Notas de Asignatura

Inteligencia Artificial I: OPTIMIZACIÓN Y METAHEURÍSTICAS

Claudia Sánchez

Elaborado por:

Profesora: Claudia Nallely Sánchez Gómez

Universidad Panamericana campus Aguascalientes
Facultad de Ingeniería
Academia de Cómputo

Contenido

1. Inteligencia Artificial

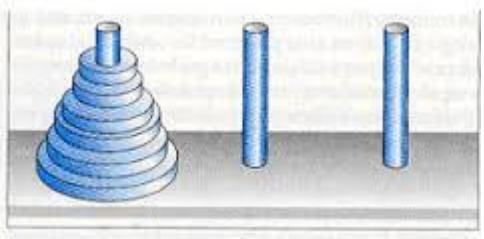
Nos hemos llamado a nosotros mismos Homo sapiens, ya que nuestras capacidades mentales son muy importantes para nosotros. Por miles de años hemos tratado de entender como pensamos, esto es, como se puede percibir, entender, predecir y manipular lo que nos rodea.
El campo de la inteligencia artificial o IA intenta no sólo entender si no crear entidades inteligentes. La IA sistematiza y automatiza tareas intelectuales y por tanto es potencialmente relevante en cualquier ámbito humano intelectual. En este sentido, es ciertamente un campo universal.

Algunas definiciones emitidas por diferentes investigadores de la IA son:

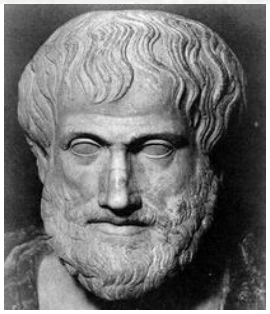
- Capacidad de razonar de un agente no vivo
- Arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia.
- Rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente
- Campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales.

1.1 Breve historia de la Inteligencia Artificial

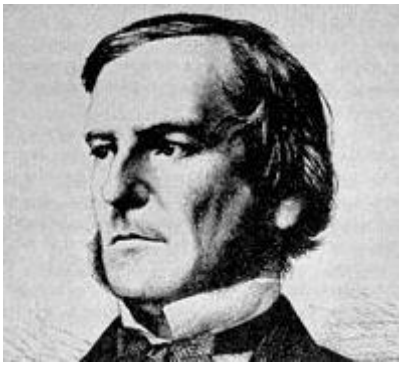
3000 a.C. Los juegos matemáticos como las Torres de Hanoi muestran el interés por la búsqueda de solución optimizando el número de movimientos empleados para resolver el problema.



~ 300 a.C. **Aristóteles** fue el primero en describir de manera estructurada un conjunto de reglas, silogismos, que describen una parte del funcionamiento de la mente humana, y que al seguirlas producen conclusiones racionales.

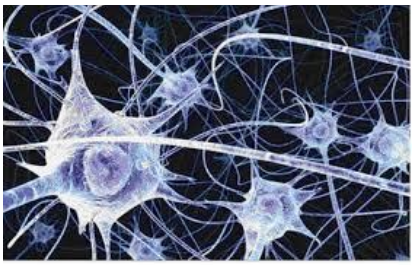


1847 George Boole estableció la lógica proposicional (booleana), mucho más completa que los silogismos de Aristóteles



1879 Gottlob Frege extiende la lógica booleana y obtiene la Lógica de Primer Orden

1937 Alan Turing. Estableció las bases teóricas de la computación, puede considerarse el origen de la informática teórica.
Introdujo el concepto de Máquina de Turing donde se formalizó el concepto de Algoritmo y fue la precursora de las computadoras digitales.
A Alan Turing se le considera uno de los padres de la Computación junto con Charles Babbage.
Él y su equipo construyeron el primer computador electromecánico.
(Ver la película Descifrando Enigma)



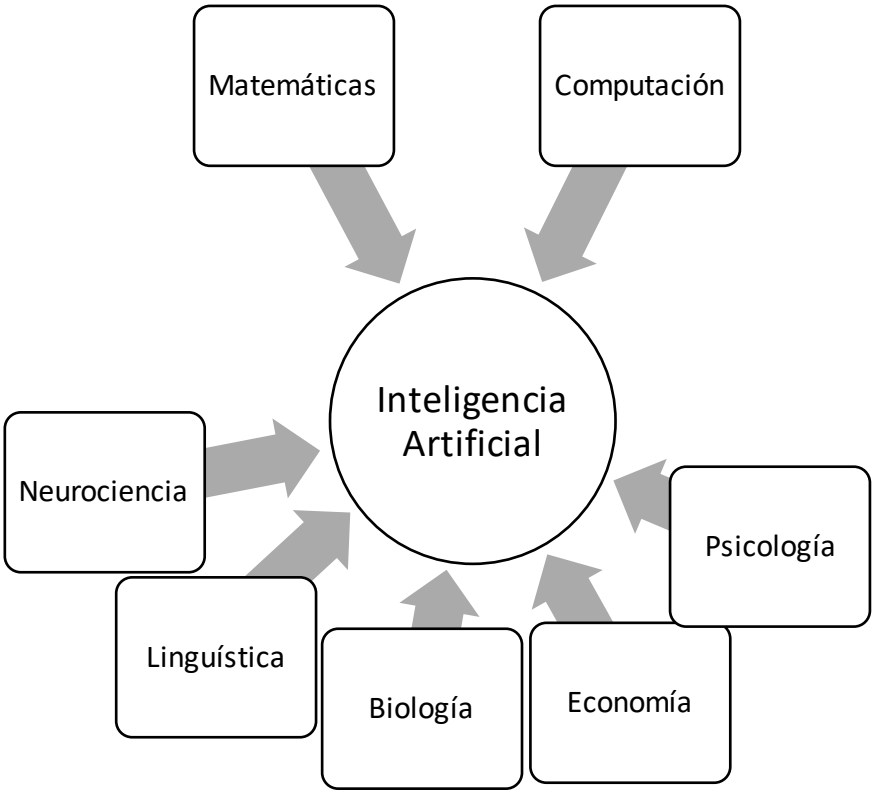
1943 Warren McCulloch y Walter Pitts presentaron su modelo de neuronas artificiales, el cual es considerado como el primer trabajo de inteligencia artificial (aunque aún no existía el término).

