



BOURBAKI

COLEGIO DE MATEMÁTICAS

Índice

01. Introducción _____ pág. **03**

 01. ¿Qué es Machine Learning? _____ pág. 03

 02. Notación matemática _____ pág. 04

02. Instalación de Python y un entorno de trabajo para MachineLearning _____ pág. **05**

 01. Instalación de Miniconda [3] _____ pág. 05

 02. Creación de un entorno virtual para análisis de datos _____ pág. 07

03. Perceptrón _____ pág. **10**

 01. Justificación teórica del algoritmo del perceptrón _____ pág. 12

04. Clasificador binario de imágenes microscópicas _____ pág. **14**

05. Redes Neuronales	_____	pág. 15
01. Capacidad expresiva	_____	pág. 15
02. Datos casi-linealmente separables	_____	pág. 16
03. Método del gradiente	_____	pág. 17
04. Margen, sobre-ajuste y regularización	_____	
	pág. 19	
05. Redes neuronales convolucionales y otras	_____	
funciones de activación	_____	pág. 20
06. Referencias	_____	pág. 22

01 Introducción

Las siguientes notas son la bitácora de un curso de 9 horas que impartimos en una alianza con Data Science Institute. Además de este documento los invitamos a consultar el Github del curso [en este link](#).

El curso es una invitación al uso de redes neuronales para problemas de clasificación binaria, nuestro ejemplo principal de algoritmo es el algoritmo del perceptrón, el curso está dividido en clases de la siguiente manera:

1. ¿Qué es Machine Learning? (una hora)
2. Un vistazo a Python (una hora)
3. Descripción formal del algoritmo del Perceptrón (dos horas)
4. Implementación del algoritmo del perceptrón para la clasificación de imágenes (dos horas)
5. Redes neuronales en general (tres horas)

¿Qué es Machine Learning?

Machine learning es un conjunto de técnicas que nos permiten hacer predicciones utilizando información del pasado. En este curso hablaremos con

detalle del perceptrón que es un primer ejemplo de un algoritmo que produce una red neuronal para hacer predicciones. El perceptrón es un algoritmo que corresponde a la fase de entrenamiento de una red neuronal de una capa, fue ideado por F. Rosenblatt.

Notación matemática

- En estas notas denotaremos por \mathbb{R} al conjunto de los números reales es decir todos los números negativos o positivos que conocemos (incluyendo algunos como π etc.).
- Un subconjunto importante de los números reales es \mathbb{N} llamada el conjunto de los números naturales y consiste en los siguientes números $\mathbb{N} = \{1, 2, 3, \dots\}$.
- Si $d \in \mathbb{N}$ es un número natural entonces denotaremos por \mathbb{R}^d al conjunto de vectores (x_1, x_2, \dots, x_d) de tamaño d con entradas en \mathbb{R} . Por ejemplo \mathbb{R}^2 es el plano cartesiano.
- Si $x = (x_1, \dots, x_d), y = (y_1, \dots, y_d) \in \mathbb{R}^d$ entonces el producto punto de x, y se denota por $\langle x, y \rangle$ y es el número real

$$\langle x, y \rangle = x_1 y_1 + \dots + x_d y_d$$

02 Instalación de Python y un entorno de trabajo para MachineLearning

La forma más rápida y eficiente de instalar Python, manejar sus librerías y evitar problemas de dependencia(por actualizaciones y versiones) es mediante el uso de Miniconda que además de la instalación de Python, instala conda[1] un sistema gestor de paquetes y de entornos virtuales¹ [2].

Una forma alternativa es utilizar Anaconda, que al igual que Miniconda, es soportada por la misma compañía Anaconda. Ambas alternativas instalan la misma versión de Python y de conda. La razón por la que se recomienda Miniconda es porque permite instalar solamente lo que se requiere para este curso, con la opción de descargar otros paquetes cuando se requieran. Anaconda por el contrario instala por defecto, una cantidad excesiva de paquetes que raramente son utilizados, lo cual requiere demasiado espacio y tiempo.

