

BOURBAKI

COLEGIO DE MATEMÁTICAS

Índice

01. Introducción _____ pág. 03

01. Frank Rosenblatt _____ pág. 04

02. ¿Qué es Machine Learning? _____ pág. 05

03. Notación matemática _____ pág. 05

04. Modelo Vs Algoritmo _____ pág. 06

02. Instalación de Python y un entorno de trabajo para MachineLearning _____ pág. 07

01. Instalación de Miniconda [3] _____ pág. 07

02. Creación de un entorno virtual para análisis de datos _____ pág. 09

03. Jupyter Notebooks para ML en la nube
pág. 12

03. Perceptrón _____ pág. 13

01. El modelo _____ pág. 13

02. El entrenamiento _____ pág. 15

03. Evaluación de modelos de clasificación	
pág. 17	
04. Justificación teórica del algoritmo del per-	
ceptrón	pág. 19
04. Clasificador binario de imágenes mi-	
croscópicas	pág. 20
01. Knowledge check	pág. 21
05. Temas selectos	pág. 22
01. Datos casi-linealmente separables	pág. 22
02. Margen, sobreajuste y regularización	pág. 23
03. La bendición de la dimensión	pág. 25
04. El perceptrón como una red neuronal	pág. 26
06. Referencias	pág. 27

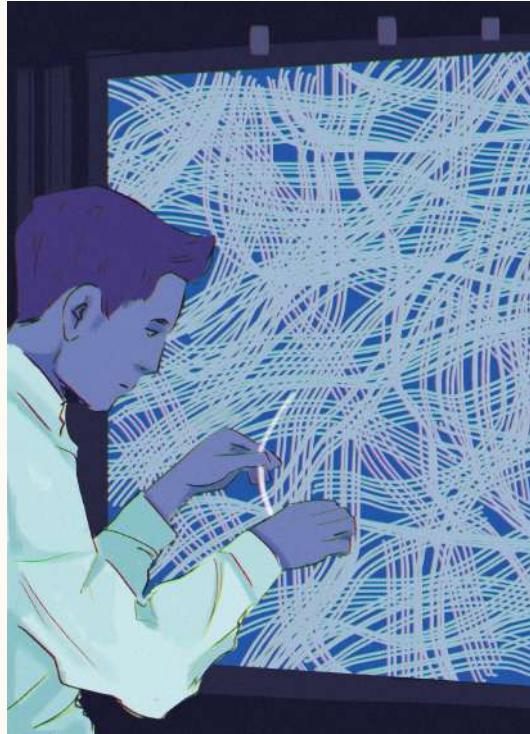
01 Introducción

Las siguientes notas son la bitácora del primer curso (de nueve en total) de 8 horas que forma parte de "Machine Learning and AI for the Working Analyst". Además de este documento los invitamos a consultar el Github del curso [en este link](#).

Este curso es una invitación al uso de modelos lineales 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. Bienvenida (una hora)
2. Un vistazo a Python (una hora)
3. ¿Qué es Machine Learning? Descripción formal del perceptrón (dos horas)
4. Implementación del algoritmo del perceptrón para la clasificación de imágenes (dos horas)
5. Knowledge check y dudas de la semana (dos horas)
6. El problema del sobre-ajuste (dos horas)

Frank Rosenblatt



Es un psicólogo estadounidense 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.

¿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

Notation 1. *A lo largo del curso utilizaremos consistentemente la siguiente notación*

- 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

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

- En una base de datos S con d características, denotaremos por N a la cantidad de registros en S .

Modelo Vs Algoritmo

En la jerga de la ciencia de datos modelo y algoritmo son palabras que pueden confundirse sin mayor inconveniente. Sin embargo, a lo largo de este curso y por cuestiones didácticas, distinguiremos dos etapas al abordar un problema. La primera será el algoritmo, que corresponde al entrenamiento o aprendizaje; y una segunda etapa a la que denominaremos modelo que ya dispone del entrenamiento para realizar predicciones.

