BOURBAKI

COLEGIO DE MATEMÁTICAS

BOURBAKI

COLEGIO DE MATEMÁTICAS

Índice

ÍNDICE

01 Introducción

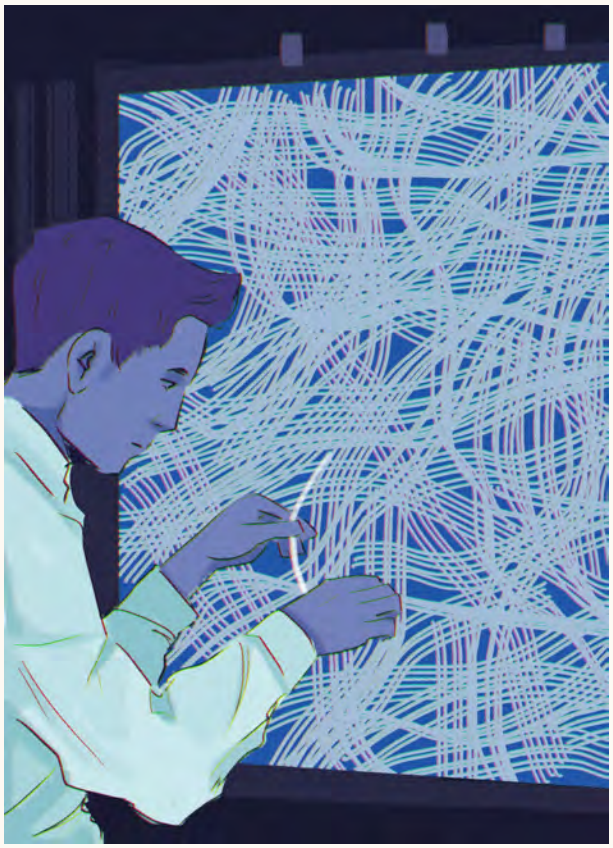
Las presentes notas son la bitácora del primero de los 6 módulos de nuestro curso Especialización en Deep Learning. En este curso trataremos modelos neuronales diversos para garantizar un mejor rendimiento dependiendo de la estructura interna de las bases de datos.

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

El curso es una invitación al Procesamiento del Lenguaje Natural sus aplicaciones a los resúmenes de texto, la organización de las clases es la siguiente:

1. Introducción a redes neuronales profundas (dos horas).
2. Introducción a Pytorch y capas densas (una hora).
3. Auto-encoders y missing values (dos horas).
4. Caso de uso sobre missing values (dos horas).
5. Dudas y complementos sobre redes neuronales (dos horas).
6. Asesoría sobre el proyecto (6 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.

Imputación de los valores faltantes

La falta de datos es un problema común que aparece en contextos reales y puede comprometer el rendimiento de la mayoría de los modelos de aprendizaje. Existen dos acercamientos ingénuos que podrían ayudarnos a imputar valores faltantes:

- Concentrándonos solo en las columnas, rellenar con el promedio de los valores no-faltantes.
- Concentrándonos solo en las columnas, rellenar con un muestreo de una distribución gaussiana con promedio y varianza calculada con los valores no-faltantes.

La desventaja de este método es que nos estamos concentando únicamente en las columnas de manera independiente y no en sus posibles interacciones, por medio de PCA y redes neuronales vamos a intentar mejorar este acercamiento.

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.

