

profiling_example

July 1, 2022

1 1A.soft - Exemple de profiling

Profiling et fonction *pdf*. Le profiling est utilisé pour mesurer le temps que passe un programme dans chaque fonction.

1.1 Bizarrerie

C'est un exemple qui m'a été envoyé par un étudiant pendant l'été pour montrer que la fonction `pdf` est plus lente qu'une réimplémentation simple qui fait à la même chose.

```
[1]: import time
      from scipy.stats import norm
      import numpy as np
```

```
[2]: debut=time.time()
      for i in range(10**3):
          norm(2,3).pdf(4)
      fin=time.time()
      fin-debut
```

```
[2]: 1.5188484191894531
```

```
[3]: def density(x,mean,sigma2):
      return np.exp(-(x-mean)**2/(2*sigma2))/(2*np.pi*sigma2)**0.5

      debut=time.time()
      for i in range(10**3):
          density(4,2,3)
      fin=time.time()
      fin-debut
```

```
[3]: 0.008997917175292969
```

Que se passe-t-il ?

Tout d'abord la fonction `pdf` comme toute les fonctions des bibliothèques numériques sont optimisées pour le calcul sur des matrices ou des vecteurs et non sur des nombres. Pour la suite, on utilise un profileur.

1.2 Profiler

```
[4]: import cProfile, io, pstats, os, sys
      def doprofile(func, filename, *l):
          pr = cProfile.Profile()
```

```

pr.enable() # début du profiling
func(*l)    # appel de la fonction
pr.disable() # fin du profiling
s = io.StringIO()
ps = pstats.Stats(pr, stream=s).sort_stats('cumulative')
ps.print_stats()
rem = os.path.normpath(os.path.join(os.getcwd(), "..", "..", ".."))
res = s.getvalue().replace(rem, "")
res = res.replace(sys.base_prefix, "").replace("\\", "/")
ps.dump_stats(filename)
return res

```

```

[5]: import numpy
x = numpy.ones((10000000, 1))*4
x.shape

```

[5]: (10000000, 1)

```

[6]: debut=time.time()
y = norm.pdf(x)
fin=time.time()
print(fin-debut, y.shape, y[0])

```

1.1308369636535645 (10000000, 1) [0.00013383]

```

[7]: r = doprofile(norm.pdf, "pdf.prof", x)
print(r)

```

106 function calls in 1.086 seconds

Ordered by: cumulative time

ncalls	totttime	percall	cumtime	percall	filename:lineno(function)
1	0.213	0.213	1.086	1.086	/lib/site-packages/scipy/stats/_distn_infrastructure.py:1619(pdf)
1	0.011	0.011	0.303	0.303	/lib/site-packages/scipy/stats/_distn_infrastructure.py:520(argsreduce)
1	0.080	0.080	0.292	0.292	/lib/site-packages/scipy/stats/_distn_infrastructure.py:545(<listcomp>)
1	0.000	0.000	0.286	0.286	/lib/site-packages/scipy/stats/_continuous_distns.py:129(_pdf)
1	0.286	0.286	0.286	0.286	/lib/site-packages/scipy/stats/_continuous_distns.py:79(_norm_pdf)
1	0.000	0.000	0.226	0.226	/lib/site-packages/numpy/lib/function_base.py:1964(place)
1	0.226	0.226	0.226	0.226	{built-in method numpy.core.multiarray._insert}
2	0.012	0.006	0.212	0.106	/lib/site-packages/numpy/lib/function_base.py:1913(extract)
2	0.000	0.000	0.127	0.063	/lib/site-packages/numpy/core/fromnumeric.py:1491(nonzero)
2	0.127	0.063	0.127	0.063	{method 'nonzero' of

```

'numpy.ndarray' objects}
  2  0.000  0.000  0.073  0.037 /lib/site-
packages/numpy/core/fromnumeric.py:56(take)
  2  0.073  0.037  0.073  0.037 {method 'take' of 'numpy.ndarray'
objects}
  1  0.032  0.032  0.032  0.032 /lib/site-
packages/scipy/stats/_distn_infrastructure.py:871(_support_mask)
  1  0.018  0.018  0.018  0.018 {built-in method
numpy.core.multiarray.putmask}
  1  0.000  0.000  0.009  0.009 /lib/site-
packages/numpy/core/fromnumeric.py:1901(any)
  1  0.000  0.000  0.009  0.009 {method 'any' of 'numpy.ndarray'
objects}
  1  0.000  0.000  0.009  0.009 /lib/site-
packages/numpy/core/_methods.py:37(_any)
  1  0.008  0.008  0.008  0.008 {method 'reduce' of 'numpy.ufunc'
objects}
  1  0.000  0.000  0.000  0.000 {built-in method
numpy.core.multiarray.zeros}
  4  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/fromnumeric.py:1384(ravel)
  1  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numerictypes.py:964(find_common_type)
  2  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numerictypes.py:942(_can_coerce_all)
  1  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/shape_base.py:9(atleast_1d)
  4  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numeric.py:414(asarray)
  7  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numeric.py:484(asanyarray)
 11  0.000  0.000  0.000  0.000 {built-in method
numpy.core.multiarray.array}
  4  0.000  0.000  0.000  0.000 {method 'ravel' of 'numpy.ndarray'
objects}
  1  0.000  0.000  0.000  0.000 {method 'reshape' of
'numpy.ndarray' objects}
  6  0.000  0.000  0.000  0.000 {built-in method
builtins.isinstance}
 14  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numerictypes.py:951(<listcomp>)
  1  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/fromnumeric.py:1575(shape)
  1  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numerictypes.py:1015(<listcomp>)
  1  0.000  0.000  0.000  0.000 /lib/site-
packages/scipy/stats/_distn_infrastructure.py:859(_argcheck)
 19  0.000  0.000  0.000  0.000 {built-in method builtins.len}
  2  0.000  0.000  0.000  0.000 {method 'append' of 'list'
objects}
  1  0.000  0.000  0.000  0.000 <string>:2(_parse_args)
  1  0.000  0.000  0.000  0.000 /lib/site-
packages/numpy/core/numerictypes.py:1016(<listcomp>)
  1  0.000  0.000  0.000  0.000 {method 'disable' of

```

```
'_lsprof.Profiler' objects}
```

```
[8]: def density(x,mean,sigma2):  
      return np.exp(-(x-mean)**2/(2*sigma2))/(2*np.pi*sigma2)**0.5  
  
debut=time.time()  
y = density(x,0.0,1.0)  
fin=time.time()  
print(fin-debut, y.shape, y[0])
```

```
0.36240410804748535 (10000000, 1) [ 0.00013383]
```

```
[9]: r = doprofile(density, "pdf.prof", x, 0, 1)  
print(r)
```

```
2 function calls in 0.341 seconds
```

```
Ordered by: cumulative time
```

```
ncalls  tottime  percall  cumtime  percall  filename:lineno(function)  
1      0.341    0.341    0.341    0.341  <ipython-  
input-15-e74f743d975f>:1(density)  
1      0.000    0.000    0.000    0.000  {method 'disable' of  
'_lsprof.Profiler' objects}
```

Quand on regarde le code de la fonction, on s'aperçoit que la fonction perd du temps dans [argsreduce](#). Elle fait aussi d'autres choses comme regarder les valeurs manquantes. En guise de conclusion, lorsqu'une fonction gère trop de cas particuliers (type, valeurs), elle est nécessairement plus lente qu'une fonction qu'on implémente soi-même.

```
[10]:
```

```
[11]:
```