BOURBAKI
FINANZAS

deep learning for finance i

Contents

01 Introducción

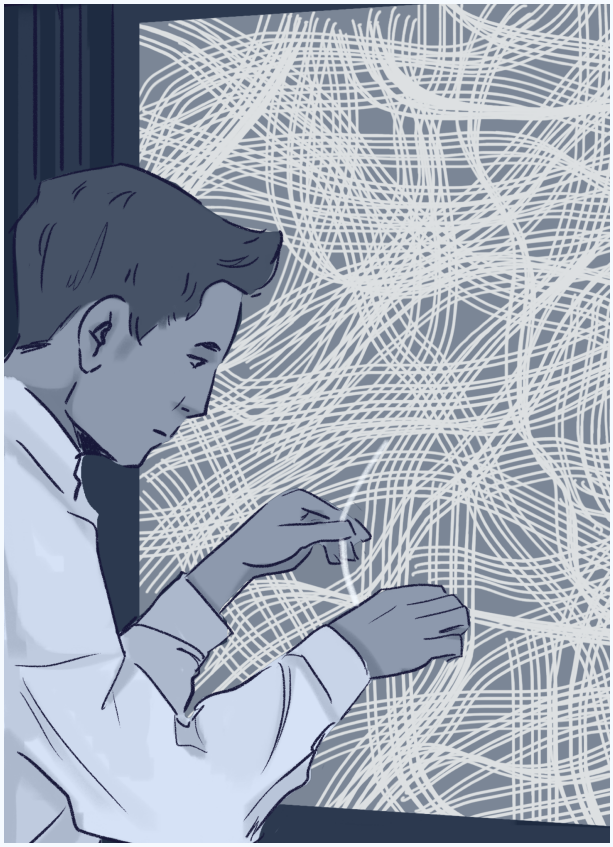
Las presentes notas son la bitácora del primero de los 6 módulos de nuestro curso Deep Learning for Finance. En este curso trataremos modelos neuronales diversos para garantizar un mejor rendimiento dependiendo de la estructura interna de las bases de datos.

Este módulo está enfocado en las redes neuronales densas y lo impartimos junto a Max Mitre. Además de este documento los invitamos a consultar el Github del curso en este link.

El curso es una invitación al uso de las redes neuronales profundas, la organización de las clases es la siguiente:

1. Introducción al problema financiero (dos horas).
2. Presentación de la arquitectura (dos horas).
3. Caso de uso (dos horas).
4. Sesión de dudas y presentación del reto (dos horas).
5. Tiempo para trabajar en el reto y dudas finales (cuatro horas).

Frank Rosenblatt



Es un psicólogo estadounidense quien es conocido como el padre del Aprendizaje Profundo, sus investigaciones en neurociencias lo acercaron a lo que hoy conocemos como la inteligencia artificial. En 1960 construyó Mark I Perceptron la primera computadora que logró aprender utilizando un algoritmo. Actualmente este modelo y algoritmo son la base de las redes neuronales, una de sus obras escritas más importantes es "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" donde resume sus investigaciones sobre este tema.

Keras

Keras es un API sobre Machine Learning escrita en Python. Fue creada con la finalidad de permitir a los usuarios experimentar, de modo rápido y sencillo, con varios modelos.

Permite al usuario la creación de algoritmos de modo sencillo para dejarle tiempo a que se enfoque en las partes importantes sobre su investigación en lugar de gastar tiempo escribiendo código de un modelo desde cero.

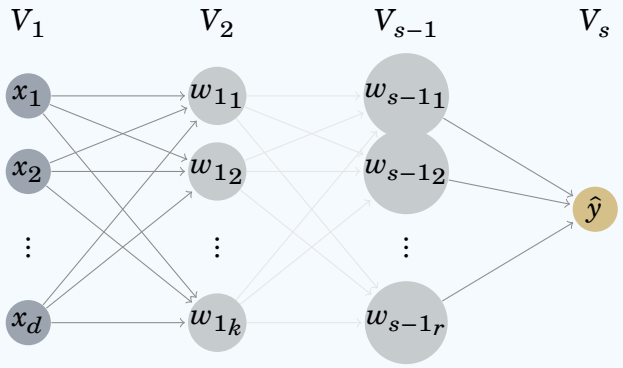
El módulo se encuentra bastante optimizado para realizar operaciones tensoriales en CPU, GPU o TPU, también permite calcular los gradientes de expresiones arbitrarias, entre otros. Además es bastante intuitiva para uso y tiene una documentación clara.