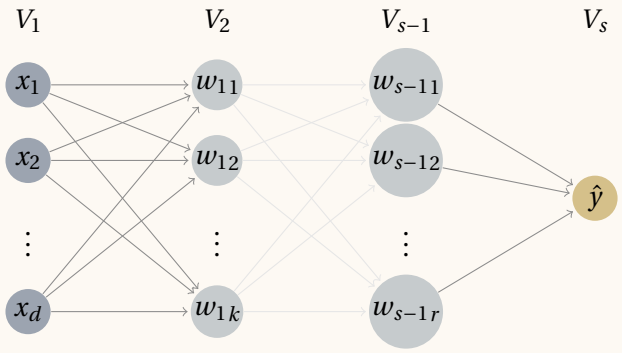
  A los elementos en V los llamaremos **neuronas** y a los elementos en E los llamaremos transformaciones lineales,

- Una función llamada **función de activación** $\rho : \mathbb{R} \to \mathbb{R}$

- Una partición disjunta del conjunto de vértices $V = V_1 \cup \ldots \cup V_s$ donde cada nodo en V_{t-1} está conectado a algún elemento de V_t.

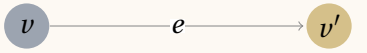
- El parámetro s será el número de **capas**,

- 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.



$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \xrightarrow{w_2(e)} \begin{pmatrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_m \end{pmatrix}$$

- Una función $f : X^d \to Y$ que puede calcularse utilizando la informa-

ción 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:

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

00.1 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:

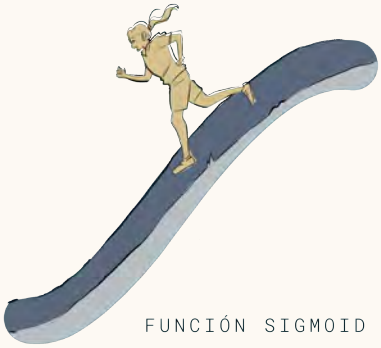
Definition 00.3. Definimos a la función sigmoide $\sigma : \mathbb{R} \to \mathbb{R}$ utilizando la siguiente fórmula:

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

Esta función es ampliamente utilizada por ejemplo en el algoritmo de la regresión logística debido a sus propiedades, por ejemplo, que es una fucnión 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:

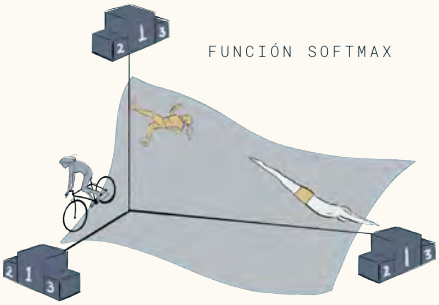
Definition 00.4. Dado un $v=(v_1,v_2,\ldots,v_d)\in\mathbb{R}^d$, definimos la función



FUNCIÓN SIGMOID

SoftMax de v como

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



FUNCIÓN SOFTMAX

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 $RELU : \mathbb{R} \to \mathbb{R}$ utilizando la siguiente fórmula:

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

Utilizando función RELU es posible definir el algoritmo de entrenamiento del perceptrón como veremos en la siguiente sección.

00.2 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

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

tal que $\left(x, y\right) \in \mathbb{R}^d \times \mathbb{R}$ y una función $f : \mathbb{R}^d \to \mathbb{R}$, definimos **el error de mínimos cuadrados** de f:

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

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

Definition 00.7. Dada una base de datos de clasificación binaria

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

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

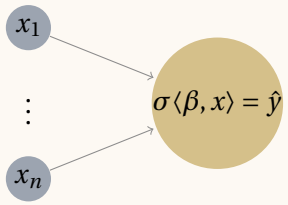
$$H\left(y, f\left(y\right)\right) = - \sum_{i \leq d} y_i log\left(f\left(y\right)_i\right)$$

00.3 Perceptrón multi-capa

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

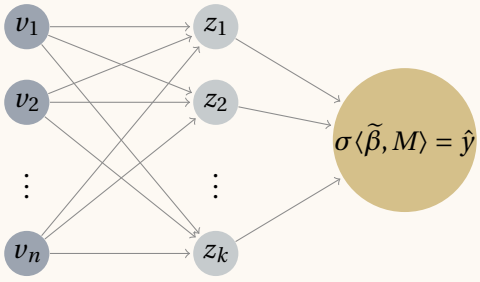
Una red neuronal densa con una capa es la arquitectura que conecta todas las d características (coordenadas) de (un vector) $x = (x_1, \dots x_d)$ con una neurona $\hat{y}$, de tal manera que a cada característica se le asocia un peso.

