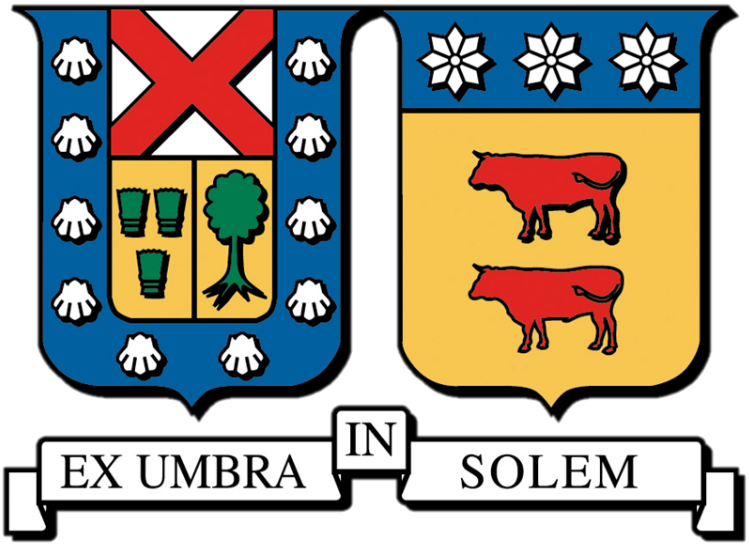

Universidad Técnica Federico Santa María
Proyecto redes neuronales avanzadas inf577

Resolución de ecuaciones diferenciales parciales vía deep learning

Mario Carlos Mallea Ruz Rol:201710515-1
mario.mallea@sansano.usm.cl

Profesor: Carlos Valle

Segundo Semestre 2021

Índice general

Capítulo 1

Introducción

En el presente trabajo se abordará la resolución de ecuaciones diferenciales parciales (EDP) utilizando una metodología de aprendizaje basado en redes neuronales, bautizado por (Sirignano and Spiliopoulos, 2018) como Deep Galerkin Method (DGM). Se expondrá en detalle el método, destacando la principal ventaja de este esquema y es que a diferencia de los métodos clásicos FEM, es libre de malla. Así, en este trabajo se resolverán EDP de diversa naturaleza de manera de estudiar y perfeccionar el método ante diferentes situaciones.

En el presente informe se presentará una breve introducción de integración de las ecuaciones diferenciales parciales y el machine learning, la revisión de trabajos relacionados, el desarrollo del método principal en detalle, presentar las cuatro ecuaciones que constituirán el marco experimental, los resultados obtenidos, un análisis general del método y finalmente las principales conclusiones y trabajo futuro.

Todos los códigos utilizados están disponibles en el siguiente repositorio.

Capítulo 2

Preliminares

2.1. Deep Learning

Motivación: Los principales desafíos de la humanidad siempre han sido el entendimiento de nuestro entorno, en particular entendernos a nosotros mismos es una tarea que esta lejos de ser resulta. Con ello ha surgido los estudios de nuestro cerebro y como es que aprendemos. En la modernidad, a través de la inteligencia artificial hemos intentado desentrañar el misterio. Sin embargo, sabemos que la inteligencia tiene muchas caras. En este trabajo usaremos la definición de inteligencia de (Mitchell, 1997), es decir diremos que un programa es inteligente o aprende de una experiencia con respecto a una tarea y una medida o performance en específico, si su desempeño en la tarea medido por la performance mejora con la experiencia.

El deep learning es una sub área del aprendizaje estadístico (o machine learning), el cual se caracteriza por modelos en profundidad llamados redes neuronales, los cuales reciben este nombre por su inspiración en el funcionamiento del sistema nervioso humano.

2.1.1. Redes Neuronales

Una red neuronal es un grafo de computación que permite transformar datos de entrada en datos de salida, es decir, implementar una función entre dos espacios (X, Y). Este grafo de computación está constituido por unidades de cálculo denominadas neuronas:

$$y_j = \phi_j \left(b_j + \sum_{i=1}^{d} w_{i,j} \mathbf{x}_i \right)$$

Matemáticamente, una neurona es el resultado de un transformación lineal por una función no lineal. En la definición, x_i representa un input (dato), $w_{i,j}$ define pesos, b_j un intercepto o sesgo, ϕ_j se denomina función de activación (usualmente no lineal), algunos ejemplos usuales son: tanh, relu, sigmoide y lineal.

En la siguiente infografía [1] vemos algunas arquitecturas populares.

Un ejemplo importante son las redes neuronales feedforward con una capa oculta (figura 2.1) las cuales se caracterizan por estar completamente conectadas.

Notar que los pesos definen si las conexiones son excitatorias o inhibitorias, el sesgo puede ser interpretado como un umbral de excitación y la activación determina el efecto de propagación de la señal.

[1] Ejemplos de arquitecturas

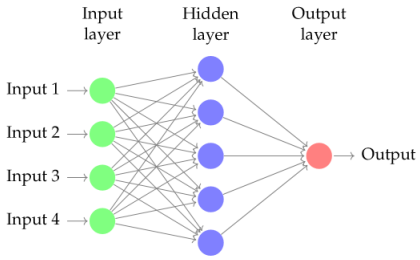

Figura 2.1: Red FF con una capa oculta

2.1.2. Aprendizaje

Estadísticamente la idea de aprender es minimizar el valor esperado de equivocarse, esto se mide a través de una función de pérdida. En virtud de la ley de los grandes números :

$$\min_{\theta} E[L(f(x), y)] = \min_{\theta} \lim_{n\to\infty} \frac{1}{n} \sum_{i} L(f(x^i), y^i) \approx \frac{1}{n} \sum_{i}^{n} L(f(x^i), y^i)$$

En un modelo neuronal, los pesos y los sesgos son los parámetros θ a encontrar. Mientras que existen otras decisiones fundamentales que son llamados hiper parámetros como por ejemplo, la arquitectura de la red (cantidad de capas y neuronas por capas), las activaciones, la loss y el tamaño de minibatch entre muchos otros. El método utilizado para encontrar los mejores parámetros que minimizen la loss es el de optimizar con descenso del gradiente estocástico (SGD) sobre sub muestras aleatorias de los datos de entrada llamados mini-batch y aprovechando el grafo de computación, las derivadas involucradas son calculadas de manera eficiente a través del método de backpropagation. En resumen:

Algorithm 1 Estimación de parámetros en una red FF

Input: Datos (x^i, y^i)
1. Definir hiper parámetros de la red
2. Minimizar el funcional de pérdida para determinar los óptimos $\boldsymbol{\theta}$:
(a) Inicializar parámetros, $\boldsymbol{\theta}_0$
(b) Muestreo aleatorio de un mini-batch de m' ejemplos de entrenamiento $\left(\mathbf{x}^i, \mathbf{y}^i\right)$
(c) Calcular la loss para este mini-batch $L\left(\boldsymbol{\theta}_i; \mathbf{x}^i, \mathbf{y}^i\right)$
(d) Calcular el gradiente $\nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}_i; \mathbf{x}^i, \mathbf{y}^i\right)$ usando backpropagation
(e) Usar la estimación del gradiente para actualizar θ_i basado en SGD:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \cdot \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}_i; \mathbf{x}^i, \mathbf{y}^i\right)$$

(f) Repetir (b) - (e) hasta que $\|\boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i\|$ sea pequeño.
Output: modelo definido por $\boldsymbol{\theta}$

En general los modelos neuronales son el estado del arte en diversos problemas de diversas áreas. Sin embargo, existe un tradeoff entre complejidad del modelo (cantidad de parámetros, principalmente debido a la profundidad y cantidad de neuronas) y el poder de generalización del modelo. Cuando este tradeoff se desequilibra de manera que los parámetros se ajustan demasiado a los ejemplos de entrenamiento, se conoce como overfitting, pero hoy en día existen diversas técnica para resolver este problema.

2.1.3. Resultados relevantes

Los mínimos entregados por el esquema de (SGD) y backpropagation son por supuesto, mínimos locales. Pero (Kawaguchi et al., 2019) demostró que si la loss es convexa y diferenciable con respecto al objetivo de la red $f(x^i)$ entonces todos los mínimos locales son globales, incluso si el objetivo de la red no es convexo con respecto a los parámetros. Intuitivamente, sucede que los mínimos de la función objetivo no son puntos aislados en el espacio de búsqueda, sino que forman un sub espacio convexo de puntos, esto producto de la alta dimensionalidad del espacio de búsqueda y la alta versatilidad de la red para reparar perturbaciones locales. Es decir a la red, le es sencillo encontrar caminos entre puntos en alta dimensionalidad.

Otro importante resultado teórico demostrado por (Cybenko, 1989) y extendido por (Hornik, 1991), es la capacidad de aprender cualquier función por parte de la red, conocido como teorema de aproximación universal. Lo que establece este teorema es que toda función continua definida en un sub conjunto compacto de $\mathbf{R}^n$ puede ser aproximada arbitrariamente bien por una red FF con solo una capa oculta. Matemáticamente se define el conjunto de aquellas redes:

Red FF de una capa oculta y n neuronas: Es una función de la forma

$$\mathfrak{C}^n = \left\{ h(t,x) : \mathbb{R}^{1+d} \mapsto \mathbb{R} : h(t,x) = \sum_{i=1}^{n} \beta_i \psi \left(\alpha_{1,i} t + \sum_{j=1}^{d} \alpha_{j,i} x_j + c_j \right) \right\}$$

Donde $\Psi : \mathbb{R} \to \mathbb{R}$ es una activación. y datos d dimensionales.

Teorema de aproximación Universal: Sea ϕ una función no trivial, acotada, monótona creciente y continua y denotamos por I_m al hipercubo unitario m-dimensional. Entonces dado $\epsilon > 0$ y cualquier función f definida en I_m, existen N, v_i, b_i, w tales que la aproximación:

$$F(x) = \sum_{i=1}^{N} v_i \phi(w \cdot x + b_i)$$

Satisface $|F(x) - f(x)| < \epsilon \quad \forall x \in I_m$.

