Optimización usando Python

Dr. Luis Gerardo de la Fraga

Correo-e: fraga@cs.cinvestav.mx
Departamento de Computación
Cinvestav Zacatenco

18 de noviembre, 2020

Contenido:

1. Introducción a python
2. Problemas no lineales
3. Cómo se resuelven los problemas no lineales
4. Solución de problemas no lineales con heurísticas:
5. El algoritmo genético
6. La evolución diferencial

¿Por qué usar python?

- Es un lenguaje de alto nivel
- Es fácil construir programas rápidamente
- Es uno de los lenguajes más usandos
- Es el lenguaje sugerido para programar en RaspBerryPi

Lenguajes de alto nivel

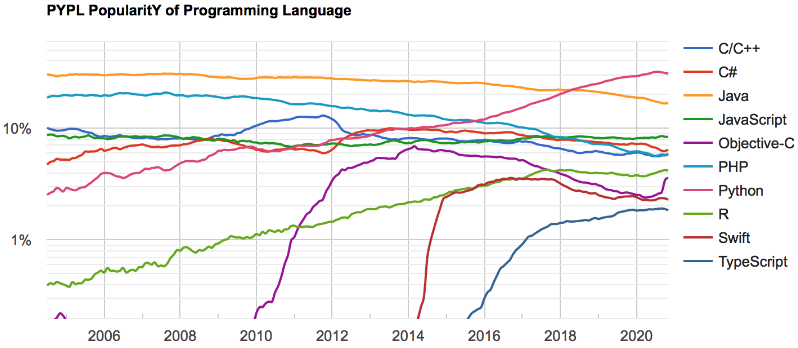
- C es un lenguaje de nivel medio
- Los leguajes de alto nivel son interpretados
- No se definen tipos de datos
- Tienen una máquina de tiempo de ejecución (para uso de memoria con un recolector de basura)
- Se puede empotrar funciones en C dentro de python si se necesita eficiencia.

Prototipado rápido

- Son ideales para realizar programas que procesan texto y generan texto
- Para generación dinámica de páginas WEB
- Si se tiene una idea, puede obtenerse un programa funcional en minutos u horas

Posición	Lenguaje	Uso %	Tendencia
1	Python	30.80 %	+1.8 %
2	Java	16.79 %	-2.3 %
3	JavaScript	8.37 %	+0.3 %
4	C#	6.42 %	-0.9 %
5	PHP	5.92 %	-0.2 %
6	C/C++	5.78 %	-0.2 %
7	R	4.16 %	+0.4 %
8	Objective-C	3.57 %	+1.0 %
9	Swift	2.29 %	-0.2 %
10	TypeScript	1.84 %	-0.0 %

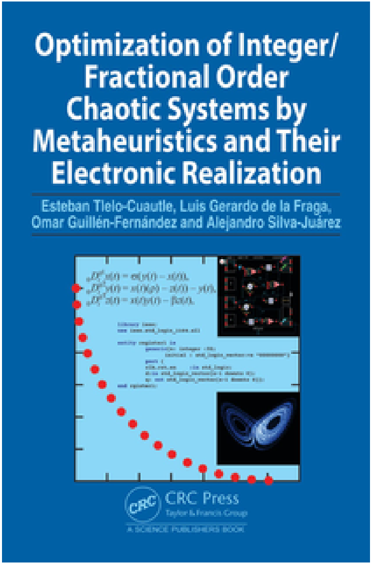
`http://pypl.github.io/PYPL.html`



http://pypl.github.io/PYPL.html

http://www.raspberrypi.org/

Se puede comprar en www.newark.com.mx



- Este libro saldrá a la venta en el año 2021.
- Los programas que veremos son parte del contenido de este libro y están disponibles en:
- `https://cs.cinvestav.mx/~fraga/OptCode.tar.gz`

Formas de usar python

- Existen dos formas principales para trabajar en python:

1. A través de una interfaz como jupyter
2. Usando la línea de comandos y un editor

Jupyter

- Jupyter es un ambiente interactive para ejecutar código dentro del navegador WEB
- Jupyter puede usarse con otros lenguajes de programación
- Dentro de las notas de jupyter se puede incorporar código, texto e imágenes

https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

En este taller usaré la línea de comandos

- Todos los programas de python son archivos
- Solo se necesita un editor (usaré "vi") y la terminal
- Este el ambiente común de desarrollo de Unix o GNU/Linux
- Esta pensado para incorporar programas dentro de otros programas