Instalación de Miniconda [3]

Descarga aquí el archivo correspondiente a tu sistema operativo, ya sea que

¹Los entornos virtuales permiten controlar las versiones de software (en nuestro caso python y sus librerías) usado para análisis o aplicaciones

sea que MacOS, Windows y Linux, eligiendo la versión de Python 3.7.

Instalación para Windows

1. Da doble clic en el archivo descargado.
2. Sigue las instrucciones que se muestren aceptando las opciones que se proponen por defecto (si es necesario se pueden cambiar después).
3. Desde el menú de Inicio de Windows abre el programa Anaconda Prompt.

Instalación para MacOS

1. En la Terminal de tu Mac, navega hasta el directorio en donde descargas el instalador y corre la siguiente línea: *bash Miniconda3-latest-MacOSX-x86_64.sh*
2. Sigue las instrucciones, aceptando las opciones por defecto (si es necesario se pueden cambiar después)
3. Cierra la Terminal y vuélvela a abrir, para que los cambios sean actualizados.

Instalación para Linux

1. En la Terminal de Linux, navega hasta el directorio en donde descarga el instalador y corre la siguiente línea: *bash Miniconda3-latest-Linux-x86_64.sh*
2. Sigue las instrucciones, aceptando las opciones por defecto (si es necesario se pueden cambiar después)
3. Cierra la Terminal y vuélvela a abrir, para que los cambios sean actualizados.

Para todos los sistemas operativos:

4. Para ver todos los paquetes que instalados por defecto en la distribución de Miniconda, corre la siguiente línea: *conda list*

Creación de un entorno virtual para análisis de datos

Un entorno virtual es un ambiente de trabajo aislado, lo que permite instalar determinadas librerías o versiones de librerías sin que afecte al resto del sistema principal o de otros ambientes.

Por defecto, Miniconda crea un entorno llamado base que estará activo cuando abramos la terminal. Este entorno contiene todos los programas enlistados cuando se utilizó el comando *conda list*. Aunque podríamos usar este entorno, una mejor práctica es crear un entorno nuevo. Para ello es conveniente

cambiar la configuración de conda para que no abra automáticamente el entorno base.

Desactivar el entorno base

En Mac y Linux, se debe correr la linea que se muestra a continuación y enseñada cerrar y volver a abrir la terminal: *conda config set auto_activate_base false*

Los usuarios de Windows deberán desactivar manualmente el entorno base utilizando *conda deactivate*

Creación de un entorno específico para análisis de datos mediante **conda-forge**

El entorno que vamos a crear lo titularemos MachineLearning, en éste instalaremos las librerías que necesitaremos para nuestros análisis.

1. Crea el ambiente de trabajo mediante la siguiente línea: *conda create -n MachineLearning*
2. Actívalo con la siguiente instrucción: *conda activate MachineLearning*
3. Agrega el canal² *conda config --env --add channels conda-forge* .

²Los repositorios desde donde podemos descargar las librerías de Python se llaman canales, Anconda, Inc provee por defecto el canal llamado default, sin embargo permite a cualquier usuario crear su propio canal si necesita otros paquetes que no están incluidos en el canal default, por esta razón existe una variedad de canales disponibles en la web desde donde se

Se recomienda utilizar el canal conda-forge, ya que es resultado de un esfuerzo colectivo con una gran variedad de librerías y paquetes que son cuidadosamente actualizados y donde se asegura tener versiones compatibles para macOS, Linux y Windows[4].

Puedes ver los canales instalados mediante la siguiente línea: *conda config -show channels*

Cada canal tiene al menos una dirección URL asociada donde se localizan los repositorios de paquetes y librerías. Puedes ver estas direcciones utilizando: *conda info*

4. Instala las librerías Pandas³, Scikit-learn⁴, Scikit-image⁵, Matplotlib⁶ y Jupyter Notebooks⁷ copiando el siguiente comando (es importante copiarlo en una sola línea, separando mediante un espacio, cada librería a ser instalada):

```
conda install pandas matplotlib notebook jupyter_contrib_nbextensions scikit-image scikit-learn
```

pueden descargar librerías de Python.