02 El lenguaje de las redes neuronales

Definition 00.1 La arquitectura de una red neuronal feed-forward es una familia de funciones que satisfacen lo siguiente:

- Sea $G = (V, E)$ un grafo dirigido, finito y acíclico; es decir, tenemos un conjunto finito de vértices $v \in V$, los elementos $e \in E$ se pueden interpretar como flechas entre vertices que poseen una dirección, además no hay una secuencia de elementos en E que empiece y termine en un vértice.

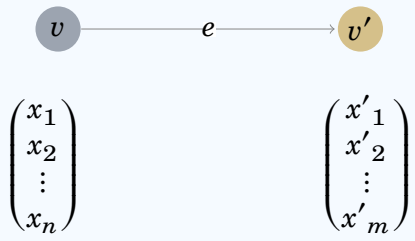
  A los elementos en V los llamaremos neuronas y a los elementos en E los llamaremos transformaciones lineales,
- Una función llamada función de activación $\rho : \mathbb{R} \to \mathbb{R}$
- Una partición disjunta del conjunto de vértices $V = V_1 \cup ... \cup V_s$ donde cada nodo en V_{t-1} está conectado a algún elemento de V_t.
- El parámetro s será el número de capas,
- V_1 es un conjunto disjunto de vértices con tamano $d+1$ y V_s tiene un solo nodo al que denotaremos como $\hat{y}$.



Definition 00.2 Dada una arquitectura de una red neuronal feed-forward, una red neuronal es lo siguiente:

- Una asignación $w_1\ (v)$ de un vector de cierta dimensión para cada neurona $v \in V$,
- Una asignación $w_2\ (e)$ de una matriz para cada arista $e \in E$,
- Las asignaciones anteriores satisfacen que si $v \in V_i, v' \in V_{i+1}$ están conectados por algún $e \in E$ entonces el tamaño de la matriz

$w_2(e)$ es $n \times m$ donde el tamaño de los vectores $w_1(v)$, $w_1(v')$ son iguales a n y m respectivamente.



$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_m \end{pmatrix}$$

- Una función $f : X^d \to Y$ que puede calcularse utilizando la información anterior en orden de izquierda a derecha V_{t-1} to V_t. La operación parcial en cada una de las neuronas se ve de la siguiente forma:

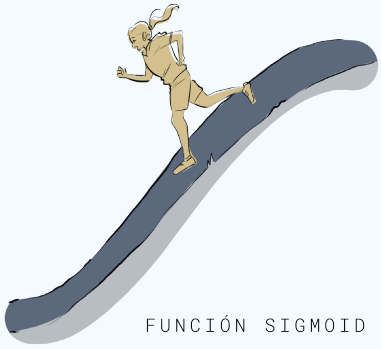
$$\rho\left(w_2(e)\, w_1(v) + b\right) \tag{02.1}$$

Funciones de activación

Como lo vimos en la definición, una red neuronal depende de una elección de las funciones de activación. En esta sección hablaremos sobre todo de dos funciones de activación:

Definition 00.3 Definimos a la función sigmoide $\sigma : \mathbb{R} \to \mathbb{R}$ utilizando la siguiente fórmula:

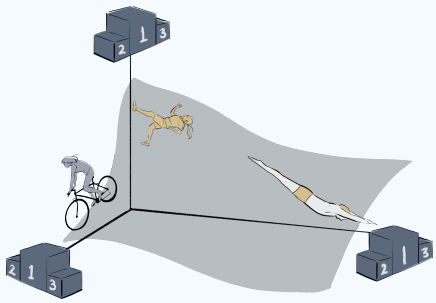
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{02.2}$$