Python en 15 minutos

```
# Mi primer programa
# en python
#
i = 0
while i < 10 :
    i += 1
    print( i )
```

- Con # se escribe un comentario
- El inicio y fin de bloque se indican con la indentación
- No hay tipos
- La variable `i` es una variable entera porque está inicializada con un valor entero

```
#
# Funciones en python
#
def f( x ) :
    v = (x-1.0)*(x-4.0)
    return v

print( f(4.0) )
```

En un archivo que se llama `funcion.py`

```
#
# Funciones en python
#
def f( x ) :
    v = (x-1.0)*(x-4.0)
    return v
```

En otro archivo:

```
import funcion

x = -1.0
while x < 4.1 :
    print( x, funcion.f( x ) )
    x += 0.5
```

Paradigmas de programación en python

1. Imperativa
2. En objetos
3. Funcional

Archivo `punto.py`

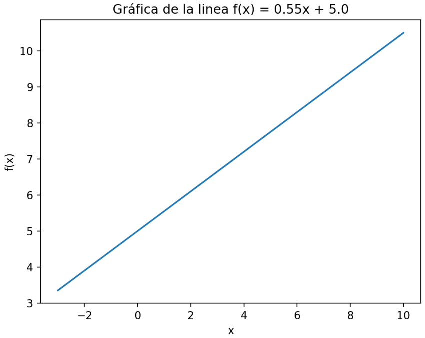
```
class Punto :
  def __init__( self, x, y ) :
    self.x = x
    self.y = y

  def __add__( self, A ) :
    p = Punto(self.x+A.x,
              self.y+A.y)
    return p

  def print( self ) :
    print( self.x, self.y )
```

En el archivo `objetos.py`

```
import punto

A = punto.Punto( 2, 3 )
A.print()

B = punto.Punto( 1, 2 )
B.print()

C = A + B
C.print()
```

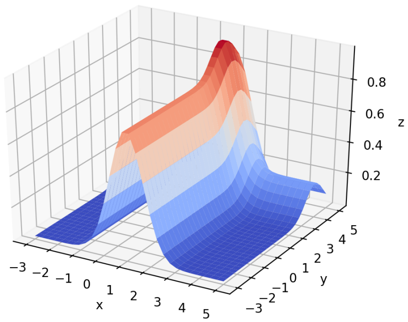
Una función lineal

$$f(x) = a_1 x + a_0$$



La dependencia puede ser lineal con respecto a los coeficientes, pero usando funciones no lineales:

$$f(x,y) = a_1\, g_1(x,y) + a_0\, g_0(x,y)$$
$$f(x,y) = a_1 \exp[-(x-1)^2] + a_0 \exp[-(y-4)^2]$$



Función $f(x,y)$ con $a_1 = 0.8$ y $a_0 = 0.2$

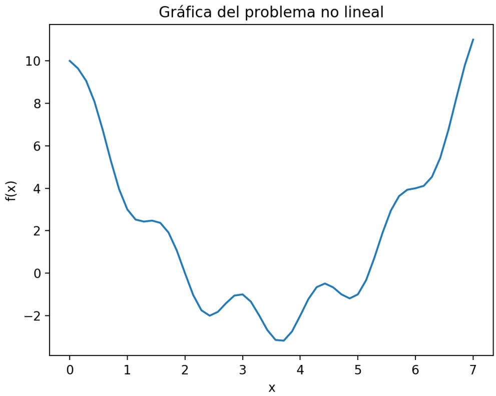
- Los problemas lineales son “fáciles” de resolver
- Su solución es equivalente al costo computacional de invertir una matriz.
- Otros problemas lineales se resuelven con la eigendescomposición de una matriz.

Problemas no lineales

- Estos son “difíciles” de resolver
- De forma clásica se necesita una solución muy cerca de la solución del problema
- El problema no lineal se lineariza usando series de Taylor y
- se itera resolviendo cada vez un problema lineal hasta que converja.

