Wydział
Informatyki

Algoritmos de la Forma Greedy

Jose Lucha (1890), Leonel Contreras (18797)

Agenda

Ideas Basicas

Tecnica de resolucion de problemas que toma decisiones localmente optimas con la esperanza de obtener una solucion globalmente optima

Funciona para problemas que satisfacen la condicion de subestructura optima. n problema tiene una subestructura optima si una solucionoptima del problema global contiene las soluciones optimas a sus sub-problemas.

Ideas Basicas

En muchos problemas, la estrategia greedy no da una solucion optima ya quetoma decisiones basado en la informacion de cada paso y no considera el pro-blema en conjunto. La eleccion echa por un algoritmo que use la tecnica greedydepende de las elecciones que ha echo hasta el momento, pero no es consientede las decisiones futuras

Por qué considerar el metodo?

Como es sabido, este metodo no aplica para varios tipos de problemas. Sin embargo, se implementa ya que aproxima una solucion optima en un tiempo considerable -o en una cantidad de pasos considerable-.

Hill Climbing

El metodo de Hill Climbing es una tecnica de optimizacion que pertence la familia de busqueda local. Esta tecnica empieza con una solucion arbitraria al problema y se mueve de solucion en solucion en el espacio de soluciones candidato al aplicar cambios locales.

El metodo Hill Climbing encuentra soluciones optimas para problemas convexos. Es decir, miniza funciones convexas y conjuntos convexos

Hill Climbing

La diferencia entre el hill climbing y los metodos de descenso gradiente en una funcion $f : R^n \longmapsto R$ es que el metodo de hill climbing ajusta uno de los componentes x_i del vactor $x \in R^n$ y determina si el cambio mejora o no $f(x)$. Mientras que los metodos de descenso gradiente ajustan todas las componentes del vector de acuerdo al gradiente.

Funcionamiento del algoritmo

1. Sean $f : R^n \longrightarrow R$ y x_0 un punto del dominio.
2. Evaluamos $f(x_0)$ y lo establecemos como la opcion optima.
3. Seleccionamos, bajo algun criterio, un puntos en la vecindad de x_0 y evaluamos la funcion en este punto.
4. Si este nuevo punto genera una mejor solucion para la funcion lo designamos como nueva opcion optima y repetimos el proceso. Si la solucion no es mejor a nuestro x_0 selecciononamos un nuevo punto.

Debilidades del metodo

Debido a la idea detras del proceso del metodo el desempeño de este depende mucho de el x_0 elegido y la geometria de la funcion.

1. Maximos y minimos locales: Sabemos que este tipo de puntos son las cotas optimas en una region, pero no necesariamente de toda la funcion. Debido a esto si el metodo hill climbing se topa con uno de estos puntos lo considerara la solucion optima para el problema.
2. Planicies: Diremos que una planicie es una region constate dentro de la funcion donde estos puntos contantes son maximo o minimo local. Debido a que el valor de la funcion no cambia dentro de esta region el metodo se detendra en esta y considerara la constante como solucion optima y algun punto de la planicie como el punto que la genera.

Hill Climbing Simple

1. Sea x_0 nuestro punto inicial y designamos $f(x_0)$ como a mejor solucion.
2. Nos desplazamos en una direccion fija y evaluamos la funcion en este punto.
3. Si la solucion en este punto es mejor designamos este nuevo punto como mejor solucion y repetimos el proceso.
4. De lo contario nos deplazamos en la direccion contaria y evaluamos de nuevo.

Implementacion

]Hill1.PNG

Rysunek: Metodo Hill Climbing Simple

Implementacion

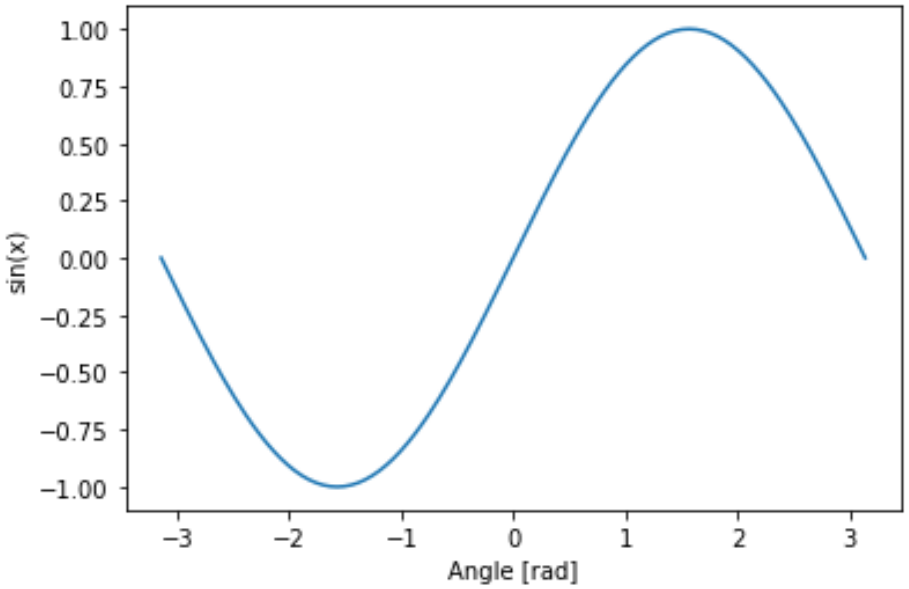
```
else:

    if f(x_0+tol)<f(x_0):
        dire = tol
        nextstep = f(x_0+dire)
        while nextstep < best:
            best = nextstep
            dire = dire+tol
            it = it + 1
            if it > it_max:
                return 'Maximo de iteraciones alcanzado. Ultima solucion: ', x_0+dire
            nextstep = f(x_0+dire)

    else:
        dire = -tol
        nextstep = f(x_0+dire)
        while nextstep < best:
            best = nextstep
            dire = dire-tol
            it = it + 1
            if it > it_max:
                return 'Maximo de iteraciones alcanzado. Ultima solucion: ', x_0+dire
            nextstep = f(x_0+dire)
print('Numero de iteraciones: ',it)
return 'La solucion optima se encuentra aproximadamente en:, ', x_0+dire
```

