UNIVERSIDAD DE CHILE

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

PRÁCTICA PROFESIONAL II

Estudio y simulación de EDP's utilizando técnicas de deep learning

Practicante: Fernando Fêtis
Evaluador: Hugo Carrillo
Investigador post-doctoral.

Periodo de práctica: 1 de enero – 31 de enero.

Otoño, 2023.

Resumen práctica profesional

En la primera práctica realizada se complementó el apunte del curso de aprendizaje de máquinas del DIM, agregando demostraciones formales a los teoremas expuestos y complementando el contenido con otras referencias. En esta segunda práctica, se continuó en la línea del aprendizaje automático pero ahora utilizando redes neuronales para simular ecuaciones en derivadas parciales.

Esta aplicación de las redes neuronales es un tópico reciente por lo que hay muy poca investigación teórica al respuesto, por lo que a lo largo de la práctica fue necesario consultar diferentes referencias del estado del arte en redes neuronales. Además, fue necesario estudiar distintas conexiones teóricas que hay entre EDP's y modelos de machine learning, con tal de obtener una visión más robusta acerca del tópico que se estaba trabajando.

Contexto

Dada la inherente complejidad de las ecuaciones en derivadas parciales y su resolución, para la mayoría de los casos se vuelve necesario conformarse con simulaciones numéricas de dichas soluciones. Es debido a esto que obtener nuevos métodos de simulación puede contribuir enormemente al entendimiento de las EDP's así como a resolver diferentes problemas de ingeniería aplicada.

A fines del año 2017 se propuesto una nueva técnica de simulación basada en redes neuronales. Esta técnica fue denominada PINN (*physics informed neural networks*) y consiste en aproximar la solución de una EDP mediante la realización de una red neuronal, donde la función de costo asociada al entrenamiento está compuesta el error cuadrático medio de aproximación tanto de la EDP como de sus restricciones (condiciones iniciales y/o de borde).

Este nuevo método no hace uso de la forma variacional de la EDP (como ocurre en otros métodos como el método de elementos o volumen finito) ni toma en consideración la regularidad del dominio de la EDP. Si bien nuestro estudio fue mayormente empírico, es esperable la obtención de resultados teóricos en un futuro que garanticen o respalden el funcionamiento de este método.

Desde un punto de vista cualitativo, el método entrega muy buenos resultados (comparables con los entregados por librerías como FeniCS) tanto en problemas directos como en problemas inverso. Además, su simplicidad de uso lo hace idóneo para personas no especializadas en EDP's que deseen realizar simulaciones para alguna tarea en específico.

Objetivos y metodología

Se pueden identificar diferentes objetivos a lo largo de la práctica, los principales son los siguientes:

- Estudiar las redes neuronales desde un punto de vista práctico y teórico. Para esto último, fue necesario revisar bibliografía moderna ya que las redes neuronales no encajan dentro de la teoría clásica del aprendizaje estadístico, por lo que se ha vuelto necesario el desarrollo de una nueva teoría de aprendizaje.
- Estudio del reciente método de PINN's, diferentes arquitecturas e hiperparámetros, así como también sus ventajas y desventajas frente a otros métodos. Durante esta práctica, se usó como método de comparación al método de elementos finitos implementado en FeniCS ya que, al ser un método más estudiado, cuenta con respaldo teórico.
- Realizar simulaciones utilizando el método de PINN para diferentes tipos de ecuaciones.
- Obtener conclusiones empíricas acerca del método.

Para el cumplimiento de estos objetivos, se trabajó con la librería de PyTorch, la cual está optimizada para hacer diferenciación automática y entrenamiento de modelos neuronales. Además, los resultados teóricos fueron obtenidos de los papers que se pueden encontrar en la bibliografía. Para los resultados prácticos, estos pueden ser consultados en detalle en el documento anexado.

A continuación se definirá el concepto de red neuronal y se explicará la metodología de PINN.

