_t,t)dt + \sigma(x_t,t)dW_t^{\mathbb{Q}} = \kappa\left(\theta x_t^{\alpha} - x_t^{\beta}\right)dt + \eta x_t^{\gamma} dW_t^{\mathbb{Q}}$$

Hence we have:

$$\mu(x_j,t_i) = k\left(\theta x_j^{\alpha} - x_j^{\beta}\right) = \kappa\left[\theta(x_0+j\Delta x)^{\alpha} - (x_0+j\Delta x)^{\beta}\right];\ k,\theta,\alpha,\beta \in \mathbb{R} \quad (2.3.2a)$$

$$\sigma(x_j,t_i) = \eta(x_0+j\Delta x)^{\gamma};\ \eta,\gamma \in \mathbb{R} \quad (2.3.2b)$$

So, we arrive at:

$$\frac{\mu(x_t,t)}{4\Delta x} = \frac{\kappa\left[\theta(x_0+j\Delta x)^{\alpha} - (x_0+j\Delta x)^{\beta}\right]}{4\Delta x}$$

$$\frac{\sigma(x_t,t)^2}{\Delta x^2} = \frac{\eta^2(x_0+j\Delta x)^{2\gamma}}{\Delta x^2}$$

Applying results to 2.2.2a/b:

$$a_j^{i+1} = \left[-\frac{\sigma(x_j,t_i)^2}{4\Delta x^2} + \frac{\mu(x_j,t_i)}{4\Delta x}\right] = \left[-\frac{\eta^2(x_0+j\Delta x)^{2\gamma}}{4\Delta x^2} + \frac{\kappa\left[\theta(x_0+j\Delta x)^{\alpha} - (x_0+j\Delta x)^{\beta}\right]}{4\Delta x}\right];$$

$$b_j^{i+1} = \left[\frac{\sigma(x_j,t_i)^2}{2\Delta x^2} - \frac{1}{\Delta t}\right] = \left[\frac{\eta^2(x_0+j\Delta x)^{2\gamma}}{2\Delta x^2} - \frac{1}{\Delta t}\right];$$

$$c_j^{i+1} = \left[-\frac{\sigma(x_j,t_i)^2}{4\Delta x^2} - \frac{\mu(x_j,t_i)}{4\Delta x}\right] = \left[-\frac{\eta^2(x_0+j\Delta x)^{2\gamma}}{4\Delta x^2} - \frac{\kappa\left[\theta(x_0+j\Delta x)^{\alpha} - (x_0+j\Delta x)^{\beta}\right]}{4\Delta x}\right];$$

$$a_j^i = -a_j^{i+1};\ b_j^i = -b_j^{i+1} - \frac{2}{\Delta t};\ c_j^i = -c_j^{i+1} \quad (2.3.2c)$$

Similarly, the problem described in 1.1.2 becomes:

$$\min_{k,\kappa,\theta,\eta,\alpha,\beta,\gamma} \frac{1}{M}\left[\overrightarrow{f^0} - \overrightarrow{f^{observed}}\right]^T\left[\overrightarrow{f^0} - \overrightarrow{f^{observed}}\right] \quad (2.3.2d)$$

Chapter 3. Mathematical Modeling + Machine Learning

3.1 Relationship between CNN & PDE

As it was clearly demonstrated in Chapter 1 & 2, learning a system using categorical inputs $\vec{x}$ implies that the artificial intelligence model is constructing a stochastic model of $\vec{x}$ which best explains the dynamics of stochastic system $f(x_t,t)$.