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 virtuales ¹ [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 MacOS, Windows y Linux, eligiendo la versión de Python 3.7.

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

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 descargaste 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 descargaste 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 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 línea que se muestra a continuación y enseguida 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 línea:

```
conda create -n MachineLearning
```

2. Actívalo con la instrucción: `conda activate MachineLearning`

3. Agrega el canal ² `conda config --env --add channels`

```
conda-forge.
```

Se recomienda utilizar el canal **conda-forge**, ya que es resultado de un

²Los repositorios desde donde podemos descargar las librerías de Python se llaman canales, Anaconda, 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 pueden descargar librerías de Python.

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 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 **Pandas**³, **Scikit-learn**⁴, **Scikit-image**⁵, **Matplotlib**⁶ y **Jupyter Notebooks**⁷ 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
```

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.

Jupyter Notebooks para ML en la nube

Una alternativa altamente recomendable es utilizar el entorno que ofrece Google Colab ya que permite ejecutar código de Python utilizando notebooks de Jupyter a cualquier persona con una cuenta de Google, sin la necesidad de tener que instalar nada en la computadora del estudiante, y ejecutar el código directamente en los servidores alojados en la nube de Google. Google Colab es altamente compatible con el repositorio del curso que se encuentra en [Github](#)

03 Perceptrón

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

$$S = \{(x_1, y_1), \dots, (x_N, y_N)\}$$

Donde $x_i \in \mathbb{R}^d$ son las variables explicativas y $y_i \in \{-1, +1\}$ es la supervisión ¹.

En la práctica, estos ejemplos son bases de datos con d características.

S:

$x_1 = (x_{1,1}, x_{1,2}, \dots, x_{1,d})$	y_1
$x_2 = (x_{2,1}, x_{2,2}, \dots, x_{2,d})$	y_2
\vdots	\vdots
$x_N = (x_{N,1}, x_{N,2}, \dots, x_{N,d})$	y_N

El modelo

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

Notation 2. *Para aligerar la notación, en esta sección supondremos que la última coordenada de todas nuestras observaciones x_i es igual a uno i.e.*

¹Utilizamos los valores 1 y -1 para clasificar nuestros registros, pero podría ser cualquier clasificación binaria

$x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d-1}, 1)$. 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} .

Example 01.1. Si nuestro conjunto de ejemplos S viviera en el plano cartesiano, la hipótesis anterior significa que en lugar de tratar a $x = (x_1, x_2) \in \mathbb{R}^2$, agregaremos una coordenada extra igual a uno, $(x_1, x_2, 1) \in \mathbb{R}^3$, lo que corresponde a trabajar en el espacio tres dimensional pero únicamente sobre el plano $Z = 1$.

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