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 $x = (v_{t-1,1} \dots v_{t-1,k})$ y $\beta \in \mathbb{R}^k$ para cada $z \in V_t$. En este punto es preciso notar que β

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, $\hat{y} = \sigma\langle\widetilde{\beta}, M\rangle$ 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

multi-capa entre ellas con función de activación ρ es una matriz $M \in \mathbb{R}^{mn}$ y un vector $b \in \mathbb{R}^m$ tales que las entradas del vector w son:

$$w_j = \rho(\langle m_j, x \rangle + b_j) \tag{02.8}$$

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 estudará 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 Auto-encoders

En este capítulo hablaremos sobre dos de los algoritmos más utilizados en ciencia de datos a saber el Análisis de Componentes Principales y los Auto-encoders. Utilizaremos estos algoritmos para la imputación de datos.

Un ejemplo juguete de la reducción de la dimensión

Comenzaremos con un ejemplo de juguete en el cual podremos explicar las ideas de visualización relacionadas los algoritmos de la reducción de la dimensión.

Veamos la siguiente tabla que contiene las calificaciones que le dan Alfonso, Bárbara, Carlos y Diana a las siguientes frutas, siendo 10 la mayor y uno la menor.

Deseamos encontrar una manera de explicar fácilmente qué factores son tomados en cuenta para que estas personas prefieran o no una fruta.

El análisis de componentes principales resuelve el problema anterior mediante una generalización de las siguientes observaciones:

Supongamos que Alfonso, Bárbara, Carlos y Diana pueden ser representados por los siguientes vectores con sus preferencias sobre las frutas: $A = (10, 1, 2, 7)$, $B =$

$(7,2,1,10)$, $C = (2,9,7,3)$, $D = (3,6,10,2)$.

Para comenzar valdrá la pena escribir que el promedio de los 4 vectores es igual a $\mu = (5.5, 4.5, 5, 5.5)$. Si fijamos los siguientes dos vectores

$$d = (3,-3,-3,3), f = (1,-1,1,-1)$$

Utilizando los vectores μ, d y f la observación fundamental de PCA es la siguiente:

$$A \sim \mu + d + f = (9.5, 0.5, 3, 7.5)$$

$$B \sim \mu + d - f$$

$$C \sim \mu - d - f = (1.5, 8.5, 7, 3.5)$$

$$D \sim \mu - d + f$$

Las ecuaciones anteriores son enormemente informativas al leerlas con cuidado. La sugerencia para la primera es la siguiente:

- Salvo una constante μ que no depende de Alfonso pues está presente en la representación vectorial del resto de las personas, las preferencias

de Alfonso se diferencían de las del resto de personas únicamente por el signo del vector d y el del vector f, en su caso ambos son positivos.

Gracias a la interpretación anterior representación de los datos se puede simplificar enormemente de la siguiente manera: $\bar{A} = (1,1)\,, \bar{B} = (1,-1)\,, \bar{C} = (-1,-1)\,, \bar{D} = (-1,1)$. Esta es la razón por la que normalmente PCA se menciona como una forma para reducir la dimensión.

Recordando el problema de la interpretabilidad que deseamos resolver, nos gustaría entender cuáles son las características de las frutas que estas 4 personas toman en cuenta para decidir cuál les gusta. La respuesta a este problema que propone PCA son las coordenadas de nuestros puntos $\bar{A} = (1,1)\,, \bar{B} = (1,-1)\,, \bar{C} = (-1,-1)\,, \bar{D} = (-1,1)$, a las coordenadas de estos puntos las llamaremos d y f.

