

Introducción a Bash y Git

Carlos Malanche

1 de febrero de 2018

1. Pequeña introducción

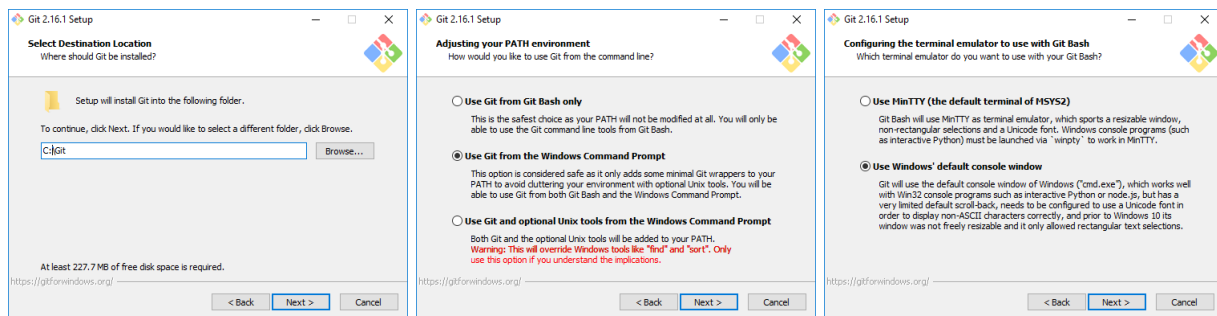
Para los que no sepan lo que es bash, es el intérprete de comandos por defecto en la gran mayoría de las distribuciones de Linux, a veces sólo le llaman *la terminal*. Su utilidad es la de navegar por el sistema e interactuar con los archivos (editarlos, copiarlos, pegarlos, hacer búsquedas, etc.) que en él viven.

Bash siempre ha sido un privilegio de los sistemas Unix (MacOS y Linux), pero desde hace unos meses hay soporte nativo para bash en Windows 10. Aquí, por cuestiones de simplicidad, no usaremos este soporte básico de bash (pues consiste en una máquina virtual, y eso lo hace un tanto menos... compatible con Windows, por así decirlo). En mi opinión, lo mejor es descargar el cliente de Git para windows, que incluye todos los comandos que utilizaremos.

2. Git

Sacado directamente de la [página de git](#): Es un sistema de control de versiones gratuito y libre (de código abierto vaya). Digamos que es una especie de Dropbox que funciona desde la terminal, y que está optimizado para trabajar con códigos fuente. Git, como ya lo había mencionado antes, viene con sus *implementaciones* de varios comandos de bash (además de una emulación de bash), comandos que veremos en esta clase.

Para los alumnos que tengan Windows, basta con descargar el cliente de Git de la página de internet e instalarlo, aceptando casi todos los valores por defecto en la instalación; Windows tiene esta *espantosa mañana* de usar espacios en los nombres de los archivos, por lo que git propondrá la ruta “C:\Program Files\Git”. Hay que cambiarla por “C:\Git”.



Con esto ya se puede utilizar git desde cmd (la terminal de windows) o Powershell. Sin embargo, yo recomiendo utilizar [cmdex](#) como emulador de una terminal de Linux. (Si alguien tiene duda de cómo se usa, me pueden mandar un correo).

Para los usuarios de MacOS es suficiente abrir una terminal (`command+space` para abrir la búsqueda, y escriben *terminal*) y ejecutar el comando `git --version`. Si Git no está instalado, les va a preguntar si quieren instalarlo.

Para los que quieran utilizar Linux, basta con ejecutar `sudo apt install git` desde la terminal.

3. Comandos de Bash

Vamos a aprender a navegar por un sistema de archivos por medio de una terminal. Esto va a ser de gran utilidad pues a la mejor un día necesitaremos correr simulaciones en servidores que no cuentan con una interfaz gráfica.

3.1. Algunos ejercicios

Para agarrar confianza con la máquina, ejecute los siguientes comandos en la terminal (las palabras en verdes son argumentos forzosos):

- **man**: Sólo disponible en Unix. Despliega información de un comando y sus usos (por ejemplo, **man ls**). Junto con *Google*, su herramienta más útil **antes** de venir a preguntarme algo que no entienden.
- **pwd**: Corto para *path of working directory*. Dice el directorio en el que nos encontramos trabajando.
- **ls**: Lista los documentos y carpetas del directorio en el que estamos. Yo siempre lo ejecuto con la opción **-la**, pues salen todos los archivos (también los ocultos) en forma de lista con permisos de ejecución.
- **cd *dirname***: Corto para *change directory*. Nos permite cambiar al directorio *dirname*. Como nota importante, todo directorio contiene dos directorios con nombres “.” y “..”. El primero es la ruta al directorio actual, y el segundo la ruta al directorio superior.
- **mkdir *dirname***: Crea un directorio bajo el nombre *dirname*.
- **rmdir *dirname***: Borra un directorio de nombre *dirname*.
- **rm *filename***: Borra un archivo
- **mv *filename otherfilename***: Mueve (o renombra) el archivo *filename* al archivo *otherfilename*.
- **cp *filename otherfilename***: Copia el archivo *filename* al archivo *otherfilename*.

Eso debiera ser suficiente por el momento. Hay muchas otras instrucciones *divertidas* en bash, pero por simplicidad, usaremos sólo lo necesario.

4. Comandos de Git

Como ya debieron haber notado, el material del curso se encuentra en la página <https://github.com/Malanche/fc-adml>. Github es un sitio donde se nos permite tener una *copi*a de un repositorio en la nube (por así decirlo). Lo equivalente al cliente web de *Google Drive* o *Dropbox*.

Git es un comando más de la terminal (si ya lo han instalado...). Si ustedes quieren tener una copia local del repositorio **fc-adml**, basta con abrir una terminal en el directorio que gusten (por ejemplo, *Mis Documentos*) y ejecuten:

```
git clone https://github.com/Malanche/fc-adml
```

El repositorio es público, por lo que no será necesario que hagan una cuenta en Github (por el momento) para ver los contenidos del repositorio en su computadora. Se va a crear un directorio bajo el nombre **fc-adml** que contiene los archivos del repositorio. Una vez que se clonó, no vuelve a haber comunicación con el servidor, por lo que (a diferencia de *Google Drive* y *Dropbox*) los archivos no estarán sincronizados con lo que hay en el servidor. Para eso hay un nuevo y divertido comando:

```
git pull
```

Si hay cambios nuevos en el servidor, llegarán a ustedes con este comando.

Ustedes deberán utilizar Git para subir sus tareas al servidor, por lo que deberían abrir una cuenta en github (es gratuito).

No es necesario ser un experto con Git, pero sí que hay que entender lo que se puede hacer con él (cuando hagan tareas en equipo verán la importancia de este software). Les dejo, por último, [este tutorial interactivo de git](#) (donde les van a explicar como crear un repositorio, añadir archivos, borrarlos, hacer *commits*, etc.).

Introducción a Python

Carlos Malanche

6 de febrero de 2018

1. El lenguaje de programación Python

Durante el curso vamos a hacer uso de herramientas un poco más sofisticadas que *excel*, o *matlab*. Todo girará alrededor de Python, un lenguaje de programación cuyo código fuente está disponible y se puede utilizar de manera gratuita. Hay varias razones por las que usaremos **Python**:

- Es muy fácil de entender y de empezar a usar, pues realiza ciertas tareas de manera automática (por ejemplo, asignación de tipo de variables).
- Es interactivo: al ser un lenguaje interpretado, es posible ejecutar instrucciones aisladas para observar el resultado inmediatamente.
- Es *aceptablemente* rápido, hay trucos (como `cpython`) que permiten obtener la velocidad de un lenguaje compilado. Sí, C++ podrá ser más rápido, pero a veces importa más que la velocidad de *implementación* de un código sea la parte más eficiente, lo cual **Python** hace muy bien.
- Hay muchos, muchos, muchos lugares en los cuales obtener ayuda en internet, ya que es un lenguaje muy popular (vaya, el sitio de la NASA, Bitbucket e Instagram están montados con Python).
- Es estable: Hay lenguajes recientes que son más eficientes al momento de programar y en su ejecución pero aún se encuentran en desarrollo. **Julia** es un buen ejemplo, aún no sale la versión 1.0 y por ello códigos que funcionan hoy pueden dejar de funcionar en 1 semana.
- El curso **no es sobre Python**, **Python** es sólo una herramienta que es conveniente en este caso.

Vamos directo a su instalación, y los códigos más pequeños y lindos para familiarizarnos con **Python**.

2. Instalación de Python

En general, todas las instrucciones para instalar **Python** se encuentran bien documentadas en [su sitio web](#). Vamos a utilizar la última versión de **Python** (3.6.4 al momento de crear este documento). Para Windows no es necesario utilizar **Anaconda**, ya que **scikit-learn**, **numpy**, **pandas** y **scipy** se pueden instalar fácilmente con la herramienta **pip** (administrador de paquetes) de **Python**. Por el momento sólo utilizaremos **Python** con **numpy**.

Si necesitan ayuda instalando **numpy**, díganme y los asistiré.

3. Los primeros pasos

Como ya es tradición en cualquier curso donde se use un lenguaje de programación por (probablemente) primera vez, vamos a escribir el programa *hello world* para **Python**.

Python es un lenguaje de programación imperativo, lo que significa que ejecuta una instrucción tras otra. Estas instrucciones pueden ser declaraciones de variables, condiciones, asignación de valores llamadas a funciones u otras cosas.

En mi caso, voy a utilizar [Visual Studio Code](#) como editor de archivos (algo así como el bloc de notas pero en su versión con *adamantium* inyectado), pero ustedes pueden utilizar cualquier herramienta con la que se sientan cómodos. Para practicar, les recomiendo hagan un folder del nombre que gusten, por ejemplo *adml-inicio* donde van a guardar los *scripts* de **Python** de esta clase. Desde la terminal, puedo editar un archivo de nombre `helloworld.py` escribiendo lo siguiente:

```
C:\Users\Carlos\Codes\adml-inicio
λ code helloworld.py
```

A continuación, escribió la siguiente línea dentro del archivo *helloworld.py*

Listing 1: Código fuente de helloworld.py

```
print("Hello_world")
```

Para ejecutar este código, desde la terminal ejecuto `python` dándole como argumento el nombre del código fuente:

```
C:\Users\Carlos\Codes\adml-inicio
λ python helloworld.py
Hello world
```

La función `print` escribe a la terminal la cadena de caracteres que le es proporcionada. Python reconoce una cadena de caracteres por estar contenida entre comillas `"`.

Python también puede ejecutarse en su versión interactiva, basta con escribir `python` en la terminal, y *Python* irá ejecutando las instrucciones que le hagamos llegar. En modo interactivo, existe un espacio de trabajo (*workspace*) en donde se van guardando variables, definiciones de funciones, etc. de cada sesión.

```
C:\Users\Carlos\Codes\adml-inicio
λ python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1.56
>>> x * 10
15.600000000000001
>>> |
```

Hmmmm... eso no es $1,56 \times 10$. Bueno, *ahí les queda de tarea* investigar [por qué](#) ocurrió eso.

3.1. Declaración de variables

Tal como se puede ver en la última captura de pantalla, en *Python* se pueden declarar variables. Para declarar una variable en *Python* basta darle un nombre y darle un valor (ya sea un entero, un flotante, una cadena de caracteres, etc.)

Listing 2: Código fuente de variabledecl.py

```
# Texto de salida
texto = "El_valor_es:_"
# Declara variable x con valor 10
x = 10
# Declara variable y con dos veces el valor de x
y = x * 2
# Muestra el resultado en la terminal
print(texto + str(y))
```

Para *Python*, las líneas de código que comienzan con `#` son ignoradas, con lo que podemos usar dicho signo para añadir comentarios al código.

El resultado de ejecutar este *Script* es:

```
C:\Users\Carlos\Codes\adml-inicio
λ python variabledecl.py
20
```

Para variables de tipo entero o flotante, se pueden realizar las operaciones de multiplicación (*), división (/), suma (+), resta (-), módulo o residuo (%), división con redondeo (//) y exponenciación (**).

Además hay operadores de comparación que devuelven como resultado verdadero (*True*) o falso (*Falso*) como la equidad (==), no equidad (!=), *mayor que* y *menor que*, *mayor o igual que* y *menor o igual que* (<, >, <=, >=).

Jueguen un rato con estas operaciones para que se familiaricen.

3.1.1. Cadenas de caracteres

Una cadena de caracteres en **Python** se declara entre comillas. Se puede acceder a los elementos de la cadena de caracteres por medio de los paréntesis cuadrados. El primer elemento lleva el índice 0.

Por defecto, podemos acceder a elementos con índices negativos en **Python**, y el comportamiento de `cadena[-i]` es igual al de `cadena[i]`

Los dos puntos nos permiten acceder a un rango de caracteres, puede ser desde un valor a hasta un valor $b - 1$ (`cadena[a:b]`), o desde un caracter hasta el final de la cadena, por ejemplo:

```
C:\Users\Carlos
λ python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x = "no voy a pasar el curso!"
>>> x[3:]
'voy a pasar el curso!'
>>> |
```

También se puede acceder desde el principio de la cadena hasta el caracter $b - 1$ (`cadena[:b]`).

Hay un tercer argumento que sirve para dar *brincos* entre caracteres. La instrucción `cadena[a:b:c]` generará la cadena de caracteres empezando en a , seguida de los caracteres en los índices $a + i * c$ (con $i > 0$ entero) tal que no se exceda ni iguale b (Los valores por defecto de a , b y c son 0, la longitud de la cadena y 1 respectivamente). Por ejemplo, si `cadena = "Hoy es Martes"`, entonces `cadena[:2]` resultará en "Hye ats".

3.1.2. Listas

Un tipo de variable que usaremos mucho son las *listas*. Son series de elementos simplemente (no tienen que ser del mismo tipo). La declaración de una lista es por medio de los paréntesis cuadrados, y con los mismos se puede acceder a los elementos de la lista, enumerados desde 0.

```
C:\Users\Carlos\Codes\adml-inicio
λ python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> milista = [1, 5, 8, 10]
>>> milista
[1, 5, 8, 10]
>>> milista[1]
5
>>> |
```

3.2. Instrucciones básicas

Al programar nos auxiliamos de muchas cosas, desde funciones complejas hasta simples instrucciones que permiten al código *tomar* decisiones. Las siguientes tres instrucciones ejecutan un bloque de código que se separa con una indentación dada una condición (en **Python** la estructura del código es importante, la indentación denota bloques de código). Además las instrucciones siguientes se anuncian con dos puntos al final.

3.2.1. if else

A veces queremos que el código tenga cierto comportamiento, dependiendo de algún parámetro como puede ser la información que un usuario nos da. Para ello existe el par **if else**. A **if** le sigue una condición *c*. Al momento de la ejecución, si la condición se satisface se ejecuta el código indentado después de la instrucción **if**. Adicionalmente, se puede agregar la instrucción **else** que es lo mismo que escribir **if !c** (*si no-c*), que le dice al python qué hacer si la condición no fue satisfecha.

Listing 3: Código fuente de par.py

```
# Pide variable al usuario y convierte en entero
x = int(input("Ingrese un valor entero: "))
# Verifica si el residuo al dividir por 2 es cero.
if (x % 2) == 0:
    print("El valor es par!")
else:
    print("El valor no es par!")
```

3.2.2. for

Existe también la instrucción **for**, que permite la ejecución repetida de una sección de código.

Listing 4: Código fuente de forloop.py

```
# Repite una instruccion 5 veces
for x in range(0,5):
    # x toma valores de 0 a 4
    print(x)
```

La función **range(start, stop)** genera una lista de valores desde **start** hasta **stop** sin incluir este último.

3.2.3. while

La instrucción **while** ejecuta repetidamente un bloque de código hasta que una condición queda satisfecha.

Listing 5: Código fuente de whileblock.py

```
import sys
# Lee x de la terminal
x = int(input("Ingrese un natural: "))
# Implementación de una verificación no óptima de la conjetura de Collatz
while x != 1:
    if x % 2 == 0:
        x = x / 2
    else:
        x = x * 3 + 1
    sys.stdout.write(str(int(x)) + ", ")

print("La cadena regresa al 1.")
```

El código de ejemplo verifica la [conjetura de Collatz](#) para el número que el usuario ingresa.

El código hace uso de una nueva instrucción, **import**, que permite el uso de funciones dentro del paquete **sys**. Un paquete es justo eso, un conjunto de herramientas disponibles al programador, en este caso utilizamos la habilidad de escribir a la terminal sin dejar saltos de línea, como lo hace **print**.

Las 3 instrucciones pasadas dependen de condiciones. Para pedir el cumplimiento de más de una condición (o de alguna condición entre varias) se utilizan las palabras **and** y **or**.

Estas 3 instrucciones brindan por sí mismas bastantes posibilidades, pero vamos a necesitar un poco más para poder hacer análisis de datos.

4. Definición de funciones

Muchas veces vamos a escribir un pedazo de código que nos gustaría ejecutar en diversos puntos de un script. Para ello se pueden definir funciones en *Python* con ayuda de la palabra **def**.