Redes neuronales

La definición de red neuronal que se trabajó fue la siguiente:

Definición 1 (Red neuronal *fully connected*)**.** *Una red neuronal fully connected viene dada por su arquitectura. Esto es, una tupla $a = (N, \rho)$, donde $N = (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$ indica el número de neuronas por capa y $\rho : \mathbb{R} \to \mathbb{R}$ es la función de activación.*
En este caso, $L \in \mathbb{N}$ indica el número de capas de la red, N_0 corresponde a la dimensión de entrada y N_L corresponde a la dimensión de salida.

Adicionalmente, se tienen las siguientes definiciones asociadas a una arquitectura en particular:

Definición 2. *Para una red neuronal $a = (N, \rho)$, se define el número de parámetros como*

$$P(N) := \sum_{l=1}^{L} N_l N_{l-1} + N_l$$

Además, se define el ancho de la red como $||N||_\infty = \max\limits_{0 \le l \le L} N_l$ y el número de capas como L. En caso de que $L > 2$, la red se dirá profunda.

Es importante notar que, de acuerdo a la definición anterior, $P(N)$ es independiente de la función de activación ρ. Esto en general no se cumple para otro tipo de redes neuronales donde la función de activación puede ser adaptativa (p.g., PReLU).

Definición 3 (Función de realización)**.** *Dada una red neuronal $a = (N, \rho)$, se define la función de realización de la red neuronal como un mapa $\Phi_a : \mathbb{R}^{N_0} \times \Theta \to \mathbb{R}^{N_L}$, donde la primera variable representa el input de la red neuronal y la segunda variable está asociada a los parámetros de la red, los cuales son de la forma*

$$\theta = \left(W^{(l)}, b^{(L)}\right)_{l=1}^{L} \in \Theta := \prod_{l=1}^{L} \left(\mathcal{M}_{N_l, N_{l-1}}(\mathbb{R}) \times \mathbb{R}^{N_l}\right) \cong \mathbb{R}^{P(N)}$$

de este modo, la función Φ_a viene dada por $\Phi_a(x, \theta) = \Phi^{(L)}(x, \theta)$ donde

$$\begin{aligned}
&\overline{\Phi}^{(0)}(x, \theta) = x \\
&\Phi^{(l)}(x, \theta) = W^{(l)} \overline{\Phi}^{(l-1)}(x, \theta) + b^{(l)}, \quad \forall l \in \{1, \dots, L\} \\
&\overline{\Phi}^{(l)}(x, \theta) = \rho\left(\Phi^{(l)}(x, \theta)\right), \quad \forall l \in \{1, \dots, L-1\}
\end{aligned}$$

en este caso, la función de activación ρ se aplica por componentes: $v \in \mathbb{R}^n \implies \rho(v) = (\rho(v_1), \dots, \rho(v_n))$.

A lo largo de este reporte, se usará el concepto red neuronal para hablar tanto de la tupla $a = (N, \rho)$ como de su función de realización. También, se le suele llamar red neuronal a la función de realización asociada a un parámetro $\theta \in \Theta$ determinado: $\Phi_a(\cdot, \theta)$.

La elección de la función de activación $\rho : \mathbb{R} \to \mathbb{R}$ juega un rol importante tanto en el desempeño de la red como en sus propiedades teóricas.

Método PINN

A lo largo de la práctica se trabajo con EDP's que se adaptan a la siguiente formulación general:

$$\mathcal{F}(u, x, t; \lambda) = 0, \quad (x, t) \in \Omega \times (0, T)$$
$$\mathcal{B}(u, x, t; \lambda) = 0, \quad (x, t) \in (\partial\Omega \times [0, T]) \cup (\Omega \times (\{t = 0\} \setminus \{t = T\})) =: \Gamma_p$$

Para el método de PINN's se busca que una red neuronal aproxime la solución de la EDP. Para su entrenamiento, se generan puntos aleatorios dentro del dominio con tal de poder evaluar la solución aproximada por la red neuronal. Para esto, se generan puntos aleatorios dentro del dominio con tal de poder estimar el error de aproximación tanto para la EDP como para sus condiciones asociadas.