Notar que del teorema se infiere que la aproximación es independiente de la función de activación y que no se establece la cantidad de neuronas necesarias para conseguir la convergencia.

2.2. Ecuaciones Diferenciales Parciales

Las ecuaciones diferenciales parciales son ecuaciones que definen una relación entre una función y sus derivadas, definidas en base a variables independientes, usualmente de tiempo y espacio. Esta capacidad permiten modelar fenómenos que se ven influenciados por varios factores, algunos problemas típicos son la propagación del sonido o del calor, la electrostática, la electrodinámica, la dinámica de fluidos, la elasticidad, la mecánica cuántica, entre muchos otros. Formalmente:

Definición EDP. Sea $k \geq 1$ y sea Ω un subconjunto abierto de $\mathbb{R}^d$. Una expresión de la forma:

$$F\left(D^k u(x), D^{k-1}u(x), \ldots, Du(x), u(x), x\right) = 0 \quad (x \in \Omega)$$

es llamada una ecuación diferencial parcial orden k, donde:

$$F : \mathbb{R}^{n^k} \times \mathbb{R}^{n^{k-1}} \times \cdots \times \mathbb{R}^n \times \mathbb{R} \times \Omega \to \mathbb{R}$$

es dado y $u : \Omega \to \mathbb{R}$ es desconocida.

Existen diversas clasificaciones de EDPs, en particular para este trabajo es de interés considerar:

EDP Quasi-lineal. Lineal en el mayor orden de derivabilidad con coeficientes que dependen de las derivadas de menor orden y un lado derecho que no depende del mayor grado de derivabilidad:

$$\sum_{|\alpha|=k} a_\alpha\left(D^{k-1}u, \ldots, Du, u, x\right) \cdot D^\alpha u + a_0\left(D^{k-1}u, \ldots, Du, u, x\right) = 0$$

Los modelos de machine learning han mostrado ser un aporte en la aproximación los modelos físicos modelados por EDPs, el siguiente survey Willard et al. (2020) recolecta algunos casos de éxito en varias disciplinas (referencias del survey): Tales como materials science [130, 220, 263], applied physics [15, 116], aquatic sciences [120], atmospheric science [185], biomedical science [245], computational biology [3], computational chemistry [61,70,83], oceanography [93], molecular dynamics[267], climate science [75,135,186], turbulence modeling [28,179,273]) entre otros. Por ejemplo las redes neuronales han demostrado exitosamente la capacidad de aproximacion de problemas que se consideran dificiles en fisica como quantum many body problem[42] y many electron Schrodinger equation[107]

Capítulo 3

Trabajos relacionados

Las técnicas de aprendizaje de máquinas han sido ampliamente aplicadas a diversas áreas, las cuales son modeladas con EDP, para tener una persepectiva global Willard et al. (2020) plantean una clasificación de trabajos que involucran tanto ecuaciones diferenciales como técnicas de aprendizaje estadístico. La taxonomía realizada consiste en una combinación según el objetivo de la tarea y el método utilizado. Los objetivos pueden consistir en: Mejorar la predicción detrás del modelo físico, parametrización, reducción del orden del modelo, reducción de escala, cuantificación de la incerteza, modelados/problemas inversos, descubrir la ecuación gobernante, resolver EDPs o generación de data. Los métodos se clasifican en: machine learning básico (clásico), pérdida, arquitectura, o inicialización guiada físicamente y modelos residuales o híbridos. De esta forma el presente trabajo se clasifica como uno de resolución de EDPs con técnicas básicas (clásicas) de machine learning, específicamente de deep learning.

Dentro de esta taxonomía los autores destacan los siguientes trabajos. Lagaris et al. (1998) como un trabajo pionero en la resolución de ecuaciones diferenciales con redes neuronales artificiales, en este, inspirados en el teorema de aproximación universal para un perceptron multicapa, se propone una red feed forward con una capa oculta de 10 neuronas y activación sigmoide con salida lineal optimizada con una técnica tipo brackpropagtion basada en el algoritmo de optimización cuadrático BFGS definiendo una pérdida única, es decir pre confeccionan un cambio de variable tal que todas las restricciones de la ecuación diferencial queden implícitamente definidas en ella. El estudio de esta idea se estudiará más en detalle más adelante. De esta manera se resuelven ecuaciones diferenciales ordinarias (incluyendo sistemas) y parciales, estableciendo una comparación del método de resolución para varias ecuaciones diferenciales con respecto al método tradicional de diferencias finitas, concluyendo que el método neuronal generaliza mejor en términos de error de interpolación [1] y que por otro lado, diferencias finitas presenta un mejor error en el conjunto de entrenamiento, siendo este ultimo un resultado esperable pues este método se basa en aproximar la solución en esos puntos (nodos de la malla).

Luego de este trabajo se vivió un pequeño invierno en la investigación que fue retomada con la proliferación de grandes aportes en el área del aprendizaje profundo, en particular el desarrollo de la diferenciación automática con tensorflow.

Seguido de ello, otro trabajo destacable fue Arsenault et al. (2014) el es un trabajo en el área llamada dynamical mean field theory, la cual a grandes rasgos estudia información no perturbada a partir de la estructura electrónica de materiales fuertemente correlacionados no localmente (propiedades electromagneticas inusuales). En este trabajo muestran empíricamente que es factible aproximar/ representar con un kernel approach la función en variable compleja de Green que se utiliza para describir las impurezas magnéticas

[1] Diferencia de error al considerar un mallado más fino

incrustadas en metales en el llamado Anderson impurity model.

Por otro lado, En Carleo and Troyer (2017) en el contexto del desafío planteado por el problema de muchos cuerpos en la física cuántica que se origina en la dificultad de describir las correlaciones no triviales codificadas en la complejidad exponencial de la función de onda de muchos cuerpos, demuestran que usando una representación de los estados cuánticos basado en una RBM (restricted boltzmann machine) de la ecuación de onda puede reducir su complejidad a una forma computacionalmente manejable para la descripción de equilibrio y propiedades dinámicas de prototypical intectacting spins models en una y dos dimensiones.

En Khoo et al. (2018) estudian la committor function el objeto central de la teoría de caminos (estudio de transición entre estados gobernados por procesos estocásticos) la cual satisface la ecuación de Fokker-Planck en alta dimensionalidad, en particular proponen una aproximación con una red neuronal de tres capas constituida por 6 neuronas cada una con activación tanh, formulada desde la forma variacional correspondiente, como un método de Galerkin no lineal, y teniendo en consideración las singularidades del dominio que puede presentar en el diseño de la arquitectura.

En Rudd and Ferrari (2014) proponen un esquema de resolución que combina los métodos clásicos del tipo Galerkin con un entrenamiento backpropagation con restricciones para obtener aproximaciones a ciertas integrales involucradas en el método clásico, que termina en la resolución de un sistemas de edos, el cual consiste en un backpropagation incremental tal que equitativamente una a través de una red neruronal una partición del conjunto de pesos es ajustado para preservar un prior Gaussian RBFs y la otra partición del conjunto de pesos es ajustado para asimilar nuevos datos via backpropagation. Los autores destacan su aplicación a dominios irregulares y muestran una significante mejora versus los métodos de elementos finitos más eficientes tanto en tiempo de computación como en precisión.

Finalmente el survey destaca también al trabajo base de este proyecto Sirignano and Spiliopoulos (2018) el cual se desarrollará en detalle más adelante.

La aplicación de métodos de deep learning para la resoluciones de EDPs es muy diverso con respecto a las áreas de aplicación. Existen diversas propuestas de pérdidas, arquitecturas y inicializaciones especificas que consideran propiedades o métodos ya diseñados para el estudio de dicha ecuación diferencial. De esta manera también han surgido librerías que buscan disponer solvers basados en deep learning de modo que sean listos para usar, por ejemplo Koryagin et al. (2019) crearon pydens una librería basada en DGM y otros métodos conocidos de forma de ser una librería lista para el uso en phyton que utiliza Batcch flow (un open source framework para la conveniente reproducibilidad de deep learning). Sin embargo, durante este trabajo no se tendrá en consideración la utilización de librerías pues la idea principal es el estudio del método en si mismo.

Para testear el problema en uno de alta dimensionalidad se utilizara como base el trabajo Raissi (2018) el cual aproxima una ecuación estocástica, que su aplicación es principalmente en areas de control estocastico, teoria economica y matematica financiera, este tipo de ecuación puede demostrarse que es equivalente a una PDE parabólica quasi lineal de 100 dimensiones por medio de una red de 5 capas FF con 256 neuronas por capa entrenando en base a 100 realizaciones del movimiento browniano correspondiente a la ecuación y pérdida definida a través de una discretización de tipo Euler-Maruyama. Destacan su aceptable capacidad de aproximación para todo tiempo, lo cual es una ventaja sobre los metodo clásicos que suelen aproximar bien solo la condición para tiempo inicial.

Capítulo 4

Deep Galerkin

Lo que motiva resolver las EDP con redes neuronales es principalmente, la alta dificultad del método de Galerkin para problemas en alta dimensionalidad. Pues la idea de establecer un mallado en un espacio de por ejemplo 200 dimensiones es inviable en memoria. Así (Sirignano and Spiliopoulos, 2018) propone el método libre de malla que bautiza como Deep Galerkin Method (DGM) que consiste en aproximar la solución de una EDP a través de una red neuronal. Como vimos en el capitulo anterior, es fundamental principalmente definir la loss que corrigirá la función a aprender y la arquitectura. Para los datos de entrenamiento, la red usa muestras aleatorias del dominio de la función. En particular, se utilizarán mini-batch (sub muestras) de manera secuencial de diferentes partes del dominio. Así el método evita la maldición de la alta dimensionalidad pues muestrea observaciones y no discretiza todo el espacio.