Listing 6: Código fuente de promedio.py

```
# Define una funcion llamada promedio que recibe un argumento
def promedio(valores):
    # Sacar el promedio de una lista
    return sum(valores)/len(valores)

misValores = [1,2,3,4,5,6,7,8,9,10]

print(promedio(misValores))
```

5. Ejercicios

5.1. suma.py: Suma de los primeros n números

Haga un programa que reciba una variable n y calcule la suma de los números desde 1 hasta n .

5.2. palindromo.py: Palíndromos

Haga un programa que pida al usuario una cadena de caracteres, y el programa diga si es o no un palíndromo.

5.3. ppalindromo.py: Producto palíndromo

Encuentre el número más grande i inferior a $n = 100000$ que al ser multiplicado con el número resultante de poner las cifras en el orden contrario genera un palíndromo. *Ejemplo:* 21 pues $21 \times 12 = 252$.

Recuerden: el internet ya llegó, y páginas como [stackoverflow](https://stackoverflow.com) son la mejor ayuda que pueden conseguir.

Un poco más de Python e iPython Notebook

Carlos Malanche

8 de febrero de 2018

La clase pasada vimos un poco de lo que se puede hacer en Python con scripts y con su sesión interactiva. Ahora vamos a utilizar una herramienta un poco más sofisticada para hacer scripts con comentarios en html, además de ver lo que es un ambiente virtual y cómo instalar paquetes de internet.

Las instrucciones aquí escritas son para Windows 10 y Ubuntu 16.04, además de una guía tentativa para MacOS. En lo que llega la siguiente clase, voy a ver como consigo derechos de administrador en las computadoras del salón para que todo el software esté disponible en las máquinas de la universidad.

1. *Virtual environments*

¿Alguna vez han instalado algo que desearían no haber instalado? A veces la computadora funciona perfectamente, hacemos una actualización y algo deja de funcionar. Con **Python** podemos evitar este tipo de situaciones gracias a que tiene soporte para ambientes virtuales. Esto quiere decir que podemos hacer una pequeña instalación local de **Python** para probar a instalar paquetes o probar ciertas configuraciones, y si algo sale mal, borramos el folder de la instalación local de **Python** y la instalación original quedará intacta.

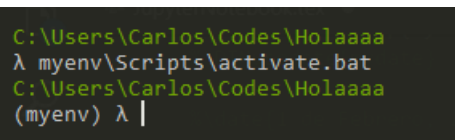
1.1. Windows

En Windows, **Python 3.6.4** ya viene con **venv**, la instrucción para generar ambientes virtuales. Para crear uno, basta con ejecutar:

```
python -m venv /ruta/al/folder
```

En particular, podemos ir a nuestro folder **adml-inicio** y dentro escribir **python -m venv myenv** para generar un ambiente virtual de **Python** en el folder **myenv**.

Una vez generado el ambiente virtual, hay que activarlo. Para ello, corremos el archivo **activate.bat** que está en **myenv/Scripts**.



```
C:\Users\Carlos\Codes\Holaaaa  
λ myenv\Scripts\activate.bat  
C:\Users\Carlos\Codes\Holaaaa  
(myenv) λ |
```

En el caso de **cmdr** podemos ver en paréntesis el nombre del ambiente virtual que estamos utilizando.

Dentro de nuestro ambiente virtual tenemos nuestros ejecutables **pip** y **pip3**, que sirven para instalar paquetes de Python. Por ejemplo, podemos instalar ahora **numpy** ejecutando **pip3 install numpy**. Numpy es un paquete que permite el manejo de matrices, vectores, y además contiene métodos numéricos usados frecuentemente en análisis numérico.


```

C:\Users\Carlos\Codes\Holaaaa
(myenv) λ pip3 install numpy
Collecting numpy
  Downloading numpy-1.14.0-cp36-none-win_amd64.whl (13.4MB)
    100% |#####| 13.4MB 49kB/s
Installing collected packages: numpy
Successfully installed numpy-1.14.0

C:\Users\Carlos\Codes\Holaaaa
(myenv) λ python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.array([1,2,3])
array([1, 2, 3])
>>> |

```

Atención: Usen Python 3.6.4 para instalar esto en Windows, la versión 3.7 aún no cuenta con los archivos *wheel*, que son archivos precompilados, entonces les pedirá que tengan Visual Studio C++14 instalado.

Con el script `deactivate.bat` se puede verificar que el paquete `numpy` sólo ha sido instalado para nosotros.

```

C:\Users\Carlos\Codes\Holaaaa
(myenv) λ myenv\Scripts\deactivate.bat
C:\Users\Carlos\Codes\Holaaaa
λ python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'numpy'
>>> |

```

1.2. MacOS

Lamentablemente con MacOS no los puedo ayudar mucho, pero con unos amigos conseguimos hacerlo funcionar siguiendo las instrucciones de [esta página](#). A diferencia de Linux y Windows, es un poco difícil hacer funcionar el comando `venv`, por lo que existe `virtualenv`. Se necesita Python 3 junto con `pip3` (administrador de paquetes de Python).

Para instalar `virtualenv` ejecuten

```
pip3 install virtualenv
```

Ahora pueden crear un ambiente virtual ejecutando:

```
virtualenv myenv
```

Para activar el ambiente virtual, basta con ejecutar `source myenv/bin/activate`. Para salir del ambiente virtual, escribimos `deactivate` en la misma terminal y ya está.

1.3. Ubuntu 16.04

Ubuntu no viene con `venv`, pero se puede instalar rápidamente ejecutando `sudo apt-get install python3-venv`.

Una vez instalado, podemos crear un ambiente virtual ejecutando:

```
python3 -m venv /ruta/al/folder
```

Una vez generado el folder con el ambiente virtual, lo podemos activar con el comando `source` y el script `activate` que está en `myenv/bin/` (es decir, `source myenv/bin/activate`). Se nos va a nunciar que funcionó pues la terminal mostrará entre paréntesis el nombre del ambiente virtual. Para dejar el ambiente virtual, basta con escribir `deactivate` en la terminal. Todo esto se puede ver en la siguiente captura de pantalla:

```

carlos@raven:~/Documents$ sudo apt-get install python3-venv
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python-pip-whl python3.5-venv
The following NEW packages will be installed:
  python-pip-whl python3-venv python3.5-venv
0 upgraded, 3 newly installed, 0 to remove and 114 not upgraded.
Need to get 1 118 kB of archives.
After this operation, 1 260 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://mx.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 python-pip-whl all 8.1.1-2ubuntu0.4 [1 110 kB]
Get:2 http://mx.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 python3.5-venv amd64 3.5.2-2ubuntu0~16.04.4 [5 998 B]
Get:3 http://mx.archive.ubuntu.com/ubuntu xenial/universe amd64 python3-venv amd64 3.5.1-3 [1 106 B]
Fetched 1 118 kB in 1s (587 kB/s)
Selecting previously unselected package python-pip-whl.
(Reading database ... 223721 files and directories currently installed.)
Preparing to unpack .../python-pip-whl_8.1.1-2ubuntu0.4_all.deb ...
Unpacking python-pip-whl (8.1.1-2ubuntu0.4) ...
Selecting previously unselected package python3.5-venv.
Preparing to unpack .../python3.5-venv_3.5.2-2ubuntu0~16.04.4_amd64.deb ...
Unpacking python3.5-venv (3.5.2-2ubuntu0~16.04.4) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.5.1-3_amd64.deb ...
Unpacking python3-venv (3.5.1-3) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up python-pip-whl (8.1.1-2ubuntu0.4) ...
Setting up python3.5-venv (3.5.2-2ubuntu0~16.04.4) ...
Setting up python3-venv (3.5.1-3) ...
carlos@raven:~/Documents$ python3 -m venv myenv
carlos@raven:~/Documents$ myenv/bin/activate
carlos@raven:~/Documents$ source myenv/bin/activate
(myenv) carlos@raven:~/Documents$ deactivate
carlos@raven:~/Documents$

```

Ahora pueden instalar los paquetes que quieran (mientras el ambiente virtual esté activo) con `pip3 install`.

1.4. Común a todos

Son 4 los paquetes que vamos a instalar: `pandas`, `numpy`, `matplotlib` y `jupyter`. Eso será suficiente para la siguiente parte del curso.

2. iPython notebook

Ahora conocido como Jupyter notebook, es una herramienta que permite ejecutar Python desde un navegador de internet. La ventaja de utilizar el navegador de internet es que podemos añadir texto en formato, además de permitirle a otras personas ejecutar nuestro código instrucción por instrucción.

Un poco más a detalle, Jupyter Notebook es un servidor de Python con el que se puede acceder por un navegador.

2.1. Instalación

Asumiendo que ya instalaron Python, basta con abrir una terminal y ejecutar `pip install jupyter` (**Ojo:** Los que usan windows tienen que añadir Python a las variables de entorno, el folder llamado Scripts, pues ahí se encuentra el comando `pip`. Los de Ubuntu pueden ejecutar `sudo apt install python3-pip`. Para los de MacOS debe haber una solución con `homebrew`).

2.2. Ejecución

En una terminal pueden escribir `jupyter notebook` para comenzar el servidor, u opcionalmente `jupyter notebook ejemplo.ipynb` para comenzar el servidor y comenzar a editar el archivo `ejemplo.ipynb` (el archivo debe existir. Si quieres empezar de cero, pueden escribir `touch ejemplo.ipynb` para generar un archivo vacío).

Al abrir jupyter sobre un documento (o hacer uno nuevo con el botón **new**) encontrarán una pequeña barra de tareas, y una barra vacía en el centro de la pantalla. Ahí pueden escribir código, texto en formato *markdown* e incluso cosas de \LaTeX (basta con rodear ecuaciones o lo que sea por símbolos de dinero, \$)

The screenshot shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "Hola Last Checkpoint: ayer a las 15:42 (unsaved changes)". On the right, there is a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for saving, adding, undo, redo, and other actions, along with a "Run" button and a "Code" dropdown menu. The main area contains three code cells:

- Ejemplo de markdown**
Ejemplo de una operación sencilla
`In [1]: 1+2+3+4+5`
`Out[1]: 15`
- Declaración de una variable
`In [2]: x = 5`
- Una operación con esa variable (por ejemplo $\sum_{i=1}^x i$)
`In [3]: sum(range(1,x+1))`
`Out[3]: 15`

Para finalizar el servidor, se ejecuta `ctrl+c` en la terminal (y `cmd+c` en MacOS, *me supongo*).

Hay algunos *atajos* que les puedo compartir: si están en una celda, `shift+enter` ejecutará el código que contiene, y `alt+enter` hará lo mismo pero generará una nueva celda debajo. Hay otros truquitos, pero no se trata de que sean expertos *desde ahorita*, pues *uno nunca deja de aprender* e internet está ahí para ayudarnos si se nos olvida un comando. Python y sus paquetes son herramientas para nosotros, cuya utilidad vamos a ir descubriendo conforme necesitemos hacer cosas más específicas. Si tienen tiempo libre, pónganse a jugar con Python y con Jupyter, les va a ser útil la siguiente clase.

Análisis de Datos

Carlos Malanche

15 de febrero de 2018

1. Qué es el análisis de datos

Directamente sacado de [Wikipedia](#), el análisis de datos consiste en tomar datos recabados de análisis estadísticos o de experimentos para inspeccionarlos, transformarlos, limpiarlos y eventualmente utilizarlos para generar conclusiones y modelos, y posiblemente comenzar a realizar predicciones del sistema del que se han obtenido los datos.

Últimamente, se ha puesto de moda el término *data science*, que engloba más etapas del proceso descrito anteriormente. La ciencia de datos incluye la manera de recabar los mismos, pues no siempre se encuentran disponibles (en el ejemplo de un experimento, donde tenemos un conjunto de sensores y nuestro proceso de captura está determinado). Al proceso de recaudar información que **ya existe** pero no se encuentra **ordenada** se le llama *minería de datos*. Este paso no existe en un experimento científico.

1.1. Un ejemplo tonto

La *minería de datos* es un proceso complicado pues se trata de darle más utilidad a la información ya existente. Un ejemplo podría ser el siguiente:

Imagine que se han realizado múltiples mediciones para determinar si el calentamiento global en verdad está ocurriendo como una consecuencia de las emisiones de ciertos gases. Por cuestiones históricas, cada laboratorio ha decidido tomar mediciones de la concentración en el aire de ciertos gases. Digamos pues, algunos laboratorios han medido Metano (CH_4) y Ozono (O_3), otros decidieron medir vapor de agua y óxido de nitrógeno N_2O y la gran mayoría se ha dedicado a medir dióxido de carbono (CO_2). Pues bien, es posible que por condiciones atmosféricas relacionadas a la región, estas medidas de manera aislada no tienen mucho valor. Sin embargo, un científico puede darse a la tarea de recolectar la información de todos los laboratorios que han hecho estas medidas, localizarlas geográficamente y temporalmente (hay registros históricos de estos gases desde hace años, y la meta es observar una tendencia de crecimiento tanto en las concentraciones de dichos gases como en la temperatura promedio de varias regiones del planeta) para comenzar a crear un modelo mucho más completo, en donde puede incluir información de otra índole (mediciones de velocidad del aire para estimar la difusión de ciertos gases, incluso información un tanto menos directa o no considerada en los estudios originales como la actividad solar y la densidad de algas en varios puntos del oceano que capturan el dióxido de carbono). Esta tarea **es** de los científicos y no es una tarea fácil. El problema de estos datos es que son **demasiados** y que se encuentran escritos para humanos, no *para máquinas*; es decir, la información no está en el formato que un programa fácilmente podría entender.

Esto ni si quiera es un problema que se pueda solucionar estableciendo un estandar para la descripción de la información en la investigación científica. Sería *irrealista* esperar que todos sigan un formato, además, la información tiene una representación óptima dependiendo el tipo de datos que se manejan y a qué público va dirigido. Incluso peor, la información puede no ser legítima (como ejemplo, es sabido que las grandes refresqueras daban dinero a varios laboratorios para publicar investigación que desmintiera los efectos negativos del azúcar en el cuerpo humano, para que su negocio pudiera crecer a costa de la salud mundial) y es la tarea de los científicos desmentir dichos estudios.

Antes de comenzar a dar el curso, se me preguntó si era pertinente con respecto al programa de estudios de la carrera de física (que le pertenece a *ciencias de la computación*, o que de *física nada hay*, que *los actuarios son los que hacen estadística*, o que al fin de cuentas son *matemáticas*). *Sí, sí es pertinente*, pues está en todos los que estamos en el área de la investigación formar un pensamiento crítico y tener nosotros mismos la capacidad de verificar que los datos que nos proporciona un estudio ajeno están bien manejados, ya sean estudios sobre comportamientos so-

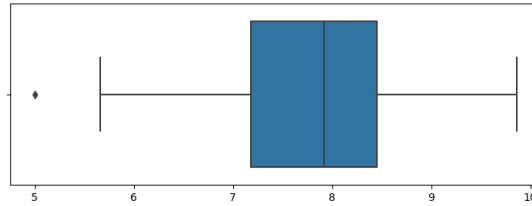


Figura 1: Gráfica de caja para los datos de la serie de promedios.

ciales, descubrimiento de partículas o especies nuevas, o lo que sea. Ya me puse muy poético y me estoy desviando...

En fin, al aceptar la idea de que la información va a estar escrito para humanos, también se acepta el reto de desarrollar sistemas que sean capaces de recolectar la información de manera automatizada sin importar el formato en el que se encuentre.

Por cuestiones de tiempo, nosotros nos vamos a saltar el paso mencionado anteriormente, y vamos a partir de tener la información disponible que podría estar corrupta. A partir de ahora, todo lo que diga se aplica a experimentos en la ciencia también.

2. Series de datos

Vamos a definir como serie de datos una secuencia finita $S = \{s_i\}_{i=1}^n$ de n elementos que pertenecen a \mathbb{R} . En este caso particular, vamos a suponer que estos valores numéricos están asociados al resultado de la mediciones $\{X_1, X_2, \dots, X_n\}$ de una variable aleatoria $X : \Omega \rightarrow \mathbb{R}$ (con X definida sobre el espacio de probabilidad (Ω, F, P)).

Como la serie no necesariamente tiene que estar ordenada, llamaré \hat{S} a la versión de esta serie cuyos elementos están acomodados en orden ascendente.

2.1. Mínimo, máximo y rango

Los elementos \hat{s}_1 y \hat{s}_n son el mínimo y el máximo de la serie respectivamente. El rango está definido como la diferencia de estos dos elementos, $\hat{s}_n - \hat{s}_1$.

2.2. Cuantiles

Los cuantiles son puntos tomados a intervalos regulares de una serie de datos para repartir la información en bloques. Uno de los más usados son los llamados *cuartiles*, los cuales distribuyen la información en 4 bloques (3 cuantiles; 0.25, 0.5 y 0.75).

En general, no hay una regla establecida para determinar el valor de estos puntos (pues el punto puede o no ser parte de la serie de datos, dependiendo si hay un número par o impar de elementos), más bien hay convenciones. Nótese que el segundo cuartil es lo mismo que la media.

