

Departamento de Física Médica - Centro atómico Bariloche - IB

Instalación e Introducción a miniconda/anaconda

Ariel Hernán Curiale
ariel.curiale@cab.cnea.gov.ar



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



Antes que nada

Erratas, bugs, comentarios,....

todos bienvenidos, y espero que los encuentren

Pero antes de comenzar con Miniconda

OBI WAN NEVER TOLD YOU

**THE POWER OF LIST COMPREHENSIONS
AND ALGEBRAIC DATA TYPES**

memegenerator.net

Listas por comprensión

- ❖ Nos quedo pendiente algo super útil y muy bonito

```
[ i for i in range(200) if i%2 ]
```

```
reverse_word = dict([(val,key) for key, val in word_index.items()])
```

```
layers = [l for l in model.layers[1:9] if 'conv' in l.name]
```

```
layers = [ l.weights[0].shape for l in model.layers[1:]  
           if len(l.weights)>0 ]
```


FOR LOOP SUCKS

USE LIST COMPREHENSION

Abrir el notebook 3 sección 2.9 para comentar
sobre estructurar el código y seguir con
manipulación de archivos

Anaconda - Miniconda

¿Qué es, y porque usarlo?

Download miniconda

- ❖ Miniconda (google download miniconda), si prefieren pueden usar anaconda o el OS con virtualenv o como sea: <https://docs.conda.io/en/latest/miniconda.html>

miniconda

- ❖ Algunos comando útiles (<https://docs.conda.io/projects/conda/en/latest/commands.html>):
 - ❖ `conda list`
 - ❖ `conda search`
 - ❖ `conda install`
 - ❖ `conda clean --all`
 - ❖ `conda update --all` o `conda update nombre_paquete`
 - ❖ `conda info --envs`
 - ❖ `conda create -n myenv python=3.6 numpy`
 - ❖ `conda remove --name myenv --all`
 - ❖ `conda -h` o `conda list -h` o `conda update -h`

miniconda

❖ **export PATH="/home/ariel/miniconda3/bin:\$PATH"**

Instalando paquetes en el base

- ❖ Si ya esta instalado miniconda crear un entorno virtual:
conda create -n deep_learning python=3.7
- ❖ Instalar los siguientes paquetes (si se creo un entorno activarlo antes):
conda install numpy matplotlib scipy scikit-image scikit-learn ipython imageio
 - ❖ GPU: **conda install -c anaconda tensorflow-gpu**
 - ❖ PC: **conda install -c anaconda tensorflow**
- ❖ Versión más completa (todos los backends): **conda install keras**

Instalando paquetes en el base

- ❖ Paquetes para trabajar
 - ❖ Consolas:
`conda install ipython`
`conda install jupyter-console` (instala el notebook)
 - ❖ IDES (entornos gráficos) estilo matlab:
`conda install spyder`
`conda install -c conda-forge jupyterlab`
`conda install -c jupyter` (notebooks, instala la consola)
- ❖ Existe una GUI para instalar cosas:
`conda install -c anaconda anaconda-navigator`