1950 Turing consolidó el campo de la IA con su artículo *Computing Machinery and Intelligence*, en que propuso una prueba concreta para determinar si una máquina era inteligente o no, **Turing's Test**, por lo que se le considera el Padre de la Inteligencia Artificial.

1956 se dio el **término Inteligencia Artificial** en la Conferencia de Dartmouth por John McCarthy. La conferencia se llevó a cabo en la universidad Dartmouth College ubicada en Nuevo Hampshire, Estados Unidos.

1959 John Henry Hollad propuso un método para la solución de problemas: los Algoritmos Genéticos.

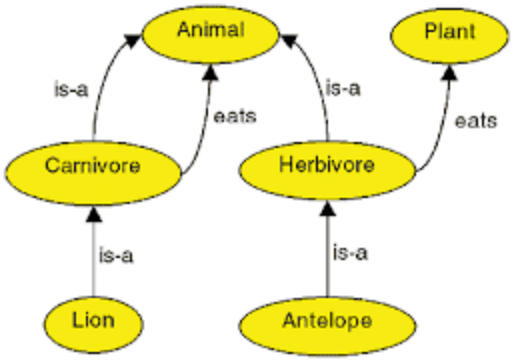
1.2 Fundamentos de la Inteligencia Artificial



1.3 Campos de la Inteligencia Artificial

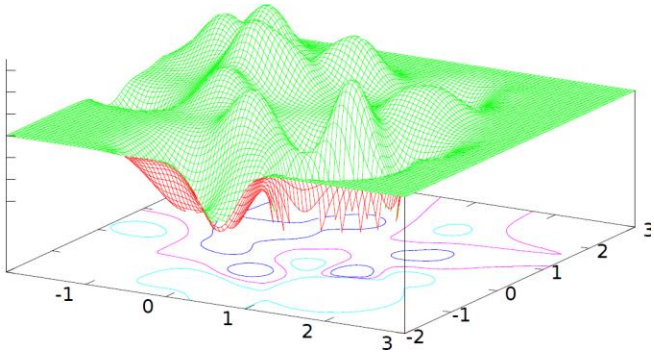
Representación del conocimiento y Razonamiento

Consiste en crear técnicas para representar información y después realizar razonamientos lógicos.



Aprendizaje de Máquina

Consiste en generar herramientas que puedan ser capaces de procesar datos históricos para entender su comportamiento y después realizar inferencias.



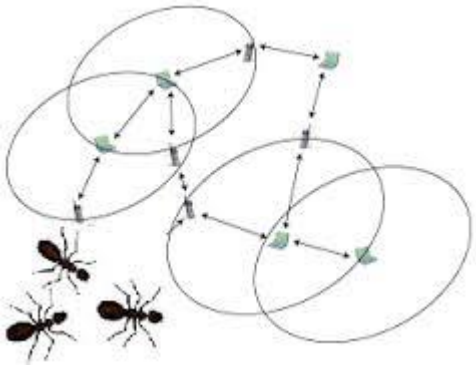
Planificación

Consiste en representar mapas mediante grafos con el objetivo de encontrar rutas óptimas.

Cómputo Evolutivo y Algoritmos BioInspirados

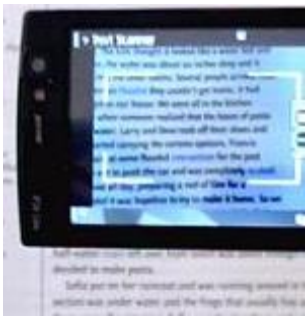
El cómputo evolutivo consiste en crear algoritmos basados en la teoría de evolución de Charles Darwin.

Los algoritmos bioinspirados se crean a partir del análisis de cómo es que la naturaleza resuelve problemas. Por ejemplo se imita el comportamiento de las hormigas, abejas, luciérnagas, parvadas, etc.



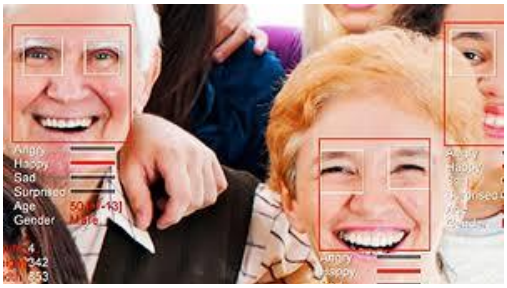
Procesamiento del Lenguaje Natural

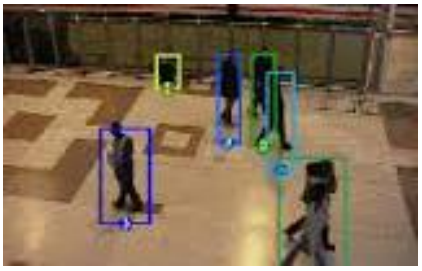
Consiste en crear una máquina o programa que sea capaz de entender el lenguaje natural de los hombres. Puede enfocarse en texto o voz.