El diagrama general que representa una PINN es el siguiente:

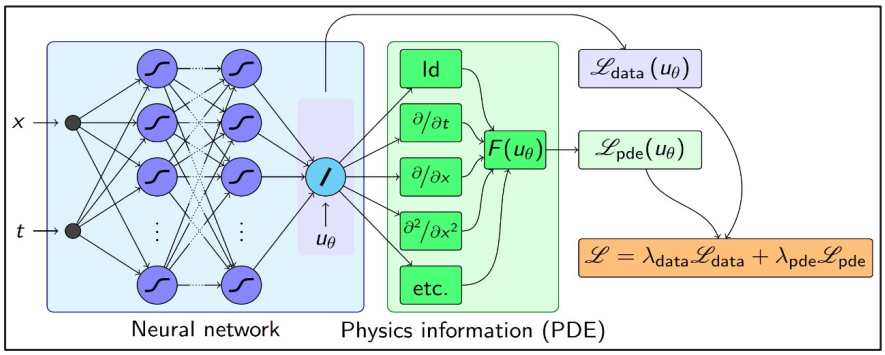

Figura 1: Diagrama general de ina PINN.

Se llama a $\{x_i,\, t_i\}_{i=1}^{N_{physics}} \subset \Omega \times (0, T)$ puntos de colocación en el dominio, que serán aquellos puntos en donde se impone la condición $\mathcal{F}(u, x, t; \lambda) = 0$. Y a $\{x_i,\, t_i,\, u_i\}_{i=1}^{N_{data}} \subset \Gamma_p \times \mathbb{R}$, los puntos donde se impondrá que $\mathcal{B}(u, x, t; \lambda) = 0$, que consideraremos como los datos conocidos del problema. Por tanto, el objetivo es minimizar la función de pérdida

$$\mathcal{L}_{physics}(\theta) = \frac{1}{N_{physics}} \sum_{i=1}^{N_{physics}} |\mathcal{F}(u_\theta(x_i,\, t_i), x_i, t_i; \lambda)|^2$$
$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |\mathcal{B}(u_\theta(x_i,\, t_i), x_i, t_i; \lambda)|^2$$
$$\mathcal{L}_{total}(\theta) = \lambda_p \mathcal{L}_{physics}(\theta) + \lambda_d \mathcal{L}_{data}(\theta)$$

Donde u_θ es una red neuronal con pesos θ que busca aproximar a la solución del problema antes expuesto. Y los pesos $\lambda_{physics}$ y λ_{data} son los ponderadores de las funciones de pérdida. Notar que no se ha explicitado si esta técnica se está aplicando a un problema inverso o uno directo.

Si estamos en el contexto de un problema directo con cantidad de objetivos n_{obj} (que sería mayor a 1 en el caso de un sistema de ecuaciones, por ejemplo), entonces $u_\theta : \overline{\Omega} \times [0, T] \mapsto \mathbb{R}^{n_{obj}}$ busca aproximar cada uno de los objetivos del problema.

En un problema inverso, los parámetros que buscamos aproximar se añaden a los objetivos, por tanto si tenemos n_{obj} cantidad de objetivos y n_{par} cantidad de parámetros a estimar, entonces simplemente se tendrá que $u_\theta : \overline{\Omega} \times [0, T] \mapsto \mathbb{R}^{n_{obj}+n_{par}}$ busca aproximar tanto a los objetivos como a los parámetros, aunque muchas veces los primeros serán, al menos, parcialmente conocidos en el problema inverso.

Desarrollo

Durante los últimos años ha crecido el interés por el análisis teórico de las redes neuronales. Esto ha permitido comprender en mayor profundidad el comportamiento de las redes neuronales y sus propiedades, pudiendo obtener cotas de error, cota para el número de parámetros y propiedades de densidad. A continuación se expone de manera informal algunos resultados modernos en redes neuronales, los cuales fueron recopilados durante la primera parte de esta práctica y permitieron tener una comprensión mucho mayor al momento de aplicar redes neuronales a ecuaciones diferenciales (ajuste de hiperparámetros, explicación de resultados patológicos, diseño de arquitecturas, etc.).

Si bien la mayoría de los resultados obtenidos son para redes *fully connected*, también se han encontrado propiedades para otras arquitecturas más complejas.