Definition 01.1. Decimos que S es separables por una función lineal o linealmente separable, si 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 $y_i = 1$ entonces $\langle \beta^*, x_i \rangle > 0$ y si $y_i = -1$ entonces $\langle \beta^*, x_i \rangle < 0$ (el caso en el que algún ejemplo x_i satisface la igualdad, corresponde al hecho geométrico de pertenecer al hiperplano definido por β^*).

Exercise 01.2. Sea S una base de datos tal que los $x_i = (x_i, y_i)$ pertenecen al plano cartesiano. Verifique que la hipótesis de separabilidad en este ca-

so significa que 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 signo $sign_{\beta^*} : \mathbb{R}^d \rightarrow \{-1, 1\}$ como

$$sign_{\beta^*}(x) = \begin{cases} 1 & \text{si } \langle x, \beta^* \rangle > 0 \\ -1 & \text{si } \langle x, \beta^* \rangle < 0 \end{cases} \quad (03.1)$$

Utilizando esta función, podemos reescribir la hipótesis de separabilidad de la siguiente manera: Sea $sign_{\beta^*}(x_i) = \hat{y}_i$ entonces la hipótesis de separabilidad significa que $y_i \cdot sign_{\beta^*}(x_i) > 0$ para todos los $i \leq N$.

El entrenamiento

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

Definition 02.1. Algoritmo del perceptrón en T épocas

1. Comenzamos con $\beta_0 = (1, 1, \dots, 1)$
2. Para cada $t \leq T$ vamos a re-ordenar los elementos en la base de datos de manera aleatoria y vamos a recorrer la base de datos completa en ese orden. Así, en la primera época denotaremos $x_i = x_{i,1}$ a los elementos

de S . En la segunda época, $t = 2$ denotaremos como $x_{i,2}$ a los elementos reordenados de S .

3. Si suponemos que $\beta_{i,t}$ está definido, buscamos el primer ejemplo (en ese orden) $(x_j, y_j) \in S$ en el que NO se cumple que: $y_j \cdot \text{sign}_{\beta_i}(x_j) > 0$.

- Si no encontramos algún índice j que cumpla los criterios de nuestra búsqueda entonces β_i es nuestra propuesta para clasificar puntos. Es decir $\beta_{i,t} = \beta_{i+1,t}$.
- Si por el contrario encontramos algún índice j que cumpla los criterios de nuestra búsqueda entonces actualizamos

$$\beta_{i+1,t} := \beta_{i,t} + y_j x_j$$

4. Volvemos al paso dos, esta vez $t + 1$.

Remark 02.1. *Notemos que si nos equivocamos en el ejemplo j , entonces*

$$y_j \langle \beta_{i+1}, x_j \rangle = y_j \langle \beta_i + y_j x_j, x_j \rangle = y_j \langle \beta_i, x_j \rangle + \|x_j\|_2^2$$

Si el algoritmo se detiene en T épocas, tenemos una red neuronal neuronal f :

$\mathbb{R}^{d+1} \rightarrow \{-1, +1\}$ de una capa y $d + 1$ neuronas definida por $f(x) = \text{sign}_{\beta_T}(x)$.

En posteriores secciones veremos que si S es linealmente separable, entonces existe una época donde el error es cero.

Evaluación de modelos de clasificación

Nos concentraremos con un problema de clasificación binaria cuyas clases corresponden a -1 , $+1$, vale la pena notar que mucho de lo que haremos para este caso, tiene sentido para problemas multi-clases.

Muchos de los algoritmos que vamos a trabajar en el curso requieren que hagamos una partición de la base de datos en dos secciones. El conjunto de entrenamiento y el conjunto test. Es importante reservar una porción de datos que el modelo no ha visto, para evaluar su desempeño en datos completamente ajenos a los utilizados en su entrenamiento.

En este ejemplo supondremos que $+1$ significa **no-churn** (clientes que no tienen desapego a un producto) y -1 significa **churn** (clientes que tienen desapego al producto).