Instalando paquetes en el base

❖ Otro paquetes útiles

`conda install -c anaconda natsort`

`conda install -c bioconda medpy`

`conda install -c conda-forge ipdb`

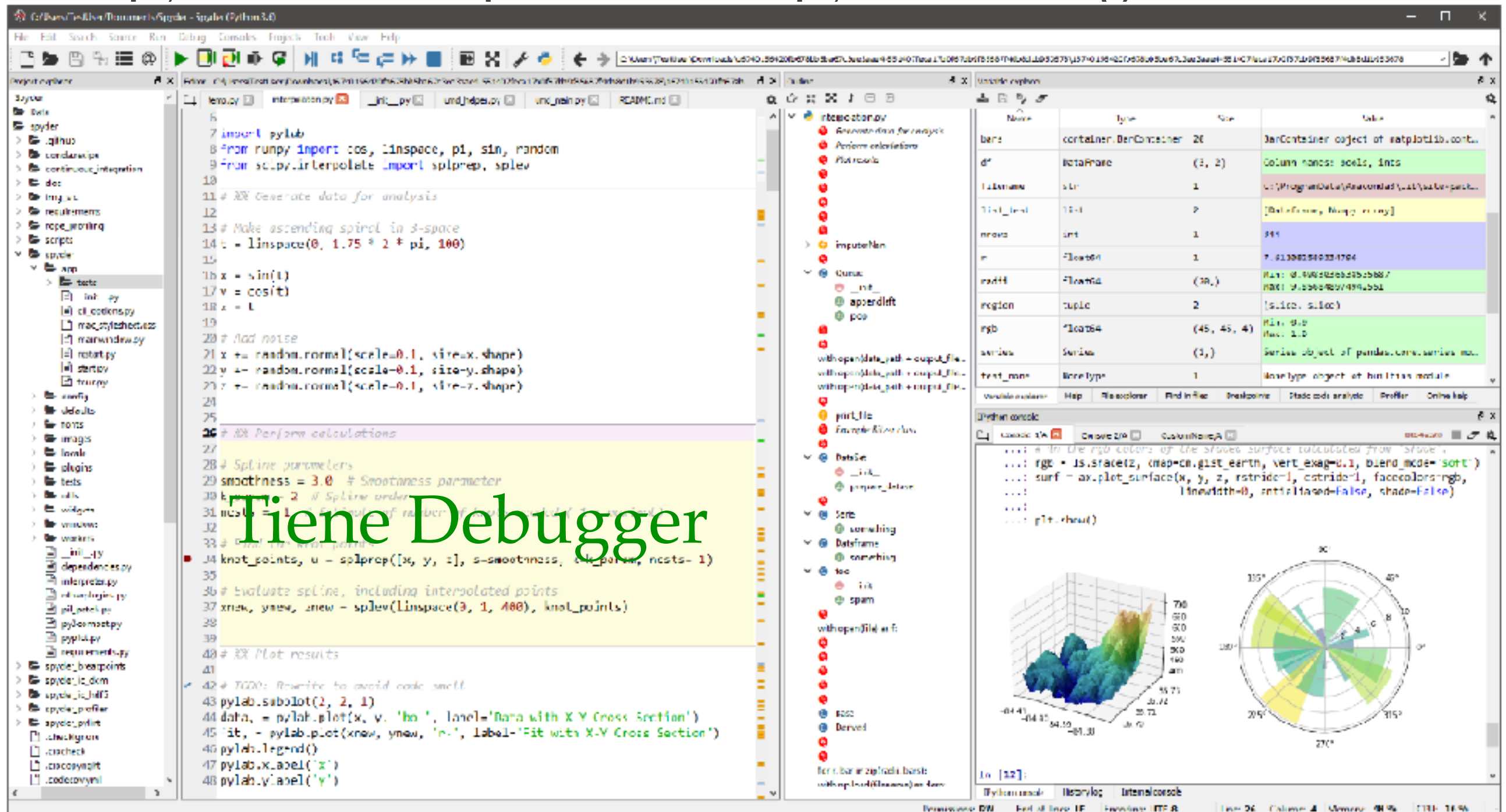
Instalando paquetes en el base

- ❖ Solo cuando no encontremos los paquetes en conda, anaconda, conda-forge podemos instalar los paquetes con pip

`pip install <paquete>`

IDE

❖ Spyder IDE: <https://www.spyder-ide.org>



IDE

❖ JupyterLab IDE: <https://jupyterlab.readthedocs.io/en/stable/>

File Edit View Run Kernel Tabs Settings Help

Files

- notebooks
- Data.ipynb (an hour ago)
- Fasta.ipynb (a day ago)
- Julia.ipynb (a day ago)
- Lorenz.ipynb (seconds ago)**
- R.ipynb (a day ago)
- iris.csv (a day ago)
- lightning.json (9 days ago)
- lorenz.py (3 minutes ago)

Running

Commands

Cell Tools

Tabs

Python 3

In this Notebook we explore the Lorenz system of differential equations:

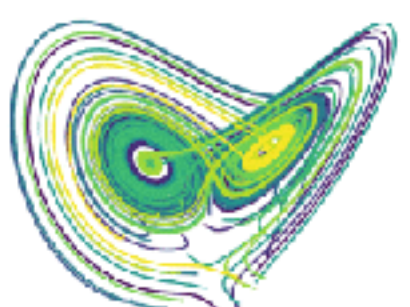
$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

In [4]: `from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=16)`

Output View

sigma 10.00
beta 2.67
rho 28.00



lorenz.py

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):  
10     """Plot a solution to the Lorenz differential equations."""  
11     fig = plt.figure()  
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')  
13     ax.axis('off')  
14  
15     # prepare the axes limits  
16     ax.set_xlim((-25, 25))  
17     ax.set_ylim((-35, 35))  
18     ax.set_zlim((5, 55))  
19  
20     def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):  
21         """Compute the time derivative of a Lorenz system."""  
22         x, y, z = x_y_z  
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]  
24  
25     # Choose random starting points, uniformly distributed from -15 to 15  
26     np.random.seed(1)  
27     x0 = -15 + 30 * np.random.random((N, 3))  
28
```

Ya tiene Debugger

Editor favorito + Consola

❖ Ipython: editor de texto + ipython