La función sigmoide es ampliamente utilizada por ejemplo en el algoritmo de la regresión logística debido a sus propiedades, por ejemplo, que es una función continua, acotada entre 0 y 1, que modela muy bien la probabilidad condicional $\mathbb{P}(x|y = 1)$. Es posible generalizar la función anterior a vectores con tamaño superior de la siguiente manera:

Definition 00.4 Dado un $v = (v_1, v_2, \dots, v_d) \in \mathbb{R}^d$, definimos la función SoftMax de v como

$$SoftMax\,(v) = \left(\frac{e^{v_1}}{\sum_{j \leq d} e^{v_j}}, \dots, \frac{e^{v_d}}{\sum_{j \leq d} e^{v_j}} \right) \tag{02.3}$$



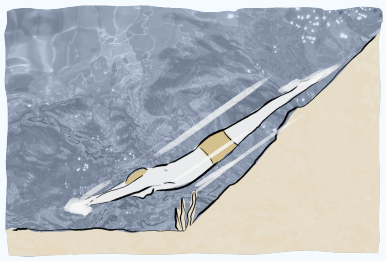
Exercise 00.1 Demuestre que si $v \in \mathbb{R}^d$ entonces:

1. $\frac{e^{v_i}}{\sum_{j \leq d} e^{v_j}} \in [0, 1]$
2. $\sum_{i \leq d} \left(\frac{e^{v_i}}{\sum_{j \leq d} e^{v_j}} \right) = 1$

Otra función de activación muy importante para la clasificación es la función RELU:

Definition 00.5 Definimos a la función $\boldsymbol{RELU} : \mathbb{R} \to \mathbb{R}$ utilizando la siguiente fórmula:

$$\boldsymbol{RELU}\,(x) = max\{0, -x\} \tag{02.4}$$



Remark 00.2 Utilizando función RELU es posible definir el algoritmo de entrenamiento del perceptrón como veremos más adelante.

Funciones de pérdida

Para pasar de la arquitectura de una red neuronal a una red neuronal, es necesario un proceso de entrenamiento utilizando algoritmos de optimización (que en su mayoría no serán convexos). A su vez para definir un problema de optimización es necesario contar con una función de pérdida.

En esta sección definiremos algunas de funciones de pérdida las más utilizadas.

Definition 00.6 Dada una base de datos de una regresión lineal

$$S = \left\{ (x_1, y_1), \dots (x_N, y_N) \right\} \tag{02.5}$$

tal que $(x, y) \in \mathbb{R}^d \times \mathbb{R}$ y una función $f : \mathbb{R}^d \to \mathbb{R}$, definimos el error de mínimos cuadrados de f como el promedio de los cuadrados de las diferencias entre $f(x_i)$ y las y_i.

$$err_S(f) = \frac{1}{N} \cdot \sum_{i \leq N} (f(x_i) - y_i)^2 \tag{02.6}$$

Esta función de pérdida normalmente se utiliza para calcular la regresión lineal.

Definition 00.7 Dada una base de datos de clasificación binaria

$$S = \left\{ (x_1, y_1), \dots (x_N, y_N) \right\} \tag{02.7}$$

tal que $(x, y) \in \mathbb{R}^d \times \{-1, +1\}$ y una función $f : \mathbb{R}^d \to \mathbb{R}$, definimos la función de pérdida de la entropía cruzada de f en (x, y):

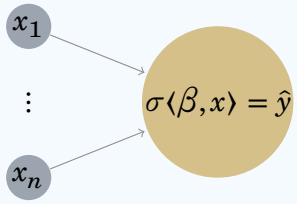
$$H(y, f(x)) = -\sum_{i \leq d} y_i \log(f(x_i)) \tag{02.8}$$

Perceptrón multicapa

Una de las redes que utilizaremos comúnmente, es la del perceptrón multicapa en la que las neuronas de una capa están conectadas con todas las neuronas de la siguiente capa.