4.0.1. Modelamiento matemático

La forma general de las EDP que resolveremos en este trabajo se pueden escribir de manera general como sigue:

Sea u una función desconocida definida en tiempo y espacio sobre la región $[0,T] \times \Omega$ con $\Omega \subset \mathbf{R}^d$ y asumimos que u satisface la EDP:

$$\begin{cases} (\partial_t + L)\, u(t,\boldsymbol{x}) = 0, & (t,\boldsymbol{x}) \in [0,T] \times \Omega \\ u(0,\boldsymbol{x}) = u_0(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \\ u(t,\boldsymbol{x}) = g(t,\boldsymbol{x}), & (t,\boldsymbol{x}) \in [0,T] \times \partial\Omega \end{cases}$$

El objetivo es aproximar u con una función $f(t,\boldsymbol{x};\boldsymbol{\theta})$ con los parámetros aprendidos desde una red neuronal.

La loss utilizada será de forma que la aproximación cumpla el operador diferencial, la condición de borde y la condición inicial:

$$L(\boldsymbol{\theta}) = \underbrace{\|(\partial_t + L)\, f(t,\boldsymbol{x};\boldsymbol{\theta})\|^2_{[0,T]\times\Omega,\nu_1}}_{\text{operador diferencial}} + \underbrace{\|f(t,\boldsymbol{x};\boldsymbol{\theta}) - g(t,\boldsymbol{x})\|^2_{[0,T]\times\partial\Omega,\nu_2}}_{\text{condición de borde}} + \underbrace{\|f(0,\boldsymbol{x};\boldsymbol{\theta}) - u_0(\boldsymbol{x})\|^2_{\Omega,\nu_3}}_{\text{condición inicial}}$$

Donde los términos de error se miden con una norma tipo L^2:

$$\|h(y)\|^2_{Y,\nu} = \int_Y |h(y)|^2 \nu(y) dy$$

Donde $\nu(y)$ denota a una densidad de probabilidad en la región Y, la cual equivale a la distribución usada para muestrear el espacio. Además, la minimización de la loss será como es usual en las redes neuronales, vía SGD y backpropagation fluyendo en la arquitectura.

Algorithm 2 Deep Galerkin con minibatch adaptativo

1. Inicializar los parámetros θ_0 y otros hiper parámetros como la tasa de aprendizaje α_n.
2. **Para cada época:**
   2.1 Generar muestras aleatorios del interior del dominio y fronteras espacio/temporales:
   - Generar (t_n, x_n) de $[0,T] \times \Omega$ según ν_1
   - Generar (τ_n, z_n) de $[0,T] \times \partial\Omega$ según ν_2
   - Generar w_n de Ω, según ν_3

   2.2. **Para k pasos de descenso:**
   2.2.1 Calcular la loss para el mini-batch correspondiente $s_n = \{(t_n, x_n), (\tau_n, z_n), w_n\})$:
   - Calcular $L_1(\boldsymbol{\theta}_n; t_n, x_n) = \left((\partial_t + L) f(\boldsymbol{\theta}_n; t_n, x_n)\right)^2$
   - Calcular $L_2(\boldsymbol{\theta}_n; \tau_n, z_n) = \left(f(\tau_n, z_n) - g(\tau_n, z_n)\right)^2$
   - Calcular $L_3(\boldsymbol{\theta}_n; w_n) = \left(f(0, w_n) - u_0(w_n)\right)^2$
   - Calcular $L(\boldsymbol{\theta}_n; s_n) = L_1(\boldsymbol{\theta}_n; t_n, x_n) + L_2(\boldsymbol{\theta}_n; \tau_n, z_n) + L_3(\boldsymbol{\theta}_n; z_n)$

   2.2.2. Tomando un paso de descenso en los puntos aleatorios s_n con paso tipo Adam* :

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_n; s_n)$$

   2.3 **Repetir**: Adaptando el minibatch hacia zonas donde la loss no disminuye lo suficiente. Hasta que $\|\theta_{n+1} - \theta_n\|$ converja.

Output: aproximación f (red) definido por $\boldsymbol{\theta}$

*: Método de tasa de aprendizaje adaptiva , que considera medias móviles e información de primer orden, en el cual los pasos o se acumulan o se pueden revertir.

Notar que DGM al igual que los problemas de machine learning, es mucho más allá del problema de optimización. Por ejemplo surge la pregunta de generalización (lo que es una teoría completa). En nuestro problema significa preguntarse, si optimizar la loss a cero nos asegura que f es una buena aproximación de la solución de la EDP, y sobre todo, los parámetros encontrados, se ajustan solamente a la muestra de puntos de entrenamiento o permite aproximar bien a puntos que no fueron vistos por la red?. En particular, estas preguntas son relevantes en problemas donde la EDP está definida en dominios no acotados, lo que imposibilita muestrear en todos lados.

Estas preguntas serán abordadas en detalle más adelante.

4.0.2. Arquitectura

La arquitectura propuesta por (Sirignano and Spiliopoulos, 2018) es similar a una red LSTM, las cuales son arquitecturas para resolver problemas de sequencias, como predicción en series de tiempo o predicción de texto (cuando los teclados del celular sugieren lo que escribirás a continuación). En nuestro caso, la arquitectura consiste en 3 capas, a las cuales les llamaremos capas tipo DGM. Destacar que esto es una propuesta y es fácilmente extensible a más capas y cada de una de ellas, es fácilmente adaptable para incorporar más información a priori.

En la figura 4.1 podemos ver el esquema general descrito. Notar que nuestro x sería el conjunto de muestras aleatorias en espacio/tiempo del dominio. Cada capa DGM se conecta con la anterior y con x, esto le da la propiedad de refuerzo por memoria a la red, pues en ningún momento olvida el input. El vector de salida y

es la aproximación de u producida por la red evaluada en cada uno de los mini-batches o sub muestras.

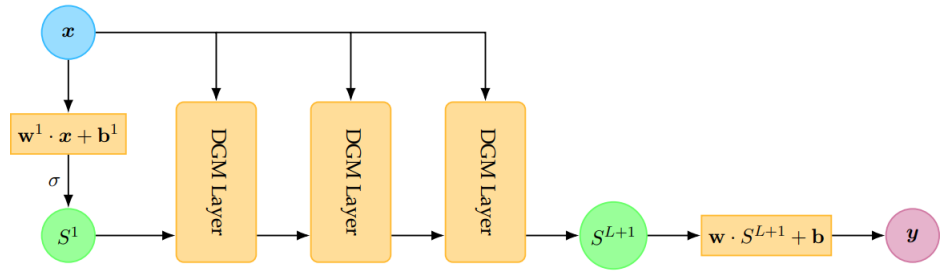

Figura 4.1: Arquitectura DGM

Cada capa DGM aplica una serie de transformaciones a x y a la salida de la capa anterior, como se ve en la figura 4.2. De manera similar a lo que hace un red neuronal LSTM, intuitivamente cada sub capa $\ell = 1, \ldots, L$ (en la 4.1, la cantidad de capas DGM es L=3) determina cuanta información debe ser traspasa a la siguiente, evitando problemas de desvanecimiento del gradiente, y también, la seguidilla de multiplicaciones element wise (multiplicar matrices elemento a elemento $\odot$) de funciones no lineales permite aprender funciones cada vez más complejas. Finalmente obtenemos la aproximación $f(t, \boldsymbol{x}; \boldsymbol{\theta})$ como otra transformación lineal de la salida S^{L+1}, de manera de darle el último tuneo a la aproximación.

Las subcapas (cuadros naranjas en 4.2) involucradas dentro de cada capa DGM:

$$\begin{aligned}
S^1 &= \sigma\left(\mathbf{w}^1 \cdot \boldsymbol{x} + \mathbf{b}^1\right) \\
Z^\ell &= \sigma\left(\mathbf{u}^{z,\ell} \cdot \boldsymbol{x} + \mathbf{w}^{z,\ell} \cdot S^\ell + \mathbf{b}^{z,\ell}\right) && \ell = 1, \ldots, L \\
G^\ell &= \sigma\left(\mathbf{u}^{g,\ell} \cdot \boldsymbol{x} + \mathbf{w}^{g,\ell} \cdot S^\ell + \mathbf{b}^{g,\ell}\right) && \ell = 1, \ldots, L \\
R^\ell &= \sigma\left(\mathbf{u}^{r,\ell} \cdot \boldsymbol{x} + \mathbf{w}^{r,\ell} \cdot S^\ell + \mathbf{b}^{r,\ell}\right) && \ell = 1, \ldots, L \\
H^\ell &= \sigma\left(\mathbf{u}^{h,\ell} \cdot \boldsymbol{x} + \mathbf{w}^{h,\ell} \cdot \left(S^\ell \odot R^\ell\right) + \mathbf{b}^{h,\ell}\right) && \ell = 1, \ldots, L \\
S^{\ell+1} &= \left(1 - G^\ell\right) \odot H^\ell + Z^\ell \odot S^\ell && \ell = 1, \ldots, L \\
f(t, \boldsymbol{x}; \boldsymbol{\theta}) &= \mathbf{w} \cdot S^{L+1} + \mathbf{b}
\end{aligned}$$

El espacio parámetrico es:

$$\theta = \left\{\mathbf{w}^1, \mathbf{b}^1, \left(\mathbf{u}^{z,\ell}, \mathbf{w}^{z,\ell}, \mathbf{b}^{z,\ell}\right)_{\ell=1}^{L}, \left(\mathbf{u}^{g,\ell}, \mathbf{w}^{g,\ell}, \mathbf{b}^{g,\ell}\right)_{\ell=1}^{L}, \left(\mathbf{u}^{r,\ell}, W^{r,\ell}, \mathbf{b}^{r,\ell}\right)_{\ell=1}^{L}, \left(\mathbf{u}^{h,\ell}, \mathbf{w}^{h,\ell}, \mathbf{b}^{h,\ell}\right)_{\ell=1}^{L}, \mathbf{w}, \mathbf{b}\right\}.$$