³Pandas es la principal librería de python utilizada Analisis de datos. Al instalar Pandas, por defecto se instala Numpy, sobre la cual funciona Pandas. Numpy es ampliamente usada en Ciencia de Datos porque permite el fácil manejo de matrices y sus operaciones.

⁴Scikit-learn es una importante librería para machine learning

⁵Es una librería para el procesamiento de imagenes

⁶Matplotlib es las principales librerías de Python usadas para graficar y visualizar información

⁷Jupyter notebook es una aplicación que nos ayudará a hacer nuestros análisis paso a paso, crear visualizaciones e incluir comentarios, como si se tratara de nuestro cuaderno de apuntes.

03 Perceptrón

A lo largo del curso supondremos que estamos en un problema clásico de aprendizaje supervisado tipo clasificación binaria en \mathbb{R}^d , donde buscamos entrenar a nuestro modelo con un conjunto de ejemplos

$$S = \{(x_1, z_1), \dots, (x_N, z_N)\}$$

cuando $x_i \in \mathbb{R}^d, z_i \in \{-1, +1\}$.

El perceptrón es un algoritmo que construye una red neuronal de una sola capa y tantos nodos como cualidades tengan nuestros ejemplos.

Para aligerar la notación, en esta sección supondremos que la última coordenada de todos nuestras observaciones x_i es igual a uno i.e. $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d-1}, 1)$. Hacer esta suposición siempre es posible si en lugar de considerar nuestras observaciones en \mathbb{R}^d lo hacemos en el hiperplano $X_{d+1} = 1$ en \mathbb{R}^{d+1} .

Exercise 00.1. *Si por ejemplo nuestro conjunto de ejemplos S viviera en el plano cartesiano, la hipótesis anterior significa en lugar de tratar a $x = (x_1, x_2) \in \mathbb{R}^2$, agregaremos una coordenada extra igual a uno, lo que corresponde, en el espacio tres dimensional a trabajar únicamente en el plano $Z = 1$.*

Haremos una suposición llamada de separabilidad que puede parecer exagerada sin embargo será necesaria para el tratamiento matemático formal, más tarde explicaremos cuándo es plausible suponerla:

- (Separabilidad) Los puntos de S son separables por una función lineal, eso quiere decir que existe un hiperplano $H \subseteq \mathbb{R}^d$ tal que de un lado de este hiperplano están todos los puntos clasificados con $+1$ y del otro lado estarán todos aquellos clasificados con -1 . Formalmente la hipótesis de separabilidad significa lo siguiente:

Existe un vector $\beta^* \in \mathbb{R}^d$ de tal forma que siempre que $z_i = 1$ entonces $\langle x_i, \beta^* \rangle > 0$ y si $z_i = -1$ entonces $\langle x_i, \beta^* \rangle < 0$ (el caso en el que algún ejemplo x_i satisfaga la igualdad corresponde al hecho geométrico de pertenecer al espacio definido por β^*).

Exercise 00.2. Sea S una base de datos tal que los $x_i = (x_i, y_i)$ pertenecen al plano cartesiano. Convencerte de que la hipótesis de separabilidad en este caso significa lo siguiente: existen números reales $m, b \in \mathbb{R}$ tales que $\langle \beta^*, (x_i, y_i, 1) \rangle = \langle (m, -1, b), (x_i, y_i, 1) \rangle$ es mayor o menor a cero.

Sea $\beta^* \in \mathbb{R}^d$ los parámetros del hiperplano que separa a nuestros ejemplos.

Definamos la función $sign_{\beta^*} : \mathbb{R}^d \rightarrow \{-1, 1\}$ tal que $x \mapsto 1$ si $\langle x, \beta^* \rangle > 0$ y $x \mapsto -1$ si $\langle x, \beta^* \rangle < 0$. Utilizando esta función nuestra hipótesis de separabilidad se puede escribir de la siguiente manera $sign_{\beta^*}(x_i) = z_i$ para cualquier $i \leq N$. Eso implica que para cualquier $i \leq N$ se tiene $z_i \cdot sign_{\beta^*}(x_i) > 0$.