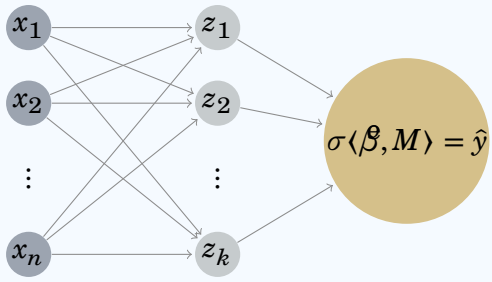
Una red neuronal densa con una capa es la arquitectura que conecta todas las d características (coordenadas) de $x = (x_1, \dots x_d)$ con una neurona $\hat{y}$, de tal manera que a cada característica se le asocia un peso.

El perceptrón (que estudiamos con profundiad en el curso de ML & IA) es una generalización de ésta idea.



En un perceptrón multicapa, los k nodos de la capa V_{t-1} están relacionados con cada uno de los nodos de V_t mediante una regresión logística, de tal modo que $z = \sigma\langle\beta, x\rangle$ para $x = (x_{t-1,1} \dots x_{t-1,k})$ y $\beta \in \mathbb{R}^k$ para cada $z \in V_t$. En este punto es preciso notar que β, es decir, los pesos, dependen de toda la capa anterior.

Si queremos hacer una predicción binaria para un conjunto de datos, como en 02.7, tendremos que añadir una capa final con un solo nodo, $\hat{y} = \sigma\langle\beta, M\rangle$ donde M es una matriz que contiene a todos los nodos de las capas internas, y σ es la función 02.2, como formalizaremos en la siguiente definición.



Definition 00.8 Si una capa intermedia tiene un vector de neuronas $x \in \mathbb{R}^n$ y la siguiente capa tiene un vector de neuronas $z \in \mathbb{R}^m$, entonces un perceptrón multicapa entre ellas con función de activación ρ es una matriz $M \in \mathbb{R}^{m \times n}$ y un vector $b \in \mathbb{R}^m$ tales que las entradas del vector w son:

$$w_j = \rho(\langle m_j, x\rangle + b_j) \tag{02.9}$$

Comúnmente lo abreviaremos con la notación: $w = \rho(Mx + b)$

La cuestión sobre cuántas capas se deben utilizar para abordar un caso como el de la predicción binaria, se estudiará más adelante y siempre depende del tipo de problema. Asímismo, hay heurísticas propias para determinar cuál función de activación será la óptima según el problema que se pretende resolver.

03 Physics-Informed Neural Networks (PINN)

En este capítulo hablaremos sobre las redes neuronales utilizadas para aproximar soluciones de ecuaciones diferenciales, estas pueden ser tanto simples como parciales.

Supongamos el caso sencillo de una ecuación diferencia de grado uno de una función $f(X)$ donde $x \in \mathbb{R}$:

$$\frac{\delta f(X)}{\delta X}(x) = F(x, f(x))$$

Con condición inicial $f(0) = C$.

La intención de este enfoque es aproximar la función $f(x)$ mediante alguna otra a la que llamaremos $\hat{f}(X)$. En este caso vamos a construir las aproximaciones mediante la siguiente fórmula:

$$\hat{f}(X) = C + Xf_\beta(X)$$

La función $f_\beta(X)$ será una red neuronal con parámetros entrenables β y una arquitectura por determinar. Debido a la fórmula anterior, $\hat{f}(X)$ satisface las condiciones iniciales.

El paso indispensable para poder aproximar a f es definir una función de pérdida adecuada:

$$l_{PINN}(\hat{f}; x_i) = \left(\frac{\delta \hat{f}(X)}{\delta X}(x_i) - F(x_i, \hat{f}(x_i)) \right)^2$$

Notemos que no es difícil calcular lo anterior ya que:

$$\frac{\delta \hat{f}(X)}{\delta X}(x_i) = \frac{\delta (C + Xf_\beta(X)))}{\delta X}(x_i) = f_\beta(x_i) + \frac{\delta f_\beta(X)}{\delta X}(x_i)$$

04 Caso de uso: valuación de opciones mediante Deep Learning

Durante el caso de uso presentaremos dos acercamientos a la valuación de opciones utilizando redes neuronales, en el primero se introduce una red neuronal densa y en el segundo una PINN. En ambos casos se busca comparar este acercamiento de deep learning contra las técnicas clásicas como la fórmula de Black y Scholes.