More specifically, in the case of 2.3.1, we can easily conclude that:

$$\mu(x_j,t_i) = 2\Delta x\left(c_j^i - a_j^i\right)$$

$$\sigma(x_j,t_i) = \sqrt{2\left(c_j^i + a_j^i\right)\Delta x^2} = \Delta x\sqrt{2\left(c_j^i + a_j^i\right)}$$

Hence, the A.I. model implies atomic dynamic feature of inputs follows:

$$dx_{j,t_i} = 2\Delta x\left(c_j^i - a_j^i\right)dt + \Delta x\sqrt{2\left(c_j^i + a_j^i\right)}dW_{t_i}^{\mathbb{Q}}$$

Conversely, thanks to Feynman-Kac, we can also conclude that:

Speculation 1: A convolutional neural network can be set up in a way, so that:
The solution to the A.I. model can be expressed in form of:

$$f(x_t, t) = E^{\mathbb{Q}}[f(x_T, T)|\mathcal{F}_t]$$

With:

$$\frac{\partial f}{\partial t} + \mu(x_t, t)\frac{\partial f}{\partial x} + \frac{1}{2}\sigma(x_t, t)^2\frac{\partial^2 f}{\partial x^2} = 0;$$
$$dx_t = \mu(x_t, t)dt + \sigma(x_t, t)dW_t^{\mathbb{Q}}$$

Where,

$$t \coloneqq \text{Normalized number of neuron layers, i.e. } t = 1,\ \Delta t = \frac{1}{number\ of\ layers}$$
$$x \coloneqq \text{Categorical inputs}$$

3.2 Multi-Dimensional Case

Extension of this relationship between CNN & PDE discussed here to multi-dimensional cases can also be bridged via multi-dimensional Feynman-Kac theorem:

$$d\begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} \mu_1(x_t, t) \\ \vdots \\ \mu_n(x_t, t) \end{bmatrix} dt + \begin{bmatrix} \sigma_{11}(x_t, t) & \cdots & \sigma_{m1}(x_t, t) \\ \vdots & \ddots & \vdots \\ \sigma_{n1}(x_t, t) & \cdots & \sigma_{n1}(x_t, t) \end{bmatrix} \cdot \begin{bmatrix} dW_1^{\mathbb{Q}}(t) \\ \vdots \\ dW_n^{\mathbb{Q}}(t) \end{bmatrix}$$

Then PDE of $f(\overrightarrow{x_t}, t)$ is given by:

$$\frac{\partial f}{\partial t} + \sum_{i=0}^{n} \mu_i \frac{\partial f}{\partial x_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} (\sigma\sigma^T)_{ij}\frac{\partial^2 f}{\partial x_i \partial x_j} - r(\overrightarrow{x_t}, t)f(\overrightarrow{x_t}, t) = 0$$

With boundary condition $f(x_T, T)$, and solution:

$$f(x_t, t) = E^{\mathbb{Q}}\left[e^{-\int_t^T r(x_u, u)du} f(x_T, T)|\mathcal{F}_t\right]$$

Thus, as long as the boundary condition (perception layer) is well defined, CNN can be constructed to explain the dynamics of system which is dependent upon a matrix of categorical inputs following stochastic processes.

3.3 Predication Stability & Convergence w.r.t A.I. Model Design

Through analogy of PDE, we can easily see that:

1. The relative robustness of CNN compared to other set up is rooted in the fact it implicitly constructs a "generic" PDE.
2. A straight forward implementation of CNN design is essentially an explicit scheme of PDE, which is more likely to suffer from grid size problem, i.e. $\Delta t\ vs.\ \Delta x$
3. A shallow network, while being theoretically as capable according to universal approximation theorem, tend to be less robust than deep network, as it suffers from large grid size by definition, i.e. large Δt.
4. A CNN constructed in such form essentially can be viewed as enabling machine to construct mathematical model.

3.4 Predication Stability & Convergence w.r.t A.I. Model Design

Through

3.5 Predication Stability & Convergence w.r.t A.I. Model Design

Through

3.6 Conclusion

So it is demonstrated that, by carefully setting up a convolutional neural network, not only can we extract and improve the predicting power of an A.I. model, but also infer the 'real' system dynamics.

Equivalently, we can treat such A.I. model as a way to allow machine to build mathematical model using categorical inputs, as long as the perception layer function is well defined (equivalent to boundary condition in PDE).