Regresando a los vectores en su dimensión original, un análisis de correlaciones nos dice que las variables Higo y Papaya están positivamente correlacionadas, de igual forma las variables Fresas y Uvas también están positivamente correlacionadas.

Bien entendido a la correlación entre Higo y Papaya la debemos asociar con la variable d, mientras que a la correlación entre Fresas y Uvas las asociamos con la variable f. Es en este momento cuando necesitamos un poco de imaginación:

1. £En qué sentido las variables Higo y Papaya están correlacionadas?

2. £En qué sentido las variables Fresas y Uvas están correlacionadas?

Nuestra respuesta que no debe de ser la última a la primera pregunta es la Dulcura (de ahí que hayamos elegido la letra D) y a la segunda pregunta es Frescura.

Gracias al análisis anterior, estas dos variables d, f son suficientes para explicar las dos características que más toman en cuenta este grupo de personas para decidir las frutas que les gustan. Más aún podemos decir que por ejemplo a Diana le gustan mucho las frutas que son frescas y le gustan menos las frutas que son dulces.

Exercise 01.1. *£Qué podemos decir sobre los gustos de Alfonso, Bárbara y Diana?*

Exercise 01.2. *Comparar los resultados del ejercicio anterior con la tabla inicial de sus calificaciones a las frutas. £Son consistentes?*

Exercise 01.3. *Además de toda la información anterior, una de las conclusiones de PCA es que la dulcura de las frutas es más importante para estas 4 personas sobre la frescura. Explicar utilizando los vectores en qué se basa esta conclusión.*

Análisis de componentes principales

En la sección anterior describimos un ejemplo muy sencillo de cómo funcio-

na la reducción de la dimensión vía PCA. En esta sección describiremos las distintas versiones de PCA y cómo puede entenderse como una técnica para reducir la dimensión. La tercera formulación que enunciaremos será la base para comprender cómo funcionan los auto-encoders.

La siguiente definición del análisis de componentes principales es bastante parecida a la definición de la regresión lineal, excepto que la manera de aproximar es distinta.

Definition 02.1. *(PCA como aproximación lineal) Sea* $X = \{x_1, \dots, x_N\} \in \mathbb{R}^d$ *y* $k < d$*. El problema de aproximación de k componentes principales corresponde con la solución del siguiente problema de minimización:*

$$V = \underset{V \subset \mathbb{R}^d, dim(V)=k}{argmin} \left(\frac{1}{N} \sum_{j \leq N} \left(d\left(x_j, V\right)^2 \right) \right)$$

Utilizando la ecuación de proyecciones ortogonales es posible re-escribir la ecuación de la definición 02.1 de la siguiente forma:

Proposition 02.2. *(PCA como aproximación de k vectores ortonormales) Sea* $X = \{x_1, \dots, x_N\} \in \mathbb{R}^d$ *y* $k < d$*. El problema de aproximación por los primeros k componentes principales* $v_1, \dots, v_k$ *corresponde con la solución del siguiente problema de minimización:*

$$(V, a_i, \dots, a_N) = \underset{V \in \mathbb{R}^{dk}, V^T V = Id_k, a_j \in \mathbb{R}^k}{argmin} \left(\frac{1}{N} \sum_{j \leq N} ||x_j - V a_j||_2^2 \right)$$

Aunque no es inmediato demostrar la siguiente proposición, es posible de-

mostrar lo siguiente:

Proposition 02.3. *(PCA como reducción de la dimensión) La solución de la ecuación en la definición 02.2 es equivalente a la siguiente:*

$$(V,U) = \underset{V\in\mathbb{R}^{dk},U\in\mathbb{R}^{kd}}{argmin}\left(\frac{1}{N}\sum_{j\leq N}||x_j - VUx_j||_2^2\right)$$

En el análisis de componentes principales las funciones que reducen la dimensión son funciones lineales sin embargo los auto-encoders utilizan funciones no lineales para modificar el espacio en el que viven nuestros datos.