Como cada una de las sub capas contienen M neuronas de cálculo (osea, la operación se repite M veces) se tiene que la activación $\sigma : \mathbb{R}^M \to \mathbb{R}^M$ es una no linealidad ϕ element wise:

$$\sigma(z) = \left(\phi(z_1), \phi(z_2), \ldots, \phi(z_M)\right)$$

Por lo que las parámetros involucrados para un problema con d dimensiones espaciales, tienen tamaño:

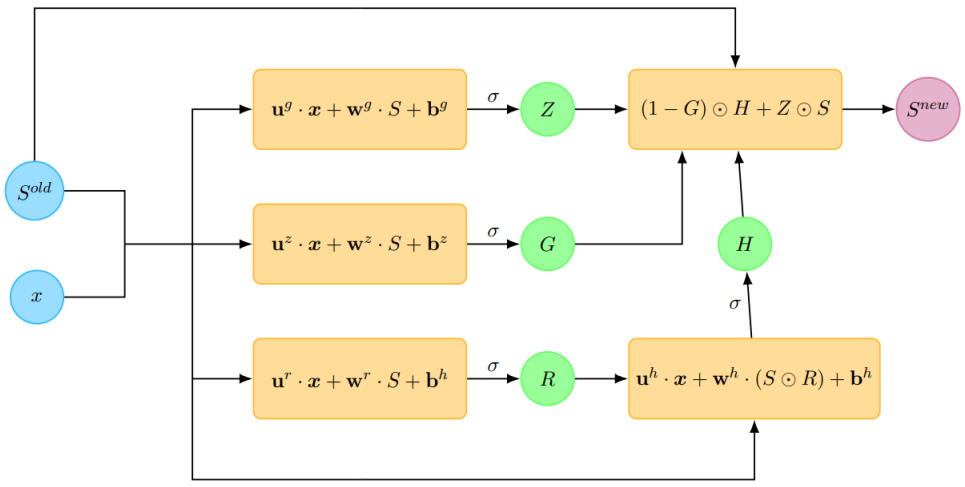

Figura 4.2: Arquitectura de una capa DGM

$$
\begin{aligned}
&\mathbf{w}^1 \in \mathbb{R}^{M\times(d+1)}, \mathbf{b}^1 \in \mathbb{R}^M \ , \\
&\mathbf{u}^{z,\ell} \in \mathbb{R}^{M\times(d+1)}, \mathbf{w}^{z,\ell} \in \mathbb{R}^{M\times M}, \mathbf{b}^{z,\ell} \in \mathbb{R}^M \ , \\
&\mathbf{u}^{g,\ell} \in \mathbb{R}^{M\times(d+1)}, \mathbf{w}^{g,\ell} \in \mathbb{R}^{M\times M}, \mathbf{b}^{g,\ell} \in \mathbb{R}^M \ , \\
&\mathbf{u}^{r,\ell} \in \mathbb{R}^{M\times(d+1)}, \mathbf{w}^{r,\ell} \in \mathbb{R}^{M\times M}, \mathbf{b}^{r,\ell} \in \mathbb{R}^M \ , \\
&\mathbf{u}^{h,\ell} \in \mathbb{R}^{M\times(d+1)}, \mathbf{w}^{h,\ell} \in \mathbb{R}^{M\times M}, \mathbf{b}^{h,\ell} \in \mathbb{R}^M \ , \\
&\mathbf{w} \in \mathbb{R}^{1\times M}, \mathbf{b} \in \mathbb{R}
\end{aligned}
$$

Es por ello que tiene sentido hablar de un esquema tipo Galerkin, pues finalmente la aproximación es una combinación de una base de funciones más sencillas, pero que con el enfoque deep, resulta ser la combinación ($\odot$) de activaciones no lineales (σ) de transformaciones lineales aprendidas adecuadamente según una arquitectura con parámetros $(\mathbf{u}, \mathbf{w}, \mathbf{b})$, osea, en base a una transformación lineal u de las sub muestras del dominio x, una transformación lineal w de su estado de cálculo anterior S y un sesgo b.

4.0.3. Resultados relevantes

Similarmente al teorema de aproximación universal. Los autores (Sirignano and Spiliopoulos, 2018) demostraron que la aproximación vía DGM converge a la solución de la EDP (para el caso en cual la EDP es del tipo parabólicas y quasi lineal) cuando el numero de neuronas de las capas ocultas tiende a infinito.

Recordemos que el objetivo de la red es minimizar la loss. La cual constituye una aproximación del operador diferencial, la condición de borde e inicial provenientes de la EDP a resolver. Entonces lo que se demostrará es que la red cumple su objetivo, es decir que: Existe $f^n \in \mathfrak{C}^n$ tal que $L(f) \to 0$ cuando $n \to \infty$. Pero es más, seguido de ello se demostrará que aquella aproximación converge de la siguiente manera a la solución u de la EDP: $f^n \to u$ cuando $n \to \infty$ fuertemente en $L^\rho([0,T]\times\Omega)$, con $\rho < 2$

Convergencia de $L(f)$ en norma L^2

Consideremos el conjunto acotado $\Omega \subset \mathbb{R}^d$ con borde suave $\partial\Omega$ y denotamos $\Omega_T = (0,T] \times \Omega$ y $\partial\Omega_T = (0,T] \times \partial\Omega$. Consideramos la clase de EDP parabólicas quasi lineales de la forma:

$$\begin{aligned} \partial_t u(t,x) - \operatorname{div}(\alpha(t,x,u(t,x),\nabla u(t,x))) + \gamma(t,x,u(t,x),\nabla u(t,x)) &= 0, \text{ for } (t,x) \in \Omega_T \\ u(0,x) &= u_0(x), \text{ for } x \in \Omega \\ u(t,x) &= g(t,x), \text{ for } (t,x) \in \partial\Omega_T \end{aligned}$$

Definimos:

$$\hat{\gamma}(t,x,u,p) = \gamma(t,x,u,p) - \sum_{i=1}^{d} \frac{\partial \alpha_i(t,x,u,p)}{\partial u} \partial_{x_i} u - \sum_{i=1}^{d} \frac{\partial \alpha_i(t,x,u,p)}{\partial x_i}$$

Además asumiremos que la EDP descrita tiene una única solución de la forma:

$$u(t,x) \in C\left(\bar{\Omega}_T\right) \bigcap C^{1+\eta/2,2+\eta}\left(\Omega_T\right) \text{ con } \eta \in (0,1) \text{ y tal que } \sup_{(t,x)\in\Omega_T} \sum_{k=1}^{2} \left|\nabla_x^{(k)} u(t,x)\right| < \infty$$

Teorema: Sea $\mathfrak{C}^n(\psi)$ con activación sigmoide, osea $\psi = \dfrac{e^x}{1+e^x} \in C^2\left(\mathbb{R}^d\right)$, acotada y no constante. Sea $\mathfrak{C}(\psi) = \bigcup_{n\geq 1} \mathfrak{C}^n(\psi)$. Asumimos que Ω_T es compacto y consideramos las medidas ν_1, ν_2, ν_3 con soporte definido en $\bar{\Omega}_T, \Omega$ y $\partial\Omega_T$, respectivamente. Además, asumimos que la EDP descrita tiene una única solución u de la forma anterior. También, asumimos que los términos no lineales $\frac{\partial \alpha_i(t,x,u,p)}{\partial p_j}$ y $\hat{\gamma}(t,x,u,p)$ son localmente C-Lipschitz en (u,p) con C tal que puede tener a lo más crecimiento polinomial con respecto a u y p, e uniforme con respecto a t y x. Entonces, para todo $\epsilon > 0$, existe una constante positiva $K > 0$ que depende de $\sup_{\Omega_T} |u|, \sup_{\Omega_T} |\nabla_x u|$ y $\sup_{\Omega_T} \left|\nabla_x^{(2)} u\right|$ tal que existe una función $f \in \mathfrak{C}(\psi)$ que satisface

$$J(f) \leq K\epsilon$$

Convergencia a la solución de la EDP

Consideremos la EDP parabólica quasilineal antes descrita pero con $g(t,x) = 0$ es decir la versión homogenea en la frontera y denotamos por $G[.]$ al operador diferencial, entonces asumiremos que cada f^n satisface la EDP:

$$\begin{aligned} G\left[f^n\right](t,x) &= h^n(t,x), \text{ for } (t,x) \in \Omega_T \\ f^n(0,x) &= u_0^n(x), \text{ for } x \in \Omega \\ f^n(t,x) &= g^n(t,x), \text{ for } (t,x) \in \partial\Omega_T \end{aligned}$$

Para algunos h^n, u_0^n, y g^n tales que satisfacen:

$$\|h^n\|_{2,\Omega_T}^2 + \|g^n\|_{2,\partial\Omega_T}^2 + \|u_0^n - u_0\|_{2,\Omega}^2 \to 0 \text{ cuando } n \to \infty$$

Además se usarán algunas condiciones adicionales:

- Existe una constante $\mu > 0$ y funciones positivas $\kappa(t,x), \lambda(t,x)$ tal que para todo $(t,x) \in \Omega_T$ se tiene

$$\|\alpha(t,x,u,p)\| \leq \mu(\kappa(t,x) + \|p\|), \text{ y } |\gamma(t,x,u,p)| \leq \lambda(t,x)\|p\|$$

  con $\kappa \in L^2\left(\Omega_T\right), \lambda \in L^{d+2+\eta}\left(\Omega_T\right)$ para algún $\eta > 0$

- $\alpha(t,x,u,p)$ y $\gamma(t,x,u,p)$ son Lipschitz continuas en $(t,x,u,p) \in \Omega_T \times \mathbb{R} \times \mathbb{R}^d$ uniformemente en compactos de la forma $\left\{(t,x) \in \bar{\Omega}_T, |u| \leq C, |p| \leq C\right\}$.
- $\alpha(t,x,u,p)$ es diferenciable con respecto a (x,u,p) con derivadas continuas.
- Existe una constante positiva $\nu > 0$ tal que:

$$\alpha(t,x,u,p)p \geq \nu|p|^2$$

y

$$\langle \alpha(t,x,u,p_1) - \alpha(t,x,u,p_2), p_1 - p_2 \rangle > 0, \text{ para todo } p_1, p_2 \in \mathbb{R}^d, p_1 \neq p_2$$

- $u_0(x) \in C^{0,2+\xi}(\bar{\Omega})$ para algún $\xi > 0$* tal que tanto su derivada como si mismo sean acotados sobre $\bar{\Omega}$.