El algoritmo del perceptrón busca descubrir los valores de β^* únicamente conociendo S , se define inductivamente de la siguiente manera:

Definition 00.1. (Algoritmo del percetrón)

1. Comenzamos con $\beta_0 = (1, 1, \dots, 1)$.

2. Si suponemos que β_i está definido, buscamos algún ejemplo $(x_j, z_j) \in S$ tal que lo siguiente NO sea cierto: $z_j \cdot \text{sign}_{\beta_i}(x_j) > 0$.
3. Si no encontramos algún índice j que cumpla los criterios de nuestra búsqueda entonces β_i es nuestra propuesta para clasificar puntos entrenada con S .
4. Si por el contrario encontramos algún índice j que cumpla los criterios de nuestra búsqueda entonces actualizamos a $\beta_{i+1} := \beta_i + z_j x_j$.
5. Volvemos al paso dos.

Si el algoritmo anterior se detiene después de m -pasos, entonces hemos encontrado una red neuronal $f : \mathbb{R}^{d+1} \rightarrow \{-1, +1\}$ de una capa y $d + 1$ neuronas definida por $f(x) = \text{sign}_{\beta_m}(x)$.

Justificación teórica del algoritmo del perceptrón

Definamos $R = \max_{i \leq N} \langle x_i, x_i \rangle$ y $B = \min_{\beta \in \mathbb{R}^d} \{\langle \beta, \beta \rangle : y_1 \cdot \langle x_1, \beta \rangle > 0, \dots, y_N \cdot \langle x_N, \beta \rangle > 0\}$.

Theorem 01.1. *Bajo la hipótesis de separabilidad sobre S el algoritmo del perceptrón se detiene (es decir encuentra alguna β que clasifica correctamente a los ejemplos en S) en $(RB)^2$ -pasos.*

Desafortunadamente la geometría del problema algunas veces implica que
tiempo en el que el perceptrón se detiene es demasiado grande.

04 Classificador binario de imágenes microscópicas

A continuación incluimos una breve descripción del data set que utilizaremos para exemplificar la implementación del Perceptrón lineal. Generaremos un modelo de clasificación binaria de un subconjunto de imágenes provenientes del Data Challenge by Mauna Kea [Link of the challenge.](#)

Para este clasificador se utilizaron solamente las imágenes de tejido sano y las imágenes de tejido con displasia/cáncer. De manera que el conjunto de datos está formado por 1,469 imágenes de tejido sano (clase 0) y 3,594 imágenes de displasia/cáncer (clase 1). Las imágenes originales fueron reducidas de 519x521 pixeles a 260x260 para reducir el tiempo y la memoria requeridos para el procesamiento. El conjunto de imágenes utilizadas están disponibles [aquí.](#)

El archivo ClasesImagenes.csv contiene una tabla donde se identifica el nombre de cada imagen y la clase a la que pertenece.

05 Redes Neuronales

La mayoría de las redes neuronales que se utilizan en la actualidad son más complejas que un perceptrón, en particular Deep Learning se caracteriza por tener diversas capas, en esta sección hablaremos brevemente de otros ejemplos de redes neuronales así como de las ventajas y desventajas del percetrón simple.

Capacidad expresiva

Ya explicamos anteriormente que el resultado del algoritmo del perceptrón es una red neuronal de una sola capa, con d neuronas en la entrada y una sola en la primera capa. Es posible construir redes neuronales con tantas capas y neuronas dentro de estas capas como uno desee, esto ayuda por ejemplo a mejorar la expresividad en la clasificación en el siguiente sentido. Supongamos que tenemos una base de datos binaria S en \mathbb{R}^2 que no es linealmente separable, por ejemplo una base de datos tal que $z_i = 1$ si y solo si $(\langle x_i, (1, -1, 0) \rangle < 0 \text{ y además } \langle x_i, (-1, -1, 0) \rangle < 0)$ o $(\langle x_i, (1, -1, 0) \rangle > 0 \text{ y además } \langle x_i, (-1, -1, 0) \rangle > 0)$.