Visión Computacional

Consiste en procesar imágenes para reconocer figuras, colores, puntos de interés, rostros humanos, etc.

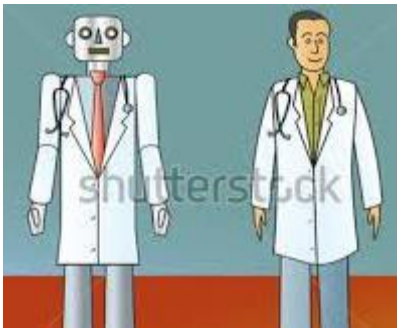




1.4 Aplicaciones de la Inteligencia Artificial

Sistemas Expertos

Consiste en crear un sistema capaz de "pensar", "razonar" o "resolver un problema" como lo haría una persona especialista. Hace uso de la Representación del Conocimiento y del Aprendizaje de Máquina.



Optimización

Consiste en la selección del mejor elemento (con respecto a algún criterio) de un conjunto de elementos posibles. Puede aplicarse para optimizar ganancias, costos, tiempos, etc. Hace uso de las Matemáticas, Cómputo Evolutivo y Algoritmos Bio Inspirados.

Robótica

Algunas de las técnicas más estudiadas en IA para robótica es la planificación de movimientos, comunicación entre robots y creación de mapas del entorno en base a lo que se percibe en cada movimiento. Utiliza:

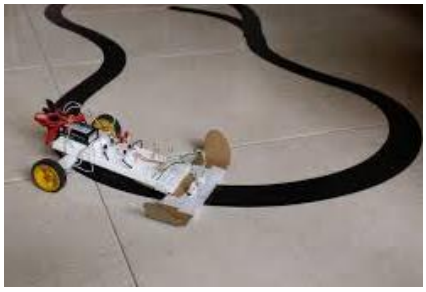
- Planificación
- Procesamiento del Lenguaje Natural
- Visión Computacional
- Aprendizaje de Máquina

Agentes Inteligentes

Consiste en crear entes que sean capaces de percibir su entorno, razonar y actuar.

Control autónomo

Consiste en generar máquinas que sean capaces de conducirse o moverse de forma autónoma. Utilizan:
- Visión computacional
- Aprendizaje de Máquina
- Representación del conocimiento
- Planificación



Minería de Datos y Análisis de Grandes Volúmenes de Datos (BIG DATA)

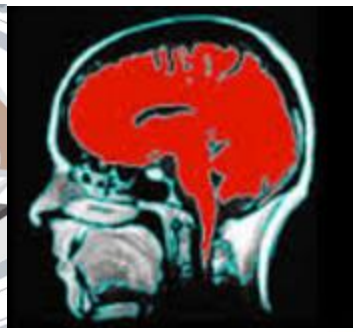
Big Data consiste en el análisis de los datos que se generan en grandes cantidades día con día como transacciones bancarias, mensajes de texto, información en redes sociales, etc. Hace uso de algunas técnicas como Aprendizaje de Máquina, Procesamiento del Lenguaje Natural, Bases de Datos, Sistemas Distribuidos, Cómputo en la Nube, etc.

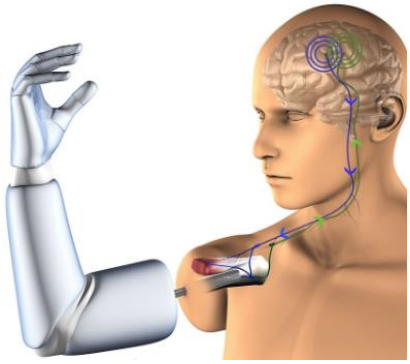
Minería de Datos consiste en el análisis de los datos que se encuentran en una base de datos, es muy parecido a Big Data pero en chiquito.

Análisis Médicos

Procesamiento de Señales del cuerpo humano o del resultado de Análisis Médico, que dan como resultado:

- Resonancias magnéticas
- Ultrasonidos
- Prótesis
- Catéter
- Etcétera





Análisis de Sentimientos

Consiste en el análisis de la información publicada en redes sociales con el objetivo de detectar las emociones o sentimientos de una población o un individuo en especial. Tiene aplicaciones interesantes como la detección de ataques terroristas, prevenir suicidios, detectar preferencias electorales, etcétera. Hace uso del Procesamiento del Lenguaje Natural, Cómputo en la Nube y Aprendizaje de Máquina.

2. Optimización

La optimización consiste en la selección del mejor elemento (con respecto a algún criterio) de un conjunto de elementos posibles.

La optimización es una tarea diaria y común, por ejemplo:

- Compañías de transporte (aerolíneas, autobuses) optimizan las asignaciones (calendarios) de las tripulaciones para reducir costos
- Las manufactureras buscan minimizar tiempo y costo para maximizar ganancias
- Buscar la ruta más corta de una ciudad a otra

Por lo tanto la optimización es importante en la toma de decisiones.

Para optimizar se debe definir una **medida de desempeño o una función objetivo** que nos permita distinguir buenas soluciones de malas soluciones. Para ello se debe definir o identificar una función objetivo, por ejemplo:

- Ganancia
- Tiempo
- Energía
- Espacio, etc

Estas cantidades se deben representar con un número y el objetivo del problema debe ser maximizarlas o minimizarlas. El resultado de la función objetivo debe depender de las características de la solución denominadas **variables o incógnitas**. La meta es encontrar los valores de las variables que optimizan el sistema.

Es común encontrar **restricciones**, por ejemplo: “el tiempo que requiere realizar una cierta actividad no puede ser negativo”, “el costo de traslado no debe exceder de $1000 pesos”.