  *: En general el espacio de Hölder $C^{0,\xi}(\bar{\Omega})$ es el espacio de Banach de funciones continuas en $\bar{\Omega}$ teniendo derivadas continuas sobre el orden $[\xi]$ en $\bar{\Omega}$ con normas uniformes correspondientes finitas y $\xi - [\xi]$ norma Hölder uniforme y finita. Análogamente, también definimos el espacio de Hölder $C^{0,\xi,\xi/2}\left(\bar{\Omega}_T\right)$ el cual además tiene finitos $[\xi]/2$ y $(\xi - [\xi])/2$ ordenes de regularidad y normas Hölder finitas de derivadas en tiempo respectivamente.
- Ω es un sub conjunto acotado y abierto de $\mathbb{R}^d$ con frontera $\partial\Omega \in C^2$.
- Para todo $n \in \mathbb{N}$, $f^n \in C^{1,2}\left(\bar{\Omega}_T\right)$. Además, $(f^n)_{n\in\mathbb{N}} \in L^2(\Omega_T)$.

Teorema: Asumimos las condiciones adicionales y las de convergencia de h^n, u_0^n, y g^n. Entonces, el problema descrito (homogéneo) tiene una única solución acotada en $C^{0,\delta,\delta/2}\left(\bar{\Omega}_T\right) \cap L^2\left(0,T; W_0^{1,2}(\Omega)\right) \cap W_0^{(1,2),2}(\Omega_T')$* para algún $\delta > 0$ y todo subdominio interior Ω_T' de Ω_T. Además, f^n converge a u la solución única del problema, fuertemente en $L^\rho(\Omega_T)$ para todo $\rho < 2$. Si, también, la sucesión $\{f^n(t,x)\}_{n\in\mathbb{N}}$ es uniformemente acotada en n y equicontinua entonces la convergencia a u es uniforme en Ω_T.

* : $W_0^{(1,2),2}(\Omega_T')$ denota el espacio de Banach el cual es clausura de $C_0^\infty(\Omega_T')$ con elementos de $L^2(\Omega_T')$ teniendo derivadas generalizadas de la forma $D_t^r D_x^s$ con r, s tales que $2r + s \leq 2$ con la norma Sobolev usual.

Capítulo 5

Implementaciones

En esta sección se presentarán cuatros EDPs de diversa complejidad, que conformarán la base de experimentación para el análisis del método DGM.

5.1. Black Scholes 1D

En finanzas cuantitativas son reconocidas la Black-Scholes EDPs introducida en 1973 para resolver el precio de varios derivados financieros. El estilo europeo afirma que los instrumentos financieros son escritos sobre un fuente de incertidumbre con una liquidación que depende del nivel del subyacente* en una fecha de vencimiento predeterminada.

*: Es un instrumento financiero cuyo precio sirve como referencia para pactar acuerdos entre dos o más participantes.

Una de las principales características es que se asume un modelo Black-Scholes de mercado simplificado, lo que repercute en una EDP lineal, donde el riesgo de un activo sigue un movimiento Browniano y tiene una cuenta en un banco sin riesgo Al-Aradi et al. (2018).

One dimensional Black-Scholes PDE (European-Style Derivate)

$$\begin{cases} \partial_t g(t,x) + rx \cdot \partial_x g(t,x) + \frac{1}{2}\sigma^2 x^2 \cdot \partial_{xx} g(t,x) = r \cdot g(t,x) \\ g(T,x) = G(x) = max\{0, x-K\} \end{cases}$$

Solución:

$$g(t,x) = x\Phi\left(d_+\right) - Ke^{-r(T-t)}\Phi\left(d_-\right)$$

$$\text{donde} \quad d_\pm = \frac{\ln(x/K) + \left(r \pm \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}$$

El objetivo es determinar *el precio de compra* (también conocido como *precio de rescate*) el cual es el precio al que el emisor de un valor exigible tiene derecho a recomprarlo a un inversor o acreedor.

El problema a resolver considera: Una tasa de interés $r = 5\,\%$, volatilidad $\sigma = 25\,\%$, tiempo de vencimiento $T = 1$ y precio de ejercicio $K = 50$.

Entonces para resolver los autores Al-Aradi et al. (2018), primero muestrean el dominio con una distribución *lognormal* uniformemente durante el tiempo (puesto que se sabe que esta es la distribución que sigue el precio de las acciones en este modelo) y también se muestrea con distribución uniforme durante el tiempo final.

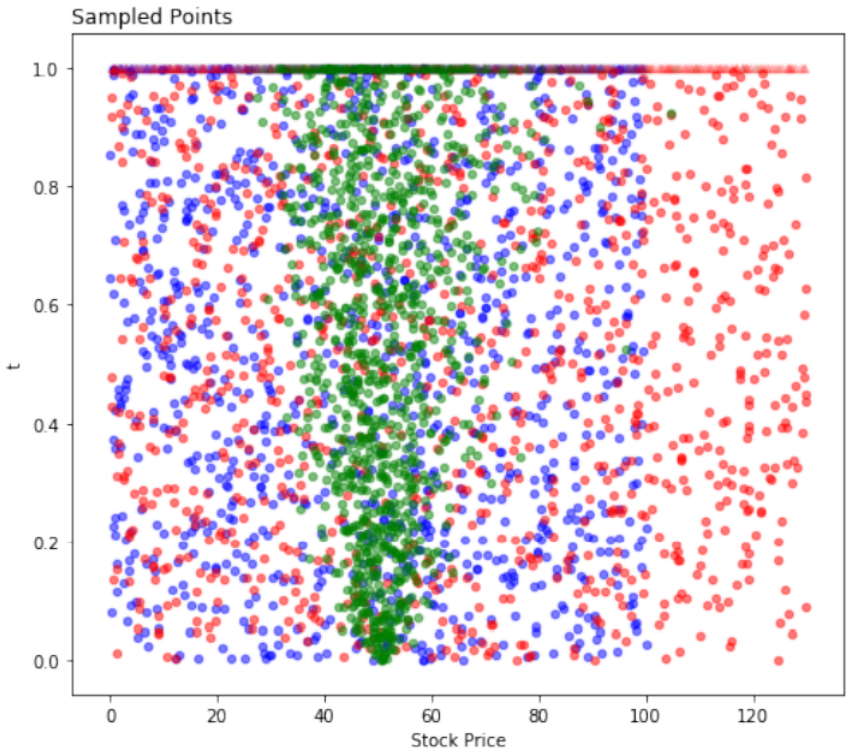

Figura 5.1: Diferentes esquemas de muestro: lognormal(verde), uniforme en $[0,1]\times[0,100]$ (azul) y uniforme en $[0,1]\times[0,130]$ (rojo)

Dado que con este esquema aparecieron problemas en los menores precios en todo tiempo, se volvió a las técnicas propuestas originalmente por (Sirignano and Spiliopoulos, 2018) y muestrear de manera uniforme la región $[0,1]\times[0,100]$. Lo cual mejoro el entrenamiento, pero presentando pequeños problema al inicio del tiempo para los mayores precios. Para mejorar aquello acertadamente los autores (Al-Aradi et al., 2018) extendieron la región con muestreo uniforme a $[0,1]\times[0,130]$. Un diagrama de esquemas de muestreo se ven en 5.1 y los resultados se presentan en 5.12.

Destacar que la loss según una metodología de reducción del riesgo empírico viene dada por $L = L1 + L2$, con:

$$L_1(\boldsymbol{\theta}) = \sum_{i=1}^{n_1}(NN_t(x,t;\boldsymbol{\theta}) + \frac{1}{2}\sigma^2 NN(x,t;\boldsymbol{\theta})^2 NN_{xx}(x,t;\boldsymbol{\theta}) + rxNN_x(x,t;\boldsymbol{\theta}))^2$$

$$L_2(\boldsymbol{\theta}) = \sum_{i=1}^{n_2}(max\{0,x-K\} - NN(x,T))^2$$

Con $(x,t) \sim (U[0,2K], U[0,1])$

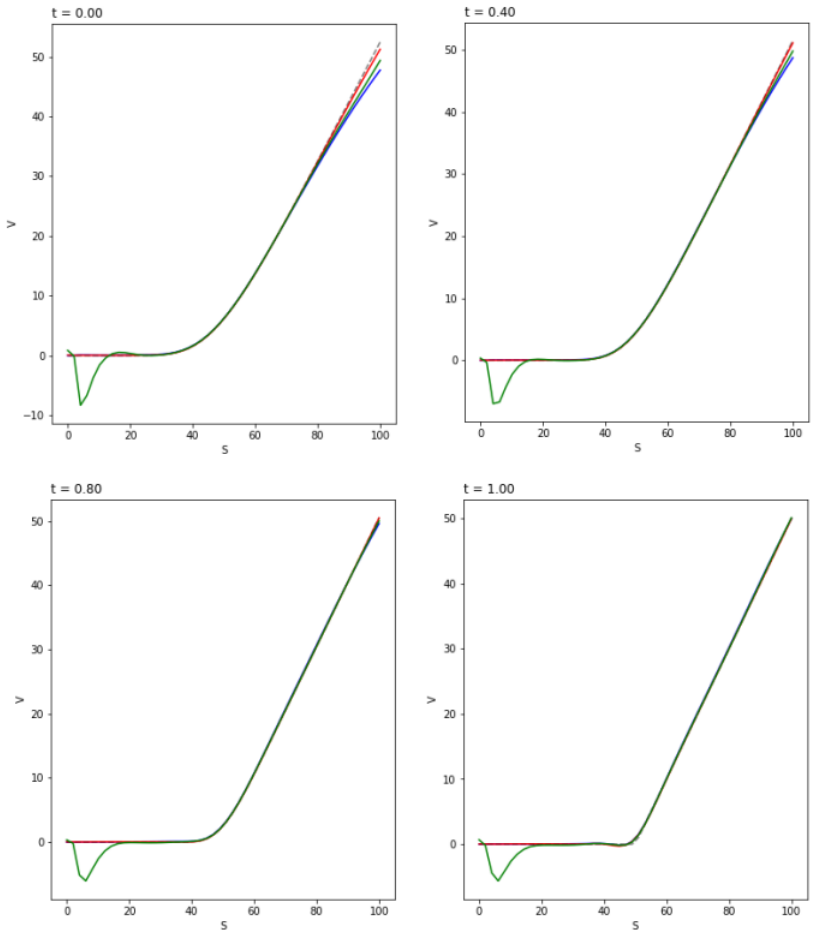

Figura 5.2: Precio de compra en función del precio del activo: La linea negra punteada es la solución real calculada con la formulas de Black-Scholes. Los colores corresponden a los esquemas de muestreo