Es posible construir una red neuronal con dos capas, dos neuronas c_1, c_2 en la primera capa y una sola neurona c_3 en la segunda capa que clasifique correctamente la base de datos anterior:

- Sea $c_1 = \text{sign}_{(1,-1,0)}(x, y, 1)$,
- Sea $c_2 = \text{sign}_{(-1,-1,0)}(x, y, 1)$,
- Sea $c_3 = c_1 \cdot c_2$.

Exercise 01.1. Construir una red neuronal que sea capaz de clasificar correctamente una base de datos en \mathbb{R}^2 tal que $z_i = 1$ si y solo si $\langle x_i, (1, -1, 0) \rangle > 0$ y además $\langle x_i, (-1, -1, 0) \rangle > 0$.

El entrenamiento que genere una red neuronal como las anteriores es un poco más complicado y requiere otro algoritmo distinto al percetrón.

Datos casi-linealmente separables

La versión del algoritmo que utilizan la mayoría de las librerías es una variante del algoritmo del percetrón anteriormente estudiado. Recuerden que su eficacia solo está garantizada en el caso cuando la información es linealmente separable. En esta sección presentaremos una variante del perceptrón útil cuando solo una pequeña porción de los datos no son linealmente separables.

Definition 02.1. Perceptrón parcial:

- (Etapa 0)

1. Comenzamos con $\beta_0 = (1, 1, \dots, 1)$.

2. Definimos un parámetro β_s y le asignamos el valor $\beta_s = \beta_0$
3. Definimos un parámetro $Ite_s(0)$ y le asignamos el valor $Ite_s(0) = 0$
 - (Etapa i)
 1. El perceptrón se actualizará de manera usual como en el algoritmo 00.1, sin embargo esta vez guardaremos el tamaño I de la cadena exitosa de pruebas de ejemplos en S (i.e. β_{i-I} clasificó correctamente a I ejemplos en S).
 2. Se actualizará $Ite_s = I$ cuando $Ite_s < I$.
 3. Se actualizará $\beta_s = \beta_{i-I}$.

Remark 02.1. *El algoritmo anterior tiene un problema, podría actualizar al vector β_s por un vector inferior a los probados anteriormente pues solo guarda la información de la última iteración.*

Theorem 02.2. *Si los coeficientes β_i son números racionales entonces el algoritmo anterior, con probabilidad uno converge a una solución óptima.*

Método del gradiente

Algunas veces no es realista suponer que tenemos acceso a una base de datos S suficientemente grande, por el contrario sí tenemos acceso a un flujo constante de información y además podemos saber cuándo nos hemos

equivocado con alguna predicción de inmediato, pensemos por ejemplo en el problema de clasificación de emails fraudulentos.

El método de entrenamiento que no supone una base de datos etiquetada sino una base de datos $S = \{x_i\}_{i \leq N}$ y el acceso a verificar si nuestra predicción z'_i es correcta o no se le llama aprendizaje en línea. Vale la pena notar que el algoritmo 00.1 en realidad es un algoritmo de aprendizaje en línea. Existe una forma alternativa de actualizar al perceptrón y es importante conocerla pues la mayoría de las redes neuronales que generalizan la que construye el perceptrón se actualizan con el método del gradiente que describiremos a continuación.

Definition 03.1. (Función de pérdida Hinge) En el tiempo i definimos la función de Hinge $H_i : \mathbb{R}^d \rightarrow \mathbb{R}$ de la siguiente manera: $H_i(\beta) = \max\{0, 1 - z_i(\beta, x_i)\}$.

El algoritmo del gradiente descendente del perceptrón consiste en minimizar la función H_i :

- $\beta_0 = (1, \dots, 1)$
- $\beta_{i+1} = \beta_i - \alpha_i \nabla H_i(\beta_i)$

Notemos que aún no hemos dicho qué significa α_i , en la siguiente sección hablaremos sobre su significado.

Margen, sobre-ajuste y regularización

En esta sección hablaremos sobre uno de los problemas más comunes y que pueden convertir machine learning en una experiencia desagradable: el sobre-ajuste.