Ejemplo 1: Problema de la mochila

Capacidad de la mochila: 5kg
Objeto tipo 1: 3kg, $10, 2 objetos
Objeto tipo 2: 2kg, $8, 4 objetos

Constantes del problema: (No se pueden cambiar, son fijas)

P_i $precio\ del\ objeto\ i$
w_i $peso\ del\ objeto\ i$
M $capacidad\ en\ peso\ de\ la\ mochila$

Variables del problema: (Son los valores que debemos encontrar)

$X_i = \{0,1\}$ Por cada objeto se debe elegir si se agrega a la mochila o no

Objetivo o medida de desempeño: Maximizar la sumatoria de los precios de los objetos en la mochila

$$Maximizar \sum_i \delta(X_i, P_i), \quad donde\ \delta(X_i, P_i) = \begin{cases} P_i & si\ X_i = 1 \\ 0 & si\ X_i = 0 \end{cases}$$

Restricciones: El peso de los objetos en la mochila no debe sobrepasar la capacidad de la mochila

$$\sum_i \delta(X_i, w_i) \leq M, \quad donde\ \delta(X_i, w_i) = \begin{cases} w_i & si\ X_i = 1 \\ 0 & si\ X_i = 0 \end{cases}$$

2.1 Tipos de optimización

De acuerdo al problema a optimizar

Discreta
- Problema de la mochila
- Problema de las monedas
- Ruta más corta de una ciudad a otra

Continua
- Calcular la función que mejor aproxime una serie de puntos
- Registro de imágenes

De acuerdo al problema a optimizar

Sin restricciones
- Ruta más corta de una ciudad a otra
- Registro de imágenes

Con restricciones
- Problema de la mochila
- Tiempos, Número de Personas (no deben ser negativos)

De acuerdo al método que se utiliza para optimizar

Determinístico
- Búsqueda primero el mejor
- Programación dinámica

Heurístico (se utiliza la probabilidad)
- Algoritmo A*
- Algoritmos genéticos

2.2 Métodos o técnicas utilizadas para optimizar

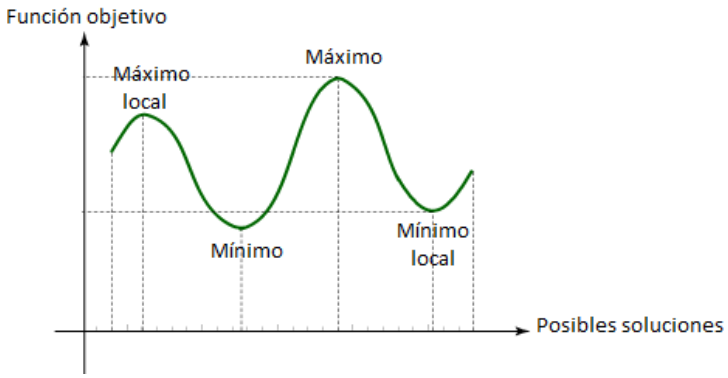
Determinísticos: Son aquellos métodos o técnicas que siempre que se ejecuten, en las mismas condiciones, darán el mismo resultado. Generalmente hacen uso de herramientas y teoremas matemáticos, en Optimización generalmente se hace uso de Cálculo Vectorial, Diferencial y Matemáticas Discretas.

Heurísticos: Según la RAE, heurística viene del griego y significa "hallar" o "inventar", y algunas de las definiciones son "Técnica de la indagación y del descubrimiento", "En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc. Generalmente decimos que un método es heurístico cuando al ejecutarlo varias veces, en las mismas condiciones, puede generar diferentes resultados.

¿Qué son las metaheurísticas?
Del griego "meta" que significa "más allá" y heurística que significa "hallar". Son algoritmos computacionales creados con el propósito de resolver problemas de optimización que generalmente NO pueden ser resueltos mediante técnicas determinísticas. Generalmente estos algoritmos se crean en base a una imitación de la naturaleza como el proceso evolutivo, comportamiento de insectos, comportamiento de los materiales, etc. No se garantizan los resultados, sin embargo presentan una tendencia muy prometedora.

3. Optimización Determinística

¿Qué se quiere lograr?
Encontrar el valor que minimiza o maximiza una función.



Dependiendo de la forma de la función, es que tan fácil o difícil se encuentran esos mínimos o máximos.

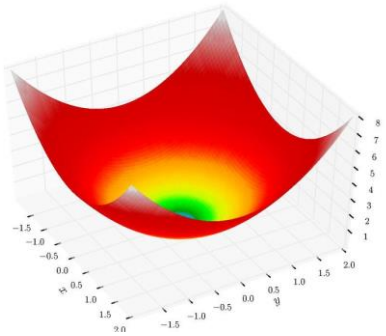
Ejemplos de funciones:

1. Esfera $$f(x) = \sum_{i=1}^{n} x_i^2$$

2. Rosenbrock $$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

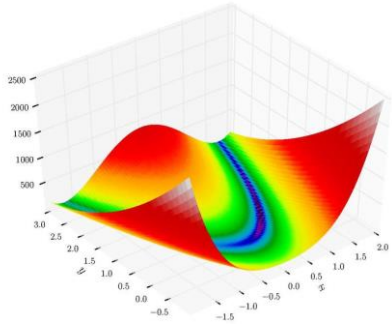
3. Ackley $$f(x) = 20 + e - 20exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right)$$

1. Esfera



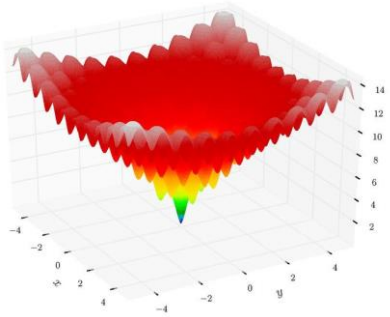
Mínimo: (0,…,0)