Al realizar experimentos, de manera eficiente, la loss se reduce y por tanto es esperable por los teoremas de aproximación, que se esté aproximando la solución de la ecuación diferencial parcial:

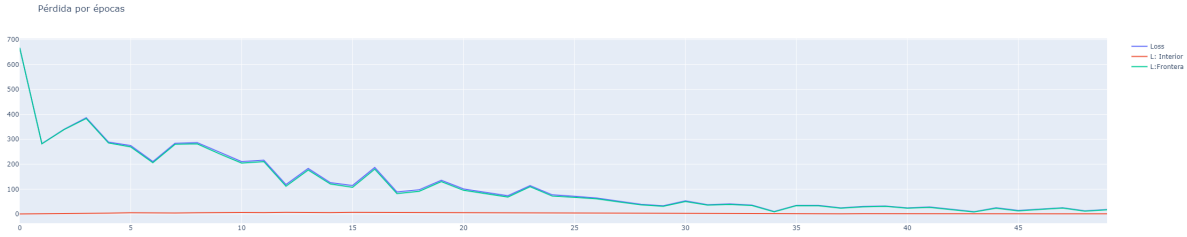

Figura 5.3: Pérdida por épocas

Se puede notar como la pérdida en la condición de frontera es una condición que le cuesta más que el interior, atribuible principalmente a que estas condiciones deben de sincronizarse para definir una única solución.

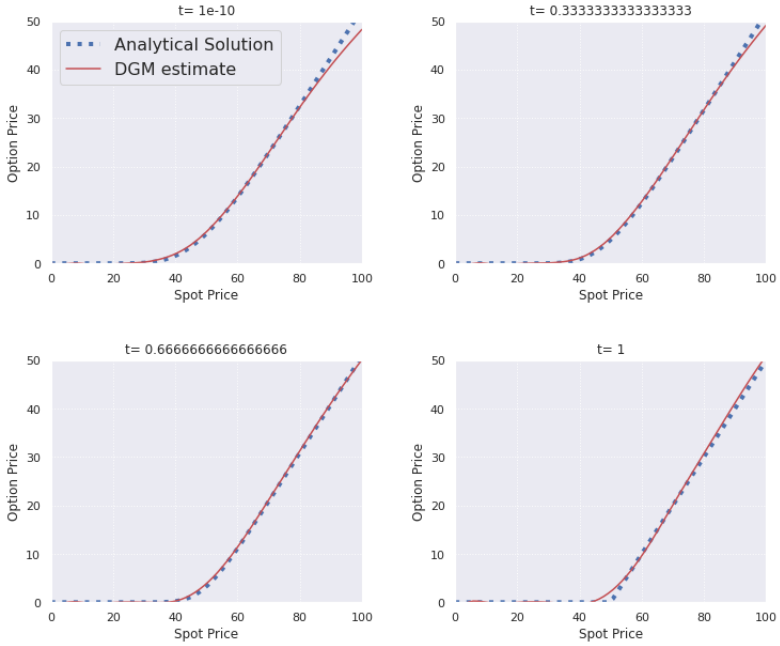

Figura 5.4: Comparación de aproximación con solución analítica

En las aproximaciones se puede notar como los principales errores están asociados a las condiciones de frontera, para tiempo inicial en 100 y tiempo final en 50. Dicha situación se puede ver también en el mapa de calor de errores.

Figura 5.5: Mapa de calor para test

Con respecto a las distribuciones de errores, se puede notar como el método generaliza bien, al tener un error menor y más concentrado en test (malla uniforme). Mientras que el error en train, que como evalúa en cada muestreo con el cual se entreno la red, existen muchas más muestras y por eso la variabilidad es mayor.

Por el lado de la modificación del minibatch en el entrenamiento, se puede lograr una mejora en la

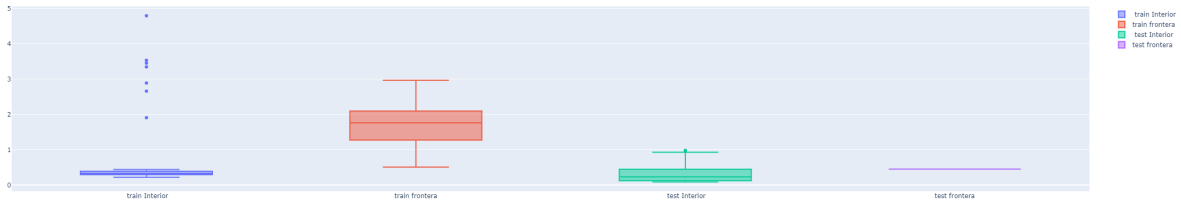

Figura 5.6: Aproximaciones obtenidas

estimación de algunos gradientes reforzando el muestreo en ciertas zonas donde la loss es más débil. Sin embargo, hay que considerar que dichas mejoras puede lograrse de manera menos eficiente con el método original.

RMSEt \ muestreo	**Normal**	**Adaptativo**
Interior	3.1	0.32
T final	0.71	0.45

5.2. Heat equation 2D

La ecuación del calor es una importante EDP del tipo parabólica que describe la distribución del calor (o variaciones de la temperatura) en una región a lo largo del transcurso del tiempo. Se utilizarán las siguiente configuración dada por una librería(Fenics) de elementos finitos:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u + f && \Omega \times (0,T] \\ u &= u_{\mathrm{D}} && \partial\Omega \times (0,T] \\ u &= u_0 && t = 0 \end{aligned}$$

Para $f = \beta - 2 - 2\alpha$, $u_D = 1 + x^2 + \alpha y^2 + \beta t$, $u_0 = 1 + x^2 + \alpha y^2$

La ecuación tiene solución $u = 1 + x^2 + \alpha y^2 + \beta t$ para $T = 2, \alpha = 3, \beta = 1{,}2$ y $\Omega = [0,1]^2$

Destacar que la loss según una metodología de reducción del riesgo empírico viene dada por $L = L1 + L2 + L3$, con:

$$L_1(\boldsymbol{\theta}) = \sum_{i=1}^{n_1}(NN_t(X,t;\boldsymbol{\theta}) - NN_{xx}(X,t;\boldsymbol{\theta}) - NN_{yy}(X,t;\boldsymbol{\theta}) - f)^2 \qquad (X,t) \sim (U(\Omega), U[0,T])$$

$$L_2(\boldsymbol{\theta}) = \sum_{i=1}^{n_2}(NN(X,t) - u_D(X,t))^2 \qquad (X,t) \sim (U(\partial\Omega), U[0,T])$$

$$L_3(\boldsymbol{\theta}) = \sum_{i=1}^{n_3}(NN(X,0) - u_0(X))^2 \qquad X \sim U(\Omega)$$

Experimentalmente, se puede observar como de manera eficiente se reduce la loss, aproximando la solución con un alto grado de precisión.

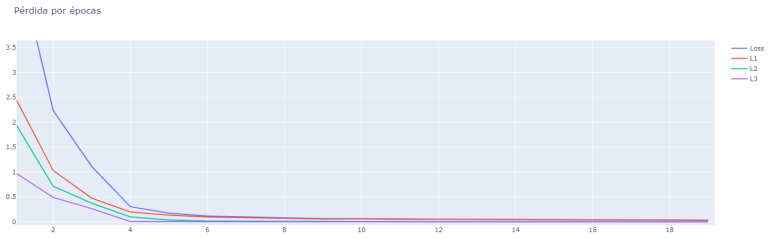

Figura 5.7: Pérdida por épocas

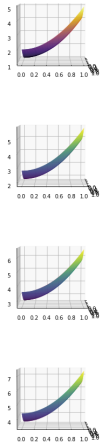

Figura 5.8: Comparación entre aproximación y solución analítica

Presentando en fase de test los siguientes errores. Nuevamente hay que considerar que la disminución del error dado por la metodología adaptativa se podría lograr también con el muestro normal pero con más épocas de entrenamiento.