Por ejemplo, supongamos la siguiente serie de datos:

Estudiante	Promedio	Estudiante	Promedio	Estudiante	Promedio	Estudiante	Promedio
1	8.41	11	8.10	21	7.24	31	9.57
2	9.10	12	6.46	22	7.12	32	7.95
3	8.30	13	5.94	23	7.14	33	7.79
4	8.70	14	7.65	24	7.98	34	8.96
5	8.58	15	9.16	25	9.48	35	7.19
6	7.49	16	7.88	26	6.10	36	5.95
7	7.42	17	7.58	27	6.94	37	7.69
8	8.25	18	5.66	28	7.19	38	8.22
9	9.86	19	8.53	29	9.27	39	6.55
10	8.02	20	8.43	30	8.22	40	5.00

La manera de determinar la media en este curso es la siguiente:

- Si el número de datos n es impar, tomar el elemento de \hat{S} con índice $r = (n - 1)/2 + 1$
- Si el número de datos n es par, tomar el promedio de los elementos de \hat{S} con índices $r_1 = n/2$ y $r_2 = n/2 + 1$

Para determinar los cuartiles 1 y 3, se toman las respectivas mitades de los datos marcadas por la media. Si el número de datos es impar, **se incluye la media en ambos grupos**. Para la tabla anterior, los tres cuartiles son:

Cuartil 1	7.17
Cuartil 2 (media)	7.91
Cuartil 3	8.45

Nótese que esto es sólo una convención, y no hay problema con que utilicen otra siempre y cuando la justifique y la mencione en su trabajo.

2.3. Momentos de una serie

Cuando se tiene una función de variable real $f(x)$, es posible definir los momentos de esta función alrededor del punto c de la siguiente manera:

$$\mu_k := \int_{-\infty}^{\infty} (x - c)^k f(x) dx \quad (1)$$

A partir de $k = 2$, estos momentos son llamados centrales si $c = \mu := \mu_1$. Si $f(x)$ es una función de densidad material, entonces los primeros tres momentos incluyendo el 0 son la masa total de un sistema, su centro de masa y su momento de inercia al rededor del punto c . Si $f(x)$ representa la función de distribución de probabilidad de una variable aleatoria X , entonces

$$\mu_k = E[(x - c)^k] \quad (2)$$

y el primer momento ($k = 1$) es el valor esperado, el segundo momento central es la varianza. Se definen como momentos normalizados, a partir de $k = 3$, las cantidades μ_k/σ^k . De utilidad son el tercer momento normalizado, conocido como asimetría (positivo si hay más elementos a la derecha del promedio), y el cuarto momento normalizado, conocido como Kurtosis (da información sobre las colas de la distribución respecto al resto de los datos).

En general, no vamos a conocer la distribución de probabilidad que originó nuestra serie de datos, y en muchas ocasiones haremos estadística inferencial en lugar de descriptiva, es decir; No tendremos todos los datos de la población disponibles, más bien un muestreo en representación de toda la población. Cuando es este el caso, se debe usar un estimador para determinar los parámetros estadísticos de una población. El estimador del k-momento de una población al rededor de un punto c basado en una serie S se define como

$$\hat{\mu}_k = \frac{1}{n} \sum_{i=1}^n (s_i - c)^k \quad (3)$$

Para la población total, se obtiene que la varianza es:

$$\sigma^2 := \mu_2 = \frac{1}{N} \sum_{i=1}^N x^2 - \mu^2 \quad (4)$$

2.3.1. Estimadores imparciales

Debería quedar claro que los momentos, son variables aleatorias en sí (dependen de una serie de valores tomados de manera independiente, al azar). Por ello, lo que estamos obteniendo es una estimación del parámetro (momento) de la población total. Se le llama estimador imparcial a un estimador cuyo valor esperado es el parámetro que se intenta estimar. Esto suena tonto, y que debiera cumplirse por sí solo, pero no es así: Tomemos el ejemplo de la varianza.

El valor esperado del estimador de la varianza es:

$$E[\hat{\sigma}^2] = E\left[\frac{1}{n} \sum_{i=1}^n (s_i - \frac{1}{n} \sum_{j=1}^n s_j)^2\right] \quad (5)$$

$$= E\left[\frac{1}{n} \sum_{i=1}^n (s_i^2 - \frac{2s_i}{n} \sum_{j=1}^n s_j + \frac{1}{n^2} \sum_{j=1}^n s_j \sum_{k=1}^n s_k)\right] \quad (6)$$

$$= \frac{1}{n} \sum_{i=1}^n \left(\frac{n-2}{n} E[s_i^2] - \frac{2}{n} \sum_{j \neq i} E[s_i s_j] + \frac{1}{n^2} \sum_{j=1}^n \sum_{k \neq j} E[s_j s_k] + \frac{1}{n^2} \sum_{j=1}^n E[s_j^2]\right) \quad (7)$$

$$= \frac{n-1}{n} \sigma^2 \quad (8)$$

El valor esperado del estimador de la varianza difiere del de la población total por un factor de $(n-1)/n$. Esto quiere decir que si hubiéramos multiplicado nuestro estimador por $n/(n-1)$ habríamos obtenido un **estimador imparcial**, pues su valor esperado es el del parámetro de la población total que se intenta estimar. A dicho factor se le conoce como corrección de Bessel.

Es valido mencionar que para n grande, este factor casi no hace diferencia alguna. Lo importante del análisis es saber que, al construir un estimador de un parámetro de una población se debe buscar que su esperanza coincida con el valor a estimar para la población total, pues ese factor pudo haber sido de mucha mayor relevancia. (aunque en este caso, ha habido un sacrificio en el error promedio cuadrático, pues cada medición va a estar más lejana al valor esperado).

2.3.2. Momentos mixtos: covarianza

Una de las herramientas más útiles para hacer análisis de datos o alimentar métodos de machine learning es deshacernos de información que no aporta nada nuevo (o en su caso, detectar relaciones entre series de datos). Este puede ser el caso cuando dos series de datos fluctúan manteniendo cierta relación.

La covarianza es un momento mixto que detecta esta relación (de manera lineal), y se define como:

$$\sigma(x, y) = E[(x - E[x])(y - E[y])] \quad (9)$$

el cual tiene su propio estimador

$$\hat{\sigma}_{xy} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (10)$$

donde ya he incluido la corrección de Bessel. Como observación trivial, $\sigma_{xx} = \sigma^2$

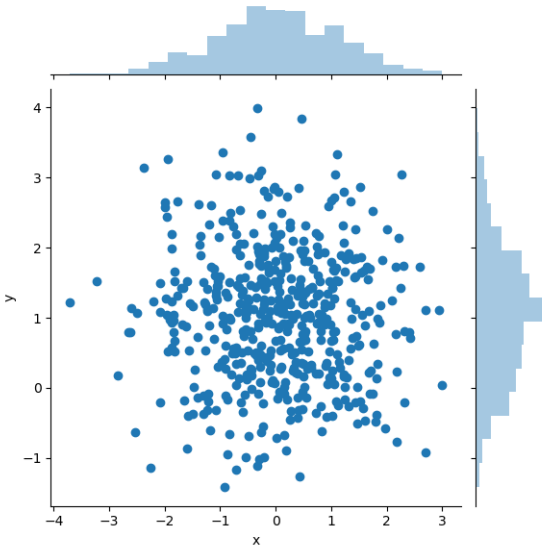


Figura 2: $\sigma_{xy} = 0,0$

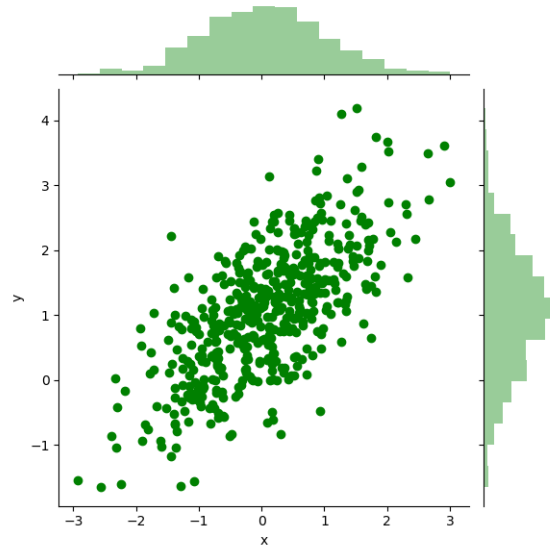


Figura 3: $\sigma_{xy} = 0,8$

Las figuras 2 y 3 son los resultados de una simulación de un proceso gaussiano multivariado de 500 mediciones. El primer proceso tiene $(\mu = (0, 1)^T, \sigma = ((1, 0)^T, (0, 1)^T))$, mientras el segundo tiene como parámetros $(\mu = (0, 1)^T, \sigma = ((1, 0, 8)^T, (0, 8, 1)^T))$. Los histogramas de cada dimensión son indistinguibles, pero es claro que hay una correlación en el segundo gráfico.

2.3.3. Matriz de covarianza

La matriz de covarianza σ (o bien, en un futuro *función de covarianza*) es una matriz cuyas entradas están dadas por:

$$\sigma_{x_i x_j} = E[(x_i - E[x_i])(x_j - E[x_j])] \quad (11)$$

Donde cada x_i es la variable aleatoria asociada a una serie de datos. La matriz contiene en la diagonal las varianzas de cada variable. Como nota, nosotros trabajaremos en su mayoría con la estimación de la matriz de covarianza:

$$\hat{\sigma}_{xy} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (12)$$

Para cualquier par de variables x e y .

2.3.4. Coeficiente de correlación

Una vez obtenida la estimación de la matriz de covarianza, se puede definir una cantidad conocida como *coeficiente de correlación*:

$$r_{xy} = \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x \hat{\sigma}_y} \quad (13)$$

Donde el denominador está compuesto de las desviaciones estándar de las variables x e y respectivamente.

Valores atípicos

Carlos Malanche

22 de febrero de 2018

Vamos primero a definir lo que es un valor atípico, pero vamos a hacerlo con una frase:

Un valor atípico es una observación que se aleja tanto de la mayoría de las otras observaciones que levanta la sospecha de no haber sido generada por el mismo mecanismo que el resto

Hay que ser muy cuidadosos con el manejo de valores atípicos, pues estos pueden tener distintas razones de fondo, por ejemplo:

- Error en la captura de la información
- Error en la transmisión de la información
- Error en el manejo de la información
- Errores experimentales
- Una *novedad*

Se le llama *novedad* a un valor atípico que se está considerando como tal por no prever su existencia, es decir, estos valores NO se deben tirar y se debe replantear la teoría sobre la que trabajamos para incluir su existencia.

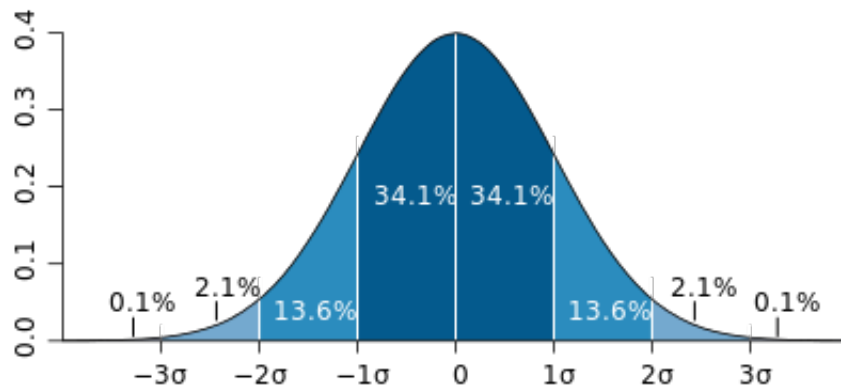
Existen diversos métodos para encontrar valores atípicos, pero nosotros nos vamos a enfocar por el momento en 3 pues 2 son los clásicos y uno algo nuevo.

1. Z-Score

Para utilizar este método, se asume que la serie de datos sigue una distribución Gaussiana, lo cual implica que la varianza es un parámetro estadístico pertinente. Dada la desviación estándar de la serie, se coloca un valor límite (usualmente un múltiplo de la desviación estándar) a partir de la cual se consideran valores atípicos. Un poco más formal, de la serie $S = \{s_i\}_{i=1}^n$ se deriva una serie de *Z-scores* $Z = \{z_i\}_{i=1}^n$ donde

$$z_i = \frac{s_i - \mu}{\sigma} \quad (1)$$

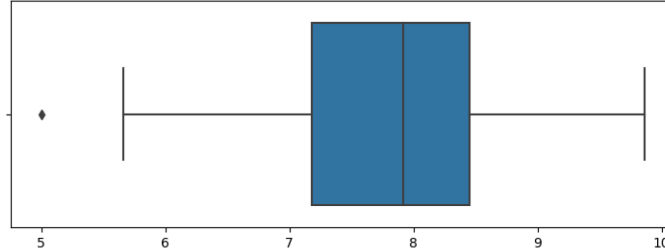
es el número de desviaciones estándar que el valor de la variable aleatoria s_i está alejado de la media.



En la figura anterior está la cantidad de información contenida bajo la curva de una distribución normal. En base a esta, valores límite populares para deshacernos de posibles valores atípicos son al menos 2.5 desviaciones estándar.

2. Interquartile range (IQR)

Como lo vimos para ver las gráficas de *caja* (en inglés *box with whiskers*), se utilizan los cuartiles para encontrar posibles outliers.



Bajo este esquema, se clasificará como valor atípico cualquier valor s_i de la serie que no esté contenido en el rango

$$(Q_1 - 1,5 * IQR, Q_3 + 1,5 * IQR) \quad (2)$$

En donde Q_1 y Q_3 son el primer y tercer cuartil respectivamente, y la región intercuartil IQR es la diferencia de los mismos ($IQR = Q_3 - Q_1$).

Si usted se pregunta *por qué exactamente 1.5 y no 1.123 ó $\pi/2$* la respuesta es muy sencilla: John Turkey, creador de la gráfica de caja, decidió arbitrariamente que 1.5 era un buen múltiplo para detectar valores atípicos. La gente lo siguió usando dado que dio *buenos resultados*, y entonces ya no se cuestiona realmente.

Sin embargo, asumiendo que se tiene una distribución normal, utilizando la función acumulativa de distribución se puede notar que $Q_3, Q_1 \approx \pm 0,68\sigma$, con eso obtenemos que $IQR \approx 1,36\sigma$, lo que nos dice que un valor atípico será todo aquel que esté más de $2,72\sigma$ lejos de la media, lo cual es masomenos el 1 % de la información bajo una curva de Bell.

3. DBSCAN

Las siglas significan *Density-Based Spatial Clustering of Applications with Noise* [1], y es un algoritmo de clustering; así es, emocionense que nuestro primer algoritmo de machine learning ha llegado.

Aunque el propósito del algoritmo es encontrar clusters de información con un poco de ruido, ha adquirido popularidad al usarse para detectar valores atípicos. Piense en el siguiente escenario: Se hacen mediciones de un evento cuya distribución de probabilidad son dos gaussianas de mismos parámetros estadísticos, con sus medias separadas por 10σ . Un valor que se encontrara a la mitad no se podría detectar asumiendo una distribución Gaussiana, y tampoco buscando un valor extremo.

La mentalidad del algoritmo es buscar la información que se encuentre *cercana* para agruparla. Si un elemento no parece tener un grupo, entonces quedará marcada como valor atípico.

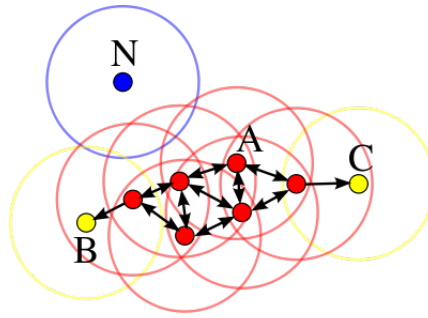
El algoritmo depende principalmente de una métrica establecida con la que se determina la distancia entre dos puntos de nuestros datos (puede ser una simple métrica euclidiana, o la de *Manhattan*), y de dos parámetros más: ϵ el radio de la vecindad conforme la métrica seleccionada y *minPts*, el número mínimo de vecinos para formar el núcleo de un *cluster*. A los datos se les asigna una métrica que los coloque espacialmente en un espacio. La métrica puede contemplar una de las variables que describen los puntos, un subconjunto del total, o todas.

Vamos a añadir las siguientes tres definiciones para el método:

- Punto núcleo: Un punto es un punto núcleo si en su vecindad de radio ϵ hay al menos tantos puntos como el parámetro *minPts* lo indica.
- Punto frontera: Un punto es un punto frontera si tiene menos puntos en su vecindad de radio ϵ de lo que *minPts* indica, pero en su vecindad contiene un punto núcleo al menos.

- Valor atípico: Los puntos que no caen en los primeros dos casos son valores atípicos.

Al calcular el número de elementos que vive en un cluster con respecto a un punto, se debe incluir el punto mismo en la cuenta.



En el ejemplo de arriba, se tiene que $minPts = 4$ (ϵ es gráfico nadamás). El algoritmo se corre en un orden específico:

- Primero se buscan todos los puntos núcleo, ignorando cualquiera que no califique.
- Con los puntos núcleo se hacen los clusters correspondientes (se ponen en un solo grupo los puntos que son vecinos).
- Los puntos restantes se separan en dos grupos: Los que se pueden anexar a un cluster, y los que son valores atípicos (*ruido*).