Rysunek: Metodo Hill Climbing Simple

Ejemplo




```
Numero de iteraciones:  25707963
('La solucion optima se encuentra aproximadamente en:, ', -1.5707963993998835)
Numero de iteraciones:  5707963
('La solucion optima se encuentra aproximadamente en:, ', 1.570796399976084)
```

Rysunek: Hill Climbing Simple Aplicado a Sin(x)

Hill Climbing Estocastico

1. Sea x_0 nuestro punto inicial y designamos $f(x_0)$ como a mejor solucion.
2. Dentro de una vecindad de x_0 seleccionamos aleatoriamente un nuevo punto y evaluamos la funcion en este punto.
3. Si la solucion es mejor a la anterior seleccionamos este puento como la nueva mejor solucion.
4. De lo contrario escoger nuevamente un numero aleatorio dentro del intervalo definido.

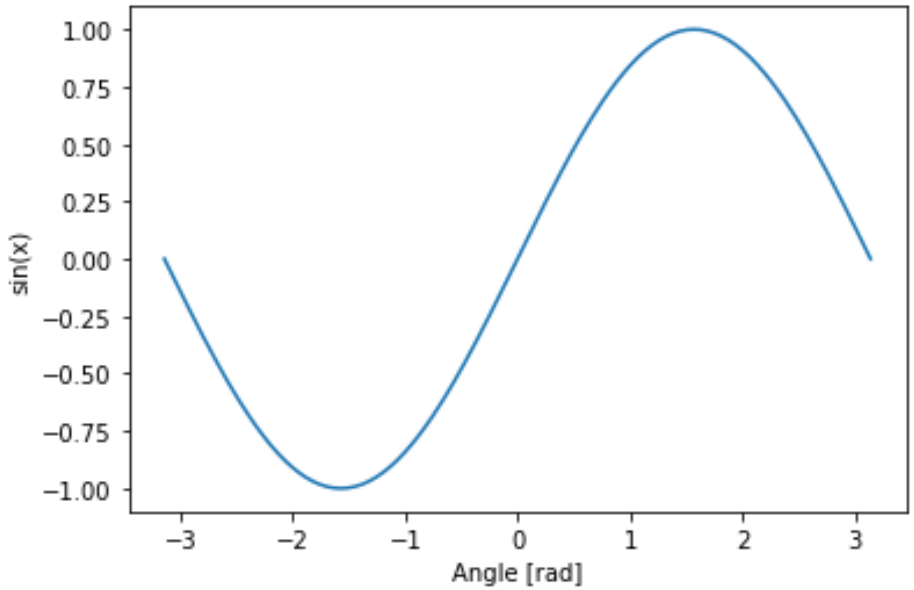
Implementacion

```python
#Hill climbing estocastico
#obj_max = False --> minimo
#obj_max = True --> Maximo
def Stochastic_hill_climbing(x_0,f,obj_max,tol,maxit):
    best = f(x_0)
    x = np.random.uniform(x_0-tol, x_0+tol)
    it = 0
    if obj_max == True:
        while it < maxit:
            if f(x)>f(x_0):
                x_0 = x
                x = np.random.uniform(x_0-tol, x_0+tol)
            else:
                x = np.random.uniform(x_0-tol, x_0+tol)
                it = it+1
    else:
        while it < maxit:
            if f(x)<f(x_0):
                x_0 = x
                x = np.random.uniform(x_0-tol, x_0+tol)
            else:
                x = np.random.uniform(x_0-tol, x_0+tol)
                it = it+1

    return 'La solucion optima se encuentra aproximadamente en:, ', x_0
```

Rysunek: Metodo Hill Climbing Estocastico

Ejemplo




```
('La solucion optima se encuentra aproximadamente en:, ', -1.570584098218456)

('La solucion optima se encuentra aproximadamente en:, ', 1.570810559730656)
```

Rysunek: Hill Climbing Estocastico Aplicado a Sin(x)

Comentarios y Recomendaciones

1. Utilizar este metodo con un punto de inicio cercano a la solucion aproximada.
2. Tomar en cuenta si se evalua una region adecuada.
3. De no obtener una respuesta satisfactoria reevaluar el punto de inicio o el numero de iteraciones.
4. La calidad de la aproximacion de la respuesta depende de la tolerancia e iteraciones utilizadas.

Referencias

1. S. Russell, P. Norvig, (2020) Artificial intelligence: a modern approach ,Prentice Hall, New Jersey
2. O. Mbaabu (16 de diciembre, 2020) Understanding Hill Climbing Algorithm in Artificial Intelligence. Obtenido de: https://www.section.io/engineering-education/understanding-hill-climbing-in-ai/