5.3. Wave equation 2D

La ecuación de onda es una EDP lineal de segundo orden para la descripción de ondas, como ocurren en la física clásica , como las ondas mecánicas (por ejemplo , ondas de agua , ondas de sonido y ondas sísmicas

RMSEt \ muestreo	Normal	Adaptativo
Interior	$2.6\ 10^{-3}$	$7.3\ 10^{-4}$
Frontera	$2.6\ 10^{-3}$	$1.1\ 10^{-4}$
T inicial	$2.4\ 10^{-3}$	$2.5\ 10^{-4}$

) u ondas de luz . Surge en campos como la acústica , electromagnética y dinámica de fluidos. Se utilizará la configuración en Rudd and Ferrari (2014):

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= c^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) && \Omega\times[0,T] \\ u(x,y,t) &= 0 && \forall (x,y)\in\partial\Omega \\ u(x,y,0) &= J_0\left(\lambda_4\sqrt{x^2+y^2}\right) && \forall (x,y)\in\Omega \\ \frac{\partial u}{\partial t}(x,y,0) &= 0 && \forall (x,y)\in\Omega \end{aligned}$$

con $\Omega = \left\{(x,y)\mid x^2+y^2\le 1\right\}$, $T=3$ y $c=2$.

Donde $J_0(\cdot)$ representa la función de Bessel de primer tipo, definida como:

$$J_0(r) = \sum_{m=0}^{\infty}\frac{(-1)^m}{(m!)^2}\left(\frac{r}{2}\right)^{2m}$$

y λ_4 representa el cuarto cero de $J_0(\cdot)$. Obteniendo como solución:

$$u(x,y,t) = J_0\left(\lambda_4\sqrt{x^2+y^2}\right)\cos\left(c\lambda_4 t\right)$$

En consecuencia, la loss según una metodología de reducción del riesgo empírico viene dada por $L = L1+L2+L3+L4$, con:

$$\begin{aligned} L_1(\boldsymbol{\theta}) &= \sum_{i=1}^{n_1}(NN_{tt}(X,t;\boldsymbol{\theta}) - c^2(NN_{xx}(X,t;\boldsymbol{\theta})+NN_{yy}(X,t;\boldsymbol{\theta})))^2 && (X,t)\sim(U(\Omega),U[0,T]) \\ L_2(\boldsymbol{\theta}) &= \sum_{i=1}^{n_2}(NN(X,t))^2 && (X,t)\sim(U(\partial\Omega),U[0,T]) \\ L_3(\boldsymbol{\theta}) &= \sum_{i=1}^{n_3}(NN(X,0) - J_0\left(\lambda_4\sqrt{x^2+y^2}\right))^2 && X=(x,y)\sim U(\partial\Omega) \\ L_4(\boldsymbol{\theta}) &= \sum_{i=1}^{n_4}(NN_t(X,0))^2 && X=(x,y)\sim U(\partial\Omega) \end{aligned}$$

Experimentalmente se puede notar como es posible reducir la loss pero eso no se condice con aproximar de manera correcta la solución de la EDP con dominio circular. Esto debido principalmente a que la base cartesiana de las redes neuronales no son suficiente para modelar un problema de geometría distinta.

Como se puede observar en la figura, la aproximación neuronal no es capaz de modelar la geometría compleja del modelo. Se trató cambiando la forma de muestreo, sin obtener mejoras notorias. La solución del problema puede venir de incluir alguna base de cálculo a la arquitectura que no sean transformaciones entre los estados sino que bases polinomiales o de kernel Gaussiano entre otras.

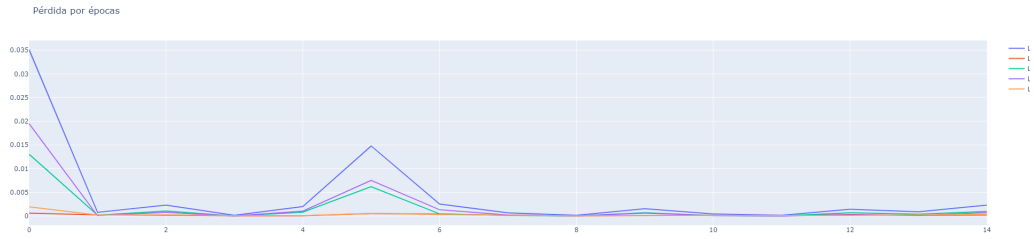

Figura 5.9: Pérdida por épocas

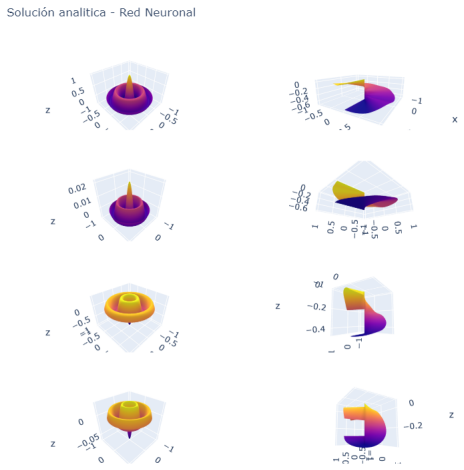

Figura 5.10: Comparación solución analítica - aproximación

Los autores Rudd and Ferrari (2014), resuelven el problema de manera precisa pero su método considera una red neuronal densa como estimación de una de las integrales en el método de elementos finitos tradicionales, incluyendo en la arquitectura una base RBF desde la formulación variacional del problema.

5.4. Black-Scholes-Barenblatt 100D

Siguiendo el trabajo de Raissi (2018) a partir de una ecuación diferencial estocástica forward-backward que tiene como principal característica que la condición terminal es aleatoria.

$$
\begin{aligned}
&dX_t = \sigma \operatorname{diag}(X_t)\, dW_t \quad t \in [0, T] \\
&X_0 = \xi \\
&dY_t = r\,(Y_t - Z_t' X_t)\, dt + \sigma Z_t' \operatorname{diag}(X_t)\, dW_t \quad t \in [0, T) \\
&Y_T = g(X_T)
\end{aligned}
$$

Donde W_t es un vector de evaluación de un movimiento Browniano, y $X_t, Y_t = u(t, X_t), Z_t = Du(t, X_t)$ procesos estocasticos asociados. Además, donde $T = 1, \sigma = 0{,}4, r = 0{,}05, \xi = (1{,}0; 5, 1{,}0; 5, \ldots, 1{,}0; 5) \in \mathbb{R}^{100}$, y $g(x) = \|x\|^2$.

La cual bajo condiciones de regularidad adecuadas puede ser descrita como una función determinista del tiempo y estado del proceso, la llamada Black-ScholesBarenblatt equation:

$$u_t = -\frac{1}{2} \operatorname{Tr}\left(\sigma^2 \operatorname{diag}\left(X_t^2\right) D^2 u\right) + r\left(u - (Du)^T x\right)$$

que admite como solución:

$$u(t, x) = \exp\left(\left(r + \sigma^2\right)(T - t)\right) g(x)$$

Los autores Raissi (2018), con una red neuronal de 5 capas y 256 neuronas resuelven el problema, que se ajusta con una loss definida sobre una aproximación tipo Euler-Maruyama de los procesos estocásticos en la EDP estocástica.

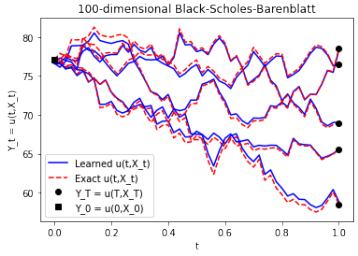

Figura 5.11: Comparación solución analítica - aproximación

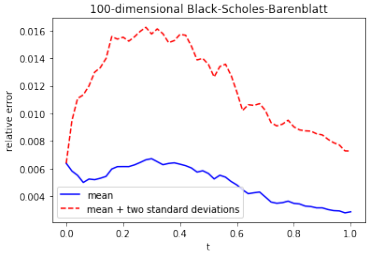

Figura 5.12: Errores asociados

Así de manera eficiente es capaz de aproximar con cierta variabilidad la solución en 100 D. La utilización de la arquitectura tipo DGM queda pendiente, pero principalmente la integración tedría que hacerse desde la formulación de la loss más que la arquitectura utilizada.

Capítulo 6

Análisis

Formalización del problema: Se introducirá el el problema de aproximación de una ecuación diferencial desde la perspectiva del aprendizaje estadístico. Sea Ω el conjunto de variables independientes de entrada, por ejemplo podría ser $\Omega \times [0,T]$ con $\Omega \subset \mathbb{R}^d$ y $T \in \mathbb{R}$, es decir un espacio de entrada que considera d variables en espacio y otra en tiempo.

Se define la función de pérdida $L = F$ en la definición de edp (preliminares), es decir como aquel funcional que caracteriza la relación de las derivadas parciales, la función y las variables independientes. Además, considere $P_{(\Omega,\mathbb{R})}$ la distribución de probabilidad que vincula el input y la solución de la edp, sobre alguna $\sigma-$algebra de $\Omega \times \mathbb{R}$, el objetivo entonces es minimizar el riesgo funcional:

$$R : \mathbb{R}^{\Omega} \to [0,\infty], f \to \int_{\mathbb{R}\times\Omega} L(f(x),x)\, dP_{(\mathbb{R},\Omega)}(y,x)$$

donde $\mathbb{R}^{\Omega}$ denota al espacio de todas las funciones de Ω a $\mathbb{R}$, es decir estamos en búsqueda de la función:

$$\hat{f} = argmin\{R(f) : f \in \mathbb{R}^{\Omega}\}$$

Específicamente, se fijará el espacio de hipótesis como el de las redes neuronales, para ello se utilizará la formalización de la definición expuesta en detalle en Gühring et al. (2020):

Definición: Sea $d = n_0, n_1, \ldots, n_{L-1}, s = n_L \in \mathbb{N}$ para algún $L \in \mathbb{N}$, y $\sigma : \mathbb{R} \to \mathbb{R}$. entonces el conjunto:

$$N_{(d,n_1,\ldots,n_{L-1},s)} := \{\Phi \text{ redes neuronales de } L \text{ capas, con } n_\ell \text{ neuronas en la capa } \ell\}$$

Además se considera la realización:

$$RN_{(d,n_1,\ldots,n_{L-1},s),\sigma} := \left\{R_\sigma(\Phi) : \Phi \in N_{(d,n_1,\ldots,n_{L-1},s)}\right\}$$

Considerar que la dimensión de entrada es dada $d \in \mathbb{N}$ y la dimensión de salida es $s = 1$

Por lo tanto la búsqueda se redefine:

$$\hat{f}_{NN} = argmin\{R(f) : f \in N_{(d,N,1)}\}$$

Considerando la notación $N = n_1, \ldots, n_{L-1}$.