2. Rosenbrock



Mínimo: (1,…,1)

3. Ackley



Mínimo: (0,…,0)

3.1 Solución, Gradiente y Hessiano

$Sea\ X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\ y\ f(X)\ una\ función\ vectorial\ f: R^n \rightarrow R$

Solución: $X^*\ es\ el\ valor\ mínimo\ si\ f(X^*) \leq f(X)\ \forall_x$

Gradiente, Jacobiano: $\nabla f(X)$
Representa a la primera derivada, está compuesto por las derivadas parciales de la función respecto a cada variable. Es un vector de tamaño nx1, donde n es el número de variables.

$$\nabla f(X) = \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Hessiano $\nabla^2 f(X)$
Representa a la segunda derivada, está compuesto por las segundas derivadas parciales de la función respecto a cada variable. Es una matriz de tamaño nxn, donde n es el número de variables, por la propiedad de $\frac{\partial^2 f}{\partial xy} = \frac{\partial^2 f}{\partial yx}$ se tiene que $\nabla^2 f(X)$ es una matriz simétrica.

$$\nabla^2 f(X) = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

X^* es un mínimo local si:

- $\left|\left|\nabla f(X^*)\right|\right| = 0$
- $\nabla^2 f(X^*)$ es positiva semidefinida, esto es, $v^T \nabla^2 f(X^*) v \geq 0\ \ \forall v$

Si recordamos de Cálculo, x es un mínimo si:

- $f'(x) = 0$
- $f''(x) > 0$

Ejemplo: Calcular Gradiente y Hessiano

$f(x,y,z) = 2x^3 + 5x^2 y - xz^4 - e^{2y}$

Gradiente

$$\nabla f(X) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} 6x^2 + 10xy - z^4 \\ 5x^2 - 2e^{2y} \\ -4xz^3 \end{bmatrix}$$

Hessiano

$$\nabla^2 f(X) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial xy} & \frac{\partial^2 f}{\partial xz} \\ \frac{\partial^2 f}{\partial yx} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial yz} \\ \frac{\partial^2 f}{\partial zx} & \frac{\partial^2 f}{\partial zy} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix} = \begin{bmatrix} 12x + 10y & 10x & -4z^3 \\ 10x & -4e^{2y} & 0 \\ -4z^3 & 0 & -12xz^2 \end{bmatrix}$$

Actividad: Calcular el Gradiente y el Hessiano de las funciones

1. $f(x,y,z) = x^2y + 3xz^3 + 2z^2$
2. $f(x_1,x_2) = 3x_1^2 + 2x_1x_2 + 4x_2^3$
3. $h(a,b) = a^4b^3 + \frac{\exp(a^3)}{4} + \frac{a^2}{2}$
4. $\Phi(a,b,c,d) = ab^3c + \frac{db^2}{2} - 3a + 14$

3.2 Diferenciación finita

Es una forma de calcular una aproximación al Gradiente y el Hessiano de una función cuando no se puede hacer de forma analítica. La única restricción es que se debe calcular en un punto específico.

Cálculo del Gradiente con Diferenciación Finita

A partir de la definición de la derivada:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Y teniendo en cuenta el gradiente está compuesto por derivadas parciales $\nabla f(X) = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$, se puede calcular una aproximación de cada derivada parcial es con la fórmula:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x+\varepsilon e_i) - f(x)}{\varepsilon} \approx \frac{f(x+\varepsilon e_i) - f(x-\varepsilon e_i)}{2\varepsilon} \approx \frac{f(x) - f(x-\varepsilon e_i)}{\varepsilon}$$

donde:

ε es un escalar pequeño

e_i es el i-ésimo vector unitario, por ejemplo $e_1 = [1,0, \ \ldots \ ,0]^T$

Ejemplo: Cálculo del Gradiente con diferenciación finita

$f(x,y) = x^2 + y^2$, Calcular $\nabla f(2,3)$

Diferencias finitas ($\varepsilon = 0.1$)

$$\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{bmatrix} = \begin{bmatrix}\frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1}\\ \frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1}\end{bmatrix} = \begin{bmatrix}\frac{f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1}\\ \frac{f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1}\end{bmatrix} = \begin{bmatrix}\frac{(2.1^2+3^2) - (2^2+3^2)}{0.1}\\ \frac{(2^2+3.1^2) - (2^2+3^2)}{0.1}\end{bmatrix} = \begin{bmatrix}4.1\\6.1\end{bmatrix}$$

Diferencias analíticas

$$\nabla f\left(\begin{bmatrix}x\\y\end{bmatrix}\right) = \begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{bmatrix} = \begin{bmatrix}2x\\2y\end{bmatrix} \qquad \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix}2(2)\\2(3)\end{bmatrix} = \begin{bmatrix}4\\6\end{bmatrix}$$

Actividad:

1. Calcular el Gradiente, utilizando diferenciación finita de las funciones:
   - $f(x,y) = 2x^2y + 8y^3, \ \nabla f(-2,4)$
   - $\varphi(x_1,x_2,x_3) = 2x_1^4x_2^2 - x_3, \ \nabla\varphi(10{,}25,-50)$
   - $\phi(i,j,k) = 3ij^3 - k^3i^2 + 5, \ \nabla\phi(2,-1,4)$
2. Crear una función que reciba como parámetro una función y un punto específico. Después debe calcular y regresar el vector del Gradiente.