Al final, quedamos con un grupo de clusters que por el momento son indistinguibles. Los clusters de 1 elemento son marcados como valores atípicos. Como lo veremos en clases posteriores, el truco de este último método está en estimar correctamente los parámetros ϵ y $minPts$. $minPts$ tendrá el efecto de definir el número mínimo de elementos en un cluster, y ϵ qué tan *similar* queremos que la información sea (asumiendo que nuestra métrica mide similitud). Algunas estrategias comunes son calcular la interdistancia de todos los elementos. Dependiendo del histograma resultante de este cómputo, se utiliza una adivinanza de ϵ para contar cuántos vecinos tiene cada elemento y ver ahora qué porcentaje de la información será tirado en función de $minPts$.

Referencias

- [1] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231.

**Instructivo de descarga de las Bases de Datos de Cómputos Distritales 2006
de la página del IFE: [http:// www.ife.org.mx](http://www.ife.org.mx)**

1. En la página inicial del Instituto Federal Electoral se encuentra el hipervínculo **Cómputos Distritales**:

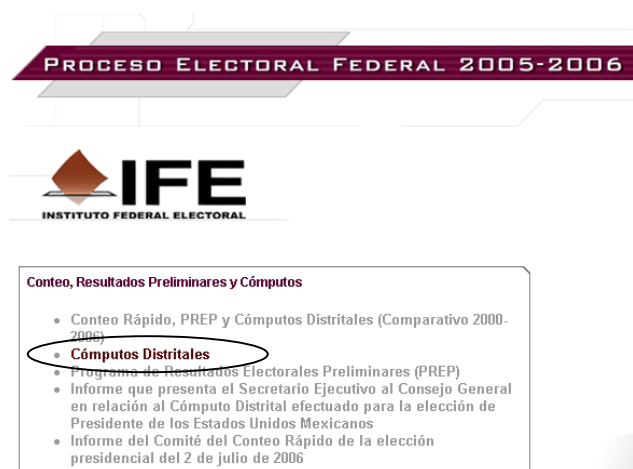


Figura 1

Este hipervínculo permite acceder a la información siguiente:

- **Cómputos Distritales**
- **Catálogo de descargas de las Bases de Datos de Cómputos Distritales 2006**

Por medio del apartado **Catálogo de descargas de las Bases de Datos de Cómputos Distritales 2006**, se pueden obtener archivos de texto "*plano*" de los:

- Cómputos de la elección de **Presidente** de los Estados Unidos Mexicanos, que incluye el cómputo del Voto de los Mexicanos Residentes en el Extranjero.
- Cómputos de la elección de Diputados por los Principios de Mayoría Relativa y Representación Proporcional.
- Cómputos de la elección de Senadores por los Principios de Mayoría Relativa y Representación Proporcional.

Por ejemplo, para obtener el archivo de texto de la elección de **Presidente** de los Estados Unidos Mexicanos, a partir de la página inicial del Instituto, se debe presionar el hipervínculo **Cómputos Distritales** y después se deben seguir las instrucciones que se describen a continuación.

2. Como resultado del paso anterior se visualiza la pantalla siguiente:

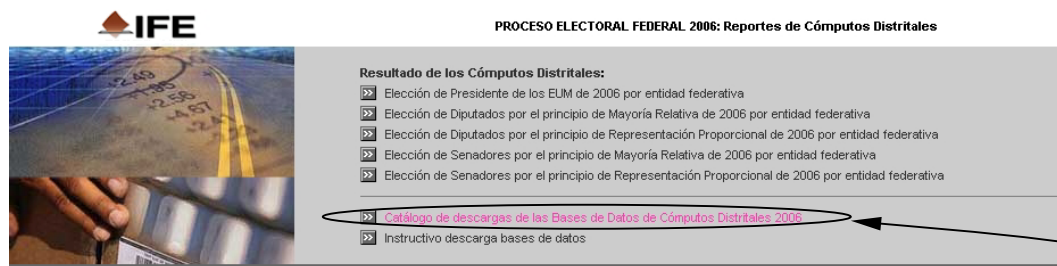


Figura 2

En esta pantalla se presiona el hipervínculo **Catálogo de descargas de las Bases de Datos de Cómputos Distritales 2006**.

NOTA IMPORTANTE: Los archivos de texto de las bases de datos se encuentran comprimidos, por lo que se requiere del programa **Winzip** para descomprimirlos.

- En la página siguiente, se debe presionar el hipervínculo que contiene los archivos de texto de la elección deseada. Continuando con el ejemplo, para obtener los de **Presidente** se debe hacer clic sobre: **Computos2006-Presidente.zip**:

Hipervínculos que permiten obtener los archivos de texto de las bases de datos de los diferentes tipos de elección.

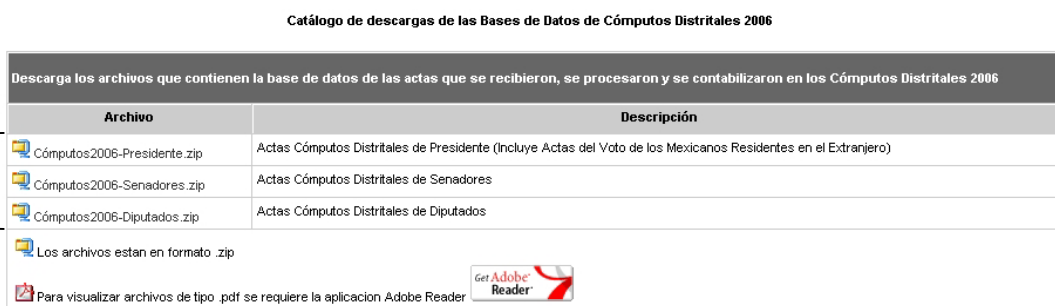


Figura 3

- Al presionar cualquiera de los tres hipervínculos señalados en la Figura 3 (para este ejemplo se presionó el de la elección de **Presidente** de los Estados Unidos Mexicanos, como ya se mencionó) se observará una ventana como la siguiente:

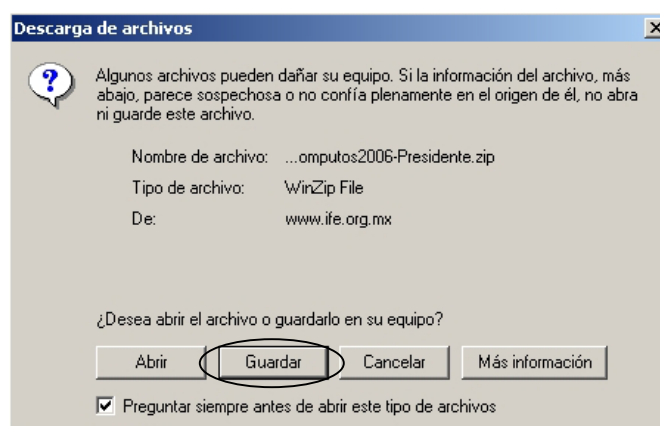


Figura 4

Para descargar el archivo comprimido, se presiona el botón **Guardar** de la ventana anterior.

5. En la ventana “**Guardar como**” que aparece, se debe seleccionar alguna ubicación en el disco duro de la computadora o bien, alguna unidad de almacenamiento de información y posteriormente presionar el botón **Guardar**:

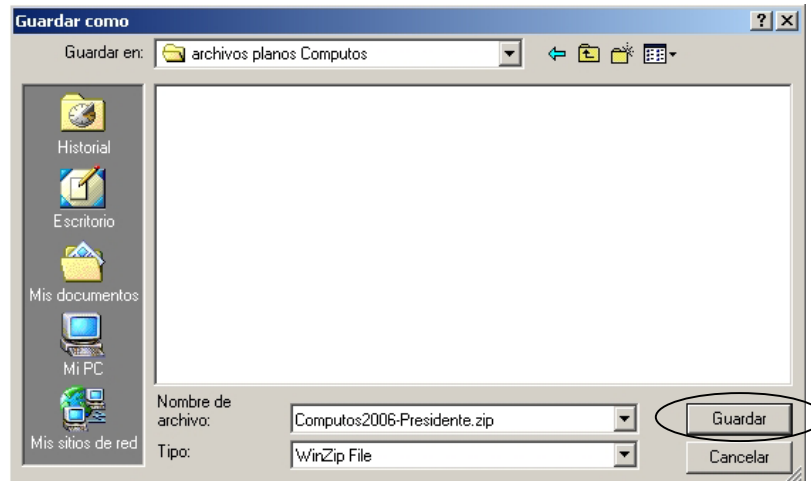


Figura 5

6. Una vez que termine de descargarse el archivo, se procede a descomprimirlo con el programa Winzip conforme a lo mencionado en la Figura 6.

1. Se ingresa a la carpeta donde almacenó el archivo.

2. Se coloca el **puntero del ratón** sobre el nombre del archivo y se presiona el botón **DERECHO**. Aparece un menú contextual del que se debe seleccionar la opción con el nombre más “largo”: **Extract to folder.....** Esta opción permite crear una carpeta de forma automática y al mismo tiempo descomprimir el archivo y depositarlo en la carpeta.

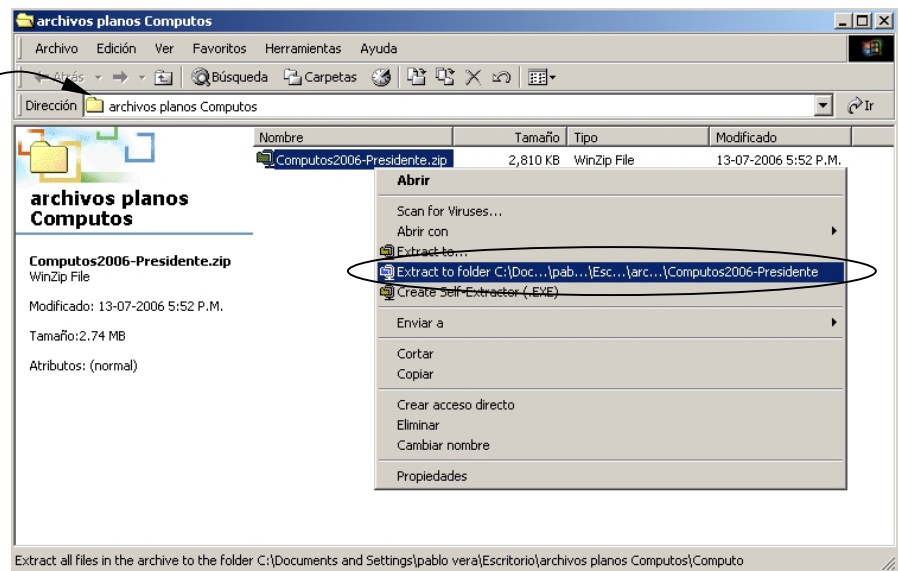


Figura 6

7. En la ventana que aparece se oprime el botón “*I agree*”.

8. El resultado se observa a continuación:

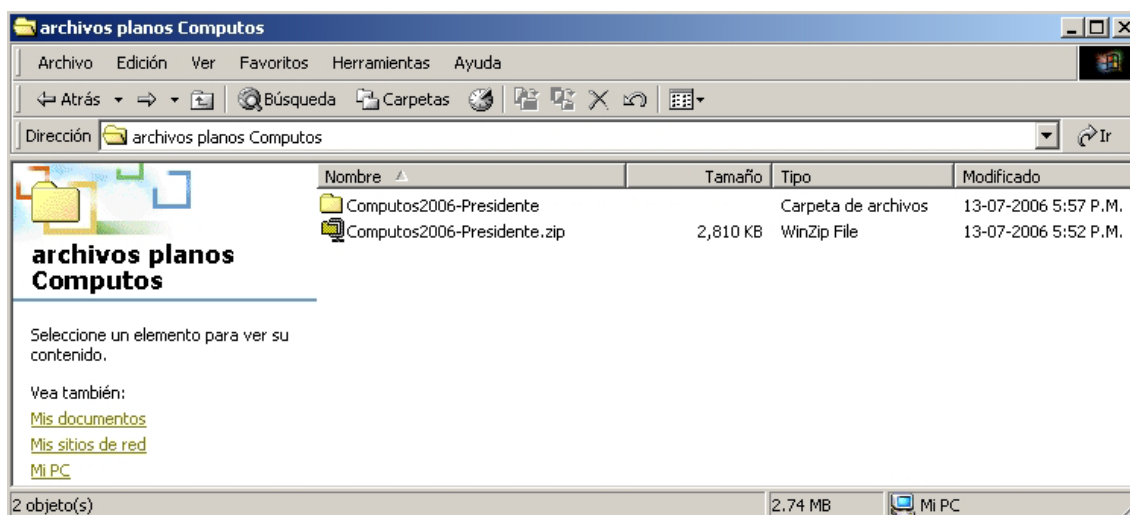


Figura 7

Si se ingresa a la carpeta que se creó: “*Computos...*” para observar su contenido, se podrán visualizar dos archivos semejantes a los que aparecen en la Figura 8.

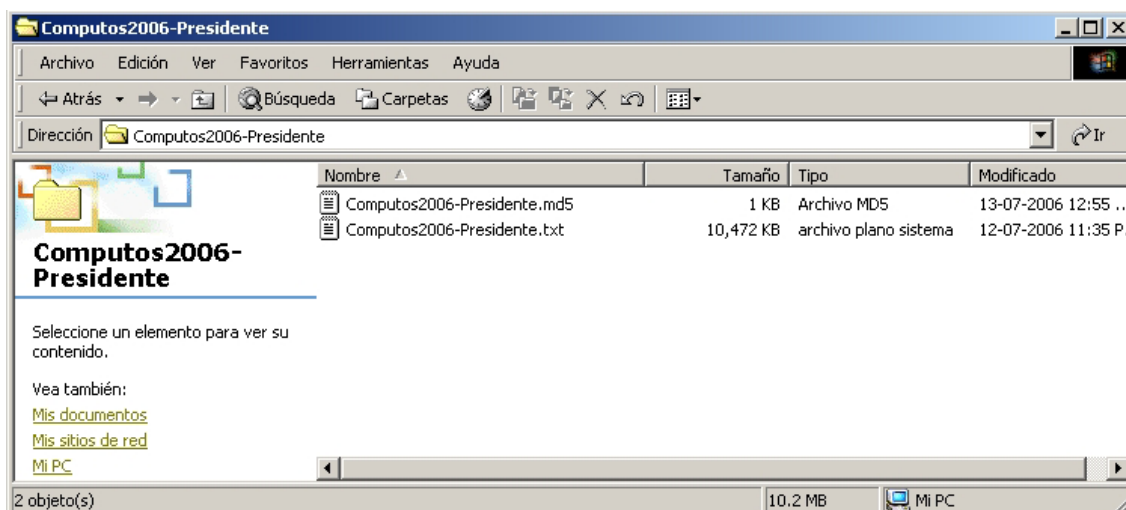


Figura 8

El archivo que contiene la base de datos es el que cuenta con la extensión **.txt**, o bien que aparece como **“Documento de texto”**.

NOTA IMPORTANTE: Una vez que se cuente con los archivos de texto plano, para visualizarlos y utilizarlos de forma conveniente, se podrá utilizar el programa “manejador” de bases de datos que se prefiera, como por ejemplo: Access, FoxPro, MySQL, etc. El símbolo DELIMITADOR que se utilizó para separar los campos de las tablas es el “|”, que se puede obtener con la combinación de teclas ALT + 124. NO se debe usar Excel debido a que la hoja de cálculo está limitada a 65,536 registros únicamente, y las bases de datos de los resultados de cada tipo de elección contienen aproximadamente 130,000 registros.

A continuación se explican brevemente los campos que contiene la base de datos.