Definition 03.1. Definimos las siguientes cantidades asociadas a un modelo de clasificación f y a un conjunto de test S de la siguiente forma:

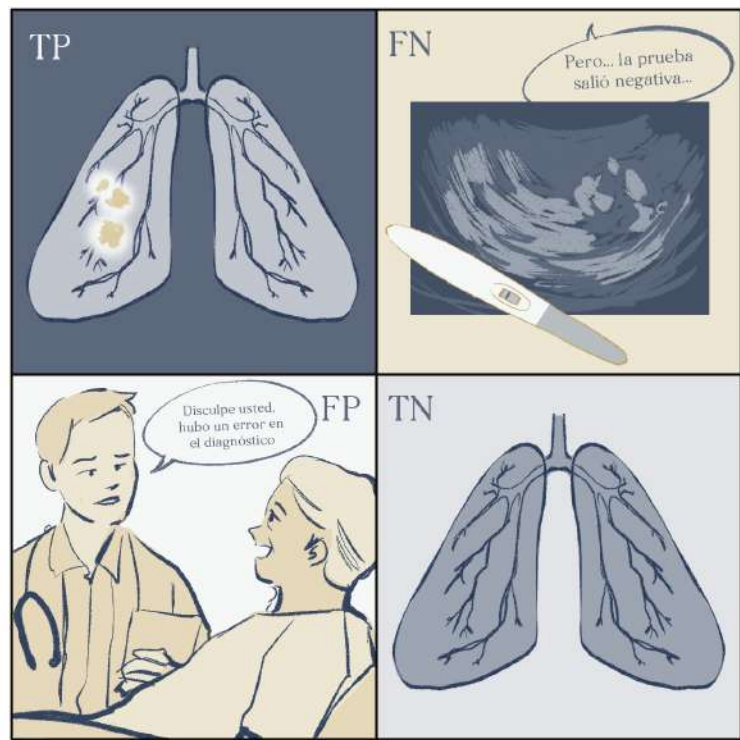
TP es la cantidad de elementos en S etiquetados como $+1$ tales que el modelo f también los etiqueta como $+1$.

TN es la cantidad de elementos en S etiquetados como -1 tales que el modelo f también los etiqueta como -1 .

FP es la cantidad de elementos en S etiquetados como -1 tales que el modelo f los etiqueta como $+1$.

FN es la cantidad de elementos en S etiquetados como $+1$ tales que el modelo f los etiqueta como -1 .

Una representación común de esta evaluación es conocida como la matriz de confusión, que usualmente se presenta como en la imagen:



Definition 03.2. Definimos las siguientes medidas de evaluación del modelo f en el conjunto de test S :

- La sensibilidad $D = \frac{TP}{TP + FN}$
- La precisión $P = \frac{TP}{TP + FP}$

En un modelo de diagnóstico médico como el de la detección de cáncer, la sensibilidad mide la capacidad del modelo para detectar como positivos a

los caso realmente positivos. Mientras que la precisión mide la disposición del modelo a no arrojar falsos positivos.

Justificación teórica del algoritmo del perceptrón

Definamos como R es el máximo del producto punto de todos los elementos de S , y B el mínimo de todas las betas para las que $y_i \langle x_i, \beta \rangle$ es positivo.

$$R = \max_{i \leq N} \langle x_i, x_i \rangle \quad (03.2)$$

$$B = \min_{\beta \in \mathbb{R}^d} \{ \langle \beta, \beta \rangle : y_1 \cdot \langle x_1, \beta \rangle > 0, \dots, y_N \cdot \langle x_N, \beta \rangle > 0 \} \quad (03.3)$$

Theorem 04.1. *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 Clasificador binario de imágenes microscópicas

A continuación incluimos una breve descripción del data set que utilizaremos para ejemplificar 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 519×521 pixeles a 260×260 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.

Exercise 00.1. *¿En el problema de la detección de cancer preferimos una alta sensibilidad o una alta precisión?*

Knowledge check

- Exercise 01.1.**
1. *¿En qué problema considera usted que se puede utilizar el algoritmo de esta semana? De preferencia describa la base de datos. Reflexione sobre el proceso de construcción de dicha base de datos y principalmente, en la supervisión de la misma.*
 2. *¿Cual sería el problema con un una base de datos desbalanceada? Es decir, una en la que haya demasiados -1 respecto de muy pocos $+1$. ¿Qué pasaría con el hiperplano que separa los puntos?*
 3. *Considerando que la cantidad de características de la base de datos determina la dimensión del hiperplano que busca el algoritmo de esta semana. ¿En qué espacio es más fácil realizar una clasificación binaria mediante un hiperplano, en un espacio de pocas dimensiones o en un espacio de muchas dimensiones?*

05 Temas selectos

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 perceptrón simple.

Datos casi-linealmente separables

La versión del algoritmo que utilizan la mayoría de las librerías es una variante del algoritmo del perceptrón anteriormente estudiado. Recordemos que su eficacia solo está garantizada 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 01.1. 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 $Ite_s(0) = 0$.