Sin embargo, $P_{(\Omega,\mathbb{R})}$ es desconocida y por tanto no es posible calcular el riesgo funcional. En cambio se pueden extraer muestras de tamaño $m \in \mathbb{N}$ independientes de Ω según alguna densidad de probabilidad v, es decir $S = S_v = \{x_i\}_{i=1}^m$, y obtener la realización de la red neuronal en la muestra. Entonces se desea minimizar el riesgo empírico $R_S(f) = \frac{1}{m}\sum_{i=1}^m L(x_i, f(x_i))$ denotado:

$$\hat{f}_{NN,S} = argmin\{R_S(f) : f \in N_{(d,N,1)}\}$$

Ahora debido a que se necesitará resolver un complejo problema de optimización no convexa para encontrar $\hat{f}_{NN,S}$ denotemos la solución aproximada $\hat{f^*}_{NN,S} \in N_{(d,N,1)}$ y planeamos la descomposición del error clásica:

$$\left|R(\hat{f}) - R\left(\hat{f}^*_{NN,S}\right)\right| \leq \underbrace{\left|R\left(\hat{f}^*_{NN,S}\right) - R\left(\hat{f}_{NN,S}\right)\right|}_{\text{training error}} + \underbrace{\left|R\left(\hat{f}_{NN,S}\right) - R\left(\hat{f}_{NN}\right)\right|}_{\text{estimation error}} + \underbrace{\left|R\left(\hat{f}_{NN}\right) - R(\hat{f})\right|}_{\text{approximation error}}.$$

En cuanto al error de aproximación o expresividad del método, como se encuentra detallado en resultados relevantes, los autores de DGM logran demostrar que una red neuronal shallow tiene la capacidad de aproximar la solución de un cierto tipo de EDP vía minimizando la loss del método, lo que motiva a seguir profundizando en técnicas de aprendizaje profundo con este método pues existe una base matemática que permite mostrar que efectivamente es posible al menos teoricamente la aproximación de la solución. Además como se revisa en detalle en Gühring et al. (2020) exisen diversos teoremas y estrategías para mostrar que las redes neuronales tienen la capacidad de aproximación en espacios de Sobolev, lo cual establece bases suficientes como para dar por sentado que las redes neuronales pueden aproximar soluciones de EDP complejas, incluso superando la maldición de la dimensionalidad para el caso de la ecuación de Black-Scholes-Barenblatt, para más detalle véase la sección 6.3 de Gühring et al. (2020).

El objetivo de este trabajo es estudiar el impacto de las metodologías clásicas tales que permitan reducir empíricamente el error de entrenamiento, es decir perfeccionar la optimización del modelo DGM. Por otro lado considerar que la reducción del error de estimación, o en otras palabras, la capacidad de generalización del modelo, depende directamente de la calidad del muestreo S , el cual se ve reflejado en el estudio del minibatch del método. El cálculo del error en el contexto de resolver una ecuación diferencial se mide como el error de interpolación, este se define como la evaluación de la red resultante sobre un conjunto independiente al de entrenamiento y que en el contexto de los métodos de elementos finitos (con mallas) se refiere al error de la solución encontrada al considerar la evaluación sobre una malla más fina. Para un método libre de malla como DGM, se requiere repensar la manera de evaluar.

Todo ello con la misión de evaluar empíricamente la capacidad de aproximación o expresividad de las redes neuronales en la resolución de EDPs.

Estudio del minibatch:

Con respecto al estudio del minibatch, notar que la pérdida depende directamente del conjunto de entrenamiento $S_v \subset \Omega$ según la densidad de probabibilidad v. En retrospectiva, la solución de la EDP debe de anular la pérdida en todos los puntos de Ω por lo que entre más denso sea S_v mejor estimador será la red neuronal. El framework de aprendizaje (algoritmo 2 Deep Galerkin) define cada minibatch como una nueva muestra del dominio. Se propone entonces un estudio de la construcción óptima del minibatch, evaluando el impacto de garantizar independencia y teniendo en cuenta el costo computacional asociado.

Estudio de la loss:

La definición de pérdida propuesta por DGM difiera de la clásica expuesta en Lagaris et al. (1998), la cual llamaremos anzats. El método anzats está basado en un cambio de variable que técnicamente siempre puede ser utilizado para integrar las condiciones adicionales a una sola ecuación diferencial. Para ilustrar

la idea consideremos la siguiente ecuación diferencial ordinaria de primer orden $f_x = G(x, f)$ donde G es un operador, además se considera la restricción $f(0) = A \in \mathbb{R}$. Luego la pérdida definida por DGM, para la realización de la red neuronal NN con parámetros θ, tendría la forma:

$$L = \|NN(x;\theta) - G(x, NN(x;\theta)\|_{L^2} + \|NN(0;\theta) - A\|_{L^2}$$

Por otro lado la pérdida anzats se definiría luego de redefinir la función a aproximar como $f_{ansatz} = A + xNN(x;\theta)$, notar que justamente $f_{ansatz}(0) = A$, luego basta definir la pérdida como:

$$L = \|f_{ansatz} - G(x, f_{ansatz})\|_{L^2}$$

De esta manera, la solución buscada es: $y = (f_{ansatz}(x) - A)/x$

Si bien el método anzats formula siempre una única pérdida a través del cambio de variable de la solución búscada, este cambio añade mayor complejidad a la solución buscada lo cual puede resultar contraproducente teniendo en cuenta que la complejidad de la solución de la ecuación diferencial. Además en un contexto más general con múltiples restricciones adicionales y posiblemente de diferente tipo, el cálculo del cambio de variable de la solución f_{anzats} puede no ser sencillo de lograr a priori. En consecuencia esta estrategia va en desmedro de la automatización de la resolución y el estudio de regularización. Por otro lado, la formulación DGM abre la puerta a la introducción de parámetros de regularización, de manera de aprender la complejidad que conlleva satisfacer cada una de las restricción de la EDP reflejadas en cada término de la pérdida.

Capítulo 7

Conclusión

En el presente trabajo se introdujo la resolución de ecuaciones diferenciales parciales a través de metodologías de deep learning, presentando una revisión del método y aplicando el mismo a diferentes contextos exponiendo ventajas y debilidades del mismo. Las principales conclusiones son:

- Una característica relevante que no cuentan los métodos de resolución clásicos es que las soluciones obtenidas via redes neuronales son naturalmente **diferenciables** y tienen una forma **(semi) analítica** que puede ser transferida a cualquier cálculo posterior. Por ejemplo el tiempo de inferencia en una red neuronal es muy bajo, a diferencia de los métodos de Runge-Kutta que ofrecen soluciones discretas o soluciones con difenciabilidad limitada como los elementos finitos.
- A diferencia de los métodos FEM, DGM puede superar la maldición de la dimensionalidad en el espacio paramétrico. Gühring et al. (2020)
- Para un problema de 200 dimensiones existen 200000 derivadas de segundo orden lo cual puede llegar a ser muy costoso para un método de diferenciación automática. Por lo que vale la pena proponer aproximaciones más rápidas de esas derivadas con montecarlo y diferencias finitas. Sirignano and Spiliopoulos (2018)
- La poca disminución de la loss puede deberse a pobre aproximaciones del gradiente, por lo que tiene sentido reforzar el muestreo en aquellas zonas.

Finalmente, se podría avanzar esta investigación aplicando el método estudiado en un contexto especifico e incorporando propiedades del sistema físico que define la ecuación diferencial. En particular algunas tareas concretas pueden ser la incorporación de una base radial, la aplicación de DGM en el problema de alta dimensionalidad, la elaboración de una estrategia de ensamblaje del método, entre otros.

Bibliografía

A. Al-Aradi, A. Correia, D. Naiff, G. Jardim, and Y. F. Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning. *arXiv: Computational Finance*, 2018.

L.-F. Arsenault, A. Lopez-Bezanilla, A. von Lilienfeld, and A. Millis. Machine learning for many-body physics: the case of the anderson impurity model. *Physical Review B*, 90, 08 2014. doi: 10.1103/PhysRevB.90.155136.

G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355:602, 02 2017. doi: 10.1126/science.aag2302.

G. Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

I. Gühring, M. Raslan, and G. Kutyniok. Expressivity of deep neural networks. *ArXiv*, abs/2007.04759, 2020.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):303–314, 1991.

K. Kawaguchi, J. Huang, and L. P. Kaelbling. Every Local Minimum Value Is the Global Minimum Value of Induced Model in Nonconvex Machine Learning. *Neural Computation*, 31(12):2293–2323, 12 2019. ISSN 0899-7667. doi: 10.1162/neco_a_01234. URL https://doi.org/10.1162/neco_a_01234.

Y. Khoo, J. Lu, and L. Ying. Solving for high dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6, 10 2018. doi: 10.1007/s40687-018-0160-2.

A. Koryagin, R. Khudorozhkov, and S. Tsimfer. Pydens: a python framework for solving differential equations with neural networks, 09 2019.

I. Lagaris, A. Likas, and D. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9:987–1000, 09 1998. doi: 10.1109/72.712178.

T. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN 9780071154673. URL https://books.google.cl/books?id=EoYBngEACAAJ.

M. Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. 04 2018.

K. Rudd and S. Ferrari. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 19, 12 2014. doi: 10.1016/j.neucom.2014.11.058.

J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.08.029. URL https://www.sciencedirect.com/science/article/pii/S0021999118305527.

J. Willard, X. Jia, S. Xu, M. S. Steinbach, and V. Kumar. Integrating physics-based modeling with machine learning: A survey. *ArXiv*, abs/2003.04919, 2020.