Calculo del Hessiano con Diferenciación Finita

Sabemos que el Hessiano es una matriz compuesta por las segundas derivadas parciales, esto es:

$$\nabla^2 f(X) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Podemos afirmar que $\nabla^2 f(X) \approx \left[\frac{\partial \nabla f(x)}{\partial x_1}, \frac{\partial \nabla f(x)}{\partial x_2}, \ldots, \frac{\partial \nabla f(x)}{\partial x_n} \right]$, es decir, $\nabla^2 f$ está formado por: la derivada del gradiente respecto a la primera variable (vector), más la derivada del gradiente respecto a la segunda variable (vector) y así sucesivamente. Para calcular la derivada parcial del gradiente con respecto a la variable i:

$$\frac{\partial \nabla f(x)}{\partial x_i} \approx \frac{\nabla f(x + \epsilon e_i) - \nabla f(x)}{\epsilon} \approx \frac{\nabla f(x + \varepsilon e_i) - \nabla f(x - \varepsilon e_i)}{2\varepsilon} \approx \frac{\nabla f(x) - \nabla f(x - \varepsilon e_i)}{\varepsilon}$$

Ejemplo: Cálculo del Gradiente con diferenciación finita

$f(x,y) = x^2 + y^2$, Calcular $\nabla^2 f(2,3)$

Diferencias finitas ($\varepsilon = 0.1$)

$$\nabla^2 f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \left[\frac{\partial \nabla f}{\partial x}, \frac{\partial \nabla f}{\partial y}\right] = \left[\frac{\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \quad , \quad \frac{\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1}\right] = \left[\frac{\nabla f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right) - \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \quad , \quad \frac{\nabla f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right) - \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1}\right]$$

$$\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{(2.1^2 + 3^2) - (2^2 + 3^2)}{0.1} \\ \frac{(2^2 + 3.1^2) - (2^2 + 3^2)}{0.1} \end{bmatrix} = \begin{bmatrix}4.1\\6.1\end{bmatrix}$$

$$\nabla f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right) = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.1\\3\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2.1\\3\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.2\\3\end{bmatrix}\right) - f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2.1\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{(2.2^2 + 3^2) - (2.1^2 + 3^2)}{0.1} \\ \frac{(2.1^2 + 3.1^2) - (2.1^2 + 3^2)}{0.1} \end{bmatrix} = \begin{bmatrix}4.3\\6.1\end{bmatrix}$$

$$\nabla f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right) = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2\\3.1\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2\\3.1\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.1\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2\\3.2\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{(2.1^2 + 3.1^2) - (2^2 + 3.1^2)}{0.1} \\ \frac{(2^2 + 3.2^2) - (2^2 + 3.1^2)}{0.1} \end{bmatrix} = \begin{bmatrix}4.1\\6.3\end{bmatrix}$$

$$\nabla^2 f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \left[\frac{\begin{bmatrix}4.3\\6.1\end{bmatrix} - \begin{bmatrix}4.1\\6.1\end{bmatrix}}{0.1}, \frac{\begin{bmatrix}4.1\\6.3\end{bmatrix} - \begin{bmatrix}4.1\\6.1\end{bmatrix}}{0.1}\right] = \left[\frac{\begin{bmatrix}0.2\\0\end{bmatrix}}{0.1}, \frac{\begin{bmatrix}0\\0.2\end{bmatrix}}{0.1}\right] = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$$

Diferencias analíticas

$$\nabla^2 f\left(\begin{bmatrix}x\\y\end{bmatrix}\right) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial xy} \\ \frac{\partial^2 f}{\partial yx} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix} \qquad \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$$

Conviene garantizar que el Hessiano numérico sea simétrico, para esto se hace lo siguiente:

$$H = \frac{H + H^T}{2}$$

Actividad:

1. Calcular el Hessiano, utilizando diferenciación finita de las funciones:

$f(x,y) = 2x^2y + 8y^3,\ \nabla^2 f(-2,4) = ¿?$

$\phi(i,k) = 3i^3 - 5k^2,\ \nabla\phi(2,-1) = ¿?$

2. Crear una función que reciba como parámetro una función y un punto específico. Después debe calcular y regresar el vector del Hessiano.

Aproximación del Gradiente y el Hessiano por un vector utilizando Diferenciación Finita

Gradiente por un vector:

$$\nabla f(x)^T d \approx \frac{f(x+\varepsilon d) - f(x)}{\varepsilon}$$

Es un vector de tamaño 1xn, por un vector de tamaño nx1, por lo que el resultado es un escalar.

Hessiano por un vector:

$$\nabla^2 f(x) d \approx \frac{\nabla f(x) - \nabla f(x - \epsilon d)}{\epsilon}$$

Es una matriz de tamaño nxn, por un vector de tamaño nx1, por lo que el resultado es un vector de tamaño nx1.

Ejemplo: Cálculo del Gradiente con diferenciación finita

$f(x,y) = x^2 + y^2,\quad \nabla f(2,3)^T \begin{bmatrix}5\\1\end{bmatrix} = ¿?,\quad \nabla^2 f(2,3) \begin{bmatrix}5\\1\end{bmatrix} = ¿?$