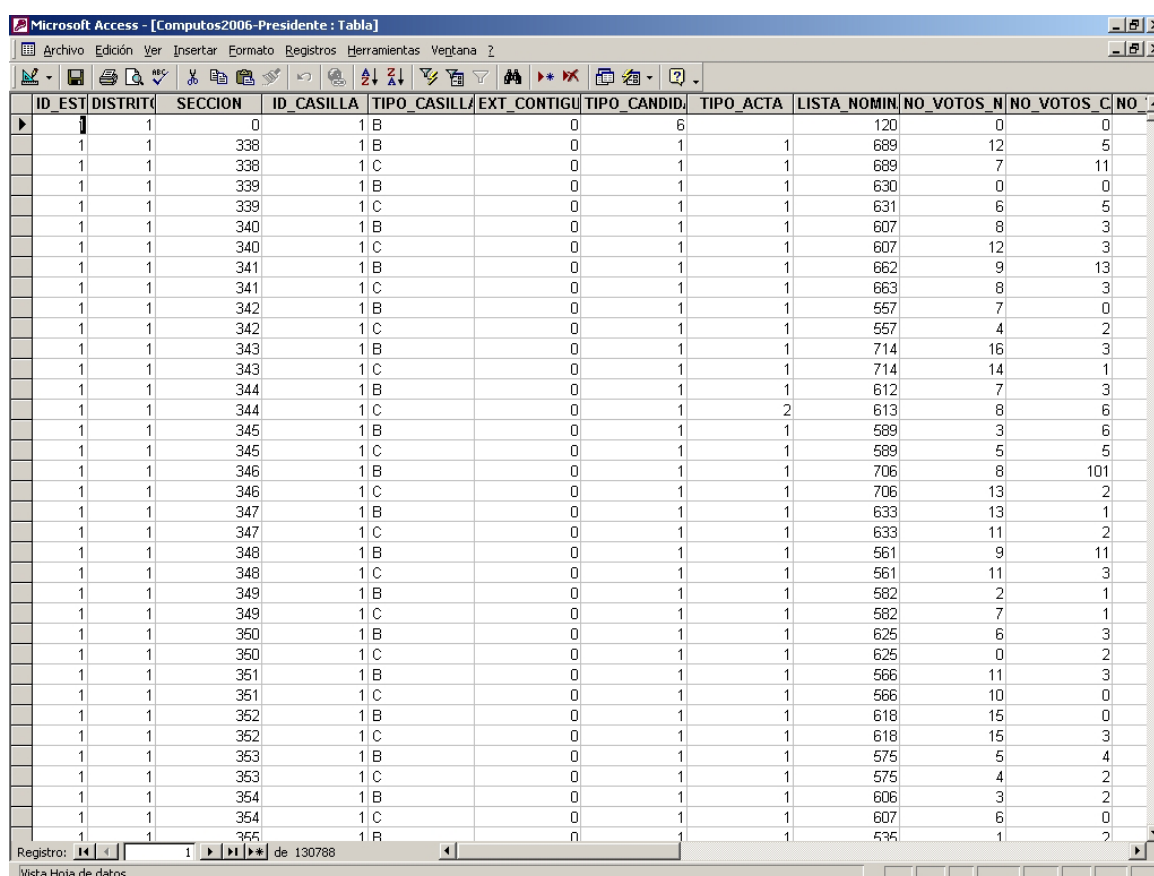
ID_ESTADO	Clave del estado
DISTRITO	Distrito electoral
SECCION	Numero de la Seccion [0 para voto en el extranjero]
ID_CASILLA	Identificador de la casilla
TIPO_CASILLA	Tipo de la casilla: B.- Básica, C- Contigua, E.- Extraordinaria, S.- Especial
EXT_CONTIGUA	Numero de la casilla contigua a la extraordinaria
TIPO_CANDIDATURA	Tipo de candidatura: 1.-Presidente, 6.-Voto en el extranjero, 2.-Senadores RP, 3.-Senadores MR, 4.-Diputados RP, 5.-Diputados MR
TIPO_ACTA	Tipo de acta: 1.- Casilla o Mesa Instalada; 2.- Consejo; 3.- No Instalada; 4.- Paquete no Entregado
LISTA_NOMINAL	Ciudadanos en lista nominal
NO_VOTOS_NULOS	Número de votos nulos
NO_VOTOS_CAN_NREG	Número de votos para candidatos no registrados
NO_VOTOS_VALIDOS	Número de votos validos
TOTAL_VOTOS	Total de votos
ORDEN	Orden en el que debe aparecer la casilla para su captura
PAN	Número de votos para el Partido Acción Nacional
APM	Número de votos para la coalición Alianza por México
PBT	Número de votos para la coalición Por el Bien de Todos
NA	Número de votos para el Partido Nueva Alianza
ASDC	Número de votos para el Partido Alternativa Social Democrática y Campesina
MUNICIPIO	Clave del municipio
PAQUETE_ENTREGADO	Paquete entregado: 0.-Sin paquete, 1= Con paquete
CASILLA_INSTALADA	Casilla instalada: 0.-No instalada, 1.-Instalada
FECHA_HORA	Fecha y hora de la última actualización en el registro

Catálogo de ESTADOS:

ID_ESTADO	ESTADO
1	AGUASCALIENTES
2	BAJA CALIFORNIA
3	BAJA CALIFORNIA SUR
4	CAMPECHE
5	COAHUILA
6	COLIMA
7	CHIAPAS
8	CHIHUAHUA
9	DISTRITO FEDERAL
10	DURANGO
11	GUANAJUATO
12	GUERRERO
13	HIDALGO
14	JALISCO
15	MEXICO
16	MICHOACAN
17	MORELOS
18	NAYARIT
19	NUEVO LEON
20	OAXACA

21	PUEBLA
22	QUERETARO
23	QUINTANA ROO
24	SAN LUIS POTOSI
25	SINALOA
26	SONORA
27	TABASCO
28	TAMAULIPAS
29	TLAXCALA
30	VERACRUZ
31	YUCATAN
32	ZACATECAS

Observe el aspecto de la base importada por medio de Access:



ID_EST	DISTRITO	SECCION	ID_CASILLA	TIPO_CASILLA	EXT_CONTIGUO	TIPO_CANDID	TIPO_ACTA	LISTA_NOMIN	NO_VOTOS_N	NO_VOTOS_C	NO_VOTOS_T
1	1	0	1 B		0	6		120	0	0	
1	1	338	1 B		0	1	1	689	12	5	
1	1	338	1 C		0	1	1	689	7	11	
1	1	339	1 B		0	1	1	630	0	0	
1	1	339	1 C		0	1	1	631	6	5	
1	1	340	1 B		0	1	1	607	8	3	
1	1	340	1 C		0	1	1	607	12	3	
1	1	341	1 B		0	1	1	662	9	13	
1	1	341	1 C		0	1	1	663	8	3	
1	1	342	1 B		0	1	1	557	7	0	
1	1	342	1 C		0	1	1	557	4	2	
1	1	343	1 B		0	1	1	714	16	3	
1	1	343	1 C		0	1	1	714	14	1	
1	1	344	1 B		0	1	1	612	7	3	
1	1	344	1 C		0	1	2	613	8	6	
1	1	345	1 B		0	1	1	589	3	6	
1	1	345	1 C		0	1	1	589	5	5	
1	1	346	1 B		0	1	1	706	8	101	
1	1	346	1 C		0	1	1	706	13	2	
1	1	347	1 B		0	1	1	633	13	1	
1	1	347	1 C		0	1	1	633	11	2	
1	1	348	1 B		0	1	1	561	9	11	
1	1	348	1 C		0	1	1	561	11	3	
1	1	349	1 B		0	1	1	582	2	1	
1	1	349	1 C		0	1	1	582	7	1	
1	1	350	1 B		0	1	1	625	6	3	
1	1	350	1 C		0	1	1	625	0	2	
1	1	351	1 B		0	1	1	566	11	3	
1	1	351	1 C		0	1	1	566	10	0	
1	1	352	1 B		0	1	1	618	15	0	
1	1	352	1 C		0	1	1	618	15	3	
1	1	353	1 B		0	1	1	575	5	4	
1	1	353	1 C		0	1	1	575	4	2	
1	1	354	1 B		0	1	1	606	3	2	
1	1	354	1 C		0	1	1	607	6	0	
1	1	355	1 B		0	1	1	535	1	2	

Figura 9

El procedimiento para importar este archivo en Access se realiza siguiendo la secuencia:

- Crear una base de datos en blanco.
- Del menú *Archivo*, seleccionar la opción *Obtener datos externos* → *Importar*.
- Localizar el archivo de texto de la base de datos y posteriormente seguir los pasos convenientes del *Asistente de importación*.

A continuación se ejemplifica la consulta de los resultados de la *votación de los mexicanos residentes en el extranjero* a partir de la base de datos correspondiente a la elección de Presidente de los Estados Unidos Mexicanos utilizando el programa Access:

A. Se crea una consulta en “vista de diseño” conforme a lo siguiente:

3. Se ejecuta la consulta presionando este icono.

1. Con un doble clic se seleccionan los campos de la tabla de la base de datos en el orden de aparición que se desea.

2. Se establecen los criterios deseados, en este caso, el ID_ESTADO=1 (AGUASCALIENTES), DISTRITO=1 (Jesús María), SECCIÓN=0, existente sólo para el voto de los mexicanos en el extranjero y TIPO_CANDIDATURA=6 (Voto en el extranjero).

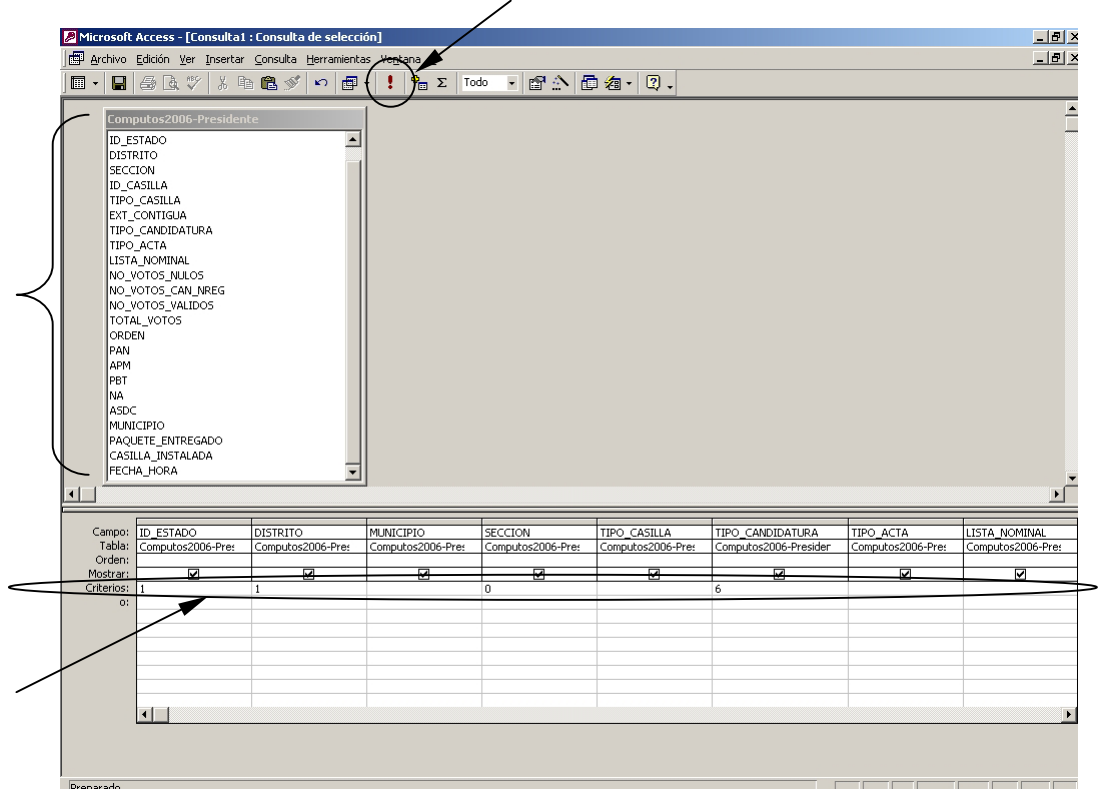


Figura 10

B. El resultado de dicha consulta se muestra enseguida:

Icono para regresar a la “vista de diseño” de la consulta.

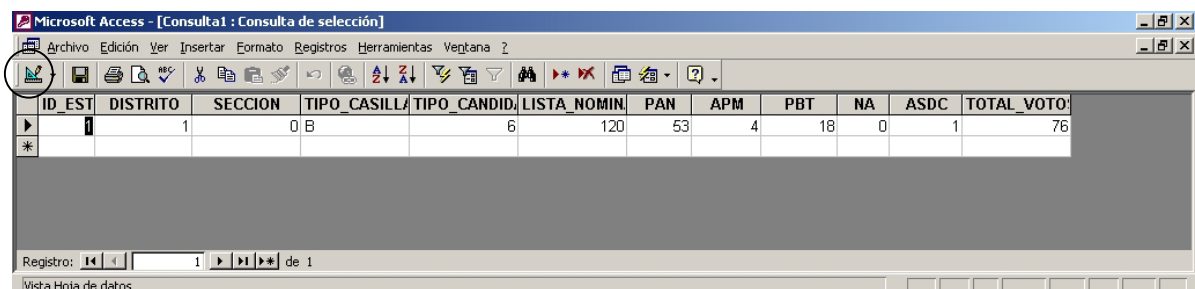


Figura 11

Es posible establecer los criterios de filtrado que se deseen, regresando a la “vista de diseño” de la consulta.

Interpolación

Carlos Malanche

8 de marzo de 2018

Ok, ahora contamos con un arreglo de datos al que ya le hemos hecho estadística y ya nos hemos encargado de entenderlo y limpiarlo un poco. Una de las primeras cosas que nos gustaría hacer es adaptarle un modelo para poder describir con precisión lo que hace

1. Definiciones

Una serie de parejas de datos estará definida como

$$D = \{x_i, y_i\}_{i=1}^n \quad (1)$$

Por simplicidad, tanto x_i como y_i serán escalares (pronto usaremos vectores).

2. Interpolación

El primer paso para hacer una predicción de información es la interpolación (la cual viene con sus desventajas). Un ejemplo muy lindo de interpolación son los polinomios de lagrange.

Dada una serie de parejas de datos D , vamos a definir como su polinomio interpolante de Lagrange

$$L(x) = \sum_{i=1}^n \ell_i(x) y_i \quad (2)$$

aquel de grado n cuyos componentes $\ell_i(x)$ están dados por

$$\ell_i(x) = \prod_{j \neq i}^n \frac{x_j - x}{x_j - x_i} \quad (3)$$

Note rápidamente que

$$\ell_i(x_j) = \begin{cases} 1, & \text{si } i = j \\ 0, & \text{o.c.} \end{cases} \quad (4)$$

El resultado es una función que interpola nuestra información de manera perfecta.

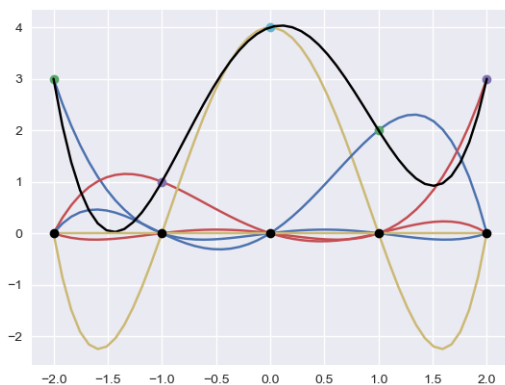


Figura 1: Polinoio de Lagrange con 5 puntos

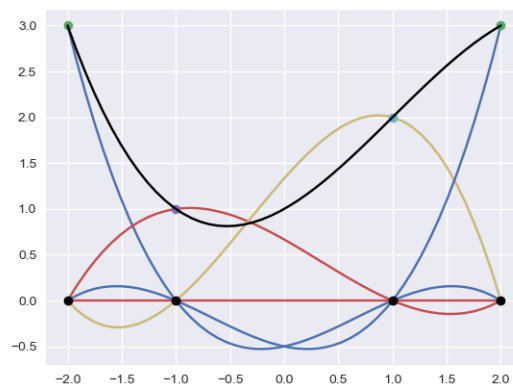
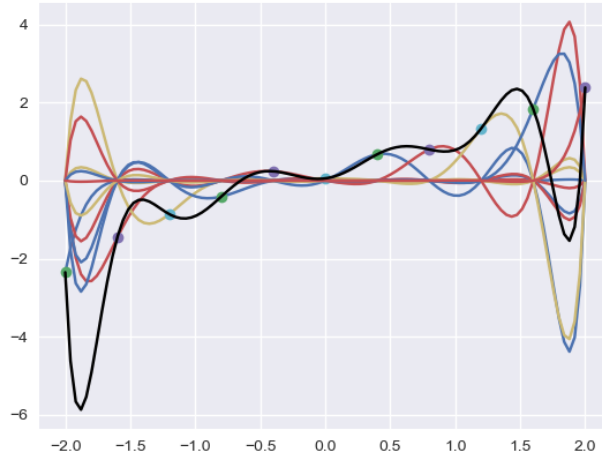


Figura 2: Polinomio de Lagrange con 4 puntos

Este resultado es demasiado inestable. Basta con retirar uno de los puntos para que los dos modelos no sean parecidos ni si quiera. A este fenómeno le llamaremos *overfitting* o sobreajuste.

El problema principal con una interpolación es que nos obligamos a tener tantos *grados de libertad* como datos tenemos. Esto convierte a nuestro modelo en uno muy complejo, lo hace dependiente de la información que le pasamos y lo hace poco flexible ante datos que aún o conocemos.



Noten que los puntos que estamos interpolando están **casi** en una línea, pero como el polinomio es de un grado muy alto comienza a dispararse cerca de las orillas.

2.1. No todo es malo

Tal y como lo he planteado, una interpolación siempre tiene desventajas. Los polinomios de Lagrange son probablemente el método de interpolación más sencillo, por lo cual no es muy recomendable usarlo pero ha aportado mucho a este campo. Una *interpolación* mucho, mucho más compleja son las series y transformada de Fourier, las cuales permiten pasar de una serie de datos equiespaciados a un conjunto de frecuencias que definen una función continua con la cual podemos hacer predicciones de valores que no se midieron en un principio.

La aplicación de esta interpolación es un tanto distinta, pero es un ejemplo en donde se asume que cada medición no contiene ruido, se asume que es una medición perfecta.