Propiedades topológicas

Para redes *fully connected* de tamaño fijo, se tienen los siguientes resultados:

- Para funciones de activación localmente Lipschitz (todas las fc. de activación comunes lo cumplen), si el conjunto de funciones alcanzables por una red neuronal es convexo, entonces la función de activación es polinomial. Es decir, recíprocamente, si el activador es no polinomial, el conjunto de realizaciones de una red neuronal es no convexo.
- Bajo hipótesis razonables sobre la función de activación y asumiendo que la entrada de las redes es un compacto $\Omega \subset \mathbb{R}^n$, entonces el conjunto de realizaciones es no cerrado en $L^p(\Omega)$ ni en $\mathcal{C}(\Omega)$.
- Si bien el resultado anterior es una propiedad no deseable de las redes neuronales, sí se tiene resultados favorables bajo una hipótesis adicional. El subconjunto de las funciones alcanzables donde se pide además que los parámetros de la red se mantengan uniformemente acotados, resulta ser un conjunto cerrado tanto en $L^p(\Omega)$ como en $\mathcal{C}(\Omega)$.
- Lo anterior permite concluir un argumento de *exploding weights*: si una función no posee un mejor aproximador dentro de la familia de funciones generadas por una arquitectura en particular, entonces al tomar una secuencia de pesos que infimice el error de aproximación, la secuencia de la norma de esos pesos es no acotada.
- Por último, se tiene un resultado de (no) continuidad inversa. Es sabido que el mapeo que transforma parámetros (de una red neuronal) en la función generada por la red neuronal es no inyectivo. Más aún, en esta investigación se probo lo siguiente: para dos funciones generables por una red neuronal y cercanas la una de la otra (i.e., $||f_1 - f_2||_{L^\infty} < \epsilon$), en general no es posible encontrar dos pesos w_1, w_2 que generen dichas funciones y que estén lo suficientemente cerca.

Propiedades de *kernel*

Desde hace unos años se han estudiado propiedades de las redes neuronales viéndolas bajo la lupa de los métodos de *kernel*, donde se han encontrado similitudes e incluso equivalencias. A continuación se citan algunos resultados importantes encontrados en los dos papers mencionados anteriormente:

- Se ha demostrado una equivalencia entre una red neuronal de ancho infinito y un proceso gaussiano con un *kernel* determinado. Esto abre la puerta a realizar inferencia bayesiana exacta sobre redes de ancho infinito. Para esta equivalencia, se han desarrollado técnicas eficientes para computar la función de covarianza de estos procesos gaussianos.
- Si bien el resultado anterior aplica para redes de ancho infinito, se obtienen resultados cualitativamente similares para redes con un número significativo de neuronas.

- Con respecto a la optimización, se ha demostrado que la evolución de los parámetros de una red neuronal cuando ese entrenada con SGD viene determinada por un *kernel*, el cual se conoce en la literatura como el Neural Tangent Kernel (NTK). Este *kernel* ha sido estudiado para redes *fully connected*, convolucionales, recurrentes e incluso para transformers.
- Este *kernel* muchas veces es utilizado para obtener resultados teóricos acerca de las redes neuronales, sobretodo en el caso límite de redes con un número infinito de neuronas ya que bajo esta hipótesis, el *kernel* se vuelve constante por lo que facilita el análisis. En este caso, la red neuronal sigue una EDO lineal durante el entrenamiento.
- Al igual que para el *kernel* asociado al proceso gaussiano, para el NKT se ha visto que el comportamiento de una red ancha es similar al límite teórico.
- El estudio del *kernel* NKT ha permitido probar que se puede alcanzar el mínimo global en tiempo polinomial siempre y cuando la red esté sobreparametrizada (sea suficientemente ancha).

Propiedades de expresividad

Una de las propiedades más conocidas de las redes neuronales es que funcionan como aproximadores universales. Más especificamente, son capaces de aproximar cualquier función continua (bajo la topología compacta) siempre y cuando se tenga libertar en el número de neuronas, o bien, en el número de capas. Este resultado ha sido refinado para diferentes espacios de funciones, logrando incluso encontrar cotas para el número de parámetros o propiedades de densidad para arquitecturas más complejas. Algunos resultados posiblemente relevantes para el estudio de EDPs utilizando redes neuronales *fully connected* son los siguientes:

- **Teorema de Mhaskar:** cualquier función en un Sobolev $W^{k,p}$ de dimensión d puede ser aproximada en norma uniforme con un error e usando una red con una única capa escondida usando $\mathcal{O}(e^{-d/r})$ neuronas. Esto muestra que la maldición de la dimensionalidad está presente cuando se quiere aproximar un Sobolev con redes neuronales ya que el número de neuronas necesarias crece exponencialmente con el número de variables. El punto siguiente muestra que esto es un problema inherente a los espacios de Sobolev.
- **Teorema de DeVore:** todo aproximador de funciones continuas que quieran poder aproximar funciones de un espacio de Sobolev $W^{k,p}$ de dimensión d necesita al menos $\Theta(e^{-d/r})$ parámetros. De esta forma, resulta que las redes neuronales están cercanas a ser aproximadores óptimos para Sobolev.
- Resultados similares se han obtenido cuando se restringen aún más los espacios de funciones donde se desea aproximar. Si bien estos resultados entregan cotas para el tamaño de la arquitectura, se debe tener en consideración que muchas veces puede ser difícil de alcanzar el aproximador óptimo debido a la no convexidad del problema de optimizaciòn. Por esto, puede ser mejor utilizar redes sobreparametrizadas ya que, como se ha probado recientemente, pueden llegar al mínimo global utilizando SGD.
- **Teorema de Montúfar:** considerando que una red ReLU aprende funciones lineales por regiones, el número de regiones que es capaz de diferenciar la red es polinomial en el número de neuronas y exponencial en el número de capas.

Simulación numérica

La segunda parte de esta práctica consistió en realizar simulaciones de EDP's utilizando redes neuronales, tratando de justificar (y/o verificar) los resultados obtenidos mediante los resultados teóricos encontrados en la primera parte. La mayor dificultad de esta parte fue la programación de los distintos métodos consultados, así como el ajuste de hiperparámetros para su óptimo entrenamiento.

A continuación se mostrará, a modo de ejemplo, algunas simulaciones realizadas utilizando esta técnica. Para más detalles se puede consultar el reporte anexado a este documento.

Problema directo

$$\begin{aligned}
\Delta u(x,y) + v(x,y) &= -2\cos(x+y) + x\cos(x) + y && (x,y) \in ([-5,5]\times[-5,5]) \\
\frac{\partial u}{\partial x}(x,y) - \Delta v(x,y) &= -\sin(x+y) - y + 2\sin(x) + x\cos(x), && (x,y) \in ([-5,5]\times[-5,5]) \\
u(x,y) &= \cos(x+y) - xy, && (x,y) \in \partial([-5,5]\times[-5,5]) \\
v(x,y) &= x\cos(x) + y, && (x,y) \in \partial([-5,5]\times[-5,5])
\end{aligned}$$

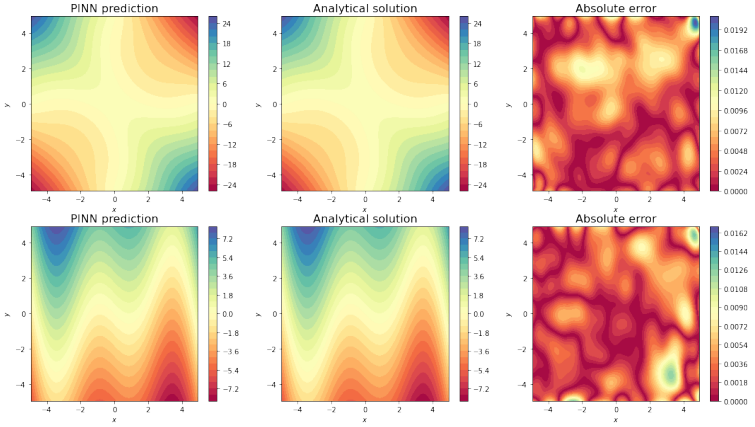

Figura 2: Simulación para u (arriba) y v (abajo) utilizando una PINN de 3 capas con 100 neuronas por capa. La red fue entrenada con el método de segundo orden L-BFGS durante 50 épocas (tiempo total: 12s).