Autoencoders

Los autoencoders son un algoritmo muy similar a la última versión que dimos sobre PCA solo que esta vez permitiremos que las transformaciones entre la dimensión original de los datos y la dimensión donde los enviamos sea no-lineal. P

Definition 03.1. *(Autoencoder) Un autoencoder es la solución al siguiente problema de minimización donde f, g son dos redes densas:*

$$(f,g) = \underset{f,g}{argmin}\left(\frac{1}{N}\sum_{j\leq N}||x_j - g(f(x_j))||_2^2\right)$$

En este caso la reducción de los datos es $f(x_j)$.

Remark 03.2. *Es importante mencionar que aunque esta vez hemos utilizado la distancia euclidiana para medir la diferencia entre nuestros datos es posible usar otras métricas.*

Estategia para imputación de datos

Supongamos que tenemos acceso a un conjunto de datos con valores faltantes al que llamaremos X_{train} al que un experto (o un histórico) le ha imputado algunos de los valores faltantes, a este nuevo conjunto lo llamaremos Y_{train}.

La estrategia consiste en llenar los datos faltantes de Y_{train} por un método ingénuo para construir Y'_{train}. Ahora entrenaremos un modelo (ya sea auto-encoder o PCA) que nos arroje una transformación f a un conjunto de variables latentes y otra transformación g al conjunto de variables originales.

Utilizando una imputación de datos ingénua para X_{train} (digamos X'_{train}) podemos evaluar nuestras funciones f, g para construir nuestras predicciones de imputación $\hat{Y}_{train}$. La evaluación del modelo se hará comparando $\hat{Y}_{train}$ con Y_{train}.

Cuando recibamos un nuevo conjuto de datos X_{test} podemos imputar datos ingénuamente y obtener X'_{test}. Con las funciones f, g lograremos hacer una predicción para los valores faltantes en X_{test}.

04 Complementos sobre entrenamiento de redes neuronales profundas

En esta sección agregaremos algunos complementos del curso que trataremos en los últimos días de la semana.

El método del gradiente

El método del gradiente es la técnica más utilizada en deep learning para entrenar a una red neuronal. Comenzamos con un caso muy sencillo para el caso de las regresiones lineales.

01.1 La derivada

En esta sección introduciremos las derivadas en una sola dimensión y más adelante serán necesarias las derivadas respecto a más de una variable.

Definition 01.1. Si $l : \mathbb{R} \to \mathbb{R}$ es una función, entonces diremos que l es diferenciable en un punto $x \in \mathbb{R}$ cuando el siguiente límite existe:

$$\lim_{\delta \to 0} \frac{l(x+\delta) - l(x)}{\delta}$$

Cuando existe ese límite, lo denotaremos $l'(x)$ y llamaremos "la derivada de l en el punto x".

A partir de ahora nos interesarán las funciones l que se utilizan como métrica para un algoritmo, por ejemplo la función que definimos en el capítulo pasado. Supongamos que $S = \{(x_i, y_i)\}_{i \leq N}$ es una base de datos asociada a una regresión univariada (i.e. $d = 1$) tal que la señal de esta función satisface: $f^*(X) = mX$, es decir que $\beta_0 = 0$, entonces

$$l(m) = err_S(m) = \frac{1}{N} \cdot \sum_{i \leq N} \left(mx_i - y_i\right)^2$$

Exercise 01.1. *Calcular la derivada de l en el ejemplo anterior.*

Definition 01.2. Si $l : \mathbb{R}^d \to \mathbb{R}$ es una función, entonces diremos que l es diferenciable en un punto $x \in \mathbb{R}^d$, con respecto a la coordenada $j \leq d$ cuando el siguiente límite existe:

$$\lim_{\delta \to 0} \frac{l(x_1, \ldots, x_{j-1}, x_j + \delta, x_{j+1}, \ldots, x_d) - l(x_1, \ldots, x_d)}{\delta}$$

Cuando exista ese límite, lo denotaremos $\frac{\delta}{\delta x_j} f(x)$ y llamaremos "la derivada parcial de l en el punto x y con dirección j".