Diferencias finitas ($\varepsilon = 0.1$)

$$\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)^T \begin{bmatrix}5\\1\end{bmatrix} = \frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}5\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} = \frac{f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} = \frac{(2.5^2 + 3.1^2) - (2^2 + 3^2)}{0.1} = 28.6$$

$$\nabla^2 f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)^T \begin{bmatrix}5\\1\end{bmatrix} = \frac{\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}5\\1\end{bmatrix}\right) - \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} = \frac{\nabla f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right) - \nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} = \frac{\begin{bmatrix}5.1\\6.3\end{bmatrix} - \begin{bmatrix}4.1\\6.1\end{bmatrix}}{0.1} = \frac{\begin{bmatrix}1\\0.2\end{bmatrix}}{0.1} = \begin{bmatrix}10\\2\end{bmatrix}$$

$$\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2\\3\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.1\\3\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{(2.1^2 + 3^2) - (2^2 + 3^2)}{0.1} \\ \frac{(2^2 + 3.1^2) - (2^2 + 3^2)}{0.1} \end{bmatrix} = \begin{bmatrix}4.1\\6.1\end{bmatrix}$$

$$\nabla f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right) = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix} + 0.1\begin{bmatrix}1\\0\end{bmatrix}\right) - f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix} + 0.1\begin{bmatrix}0\\1\end{bmatrix}\right) - f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{f\left(\begin{bmatrix}2.6\\3.1\end{bmatrix}\right) - f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right)}{0.1} \\ \frac{f\left(\begin{bmatrix}2.5\\3.2\end{bmatrix}\right) - f\left(\begin{bmatrix}2.5\\3.1\end{bmatrix}\right)}{0.1} \end{bmatrix} = \begin{bmatrix} \frac{(2.6^2 + 3.1^2) - (2.5^2 + 3.1^2)}{0.1} \\ \frac{(2.5^2 + 3.2^2) - (2.5^2 + 3.1^2)}{0.1} \end{bmatrix} = \begin{bmatrix}5.1\\6.3\end{bmatrix}$$

Diferencias analíticas

$$\nabla f\left(\begin{bmatrix}2\\3\end{bmatrix}\right)^T \begin{bmatrix}5\\1\end{bmatrix} = [2x\ \ 2y] \begin{bmatrix}5\\1\end{bmatrix} = [4\ \ 6] \begin{bmatrix}5\\1\end{bmatrix} = 26 \qquad \nabla^2 f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) \begin{bmatrix}5\\1\end{bmatrix} = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix} \begin{bmatrix}5\\1\end{bmatrix} = \begin{bmatrix}10\\2\end{bmatrix}$$

Actividad:

1. Calcular:

$f(x,y) = 2x^2y + 8y^3,\quad \nabla f(-2,4)^T \begin{bmatrix}3\\1\end{bmatrix} = ¿?,\quad \nabla^2 f(-2,4)^T \begin{bmatrix}3\\1\end{bmatrix} = ¿?$

$\phi(i,k) = 3i^3 - 5k^2,\ \nabla\phi(2,-1,4)^T \begin{bmatrix}5\\2\end{bmatrix} = ¿?,\qquad \nabla\phi(2,-1,4)^T \begin{bmatrix}5\\2\end{bmatrix} = ¿?$

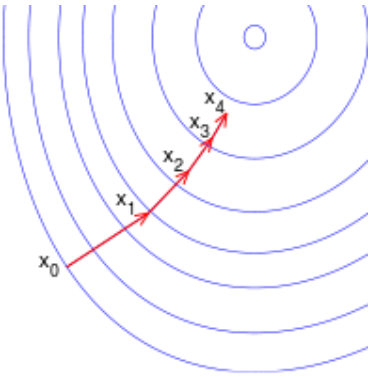
2. Crear un una función que reciba como parámetro una función, un vector (punto específico) y otro vector v, luego debe calcular y regresar el resultado de multiplicar el gradiente por el vector v.
3. Crear un una función que reciba como parámetro una función, un vector (punto específico) y otro vector v, luego debe calcular y regresar el resultado de multiplicar el hessiano por el vector v.

3.3 Descenso del Gradiente

Es un algoritmo que busca el mínimo local en una función. Lo que hace es: dado un punto (o solución) inicial el algoritmo mejora dicha solución en una secuencia de iteraciones que termina cuando no hay progreso o se ha encontrado una solución relativamente buena.

$$x_{k+1} = x_k + \alpha_k \, p_k$$

En cada iteración se debe determinar:
- La dirección hacia donde se debe mover p_k (p_k debe ser un vector unitario)
- El tamaño de paso, el cual indica que tanto se debe mover α_k (α_k debe ser un escalar)



Algoritmo

Recibe como parámetro el punto inicial: X_0
Mientras $\left(k < \text{MaxIteraciones} \;\&\&\; ||\nabla f(X_k)|| > \varepsilon\right)$
    Calcular el tamaño de paso α_k
    Calcular la dirección P_k
    $X_{k+1} = X_k + \alpha_k P_k$
    $k = k + 1$
Fin mientras
Regresar el valor de X_k

Cálcular la dirección P_k

La esencia del Descenso del Gradiente es moverse en cada iteración en la dirección contraria al gradiente. La dirección debe representarse como un vector unitario, para que su magnitud no afecte el tamaño de paso. Se puede calcular de la siguiente forma:

$$P_k = \frac{-\nabla f(X_k)}{||\nabla f(X_k)||}$$

donde $||V|| = \sqrt{V_1^2 + V_2^2 + \cdots + V_n^2}$.