```
1 File: p0.py
2 Author: Ariel Hernán Curiale
3 Email: curiale@gmail.com
4 Github: https://github.com/Curiale
5 Description:
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 # ----- [1]
11 A = np.array([[1, 0, 1], [2, -1, 1], [-3, 2, -2]])
12 b = np.array([-2, 1, -1])
13
14 # Ax=b -> x = A^-1 b
15 x = np.dot(np.linalg.inv(A), b)
16
17 # También se puede usar solve
18 x = np.linalg.solve(A, b)
19
20 # ----- [2]
21 data = np.random.gamma(3,2,1000)
22
23 hist = np.histogram(data)
24
25 plt.figure(), plt.bar(hist[1][1:], hist[0], align="center", width=1, alpha=0.5)
26
27 mean = np.mean(data) # mean = K*theta = 3*2 = 6
28 var = np.var(data) # var = K*theta**2 = 3*2**2 = 12
29
30 # ----- [3]
31 def second_order_roots(a, b=0, c=0):
32     discriminante = (b**2 - 4*a*c)+0.5
33     res = np.array([(-b + discriminante)/(2*a), (-b - discriminante)/(2*a)])
34     return res
35
36 # ----- [4]
37 # coeff = np.array([1/4, 3/4, -3/2,-2])
38 # r = np.roots(coeff)
39
40 r = second_order_roots(3/4, b=-3/2, c=-2)
41
42 x = np.linspace(-4.5, 4, 100)
43 # data = 1/4*(x**3 + 3*x**2 - 8*x - 8)
44 data = 1/4*(3*x**2 - 6*x - 8)
45 plt.figure()
46 plt.plot(x, data)
47 # plt.plot(r, [0, 0, 0], 'r')
48 plt.plot(r, [0, 0], 'r')
49
50
51 # ----- [5]
52
53 class Linear:
54
55     def __init__(self, a, b):
56         self.a = a
57         self.b = b
58
59     def calcular(self, x):
60         return self.a*x + self.b
61
```

```
ariel@Goku:~/Docencia/Dalseiro/Materias/CV_DeepLearning/Practica/problemas_resueltos$ ipython
Python 3.6.8 [Anaconda, Inc.] (default, Dec 29 2018, 19:04:46)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.6.1 -- An enhanced Interactive Python. Type '?' for help.
Warning: disable autoreload in ipython_config.py to improve performance.

In [1]:
```

Tiene debugger de consola (ipdb)

Editor favorito + Consola

- ❖ Jupyter console: editor de texto + jupyter console



The screenshot displays a Jupyter Notebook interface. The left pane shows a Python script in a text editor. The script includes comments, imports for numpy and matplotlib, and several code blocks for matrix operations, random data generation, histogram plotting, mean and variance calculation, solving a quadratic equation, and plotting a function. The right pane shows the Jupyter console output, which includes the Python version (3.6.8), the Jupyter console version (6.0.0), and the execution of the first code block, resulting in an empty list.