Este es un problema no lineal:

$$\text{Minimizar: } f(x) = (x-2)(x-5) + \text{sen}(1.5\pi x)$$



Para resolver un problema no lineal:

- Necesitamos una aproximación de la solución muy cerca de la solución del problema (¿Esto es una contradicción?)
- Podemos usar el método de Newton que aproxima la función no lineal con una aproximación lineal (usando los dos primeros términos de la serie de Taylor).
- Iteramos hasta la **convergencia**.

La series de Taylor

$$f(\mathbf{x}) = \sum_{i=0}^{\infty} \frac{(\mathbf{x}-\mathbf{a})^i f^{(i)}(\mathbf{a})}{i!} \tag{1}$$

Usando los dos primeros términos de (1), obtenemos:

$$f(\mathbf{x}) \approx f(\mathbf{a}) + (\mathbf{x}-\mathbf{a})f'(\mathbf{a}) = f(\mathbf{a}) + \Delta\mathbf{a} f'(\mathbf{a}) \tag{2}$$

Y esta es una aproximación lineal de la función f alrededor del punto $\mathbf{a}$

El método de Newton

- Donde f es mínima, está función es plana y su derivada debe ser igual a cero.
- Para econtrar el punto mínimo entonces calculamos la derivada de f e igualamos a cero para encontrar el valor de $\mathbf{x}$.
- La aproximación lineal de la derivada de f es:

$$f'(\mathbf{x}) \approx f'(\mathbf{a}) + (\mathbf{x} - \mathbf{a})f''(\mathbf{a}) = f'(\mathbf{a}) + \Delta\mathbf{a}\ f''(\mathbf{a}) \tag{3}$$

  Igualando a cero se obtiene:

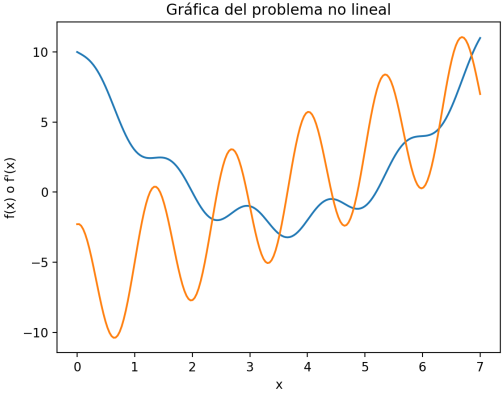
$$\Delta\mathbf{a} = \frac{-f'(\mathbf{a})}{f''(\mathbf{a})} \tag{4}$$

Y se itera $\mathbf{a}_{i+1} = \mathbf{a}_i + \Delta\mathbf{a}$, comenzando con un $\mathbf{a}_0$

```python
def f1(x):
    return (x-2.0)*(x-5.0) + math.sin( 1.5*math.pi*x )

def df1(x):
    return 2.0*x - 7.0 + 1.5*math.pi * math.cos( 1.5*math.pi*x )

def ddf1(x):
    return 2.0 - 1.5*math.pi*1.5*math.pi*math.sin(1.5*math.pi*x)

print( "0", x )
i = 1
while i < 20 :  # 20 maximum iterations
    Deltaa = -df1(x)/ddf1(x)
    x += Deltaa
    print( i, x )

    if abs(Deltaa) < 1e-10 :
        break

    i += 1
```

Este es un problema no lineal:

$$\text{Minimizar: } f(x) = (x-2)(x-5) + \operatorname{sen}(1.5\pi x)$$



Una prueba del método de Newton

Usando el siguiente algoritmo:

1. Generar un número aleatorio en $[0, 7]$
2. Usar este número como a_0
3. Si la solución difiere en 0.001 del valor óptimo, se halló la solución al problema
4. Ir al paso 1

Se ejecutó 500 veces este algoritmo y tuvo un 13.8 % de éxito.

Para el tipo de problemas no lineales y multimodales se justifica usar heurísticas.

El algoritmo genético

- Utiliza un conjunto de soluciones (se le llama población)
- Combina soluciones para generar otras
- La solución que sobrevive es la mejor (la más apta)
- Usa los operadores de selección, cruza, mutación y elitismo
- La población guarda la "inteligencia" del algoritmo

Pseudocódigo para un algoritmo genético

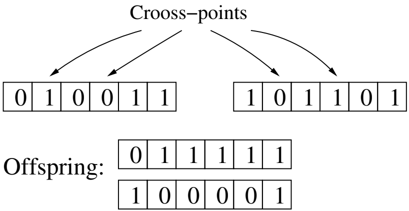
Require: Tamaño de la población μ, número de generaciones g, probabilidad de cruza p_c, probabilidad de mutación p_m. La función f a optimizar.