- Etapas i
1. El perceptrón se actualizará de manera usual como en el algoritmo 02.1, 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 01.1. *El algoritmo anterior tiene un problema pues podría actualizar al vector β_s por un vector inferior a los probados anteriormente ya que solo guarda la información de la última iteración.*

Theorem 01.2. *Si los coeficientes β_i son números racionales entonces el algoritmo anterior, con probabilidad uno converge a una solución óptima.*

Margen, sobreajuste y regularización

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

Intuitivamente, el sobreajuste se puede ver como una diferencia sustancial entre el error del entrenamiento y el error del test. Éste último se puede calcular fácilmente como $Err_{test} = \frac{FP + FN}{TP + FN + FP + TN}$. Más precisamente, tendremos sobreajuste cuando el error en el entrenamiento es mucho más pequeño que el error en el test.

Decimos que el modelo **subajusta** cuando el error en el entrenamiento es muy grande. En este caso el error del test es irrelevante puesto que un modelo tal no es confiable.

Definition 02.1. 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 el modelo M ha incurrido en **sobreajuste 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 sobreajuste, 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 02.2. 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 04.1.

Theorem 02.1. *Supongamos que S es separable linealmente con un margen δ , entonces el algoritmo regularizado del perceptrón se detiene (es decir encuentra alguna β que clasifica correctamente a los ejemplos en S) en $\left(\frac{R}{\delta} B\right)^2$ pasos.*

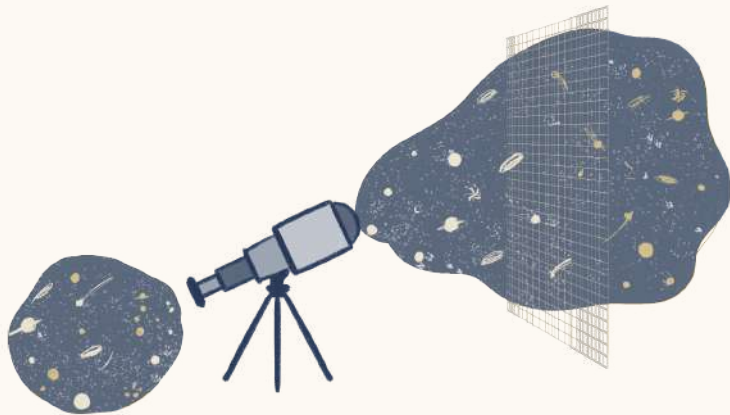
Es importante notar que no existe una relación sencilla de explicar entre el margen δ y α .

La bendición de la dimensión

En la siguiente semana hablaremos sobre la maldición de la dimensión en el caso de los árboles de decisión y su relación con el sobre-ajuste.

Afortunadamente las altas dimensiones en los datos no siempre son un grave problema para los problemas de machine learning como lo expresa el siguiente teorema:

Theorem 03.1. *Supongamos que S es un conjunto de datos de aprendizaje supervisado cuya separación binaria es un problema complicado. Sea $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ una función no-lineal en un espacio suficientemente grande $d' > N$, entonces con alta probabilidad la imagen de S es linealmente separable.*



El perceptrón como una red neuronal

Es posible pensar en el perceptrón como una red neuronal con una sola capa cuya función de activación es la más sencilla posible. Existen muchas otras redes neuronales con más capas y otras funciones de activación de las cuales hablaremos más adelante en el curso.

06 Referencias

[1] Conda conda 4.8.3.post5+125413ca documentation.

<https://docs.conda.io/projects/conda/en/latest/> (accedido mar. 26, 2020).

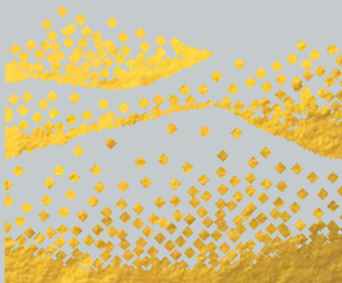
[2] Anaconda is Bloated - Set up a Lean, Robust Data Science Environment with Miniconda and Conda-Forge.

[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).



BOURBAKI
COLEGIO DE MATEMÁTICAS

escuela-bourbaki.com