```
1 """
2 File: p0.py
3 Author: Ariel Hernán Curiale
4 Email: curiale@gmail.com
5 Github: https://gitlab.com/Curiale
6 Description:
7 """
8 import numpy as np
9 import matplotlib.pyplot as plt
10 # ----- [1]
11 A = np.array([[1, 0, 1], [2, -1, 1], [-3, 2, -2]])
12 b = np.array([-2, 1, -1])
13
14 # Ax=b -> x = A^-1 b
15 x = np.dot(np.linalg.inv(A), b)
16
17 # Tambien se puede usar solve
18 x = np.linalg.solve(A, b)
19
20 # ----- [2]
21 data = np.random.randn(3,2,1000)
22
23 hist = np.histogram(data)
24
25 plt.figure(), plt.bar(hist[1][1:], hist[0], align="center", width=1, alpha=0.5)
26
27 mean = np.mean(data) # mean = K*theta = 3*2 = 6
28 var = np.var(data) # var = K*theta**2 = 3*2**2 = 12
29
30 # ----- [3]
31 def second_order_roots(a, b=0, c=0):
32     discriminante = (b**2 - 4*a*c)**0.5
33     res = np.array((-b + discriminante)/(2*a), (-b - discriminante)/(2*a))
34     return res
35
36 # ----- [4]
37 # coeff = np.array([1/4, 3/4, -3/2,-2])
38 # r = np.roots(coeff)
39
40 r = second_order_roots(3/4, b=-3/2, c=-2)
41
42 x = np.linspace(-4.5, 4, 100)
43 # data = 1/4*(x**3 + 3*x**2 -6*x - 8)
44 data = 1/4*(3*x**2 -6*x - 8)
45 plt.figure()
46 plt.plot(x, data)
47 # plt.plot(r, [0, 0, 0], 'r')
48 plt.plot(r, [0, 0], 'r')
49
50
51 # ----- [5]
52
53 class Linear:
54
55     def __init__(self, a, b):
56         self.a = a
57         self.b = b
58
59     def calcular(self, x):
60         return self.a*x + self.b
61
```

ariel@Goku: ~/Docencia/Balseiro/Materias/CV_DeepLearning/Practica/problemas_resueltos (Vim)

ariel@Goku:~/Docencia/Balseiro/Materias/CV_DeepLearning/Practica/problemas_resueltos jupyter console
Jupyter console 6.0.0

Python 3.6.8 [Anaconda, Inc.] (default, Dec 29 2018, 19:04:46)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.6.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: []

Tiene debugger de consola (ipdb)

NORMAL p0.py python utf-8[unix] 0% 1/154 ln : 1 Et (86/53)

Local console kernel remoto

- ❖ Con Jupyter console podemos levantar de forma local una consola que realmente ejecuta en un servidor

1. Averiguamos el directorio donde corre jupyter

```
[server] $ jupyter --runtime-dir  
/run/user/1000/jupyter
```

2. Iniciamos el ipython

```
[server] $ ipython kernel  
[IPKernelApp] To connect another client to this kernel, use:  
[IPKernelApp] --existing kernel-1234.json
```

3. Copiamos el .json

```
[client] $ scp server:/run/user/1000/jupyter/kernel-1234.json ./
```

4. Lanzamos la consola en el cliente de ipython

4.1 console

```
[client] $ jupyter console --existing ./kernel-1234.json --ssh user@server
```

4.2 qtconsole

```
[client] $ ipython qtconsole --existing ./kernel-1234.json --ssh user@server
```


Editor favorito + consola + kernel remoto

```
ariel@Goku: ~ (python3.6)