Exercise 01.2. *Calcule la derivada de l en el ejemplo anterior, esta vez cuando* $\beta_0 0$.

01.2 El método del gradiente de Cauchy

El método del gradiente es un algoritmo que es comúnmente utilizado en Machine Learning para el proceso de entrenamiento de diversos modelos. En esta sección supondremos que $l : \mathbb{R}^d \to \mathbb{R}$ es una función que entenderemos como la función de error en algún modelo, por ejemplo podríamos suponer que $l(\beta) = \frac{1}{N} \sum_{i \leq N} \left(y_i - \langle \beta, x_i \rangle\right)^2$ en el caso de una regresión lineal.

Definition 01.3. Sea f como en el párrafor anterior y $\beta \in \mathbb{R}^d$, para $\nu > 0$ definimos el algoritmo del gradiente descendente de la siguiente manera:

1. $\beta_0 = (1, 1, \ldots, 1)$
2. $\beta_{t+1} = \beta_t - \nu \nabla l(\beta_t)$

Si desenmascaramos con cuidado la definición anterior obtenemos lo siguiente:

$$\beta_{t+1,j} = \left(\beta_{t,1} - \nu \frac{\delta f}{\delta \beta_{t,1}}(\beta_t), \ldots, \beta_{t,j} - \nu \frac{\delta f}{\delta \beta_{t,j}}(\beta_t), \ldots, \beta_{t,d} - \nu \frac{\delta f}{\delta \beta_{t,d}}(\beta_t)\right)$$

La idea principal detrás del método del gradiente se podría resumir para el caso uno-dimensional de la siguiente manera, como lo mencionamos anteriormente estamos buscando un β tal que $l(\beta) = 0$. Cuando $d = 1$ y sabemos que $\beta_0 = 0$ entonces la función de pérdida l depende únicamente de un parámetro. Si hemos encontrado un $\beta' \in \mathbb{R}$ que aún no nos satisface, entonces

podrían ocurrir los siguientes dos casos:

$$l(\beta') \geq 0, l(\beta') \leq 0$$

Sabemos que estos casos corresponden respectivamente a cuando al aumentar el valor de β', el error $l(\beta')$ aumenta y cuando al disminuir el valor de β', el error $l(\beta')$ aumenta. Por eso en el primero de los casos deseamos restarle una cantidad positiva a β', a saber $l'(\beta')$ y en el segundo caso deseamos sumarle una cantidad positiva, a saber $-l'(\beta')$.

Remark 01.3. *Notemos que por cálculo clásico sabemos que si una función es diferenciable un punto, entonces al rededor de ese punto se tiene la siguiente aproximación lineal, si β es cercana a β_t, entonces*

$$f\left(\beta\right) \sim f\left(\beta_t\right) + \langle \beta - \beta_t, \nabla f\left(\beta_t\right) \rangle$$

El parámetro ν deberá garantizar que la aproximación no se aleje demasiado de β.

Cuando la función f satisface algunas condiciones de regularidad y convexidad, es posible garantizar rápidamente la convergencia del método del gradiente, sin embargo algunas veces podría ser problemático en teoría, afortunadamente para las redes neuronales profudnas estas técnicas funcionan muy bien.

01.3 Método del gradiente estocástico

Uno de los casos principales cuando el método del gradiente podría no converger es cuando la cantidad de datos es demasiado extensa i.e. $N >> 1$, en este caso el método del gradiente estocástico permite solucionar la velocidad de convergencia.