3. Una función más compleja

La interpolación es una buena solución cuando tenemos certeza de los datos con los que contamos. Ahora vamos a suponer que los datos que medimos fueron generados por una función $f(x)$ **la cual es imposible de encontrar**, y asumiremos también que en el proceso de medición hubo un poco de ruido involucrado, con lo que cualquier medición se podría describir como:

$$y = f(x) + \epsilon \quad (5)$$

donde ϵ es una variable aleatoria de distribución de gauss, centrada en cero y con varianza σ^2 . Tan rápido como hemos optado por este modelo, es claro que *no tiene sentido buscar una interpolación* pues la probabilidad de que $y = f(x)$ es cero. Si no buscamos una interpolación, buscamos entonces una función estimadora $\hat{f}(x)$ que aproxime *suficientemente bien* nuestros datos...

Antes de intentar arreglar un problema, hay que entender bien el problema y definir un marco para solucionarlo. En este caso, hay varias maneras de *medir* qué tan buenas son nuestras funciones estimadoras, lo haremos por medio de funciones conocidas como *funciones de costo*, que por lo regular serán denotadas por una J . Como casi todo en este campo, no hay una regla única para optar por una función de costo, pero la gran mayoría de los métodos de aprendizaje estadístico utilizan una función de costo cuadrática (en busca de obtener un problema de optimización

convexo).

Definimos así pues, la función de pérdidas cuadrática como:

$$J(\vec{f}) = \frac{1}{2N} \sum_{i=1}^n \|\vec{f}(\underline{x}_i) - \underline{y}_i\|^2 \quad (6)$$

donde en un abuso de notación he convertido a \vec{f} en el parámetro de la función de costo.

3.1. Pequeña nota sobre la interpolación

La interpolación es un modelo de muchos parámetros, supongamos pues $\vec{f} := L(x)$. Resulta pues que en este modelo la varianza es muy alta respecto a la parcialidad (o el sesgo). Para ver esto, estimemos el error cuadrático generado por un dato que no está contenido en la interpolación $\{x_0, y_0\}$.

Trivialmente, $\text{Var}[y_0] = \sigma^2$ (por el comportamiento determinístico de f). Haciendo un cálculo pequeño, se tiene que de la definición de la varianza de una variable aleatoria X

$$\text{Var}[X] = E[X^2] - E[X]^2 \quad (7)$$

Para acortar un poquito la notación, $\hat{f}_0 := \vec{f}(x_0)$ (lo mismo sin el gorrito).

$$\begin{aligned} E[(y_0 - \hat{f}_0)^2] &= E[y_0^2 + \hat{f}_0^2 - 2y_0\hat{f}_0] \\ &= E[y_0^2] + E[\hat{f}_0^2] - 2E[y_0\hat{f}_0] \\ &= \text{Var}[y_0] + E[y_0]^2 + \text{Var}[\hat{f}_0] + E[\hat{f}_0]^2 - 2E[y_0]E[\hat{f}_0] \\ &= \text{Var}[y_0] + \hat{f}_0^2 + \text{Var}[\hat{f}_0] + E[\hat{f}_0]^2 - 2\hat{f}_0E[\hat{f}_0] \\ &= \text{Var}[y_0] + \text{Var}[\hat{f}_0] + (\hat{f}_0^2 - 2\hat{f}_0E[\hat{f}_0] + E[\hat{f}_0]^2) \\ &= \sigma^2 + \underbrace{\text{Var}[\hat{f}_0]}_{\text{Varianza}} + \underbrace{(\hat{f}_0 - E[\hat{f}_0])^2}_{\text{Sesgo}^2} \end{aligned}$$

A lo mejor es un poco complicado de entenderlo así nadamás de verlo, pero imaginemos que los métodos de predicción son elegidos al azar (o más bien, sus parámetros son elegidos al azar). Resulta que el error esperado está compuesto por una parte irreducible, que es la varianza de la misma medición, y dos términos que representan la varianza de la función de predicción (se interpreta como qué tan diferentes son las funciones entre sí dadas la elección aleatoria de sus parámetros) y su sesgo (se interpreta como qué tan lejos va a estar la media de la evaluación de x_0 en \vec{f} del valor sin ruido real de la medición $f(x_0)$).

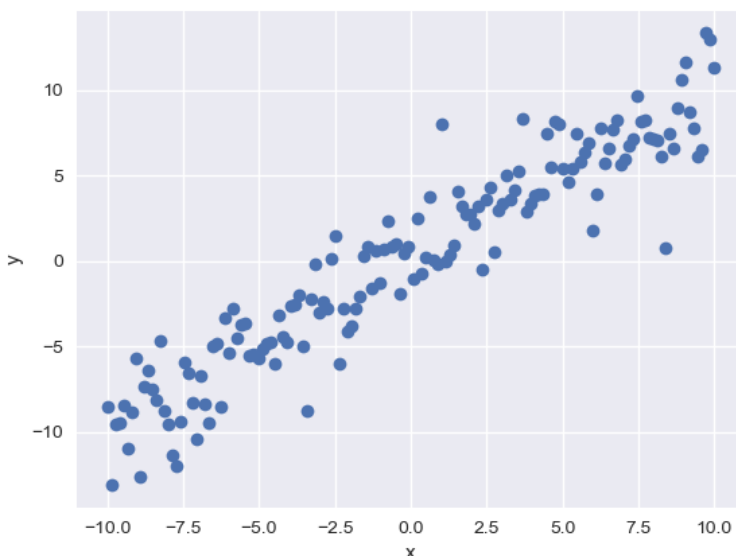
Por el momento tendrán que tomar mi palabra, pero estas dos tienen una correlación negativa: si la varianza es alta, el sesgo es bajo y al revés. El método de interpolación polinómica es de varianza muy elevada, pues cada polinomio es muy diferente con sólo modificar un punto.

Regresión Lineal

Carlos Malanche

13 de marzo de 2018

Ok, regresemos a lo que se tenía la clase pasada. El problema es encontrar una función estimadora \hat{f} que aproxime f dadas las parejas de datos $\{x_i, y_i = f(x_i) + \epsilon\}_{i=1}^n$ donde, sin conocimiento previo, ϵ es ruido de distribución Gaussiana con varianza σ^2 y primer momento $\mu = 0$. Bien, pues supongamos que tomamos nuestra serie de pares de datos, y la graficamos, obteniendo lo siguiente:



Pues, a reserva de abstenciones, yo voto que es una línea con ruido, sí o no raza?

1. Caso de una variable

Por eso mismo sería una buena idea proponer $\hat{f}(x) = mx + b$. Ahora, dada la serie de datos, cómo estimamos los parámetros m y b ?

En la clase anterior vimos que la función de pérdidas cuadrática (para funciones escalares) está definida como

$$J(\hat{f}) = \frac{1}{2n} \sum_{i=1}^n (\hat{f}(x_i) - y_i)^2 \quad (1)$$

En nuestro caso ya no es necesario escribir J como un funcional, ya sabemos de qué variables va a depender el costo

$$J(m, b) = \frac{1}{2n} \sum_{i=1}^n (mx_i + b - y_i)^2 \quad (2)$$

Suena como una buena idea derivar la función de costo respecto a cada variable e igualar a cero para encontrar un *mínimo*. Primero respecto a m

$$\frac{\partial J}{\partial m}(m, b) = \frac{1}{n} \sum_{i=1}^n (mx_i + b - y_i)x_i = 0 \quad (3)$$

Y ahora respecto a b

$$\frac{\partial J}{\partial b}(m, b) = \frac{1}{n} \sum_{i=1}^n (mx_i + b - y_i) = 0 \quad (4)$$

Esto es fácil de resolver, pues tenemos un sistema de dos ecuaciones con dos incógnitas, siendo la ecuación lineal en ambas variables. Abriremos la suma y factorizaremos para que sea más obvio (además de multiplicar ambas ecuaciones por n).

$$\left(\sum_{i=1}^n x_i^2\right)m + \left(\sum_{i=1}^n x_i\right)b = \sum_{i=1}^n x_i y_i \quad (5)$$

$$\left(\sum_{i=1}^n x_i\right)m + (n)b = \sum_{i=1}^n y_i \quad (6)$$

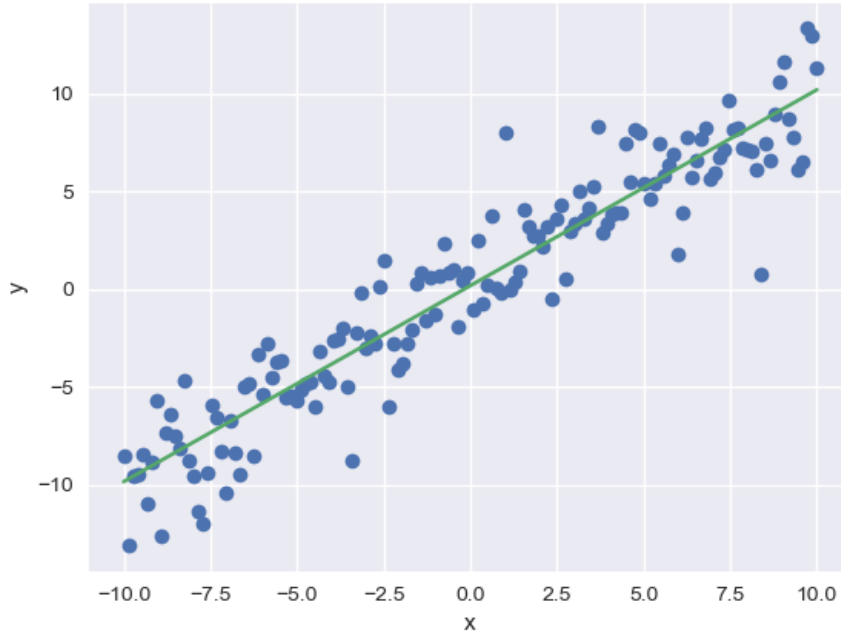
Pueden usar el método que más les guste para resolver este sistema. Por determinantes por ejemplo

$$m = \frac{n \sum_{i=1}^n x_i y_i - \left(\sum_{i=1}^n x_i\right)\left(\sum_{i=1}^n y_i\right)}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i\right)^2} = \frac{Cov(x, y)}{Var(x)} \quad (7)$$

Qué curioso resultado. Ahora veamos que pasa con b

$$b = \frac{\left(\sum_{i=1}^n x_i^2\right)\left(\sum_{i=1}^n y_i\right) - \left(\sum_{i=1}^n x_i\right)\left(\sum_{i=1}^n x_i y_i\right)}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i\right)^2} = \frac{\overline{(x \odot x)}\bar{y} - \bar{x}\overline{(x \odot y)}}{Var(x)} \quad (8)$$

Con eso, podemos construir la línea de predicciones del modelo, la cual se ve así



2. Caso multivariado

Antes de seguir, hay que definir las derivadas de escalares respecto a vectores

Definición. La derivada de una función vectorial $\mathbf{y}(x)$ respecto a un escalar x será un vector renglón

$$\frac{\partial \mathbf{y}}{\partial x} = \left[\frac{\partial y_1}{\partial x}, \frac{\partial y_2}{\partial x}, \dots, \frac{\partial y_m}{\partial x} \right]$$

Definición. La derivada de una función escalar $y(\mathbf{x})$ respecto a un vector \mathbf{x} será un vector columna

$$\frac{\partial y}{\partial \mathbf{x}} = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \dots, \frac{\partial y}{\partial x_m} \right]^T$$

Definición. De las anteriores dos definiciones, se sigue que la derivada de una función vectorial $\mathbf{y}(\mathbf{x})$ respecto a un vector \mathbf{x} es una matriz cuyas entradas están dadas por

$$\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)_{ij} = \frac{\partial y_j}{\partial x_i}$$

Una implicación directa de esta convención (conocida como *denominator layout*, o formulación Hessiana) es que al derivar un producto de matrices, aparece una operación de transposición:

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{A}\mathbf{x}) = \frac{\partial \mathbf{x}}{\partial \mathbf{x}} \mathbf{A}^T = \mathbf{A}^T$$

Bien, lo que sigue es escribir la generalización del problema para \mathbf{x} un vector en \mathbb{R}^m en lugar de un escalar. Podemos escribir un plano en m dimensiones como el siguiente producto interior

$$\hat{f}(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x} = \mathbf{c}^T \mathbf{x} \quad (9)$$

donde \mathbf{c} contiene los coeficientes a determinar. Falta decir algo sobre esta función, pero lo arreglaremos después de hacer un poco de trabajo.

Nuestra función de costo nuevamente ya tiene variables de las que depende, y podemos escribirla como

$$J(\mathbf{c}) = \frac{1}{2n} \sum_{i=1}^n (\mathbf{c}^T \mathbf{x}_i - y_i)^2 \quad (10)$$

Con un poco de imaginación, podemos ver que la suma de la función de costo es cuadrado de la norma de un vector, un vector cuyas entradas son

$$\left[\mathbf{c}^T \mathbf{x}_1 - y_1, \mathbf{c}^T \mathbf{x}_2 - y_2, \dots, \mathbf{c}^T \mathbf{x}_n - y_n \right]$$

Pues, si inteligentemente definimos la matrix \mathbf{X} como

$$\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \right]^T$$

entonces, definiendo $\mathbf{y} := [y_1, y_2, \dots, y_n]^T$ tenemos que la función de costo se puede escribir como

$$J(\mathbf{c}) = \frac{1}{2n} \|\mathbf{X}\mathbf{c} - \mathbf{y}\|^2 = \frac{1}{2n} (\mathbf{X}\mathbf{c} - \mathbf{y})^T (\mathbf{X}\mathbf{c} - \mathbf{y}) \quad (11)$$

Vamos ahora a buscar el mínimo de esta función, encontrando el gradiente $\partial J_{\mathbf{c}}$ e igualándolo a cero. Para esto necesitamos la regla de la cadena. Noten porfavor que por la formulación Hessiana, se tiene que para una función $u : \mathbb{R}^n \rightarrow \mathbb{R}$, una función $\mathbf{g} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ y una variable $\mathbf{c} \in \mathbb{R}^m$, la regla de la cadena es:

$$\frac{\partial}{\partial \mathbf{c}} u(\mathbf{g}(\mathbf{c})) = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial u}{\partial \mathbf{g}} \quad (12)$$

Ahora basta con aplicar la regla de la cadena para encontrar el gradiente de J

$$\frac{\partial J}{\partial \mathbf{c}} = \frac{1}{n} \mathbf{X}^T (\mathbf{X}\mathbf{c} - \mathbf{y}) \quad (13)$$

Al igualar esta expresión a cero se puede resolver para \mathbf{c}

$$\mathbf{c} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (14)$$

Al conjunto de ecuaciones descritas por la ecuación matricial (14) se les conoce como **ecuaciones normales de la regresión lineal**.

2.1. Interpretación geométrica de las ecuaciones normales

Desde haber escrito la función de costo en forma matricial, el problema se convirtió en minimizar el tamaño del vector $\mathbf{X}\mathbf{c} - \mathbf{y}$, es decir, encontrar un vector ortogonal al espacio generado por las columnas de \mathbf{X} , cuya longitud está dada por la longitud de \mathbf{y} y su ángulo a dicho espacio. Cada vector columna de la matriz \mathbf{X} es un vector correspondiente a una de las variables que estamos midiendo, y todas sus entradas son las múltiples mediciones que hicimos.

2.2. La matriz de Gram

Como una curiosa observación, pongámonle un poco más de atención a la matriz $\mathbf{X}^T\mathbf{X}$, la cual es conocida como *matriz de Gram*. Esta matriz puede dar problemas si resulta no ser invertible, por lo que diremos lo siguiente

Teorema. *La matriz de Gram es invertible si y sólo si \mathbf{X} es de rango completo.*

Demostración. (\Leftarrow) Suponga que \mathbf{X} no es de rango completo, entonces existe un vector $\mathbf{v} \neq \mathbf{0}$ tal que $\mathbf{X}\mathbf{v} = \mathbf{0}$, con lo que se tiene que $\mathbf{X}^T\mathbf{X}\mathbf{v} = \mathbf{0}$, lo que implica que $\mathbf{X}^T\mathbf{X}$ tampoco es de rango completo y por lo tanto no es invertible.

(\rightarrow) Suponga que $\mathbf{X}^T\mathbf{X}$ no es invertible, siendo ese el caso existe un vector $\mathbf{v} \neq \mathbf{0}$ tal que $\mathbf{X}^T\mathbf{X}\mathbf{v} = \mathbf{0}$. Se sigue entonces que

$$\mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} = \|\mathbf{X}\mathbf{v}\|^2 = 0$$

Esto implica que hay dependencia lineal en las columnas de \mathbf{X} y por tanto es de rango deficiente □

El resultado de esto es que las columnas de la matriz \mathbf{X} deben ser linealmente independientes. Ojo que las columnas de la matriz no son los vectores de cada medición; hay un vector por cada variable en la que se hacen mediciones. Independencia lineal de estos vectores quiere decir que necesitamos que ni una de las variables que medimos sea la combinación lineal de otras. Aquí viene la importancia de la correlación! Y aunque la correlación entre columnas no sea 1, columnas con alta correlación volverán computacionalmente inestable a la inversión de la matriz de Gram. Cuidado!

Por último, esta matriz resulta ser positiva semidefinida. La prueba la pueden hacer ustedes.