Problema inverso

$$\nabla\cdot(\gamma(x,y)\nabla u(x,y)) = -5(\sin(xy)+2)\cos(x+2y) - (2x+y)\sin(x+2y)\cos(xy), \qquad (x,y) \in [-\pi,\pi]\times[-\pi,\pi]$$

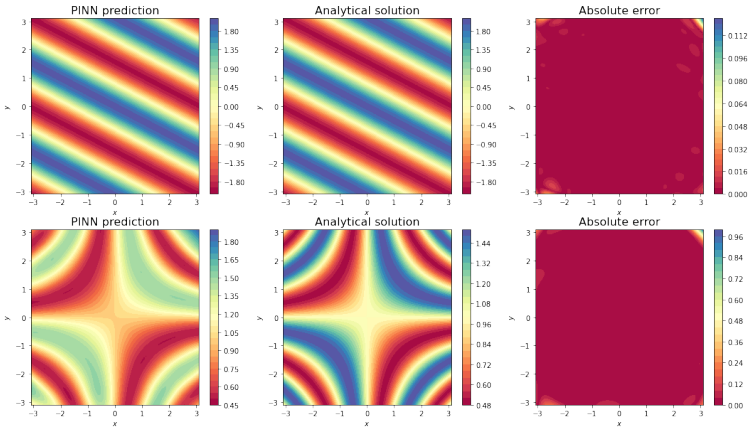

Figura 3: Simulación para u (arriba) y γ (abajo) utilizando una PINN de 5 capas con 250 neuronas por capa. La red fue entrenada con el método de segundo orden L-BFGS durante 300 épocas (tiempo total: 14m, 12s).

Para otras simulaciones y problemas aplicados, revisar el jupyter Notebook adjunto.

Conclusiones

Si bien se realizaron trabajos adicionales (estudio de la continuidad única, análisis de incertidumbre, problema en dominios no rectángulares, etc), aquí se anexó la línea de trabajo principal de la práctica. El reporte adjunto consiste en el reporte final entregado a la empresa y ahí se indican todos los trabajos realizados.

Con respecto a las conclusiones, se listan a continuación las principales:

- Con respecto a la arquitectura de la red neuronal, observamos que es mejor una red ancha que una red profunda. También se puede afirmar que no fue necesaria una arquitectura más compleja ya que se entrenó con modelos residuales y con transformers pero en ningún experimento se obtuvo un mejor resultado. Además, la función de activación que mejor función fue la tangente hiperbólica, mostrando diferencias a lo usado en otras áreas como visión computacional, donde se usa ReLU.
- Para el entrenamiento de los modelos, los métodos de segundo orden mejoran considerablemente la velocidad de entrenamiento, bajando de tiempos mayores a 10 minutos a unos cuantos segundos. Por esto, todos los modelos finales fueron entrenados con L-BFGS.
- El fenómeno de spectral bias influye considerablemente en el desempeño de la red y puede dar indicios acerca de la complejidad que necesitará la red neuronal. En particular, para funciones altamente oscilantes, se vuelve necesario una red más compleja y un mayor número de épocas de entrenamiento.
- No hay mucha diferencia en cambiar los ponderaciones de las funciones de pérdida. Dándole el mismo peso a todas las funciones de pérdida se obtienen resultados buenos.
- El problema depende fuertemente de las condiciones iniciales. Por esto, se utilizó una inicialización correcta (Xavier o He) dependiendo de la función de activación utilizada.
- Las EDP's hiperbólicas también tienen problemas de simulación con esta nueva técnica. Esto también se observa en otros métodos como métodos de elementos finitos.
- Para solucionar lo anterior, se probaron arquitecturas diferentes buscando disminuir el error de aproximación. Se observó que una arquitectura basada en atención puede mejorar la aproximación de la solución en tiempos largos.
- La red neuronal es capaz de aprender aún cuando el problema esté mal puesto. Por lo tanto, queda pendiente analizar qué es lo que aprende la red al ser entrenada. Es posible que busque una solución minimal (en algún sentido) dentro del espacio de soluciones.

Por último, es bueno insistir en el potencial que tiene este método debido a su simplicidad y generalidad para resolver todo tipo de ecuaciones. Se espera que en un futuro se estudie más en detalle su convergencia y posibles falencias.