Ensure: Una solución $\mathbf{x}$ que minimiza $f(\mathbf{x})$

Aletoriamente crear la población inicial $P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mu\}$.
Calcular la aptitud de los individuos.
for $i \in [1, g]$ **do**
  for $j \in [1, \mu/2]$ **do** ▷ μ debe ser un número par
    Seleccionar dos individuos (llamados padres) que se reproducirán.
    Con probabilidad p_c, aplicar el operador de cruza
    Aplicar el operador de mutación con probabilidad p_m.
    Calcular la función de aptitud.
    Aplicar el mecaniso de elitismo.
  end for
  La nueva población de hijos reemplaza a los padres.
end for
Reportar el individuo con la mejor aptitud.

Operadores genéticos

1. Selección: se usar el **torneo binario**, la población se barajea aleatoriamente y dos padres consecutivos se van escogiendo. Gana el de mejor aptitud.
2. Cruza de dos puntos:



Prueba del algoritmo genético

- El espacio de búsqueda fue $[0, 7]$
- 20 individuos y 30 generaciones
- Se codificó cada individuo en 10 bits
- La solución es exitosa si $|x - 3.65288744| < 0.05$.
- Tuvo una tasa de éxito de 97.2 %
- Esto requirió $20 \times 30 = 600$ evaluaciones de la función objetivo
- El AG solo requiere la función objetivo

La evolución diferencial

- Usa números reales en su representación
- Realiza un mejor trabajo que el algoritmo genético
- Se puede usar una condición automática de paro

initialize($P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mu\}$)
evaluate(P)
$k = 0$
repeat
  for $i = 1$ to μ **do**
    Let r_1, r_2 and r_3 be three random integers in $[1, \mu]$, such that $r_1 \neq r_2 \neq r_3$
    Let j_{rand} be a random integer in $[1, n]$
    for $j = 1$ to n **do**
      $$x'_j = \begin{cases} x_{r_3,j} + F(x_{r_1,j} - x_{r_2,j}) & \text{if } U(0,1) < R \text{ or } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise} \end{cases}$$
      if $x'_j < l_i$ or $x'_j > u_i$ **then** ▷ Check bounds
        $x'_j = U(0,1)(u_i - l_i) + l_i$
      end if
    end for
    if $f(\mathbf{x}') < f(\mathbf{x}_i)$ **then**
      $\mathbf{x}_i = \mathbf{x}'$
    end if
  end for
  for $i = 2$ to μ **do**
    mín = mejor individuo
    máx = peor individuo
  end for
until (máx − mín) $< s$ or $k > g$

Una prueba

- Con 16 individuos
- 30 generaciones
- Constante de diferencias igual a 0.8
- Constante de recombinación igual a 0.6
- Intervalo de búsqueda en $[0, 7]$
- Después de 500 ejecuciones obtuvo una tasa de éxito del 100 %
- Un exito se cuenta si $|x - 3.65288744| < 10^{-4}$.

Resumen

1. Se estudió el lenguaje de programación python
2. Se describió un problema no lineal y multimodal
3. Se revisó el método de Newton,
4. el algoritmo genético y
5. la evolución diferencial

Resultados

Método	Tasa éxito	Éxito
Newton con inicio aleatorio	13.8 %	$\lvert x - 3.65288744\rvert < 0.001$
Algoritmo genético	97.2 %	$\lvert x - 3.65288744\rvert < 0.05$
Evolución diferencial	100.0 %	$\lvert x - 3.65288744\rvert < 10^{-4}$

Más material

- Mi correo-e: fraga@cs.cinvestav.mx
- Página web: `https://delta.cs.cinvestav.mx/~fraga`
- Programas del libro: `https://delta.cs.cinvestav.mx/~fraga/OptCode.tar.gz`
- pymoo (`https://pymoo.org`), para resolver problemas multiobjetivo en python. Este paquete provee varios algoritmos evolutivos y una guía de uso para resolver los problemas que tengamos a mano.

¡Gracias!