2.3. Una manera de arreglar la inestabilidad

Cuando la inversión de la matriz de Gram es inestable, a veces se recurre a añadir términos de regularización a la función de costo. Es para que se queden *picados*, viene en las siguientes clases. Ah, también nunca verificamos que el problema es *convexo* y de minimización. Eso queda pendiente!

2.4. Por cierto...

El vector \mathbf{x} de parámetros no puede por sí mismo representar todos los hiperplanos que nos interesan para generalizar la regresión lineal que teníamos en el caso de una dimensión, donde había dos grados de libertad. Para conseguir este grado de libertad extra, lo que se hace es aumentar una entrada al vector de mediciones, quedando así como

$$\mathbf{x} := [1, \mathbf{x}^T]^T \tag{15}$$

Con lo que recuperamos la ecuación de un plano en m dimensiones. Esta decisión se puede hacer en general para añadir un grado de libertad (en caso de que haya una dependencia lineal con el tamaño del vector de entrada) por lo que a partir de ahora no se escribirá explícitamente.

Problemas Convexos

Carlos Malanche

20 de marzo de 2018

1. Máxima verosimilitud (*Maximum Likelihood*)

Podemos ver el problema de una manera alterna y un tanto complicada, es una manera de ver las cosas basados en probabilidad.

Como ya lo habíamos dicho antes, podemos suponer que nuestra información fue generada por una parte determinística más una parte de ruido

$$y = f(\mathbf{x}) + \epsilon \quad (1)$$

Con una restricción más, supongamos que la función determinística es una función lineal, tal y como lo hicimos con mínimos cuadrados

$$y = \mathbf{x}^T \mathbf{c} + \epsilon \quad (2)$$

Dado un \mathbf{c} fijo, podemos preguntarnos cuál es la probabilidad de haber obtenido las mediciones y .

$$P(y|\mathbf{x}, \mathbf{c}) = \mathcal{N}(y|\mathbf{x}^T \mathbf{c}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mathbf{x}^T \mathbf{c})^2}{2\sigma^2}} \quad (3)$$

En el caso de \mathbf{y} un vector, la expresión es un tanto más complicada

$$P(\mathbf{y}|\mathbf{X}, \mathbf{c}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{c}, \Sigma) = \frac{1}{\sqrt{(2\pi)^m \det(\Sigma)}} \exp\left[-\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{c})^T \Sigma^{-1}(\mathbf{y} - \mathbf{X}\mathbf{c})\right] \quad (4)$$

La idea de *maximum likelihood* es encontrar el vector de parámetros \mathbf{c} que maximiza la probabilidad de haber obtenido el vector de *outcomes* \mathbf{y} .

Para resolver esto: ¿La probabilidad de obtener \mathbf{y} dado \mathbf{X} como función de \mathbf{c} qué forma tiene? ¿Es un problema convexo? ¡No! y tampoco es cóncavo. Primero vamos a arreglar lo primero. Hay una forma cuadrática en el argumento de la exponencial, la cual ya sabemos será convexa (excepto por el signo menos...). Vamos a intentar sacarla aplicando un logaritmo a la probabilidad.

$$\log(P(\mathbf{y}|\mathbf{X}, \mathbf{c})) = -\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{c})^T \Sigma^{-1}(\mathbf{y} - \mathbf{X}\mathbf{c}) + \text{cnst} \quad (5)$$

En donde hemos decidido abreviar esa constante

$$\text{cnst} = -\frac{1}{2}m\log((2\pi)^m \det(\Sigma))$$

Basta con multiplicar toda la expresión por un menos y deshacernos de la constante (no va a modificar el problema) para movernos a un marco muy similar al que estábamos utilizando anteriormente (un problema de minimizar en lugar de maximizar).

$$\arg \min_{\mathbf{c}} \left\{ \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{c})^T \Sigma^{-1}(\mathbf{y} - \mathbf{X}\mathbf{c}) \right\} \quad (6)$$

El problema es casi idéntico al de mínimos cuadrados! En realidad, es una generalización, pues basta con hacer $\Sigma = n\mathbb{I}$ para recuperar los mínimos cuadrados. Hacer esto le quitaría el sentido estadístico a nuestra interpretación así que no lo haremos.

En lugar de ello, todo se simplifica cuando la matriz de covarianza es diagonal, pues implica que cada una de las medidas fue independiente (lo cual es algo relativamente seguro de hacer en la mayoría de los casos). Incluso

podemos asumir que la desviación estándar σ es la misma para cada uno de los parámetros, con lo que obtenemos casi la función de pérdidas cuadrática

$$\arg \min_{\underline{c}} \left\{ \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\mathbf{c}\|^2 \right\} \quad (7)$$

2. Regresión regularizada

Recordemos que el objetivo es encontrar un vector de parámetros (los cuales definen una función) que minimice el error de predicción sobre una serie de datos conocida. Si seguimos denotando con $J_{\hat{f}}(\underline{c})$ la función de costo adaptada a nuestra función estimadora \hat{f} , definiremos pues la función objetivo como

$$\mathcal{L}(\underline{c}) := J_{\hat{f}}(\underline{c}) \quad (8)$$

se ve un poco redundante pero tiene un propósito inventar esta nueva función. Lo que buscamos es minimizar la función objetivo

$$\arg \min_{\underline{c}} \mathcal{L}(\underline{c}) \quad (9)$$

Hemos visto que cuando J es la función de pérdidas cuadrática (FPC) la solución es analítica y está dada por las ecuaciones normales. Sin embargo, el problema puede volverse inestable dependiendo del rango de la matriz de observaciones \mathbf{X} (columnas casi linealmente independientes convierten en casi-singular a la matriz, y su inversión es inestable). Llegan a salvarnos los **regularizadores**, que son funciones que se añaden a la función objetivo para poner imponer condiciones.

$$\mathcal{L}(\underline{c}) := J_{\hat{f}}(\underline{c}) + \Omega(\underline{c}) \quad (10)$$

Esto es todo un campo de investigación, pues no basta con pasarle condiciones a la función objetivo a diestra y siniestra, no tiene caso hacerlo si no sabemos como resolver el problema. Un ejemplo de un *regularizador* es esta modificación a la función indicadora

$$\Omega(\underline{c}) = 1_S := \begin{cases} 0, & \text{si } \underline{c} \notin S \\ \infty, & \text{o.c.} \end{cases} \quad (11)$$

Basta con que la función de costo no sea infinita para una $\underline{c} \in S$ para que el vector solución \underline{c} no pueda estar fuera de S .

Por el momento nos vamos a enfocar en regularizadores con un comportamiento un poco menos agresivo (en general, derivables) que de todos modos nos pueden ayudar, en particular a que la matriz de las ecuaciones normales no quede mal condicionada.

2.1. Regularización de Tikhonov

Definamos los regularizadores de Tikhonov como aquellas función objetivo en donde el regularizador Ω tiene la forma

$$\Omega(\underline{c}) := \|\mathbf{\Gamma}\mathbf{c}\|^2 \quad (12)$$

Donde $\mathbf{\Gamma}$ es una matriz que podemos elegir para sacar características lineales del vector de parámetros. Busquen cuál es el resultado de encontrar el gradiente del problema con regularización de Tikhonov, es muy sencillo.

2.2. Regularización de arista

Caso particular, tomaremos

$$\Omega(\underline{c}) := \frac{\gamma}{2n} \|\underline{c}\|^2 \quad (13)$$

es decir, $\mathbf{\Gamma} = (\gamma/2n)^{1/2}\mathbb{I}$. Vamos a repetir el ejercicio de la clase pasada, y vamos a tratar de encontrar un punto óptimo para este caso

$$\nabla \mathcal{L}(\underline{c}) = \nabla \left(\frac{1}{2n} \|\mathbf{y} - \mathbf{X}\mathbf{c}\|^2 \right) + \frac{\gamma}{2n} \nabla \|\underline{c}\|^2 \quad (14)$$

$$= \frac{1}{n} \mathbf{X}^T (\mathbf{X}\mathbf{c} - \mathbf{y}) + \frac{\gamma}{n} \mathbf{c} \quad (15)$$

Ya sólo igualamos a cero y resolvemos para \mathbf{c}

$$\mathbf{c} = (\mathbf{X}^T \mathbf{X} + \gamma \mathbb{I})^{-1} \mathbf{X}^T \mathbf{y} \quad (16)$$

Para los atentos, el efecto de sumarle ese múltiplo de la identidad a una matriz semidefinida positiva es el de levantar los eigenvalores, con lo que podemos obligar a la matriz completa a ser invertible

Teorema. Sea \mathbf{G} una matriz de Gram generada por la matriz \mathbf{X} . Entonces los valores propios de $(\mathbf{G} + \gamma \mathbb{I})$ se elevan por γ

Demostración. Consideremos la diagonalización de la matriz de Gram, dada por las matriz de vectores propios \mathbf{V} y la matriz diagonal de valores propios \mathbf{P} .

$$\mathbf{G} = \mathbf{V}^T \mathbf{P} \mathbf{V}$$

Con ella, escribimos de nuevo nuestra matriz por invertir

$$\mathbf{X}^T \mathbf{X} + \gamma \mathbb{I} = \mathbf{V}^T \mathbf{P} \mathbf{V} + \gamma \mathbb{I} = \mathbf{V}^T \mathbf{P} \mathbf{V} + \gamma \mathbf{V}^T \mathbf{V} = \mathbf{V}^T (\mathbf{P} + \gamma \mathbb{I}) \mathbf{V} \quad (17)$$

Nota: Esta demostración se auxilió de saber que las matrices de vectores propios de una matriz de Gram son ortonormales, lo cual se puede ver con la descomposición de valores singulares de su matriz generadora. \square

Casi por diversión, de una manera alternativa podemos extraer los valores propios usando el coeficiente de *Rayleigh*

$$\frac{\mathbf{v}^T (\mathbf{X}^T \mathbf{X} + \gamma \mathbb{I}) \mathbf{v}}{\mathbf{v}^T \mathbf{v}} = \frac{\mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v}}{\mathbf{v}^T \mathbf{v}} + \gamma \frac{\mathbf{v}^T \mathbf{v}}{\mathbf{v}^T \mathbf{v}} = \frac{\mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v}}{\mathbf{v}^T \mathbf{v}} + \gamma$$

Añadimos el conocimiento de que esa matriz de Gram es semidefinida positiva, lo cual quiere decir que los valores propios son iguales o mayores a cero, con lo que podemos concluir que los valores propios de la matriz por invertir **son al menos** γ .

2.3. Regularización *Lasso*

Otro caso divertido es la regularización de *Lasso* (por sus siglas *least absolute shrinkage and selection operator*), en donde la función de regularización es la norma en L_1 , es decir

$$\Omega(\underline{c}) = \|\underline{c}\|_{L_1}$$

léase, la suma de los parámetros que definen al modelo. Lo curioso de este regularizador es que favorece soluciones con muchas entradas en cero. Piensen en un caso simple de dos dimensiones, y se podrán imaginar por qué pasa esto.

3. Problemas convexos

Nosotros agarramos y minimizamos el problema asumiendo que era convexo pero no lo probamos. Pues bien, definamos lo que es un problema convexo

Definición. Una función $f : U \rightarrow \mathbb{R}$ definida en un conjunto convexo y abierto es convexa si para cualesquiera dos puntos $\mathbf{s}_1, \mathbf{s}_2 \in U \subset \mathbb{R}^m$ y $\lambda \in [0, 1]$ se cumple que

$$f(\lambda \mathbf{s}_1 + (1 - \lambda) \mathbf{s}_2) \leq \lambda f(\mathbf{s}_1) + (1 - \lambda) f(\mathbf{s}_2)$$

La definición es muy intuitiva, pues nos dice que el resultado de evaluar f en cualquiera de los puntos sobre la recta que une un par de puntos $\underline{s}_2, \underline{s}_2 \in S$ (donde S es el dominio de la función) es inferior al resultado de evaluar el mismo punto en la línea que une $f(\underline{s}_1)$ con $f(\underline{s}_2)$.

Para el caso de mínimos cuadrados, se tendría que verificar que la siguiente desigualdad es cierta

$$\frac{1}{2n} \|\mathbf{X}(\lambda \mathbf{s}_1 + (1 - \lambda) \mathbf{s}_2) - \mathbf{y}\|^2 \leq \lambda \frac{1}{2n} \|\mathbf{X} \mathbf{s}_1 - \mathbf{y}\|^2 + (1 - \lambda) \frac{1}{2n} \|\mathbf{X} \mathbf{s}_2 - \mathbf{y}\|^2 \quad (18)$$

Ese ejercicio queda al lector. Vamos a intentar llegar a una definición alterna.

$$\begin{aligned} f(\lambda \mathbf{s}_1 + (1 - \lambda) \mathbf{s}_2) &\leq \lambda f(\mathbf{s}_1) + (1 - \lambda) f(\mathbf{s}_2) \\ f(\lambda(\mathbf{s}_1 - \mathbf{s}_2) + \mathbf{s}_2) &\leq \lambda(f(\mathbf{s}_1) - f(\mathbf{s}_2)) + f(\mathbf{s}_2) \\ \frac{f(\lambda(\mathbf{s}_1 - \mathbf{s}_2) + \mathbf{s}_2) - f(\mathbf{s}_2)}{\lambda} &\leq f(\mathbf{s}_1) - f(\mathbf{s}_2) \end{aligned}$$

Sin pérdida de generalidad, vamos a tomar el límite cuando λ tienda a cero (ya suficiente libertad tenemos con permitir a \underline{s}_1 y a \underline{s}_2 ser cualesquiera dos puntos en el dominio)

$$\begin{aligned} \lim_{\lambda \rightarrow 0} \frac{f(\lambda(\mathbf{s}_1 - \mathbf{s}_2) + \mathbf{s}_2) - f(\mathbf{s}_2)}{\lambda} &\leq f(\mathbf{s}_1) - f(\mathbf{s}_2) \\ (\mathbf{s}_1 - \mathbf{s}_2)^T \nabla f(\mathbf{s}_2) &\leq f(\mathbf{s}_1) - f(\mathbf{s}_2) \end{aligned}$$

Donde hemos hecho uso de una de las definiciones de derivada direccional (una que no se encuentra normalizada). Haciendo un último despeje, tenemos esta definición alterna para funciones diferenciables:

Definición. Una función $f : \mathbb{R}^m \rightarrow \mathbb{R}$ diferenciable es convexa si para cualesquiera dos puntos $\mathbf{s}_1, \mathbf{s}_2 \in S \subset \mathbb{R}^m$ se cumple que

$$f(\mathbf{s}_1) - f(\mathbf{s}_2) \geq (\mathbf{s}_1 - \mathbf{s}_2)^T \nabla f(\mathbf{s}_2)$$

Problemas Convexos

Carlos Malanche

13 de marzo de 2018

1. Optimización

1.1. Búsqueda en retícula (*grid search*)

La búsqueda en retícula es el método más simple (y más ineficiente) para encontrar un punto óptimo. Consiste en discretizar el dominio de los parámetros (muchas veces en puntos equidistantes) y evaluar con fuerza bruta el valor de la función objetivo para poder elegir el *mínimo*. Puesto de manera formal, si la partición de la i -dimensión en el espacio de parámetros de \underline{c} la denotamos por P_{c_i} el resultado será

$$\underline{c}_0 = \min\{\underline{c} | c_i \in P_{c_i} \forall i = 1, \dots, m\} \quad (1)$$

En el modestísimo caso en el que hay $m = 32$ parámetros (imaginen que podemos pasar una imagen, con millones de píxeles, siendo cada uno una *medición*), tendríamos que realizar 2^{32} evaluaciones de la función objetivo para buscar en los extremos de las particiones nadamás, y esto sólo dará buenos resultados bajo la suposición de que la función objetivo es lineal en todos los parámetros (nunca pasará).

1.1.1. La versión estocástica

La versión estocástica de este problema consiste en tomar puntos al azar, evaluarlos y quedarnos con el mejor. El panorama no es alentador. Todo es por algo conocido como *la maldición de la dimensionalidad*. Tiene un nombre demasiado alarmista para lo que es. Sólo consiste en observar el crecimiento exponencial del espacio de muestreo con el crecimiento lineal de los parámetros de los que está compuesto.

Sin embargo, es curioso ver que una búsqueda aleatoria tiene una *alta* probabilidad de quedar cerca de un óptimo: Imaginemos el espacio de búsqueda, creado a partir de los extremos de las particiones de cada variable en la búsqueda en malla. Asumiremos que vamos a tomar puntos de manera aleatoria bajo una distribución uniforme en cada dimensión.

Si esto es así, la probabilidad de que un punto quede en el 5 % del espacio que rodea al óptimo sería 0.05. Ahora, nos preguntamos: Después de k intentos, cuál es la probabilidad de que **ninguno** de los k puntos se encuentre en el 5 % al rededor del óptimo? Debería resultar claro que la respuesta es

$$(1 - 0,05)^k$$

Si pedimos que su complemento en probabilidad (es decir, que al menos uno haya caído en el 5 %) sea al menos 95 % (considerablemente bueno, digamos), podemos resolver para k , lo que nos da el bellísimo resultado de

$$k \geq \frac{\log(0,05)}{\log(0,95)} \approx 60 \quad (2)$$

Es decir, independientemente del número de dimensiones, en sólo 60 evaluaciones hay un 95 % de probabilidades de estar cerca del máximo/mínimo. Léase: Es mejor utilizar una búsqueda aleatoria para encontrar al menos un buen punto de partida.