57 """
58 File: 07_TrainingNetworks_Part1.py
59 Author: Ariel Hernán Curiale
60 Email: curiale@gmail.com
61 Github: https://gitlab.com/Curiale
62 Description:
63 La implementación la saque del curso de Fei-Fei Li. Vamos a ver como quedan
64 en las capas internas la activación de las neuronas si hacemos una
65 inicialización pequeña normal.
66 """
67
68 import numpy as np
69 import matplotlib.pyplot as plt
70
71 def small_normal(fan_in, fan_out):
72     W = np.random.randn(fan_in, fan_out) * 1e-2
73     return W
74
75 def glorot_normal(fan_in, fan_out):
76     """Xavier initialization Glorot et al., 2010"""
77     W = np.random.randn(fan_in, fan_out) * (2 / np.sqrt(fan_in + fan_out))
78     return W
79
80 def glorot_uniform(fan_in, fan_out):
81     """Xavier initialization Glorot et al., 2010"""
82     limit_uniform = [np.sqrt(6 / (fan_in + fan_out)), np.sqrt(6 / (fan_in + fan_out))]
83     W = (limit_uniform[1] - limit_uniform[0]) * np.random.random((fan_in, fan_out)) + limit_uni
84     form[0]
85     return W
86
87 def he_normal(fan_in, fan_out):
88     """He et al., 2015"""
89     W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in/2)
90     return W
91
92 n_layers = 10
93 n_neurons=500
94 n_samples = 10000
95 D = np.random.randn(n_samples, n_neurons)
96 n_neurons_by_layer = [n_neurons] * n_layers
97 act = {'relu': lambda x: np.maximum(0, x), 'tanh': lambda x: np.tanh(x)}
98 # Como todas las capas tienen la misma cantidad de neuronas puedo agrupar todo
99 # en una gran matriz
100 Hs = np.zeros((n_samples, n_neurons, n_layers))
101 for i in range(n_layers):
102     X = D if i==0 else Hs[... , i-1]
103     fan_in = X.shape[-1] # neuronas de la capa anterior
104     fan_out = n_neurons_by_layer[i] # neuronas de la capa i
105     # Inicializamos con media cero y std 1e-2
106     # W = small_normal(fan_in, fan_out)
107     # W = glorot_uniform(fan_in, fan_out)
108     # W = glorot_normal(fan_in, fan_out)
109     # He et. al 2015
110     W = he_normal(fan_in, fan_out)
111
112 H = X.dot(W) # producto interno
113 H = act['relu'](H)
114 Hs[... , i] = H
115
116 """
```

Executing transaction: done
ariel@Goku:~\$

miniconda3

Applications	Downloads	Logos	Phd	Scripts
Datos	Dropbox	Movies	Pictures	Sites
Desktop	GoogleDrive	Music	Proyectos	kernel-7878-ssh.j
son				kernel-7878.json
Docencia	Latex resources	OneDrive	Public	

ariel@Goku:~\$ rm kernel-7878-ssh.json
ariel@Goku:~\$ jupyter console --existing ../kernel-7878.json --ssh 10.73.27.83
[ZMQTerminalIPythonApp] Forwarding connections to 127.0.0.1 via 10.73.27.83
SSH Password for 10.73.27.83:
[ZMQTerminalIPythonApp] To connect another client via this tunnel, use:
[ZMQTerminalIPythonApp] --existing kernel-7878-ssh.json
Jupyter console 6.0.0

Python 3.7.3 (default, Mar 27 2019, 22:11:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.7.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:

ipython kernel
NOTE: When using the 'ipython kernel' entry point, Ctrl-C will not work.

To exit, you will have to explicitly quit this process, by either sending
"quit" from a client, or using Ctrl-\ in UNIX-like environments.

To read more about this, see <https://github.com/ipython/ipython/issues/2049>

To connect another client to this kernel, use:
--existing kernel-7878.json

OMP: Info #212: OMP_AFFINITY: decoding x2APIC ids.
OMP: Info #218: OMP_AFFINITY: Affinity capable, using global cpuid leaf 11 info
OMP: Info #154: OMP_AFFINITY: Initial OS proc set respected: 0-11
OMP: Info #156: OMP_AFFINITY: 12 available OS procs
OMP: Info #157: OMP_AFFINITY: Uniform topology
OMP: Info #179: OMP_AFFINITY: 1 packages x 6 cores/pkg x 2 threads/core (6 total cores)
OMP: Info #214: OMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: OMP_AFFINITY: OS proc 0 maps to package 0 core 0 thread 0
OMP: Info #171: OMP_AFFINITY: OS proc 6 maps to package 0 core 0 thread 1
OMP: Info #171: OMP_AFFINITY: OS proc 1 maps to package 0 core 1 thread 0
OMP: Info #171: OMP_AFFINITY: OS proc 7 maps to package 0 core 1 thread 1
OMP: Info #171: OMP_AFFINITY: OS proc 2 maps to package 0 core 2 thread 0
OMP: Info #171: OMP_AFFINITY: OS proc 8 maps to package 0 core 2 thread 1
OMP: Info #171: OMP_AFFINITY: OS proc 3 maps to package 0 core 3 thread 0
OMP: Info #171: OMP_AFFINITY: OS proc 9 maps to package 0 core 3 thread 1
OMP: Info #171: OMP_AFFINITY: OS proc 4 maps to package 0 core 4 thread 0
OMP: Info #171: OMP_AFFINITY: OS proc 10 maps to package 0 core 4 thread 1
OMP: Info #171: OMP_AFFINITY: OS proc 5 maps to package 0 core 5 thread 0
OMP: Info #171: OMP_AFFINITY: OS proc 11 maps to package 0 core 5 thread 1
OMP: Info #250: OMP_AFFINITY: pid 7878 tid 8278 thread 0 bound to OS proc set 0
OMP: Info #250: OMP_AFFINITY: pid 7878 tid 8225 thread 1 bound to OS proc set 1
OMP: Info #250: OMP_AFFINITY: pid 7878 tid 8226 thread 2 bound to OS proc set 2
OMP: Info #250: OMP_AFFINITY: pid 7878 tid 8227 thread 3 bound to OS proc set 3
OMP: Info #250: OMP_AFFINITY: pid 7878 tid 8228 thread 4 bound to OS proc set 4
OMP: Info #250: OMP_AFFINITY: pid 7878 tid 8229 thread 5 bound to OS proc set 5

NORMAL 07_TrainingNetworks_Part1.py[+] python utf-8[unix] 29% 58/195 ln : 1 E: (148/41)

Jupyter notebooks



Por último

- ❖ En ipython y jupyter existen los magics commands. Son comando que comienzan con %

%paste

%run

%timeit

%who

%reset

%autoreload

.....

<https://ipython.readthedocs.io/en/stable/interactive/magics.html>

¿Mostramos como usar ipdb?