Definition 04.1. Si hemos fijado un tiempo t y una base de datos S_t , supongamos que utilizaremos un algoritmo A para predecir un modelo M , se dice que el modelo M ha incurrido en sobre-ajuste relativo a la información anterior cuando M incluye sistemáticamente el ruido estocástico (aleatorio) de la base S_t .

El proceso de regularización en Machine Learning es un método para evitar el sobre-ajuste, existen distintas maneras de intentarlo y su eficacia no solo depende de nuestras bases de datos sino del algoritmo que estamos utilizando, en el caso del perceptrón la regularización se puede interpretar como un margen entre el hiperplano que predecimos y nuestra base de datos.

Definition 04.2. El algoritmo del perceptrón regularizado con un parámetro $\alpha > 1$:

1. Comenzamos con $\beta_0 = (1, 1, \dots, 1)$.
2. Si suponemos que β_i está definido, buscamos algún ejemplo $(x_j, z_j) \in S$ tal que lo siguiente NO sea cierto: $z_j \cdot \text{sign}_{\beta_i}(x_j) > 0$.
3. Si no encontramos algún índice j que cumpla los criterios de nuestra

búsqueda entonces β_i es nuestra propuesta para clasificar puntos entrenada con S .

4. Si por el contrario encontramos algún índice j que cumpla los criterios de nuestra búsqueda entonces actualizamos a $\beta_{i+1} := \beta_i + \alpha z_j x_j$.
5. Volvemos al paso dos.

Notemos que si $\alpha = 1$ entonces el algoritmo del perceptrón regularizado es el algoritmo usual. La regularización también tiene ventajas en la eficiencia computacional del perceptrón como lo muestra el siguiente resultado que generaliza el teorema de la sección 01.

Theorem 04.1. *Supongamos que S es separable linealmente con un margen δ el algoritmo regularizado del perceptrón se detiene (es decir encuentra alguna β que clasifica correctamente a los ejemplos en S) en $(\frac{R}{\delta}B)^2$ -pasos.*

Es importante notar que no existe una relación sencilla de explicar entre el margen δ y α .

Redes neuronales convolucionales y otras funciones de activación

En la sección 5.1 ejemplificamos cómo utilizando más capas en una red neuronal (y neuronas) es posible aumentar la capacidad expresiva de una red

neuronal, no es difícil convencerse y demostrar matemáticamente que si uno aumenta la cantidad de capas suficientemente es posible aproximar cualquier relación funcional entre las entradas x_i de una base de datos y el etiquetado z_i . Lo anterior aunque podría parecer deseable no es ideal si se hace sin cuidado, es decir aumentar la cantidad de capas y neuronas sin control, el problema de hacerlo es que rápidamente podríamos generar overfitting en nuestras predicciones i.e. nuestra red neuronal se entrenó muy bien en la base de datos S sin embargo no tiene buena predicción en nuevos ejemplos. Para remediarlo se consideran redes neuronales que por un lado son profundas i.e. tienen varias capas y muchas neuronas sin embargo no todas las neuronas deben estar conectadas entre sí. Un ejemplo de tales redes son las Redes Neuronales Convolucionales las cuales son el state of the art de los problemas de clasificación.

Otro aspecto importante en la construcción de redes neuronales más complejas es la utilización de otras funciones de activación, en este curso únicamente hemos estudiado las funciones *sign*, *Hinge* sin embargo existen muchas otras, en el curso sobre Regresiones Lineales por ejemplo utilizaremos las funciones σ .

06 Referencias

- [1] «Conda conda 4.8.3.post5+125413ca documentation». <https://docs.conda.io/projects> (accedido mar. 26, 2020).
- [2] Anaconda is Bloated - Set up a Lean, Robust Data Science Environment with Miniconda and Conda-Forge.
- [3] «Miniconda Conda documentation». <https://docs.conda.io/en/latest/miniconda.html> (accedido mar. 26, 2020).
- [4] «A brief introduction conda-forge 2019.01 documentation». <https://conda-forge.org/docs/user/introduction.html> (accedido mar. 26, 2020).



escuela-bourbaki.com