Supongamos que $f(\beta)$ es una función que podemos escribir de la siguiente manera:

$$f(\beta) = \frac{1}{N} \sum_{i \leq N} f_i(\beta)$$

Por ejemplo la funición de error de las regresiones lineales lo satisface, en este caso es inmediato que

$$\nabla f(\beta) = \frac{1}{N} \sum_{i \leq N} \nabla f_i(\beta)$$

En este caso el algoritmo de actualización del gradiente estocástico se define de la siguiente manera:

1. $\beta_0 = (1, 1, \ldots, 1)$
2. $\beta_{i+1} = \beta_i - \nu \nabla f_j(\beta_i)$

Para algún $j \leq N$, por su puesto que la pregunta más importante en este caso es la siguiente: £cómo elegir j?

Supongamos por ejemplo que elejimos j de manera aleatoria y uniforme en $\{1,2,3,\ldots,N\}$. En este caso es claro que β_{i+1} será una variable aleatoria que depende de j, calculando su esperanza:

$$\mathbb{E}\left(\beta_{i+1}\right) = \frac{1}{N}\sum_{j\leq N}\left(\beta_i - \nu\nabla f_j\left(\beta_i\right)\right) = \beta_i - \nu\nabla f\left(\beta_i\right)$$

Lo anterior es muy interesante porque significa que en promedio el efecto del método del gradiente estocástico será igual al del gradiente determinista, esta vez con la enorme ventaja de solo requerir d cálculos.

05 Caso de uso

A medida que aumenta el número de inversores y de personas dispuestas a comprometerse con las preocupaciones medioambientales y sociales, las características extrafinancieras de las empresas (también denominadas ESG por Environment, Social and Governance) cobran cada vez más importancia.

Impactfull es una empresa que ofrece asesoramiento sobre finanzas sostenibles y datos extrafinancieros basados en datos brutos verificados procedentes de las empresas. Sus datos se componen de más de 30 indicadores (dentro de las categorías medioambiental, social y de gobernanza) y se extraen de los informes de sostenibilidad.

Pladifes es un proyecto de investigación alojado en el Institut Louis Bachelier, una asociación sin ánimo de lucro que promueve la investigación en economía y finanzas.

Este reto es el resultado de una colaboración entre las dos partes mencionadas, como una oportunidad para ambas de ganar visibilidad y permitir a los estudiantes trabajar con (esperemos) interesantes datos financieros adicionales.

El equipo de Pladifes también tiene previsto incluir la mejor propuesta del reto en su base de datos, lo que permitirá a los investigadores utilizarla para proyectos académicos.

Objetivos del reto

El objetivo del reto es predecir los valores que faltan para 15 indicadores extrafinancieros corporativos (hasta un 96 porciento de valores perdidos). Estos indicadores están disponibles durante tres años (2018, 2019, 2020) y provienen de divulgaciones de sostenibilidad.

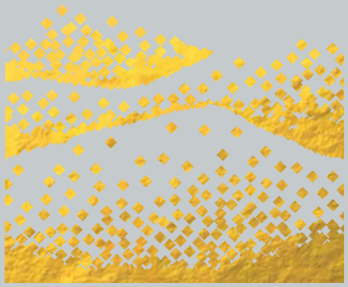
El objetivo es, por tanto, entrenar un modelo de imputación de valores perdidos en los datos de entrenamiento y utilizarlo en los datos de prueba para rellenar los huecos.

Descripción de los datos

Los datos de entrada contienen 15 indicadores financieros adicionales sobre 10 000 empresas durante un máximo de tres años (2018, 2019, 2020).

Cada línea está definida por un "ID" único y corresponde a una empresa determinada y a un año determinado. Los datos se han seleccionado de forma que no haya más de un 96 porciento de valores perdidos para un indicador determinado. Las empresas son anónimas y se dividen en un conjunto de entrenamiento y otro de prueba, de modo que una empresa determinada sólo puede encontrarse en el conjunto de entrenamiento o en el de prueba.

La primera línea del archivo de entrada contiene la cabecera, y las columnas están separadas por comas. El tamaño total es inferior a 10Mo.



escuela-bourbaki.com