1.2. Descenso de gradiente (*gradient descent*)

De lo que se trata el algoritmo es de iterativamente caminar en dirección contraria al gradiente, en algún paso moderado. Si denotamos con $\nabla \mathcal{L}(\underline{c})$ el gradiente de la función objetivo evaluado en \underline{c} , y suponemos \underline{c}_0 una *buen adivinanza* de punto de comienzo, la iteración número $i + 1$ estará dada por

$$\mathcal{C}_{i+1} = \mathcal{C}_i - \eta \nabla \mathcal{L}(\mathcal{C}_i) \quad (3)$$

Al parámetro η se le conoce como el *paso* del algoritmo. Hay varios estudios al respecto de este parámetro, pues de él dependerá el ritmo de convergencia y la estabilidad del descenso de gradiente.

1.3. Convergencia

El análisis de convergencia del algoritmo es algo complicado, es por ello que veremos el caso en el que se utiliza un tamaño de paso constante.

Definición. Sea $f : U \rightarrow \mathbb{R}$ una función. Se dice que la función es Lipschitz continua si existe un valor L tal que

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|$$

Esto acota en cierta medida el ritmo de cambio de una función. Tomemos $f := \nabla \mathcal{L}$ y supongamos que el gradiente es Lipschitz-continuo de constante L , además de suponer que nuestra función es *convexa*. Vamos a probar que el ritmo de convergencia va como el inverso del tamaño del paso.

Al suponer que el gradiente tiene constante de Lipschitz L , ocurre que el Hessiano queda acotado por L , que se puede escribir de otro modo como

$$\nabla^2 \mathcal{L} - LI \geq 0 \quad (4)$$

(una manera de probarlo es utilizar el teorema del valor medio y la definición de Lipschitz continuo). Usando la definición de semidefinido positivo, tomamos convenientemente el vector $(x - y)$ para escribir

$$(x - y)^T (\nabla^2 \mathcal{L} - LI)(x - y) \leq 0 \quad (5)$$

$$(x - y)^T \nabla^2 \mathcal{L}(x - y) \leq L\|x - y\|^2 \quad (6)$$

Usando el teorema del residuo de Taylor, podemos encontrar $z \in [x, y]$ (donde estoy representando con ese intervalo la línea que une x con y) tal que

$$\begin{aligned} \mathcal{L}(y) &= \mathcal{L}(x) + \nabla \mathcal{L}(x)^T (y - x) + \frac{1}{2} (x - y)^T \nabla^2 \mathcal{L}(z) (x - y) \\ &\leq \mathcal{L}(x) + \nabla \mathcal{L}(x)^T (y - x) + \frac{L}{2} \|y - x\|^2 \end{aligned}$$

hagamos que $y := x^+ = x - \eta \nabla \mathcal{L}(x)$ (es decir, y es el paso inmediato del descenso de gradiente al haber evaluado x).

$$\begin{aligned} \mathcal{L}(x^+) &\leq \mathcal{L}(x) + \nabla \mathcal{L}(x)^T (x - \eta \nabla \mathcal{L}(x) - x) + \frac{L}{2} \|x - \eta \nabla \mathcal{L}(x) - x\|^2 \\ &= \mathcal{L}(x) - \eta \nabla \mathcal{L}(x)^T \nabla \mathcal{L}(x) + \frac{\eta^2 L}{2} \|\nabla \mathcal{L}(x)\|^2 \\ &= \mathcal{L}(x) - (1 - \frac{\eta L}{2}) \eta \|\nabla \mathcal{L}(x)\|^2 \end{aligned}$$

Tomemos $0 < \eta < 1/L$, con lo que podemos ver que $-(1 - \frac{\eta L}{2}) = \frac{\eta L}{2} - 1$ que en el mejor caso (es decir $\eta = 1/L$) el valor queda acotado por $1/2$.

$$\mathcal{L}(x^+) \leq \mathcal{L}(x) - \frac{\eta}{2} \|\nabla \mathcal{L}(x)\|^2 \quad (7)$$

Primer observación: bajo las condiciones establecidas sobre \mathcal{L} y sobre η , el descenso de gradiente siempre conduce a un valor mejor (más pequeño). Recordemos, además, que $\mathcal{L}(x)$ es convexa. Esto implica que

$$\mathcal{L}(x) \leq \mathcal{L}(x^*) + \nabla \mathcal{L}(x)^T (x - x^*) \quad (8)$$

(la expresión es la misma que la de la clase pasada pero multiplicada por -1 , y hemos bautizado a x^* como el valor óptimo). Bien, usemos esta definición dentro del resultado anterior

$$\begin{aligned}
\mathcal{L}(x^+) &\leq \mathcal{L}(x) - \frac{\eta}{2} \|\nabla \mathcal{L}(x)\|^2 \\
&\leq \mathcal{L}(x^*) + \nabla \mathcal{L}(x)^T (x - x^*) - \frac{\eta}{2} \|\nabla \mathcal{L}(x)\|^2 \\
&= \mathcal{L}(x^*) + \frac{1}{2\eta} (\|x - x^*\|^2 - \|x - x^* - \eta \nabla \mathcal{L}(x)\|^2) \\
&= \mathcal{L}(x^*) + \frac{1}{2\eta} (\|x - x^*\|^2 - \|x^+ - x^*\|^2)
\end{aligned}$$

Como última observación, ya que el descenso de gradiente es no creciente, podemos concluir que en el paso número k , la distancia al óptimo será mejor que el promedio de todos los pasos anteriores

$$\mathcal{L}(x^{(k)}) - \mathcal{L}(x^*) \leq \frac{1}{k} \sum_{i=1}^k (\mathcal{L}(x^{(i)}) - \mathcal{L}(x^*)) \quad (9)$$

Esa suma se convierte en una suma telescópica, pues

$$\begin{aligned}
\sum_{i=1}^k (\mathcal{L}(x^{(i)}) - \mathcal{L}(x^*)) &\leq \sum_{i=1}^k \frac{1}{2\eta} (\|x^{(i-1)} - x^*\|^2 - \|x^{(i)} - x^*\|^2) \\
&= \frac{1}{2\eta} (\|x^{(0)} - x^*\|^2 - \|x^{(k)} - x^*\|^2) \\
&\leq \frac{1}{2\eta} \|x^{(0)} - x^*\|^2
\end{aligned}$$

Entonces, el ritmo de convergencia depende inversamente del tamaño del paso, y de la distancia al óptimo en un principio (además del número de iteraciones)

$$\mathcal{L}(x^{(k)}) - \mathcal{L}(x^*) \leq \frac{\|x^{(0)} - x^*\|^2}{2\eta k} \quad (10)$$

Noten porfavor que este ritmo de convergencia salió de poner condiciones bastante estrictas a la función objetivo.

1.3.1. Subgradiente

Increíblemente, podemos seguir aplicando nuestro descenso de gradiente aún cuando la función no es diferenciable. Pero para ello es necesario definir un *subgradiente*.

Definición. Sea $f : U \rightarrow \mathbb{R}$ una función definida sobre un conjunto convexo y abierto, subconjunto de \mathbb{R}^m . Un vector \underline{v} es un subgradiente de f en el punto \underline{x}_0 si para toda $\underline{x} \in U$

$$f(\underline{x}) - f(\underline{x}_0) \geq \underline{v} \cdot (\underline{x} - \underline{x}_0)$$

Hay un gran parecido entre la definición de función convexa diferenciable y la de un subgradiente. Como última nota, si la función es diferenciable, el gradiente común y corriente es el único subgradiente de f .

Ya podemos escribir nuestro algoritmo, que queda de la siguiente manera (denotando con $\hat{\nabla} \mathcal{L}$ el subgradiente de la función objetivo)

$$\underline{c}_{i+1} = \underline{c}_i - \mu \hat{\nabla} \mathcal{L}(\underline{c}_i)$$

1.4. Descenso de gradiente estocástico (*stochastic gradient descent*)

Muchas veces los cantidad de puntos de *entrenamiento* será muy grande y afectará mayormente el desempeño de nuestros algoritmos. En el caso del descenso del gradiente, calcular el gradiente puede tomar mucho tiempo. Una opción es tirar parte de la información que tenemos para hacer el cómputo del gradiente más rápido, pero si no sabemos *cuál es la información más útil* esto es una mala idea. Llega el descenso de gradiente estocástico al rescate.

Simplemente consiste en tomar uno de los datos de manera aleatoria y avanzar en la dirección del gradiente que este dato proporciona. De esta manera, para un punto fijo

$$E[\nabla_s \mathcal{L}] = \nabla \mathcal{L} \quad (11)$$

Regresión Logística

Carlos Malanche

10 de abril de 2018

1. Clasificación

No todos los problemas son sobre predecir valores sobre la recta real. En algunos casos, vamos a tener medidas en que los elementos de la serie de datos $D = \{\underline{x}_i, y_i\}_{i=1}^n$ cumplen $\underline{x}_i \in \mathbb{R}^n$, $y_i \in \{\text{Clase 1, Clase 2}\}$ (por ejemplo al tratar de caracterizar especies, los biólogos deben utilizar categorías). Vamos a comenzar por intentar resolver el problema para dos categorías.

1.1. Clasificación binaria

El primer paso (intuitivamente) es convertir las clases a valores que un modelo lineal pueda predecir, es decir, números. Digamos pues que $y_i \in \{0, 1\}$. Qué ocurre si entrenamos un modelo lineal para predecir esta variable? Recordemos que la predicción de un modelo lineal con vector de parámetros \underline{c} , para una nueva observación \underline{x}_0 se obtiene así

$$y_0 = \langle \underline{x}_0, \underline{c} \rangle \quad (1)$$

No es muy difícil de ver que bastará con que \underline{c} tenga al menos dos entradas distintas de cero para obtener y_0 no sólo diferente a 0 o a 1, si no posiblemente fuera del rango de estas dos. Si el valor obtenido estuviera limitado al intervalo $[0, 1]$, podríamos pensar en interpretar el valor de y como una probabilidad... vaya, si tan sólo hubiera una manera de mapear el intervalo $(-\infty, \infty)$ (rango del operador $\langle \cdot, \cdot \rangle$) al $(0, 1)$...

1.2. Regresión Logística

Definamos como función sigmoide (a veces llamada función logística) la siguiente función

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

Si han leído la sección anterior, sabrán qué busca cumplir esta función: Mapear el rango del producto interno al intervalo abierto $(0, 1)$.

Ahora podemos interpretar el resultado de nuestro predictor lineal, al pasar por la función sigmoide, como una probabilidad de que la medición \underline{x}_0 pertenece a una de las dos clases (es decir, $P(y_0 \in \text{Clase 2}) = \sigma(\langle \underline{x}_0, \underline{c} \rangle)$ y por ser una probabilidad binomial, $P(y_0 \in \text{Clase 1}) = 1 - \sigma(\langle \underline{x}_0, \underline{c} \rangle)$).

Todo muy bien hasta ahora, pero hay que preguntarnos: Si vamos a hacer esto... no hay que entrenar el modelo de manera diferente? La respuesta es sí, pues hemos modificado la función de costo:

$$J(\underline{c}) = \frac{1}{2n} \sum_{i=1}^n (\sigma(\underline{x}_i \cdot \underline{c}) - y_i)^2 \quad (3)$$

Noten que otra manera de derivar esta función de costo es utilizar máxima verosimilitud.

$$\begin{aligned} P(\mathbf{y}|\mathbf{X}, \mathbf{c}) &= \prod_{n=1}^m p(y_n, \mathbf{x}_n) \\ &= \prod_{n:y_n=1} p(y_n|\mathbf{x}_n) \prod_{n:y_n=0} p(y_n|\mathbf{x}_n) \\ &= \prod_{n=1}^N \sigma(\mathbf{x}_n^T \mathbf{c})^{y_n} (1 - \sigma(\mathbf{x}_n^T \mathbf{c}))^{1-y_n} \end{aligned}$$

Usando un truco similar al de la vez pasada, convertimos esto una función compuesta de sumandos al aplicar un logaritmo y agregando un (-)

$$\begin{aligned}
\mathcal{L}(\mathbf{x}) &= -\log\left(\prod_{n=1}^N \sigma(\mathbf{x}_n^T \mathbf{c})^{y_n} (1 - \sigma(\mathbf{x}_n^T \mathbf{c}))^{1-y_n}\right) = -\sum_{n=1}^N (y_n \log(\sigma(\mathbf{x}_n^T \mathbf{c})) + (1 - y_n) \log(1 - \sigma(\mathbf{x}_n^T \mathbf{c}))) \\
&= -\sum_{n=1}^N \log(1 - \sigma(\mathbf{x}_n^T \mathbf{c})) + y_n \log\left(\frac{1 - \sigma(\mathbf{x}_n^T \mathbf{c})}{\sigma(\mathbf{x}_n^T \mathbf{c})}\right) \\
&= \sum_{n=1}^N \log(1 + e^{\mathbf{x}_n^T \mathbf{c}}) - y_n \mathbf{x}_n^T \mathbf{c}
\end{aligned}$$

Para poder encontrar un mínimo, es necesario derivar y encontrar el gradiente. Notemos que

$$\frac{\partial}{\partial x} \log(1 + e^x) = \frac{e^x}{1 + e^x} = \sigma(x) \quad (4)$$

Con ello, la derivada de la función objetivo es

$$\begin{aligned}
\nabla \mathcal{L}(\mathbf{c}) &= \sum_{n=1}^N \mathbf{x}_n \sigma(\mathbf{x}_n^T \mathbf{c}) - y_n \mathbf{x}_n \\
&= \sum_{n=1}^N \mathbf{x}_n (\sigma(\mathbf{x}_n^T \mathbf{c}) - y_n) \\
&= \mathbf{X}^T (\sigma(\mathbf{X} \mathbf{c}) - \mathbf{y})
\end{aligned}$$

En donde al final se utilizó notación matricial para quitar la suma (noten que σ aplicada a un vector es lo mismo que aplicar la función en cada componente).

Igualar a cero este gradiente nos deja con un problema que no tiene solución analítica. Se puede usar un descenso de gradiente pues la función es convexa... lo es?

1.3. Convexidad

Observemos que una línea es convexa (pero no estrictamente convexa). La suma de funciones convexas es convexa. Podemos decir que las últimas N funciones forman una función convexa.

Ahora, para el logaritmo: para la función $\log(1 + e^x)$ tenemos que su primer derivada es la función sigmoide. Su derivada es

$$\frac{\partial}{\partial x} \sigma(x) = \frac{\partial}{\partial x} (1 + e^{-x})^{-1} = (1 + e^{-x})^{-2} e^{-x} = \sigma(x) \frac{e^{-x}}{1 + e^{-x}} = \sigma(x)(1 - \sigma(x)) \quad (5)$$

valor que siempre es positivo. Esto implica que la función es convexa. Por último, el argumento de la exponencial es una función lineal, y la composición de dos funciones convexas es convexa si la externa es no-decreciente. (la primer derivada es mayor o igual a cero). Todos los términos de la función objetivo son convexas!

k-Vecinos más cercanos

Carlos Malanche

12 de abril de 2018

El método de *machine learning* de hoy es uno de los más intuitivos, sólo que lo trataré de presentar de manera formal.

1. Clasificación múltiple

Ahora nos enfrentamos a un problema más duro. Tenemos una serie de pares de datos $S = \{y_i, x_i\}_{i=1}^N$, y sabemos que $y_i \in C = \{\text{Clase1}, \text{Clase2}, \dots, \text{ClaseH}\}$, es decir, hay H clases a las que cada y_i puede pertenecer.

2. k-Vecinos más cercanos (*k-Nearest Neighbours*, *k-NN*)

El único requisito para que este método funcione es que las variables x_i vivan en algún espacio U al que se le pueda asignar una métrica, de tal modo que la cantidad $D(x_i, x_j)$, la distancia entre dos elementos del espacio, esté bien definida para cualesquiera dos elementos $x_i, x_j \in U$. Definimos por $\dot{S}_{\underline{x}}$ la permutación de la serie S en donde se cumple que

$$D((S_{\underline{x}})_i, \underline{x}) \leq D((S_{\underline{x}})_{i+1}, \underline{x}), \text{ para } i = 1, \dots, N-1$$

En donde $(\dot{S}_{\underline{x}})_i$ es el elemento número i de la lista. En el caso de que todas las componentes del vector \underline{x} estén en los reales, $D(x_i, x_j) := \|x_i - x_j\|$. Por último, denotamos por $\dot{S}_{\underline{x}}^k$ al conjunto que contiene los primeros k elementos de $\dot{S}_{\underline{x}}$. Un estimador de k-NN queda escrito entonces como:

$$\hat{f}(\underline{x}) = \frac{1}{k} \sum_{i: x_i \in \dot{S}_{\underline{x}}^k} y_i \quad (1)$$

Es decir, nuestra estimación es el promedio de los k elementos más cercanos a \underline{x} , según la métrica establecida. Cuando se trata de clasificaciones, basta con tomar el elemento de voto mayoritario.