Cálcular el tamaño de paso α_k

Formas de calcular el tamaño de paso:

- **Tamaño de paso constante**, por ejemplo:

$$\alpha_k = 0.01 \quad o \quad \alpha_k = 1x10^{-5}$$

- **Paso de newton,** del Teorema de Taylor se deduce que:

$$f(x_k + \alpha_k p_k) = f(x_k) + \nabla f(x_k)^T(\alpha_k p_k) + \frac{(\alpha_k p_k)^T \nabla^2 f(x)(\alpha_k p_k)}{2} + \cdots$$

Para optimizar con respecto a α_k

$$\frac{d\, f(x_k + \alpha_k p_k)}{d\alpha_k} = \nabla f(x_k)^T p_k + \alpha_k p_k^T \nabla^2 f(x) p_k = 0$$

$$\alpha_k p_k^T \nabla^2 f(x) p_k = -\nabla f(x_k)^T p_k$$

$$\boldsymbol{\alpha_k = \frac{-\nabla f(x_k)^T p_k}{p_k^T \nabla^2 f(x_k) p_k}}$$

- **Backtracking,** pide que se cumpla la condición de Wolfe:

$$f(X_k + \alpha_k P_k) \leq f(X_k) + c\nabla f(X_k)^T P_k \quad \text{con } 0 < c < 1$$

Algoritmo preliminar

Recibe como parámetro: $\alpha > 0, \quad 0 < \rho, c < 1$
$\alpha_k = \alpha$
Repetir hasta $f(X_k + \alpha_k P_k) \leq f(X_k) + c\nabla f(X_k)^T P_k$
$\alpha_k = \rho\alpha_k$
Regresar α_k

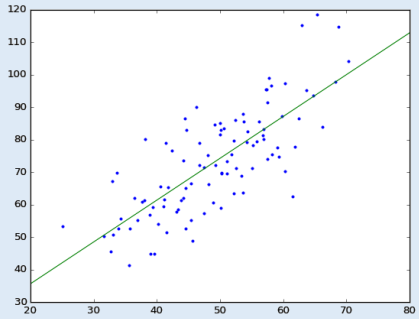
Algoritmo Backtracking con inercia

// a y m son variables globales se inicializan m=0, a=1
Recibe como parámetro: a y m
$\alpha_k = a$
Repetir hasta $f(X_k + \alpha_k P_k) \leq f(X_k) + c_0 \nabla f(X_k)^T P_k$ // Si α_k no cumple, hay que hacerlo más chico
$m = 0$
$\alpha_k = \frac{\alpha_k}{c_1}$
Fin repetir
$m = m + 1$ // Para contar el número de veces que se ha utilizado el mismo α_k
Si $m > c_2$ // Si c_2 veces se ha utilizado el mismo tamaño de paso, hay que hacerlo más grande
$m = 0$
$a = c_3 \alpha_k$
Si no // Guardar el tamaño de paso actual
$a = \alpha_k$
Regresar α_k

Valores típicos

$a = 1, c_0 = 1x10^{-4}, c_1 = 2, c_2 = 5, c_3 = 3$

Actividad: Regresión lineal simple



Línea que se quiere encontrar:

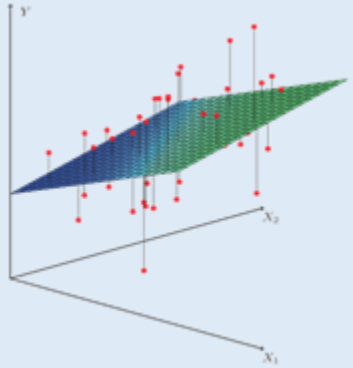
$$y = mx + b$$

La función a optimizar es:

$$\min_{m,b} \theta = \frac{1}{2} \sum_{i=1}^{M} \left(Y^i - (mX^i + b) \right)^2$$

$$\nabla\theta = \begin{bmatrix} \sum_{i=1}^{M} \left(Y^i - \left(mX^i + b \right) \right) \left(-X^i \right) \\ \sum_{i=1}^{M} \left(Y^i - \left(mX^i + b \right) \right) (-1) \end{bmatrix} \qquad \nabla^2\theta = \begin{bmatrix} \sum_{i=1}^{M} \left(X^i \right)^2 & \sum_{i=1}^{M} \left(X^i \right) \\ \sum_{i=1}^{M} \left(X^i \right) & M \end{bmatrix}$$

En General, se puede hacer regresión lineal en N dimensiones. En la siguiente figura se muestra una regresión en 2 dimensiones:



Función del hiperplano:

$$f(X) = B_0 + B_1 X_1 + \cdots + B_N X_N$$

N es la dimensión de los datos

B son los valores a optimizar

La función a optimizar es:

$$\min_{B} \theta(B) = \frac{1}{2} \sum_{i=1}^{M} \left(f\left(X^i \right) - Y^i \right)^2,$$

M es el número de muestras que se tienen

Gradiente

$$\nabla\theta = \begin{bmatrix} \frac{\partial\theta}{\partial B_0} \\ \frac{\partial\theta}{\partial B_1} \\ \vdots \\ \frac{\partial\theta}{\partial B_N} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{M} (f(X^i) - Y^i) \\ \sum_{i=1}^{M} (f(X^i) - Y^i) \left(X_1^i \right) \\ \vdots \\ \sum_{i=1}^{M} (f(X^i) - Y^i) \left(X_N^i \right) \end{bmatrix}$$

Hessiano

$$\nabla^2\theta = \begin{bmatrix} \frac{\partial^2\theta}{\partial B_0^2} & \frac{\partial^2\theta}{\partial B_0 B_1} & \cdots & \frac{\partial^2\theta}{\partial B_0 B_N} \\ \frac{\partial^2\theta}{\partial B_1 B_0} & \frac{\partial^2\theta}{\partial B_1^2} & \cdots & \frac{\partial^2\theta}{\partial B_1 B_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\theta}{\partial B_N B_0} & \frac{\partial^2\theta}{\partial B_N B_1} & \cdots & \frac{\partial^2\theta}{\partial B_N^2} \end{bmatrix} = \begin{bmatrix} M & \sum_{i=1}^{M} X_1^i & \cdots & \sum_{i=1}^{M} X_N^i \\ \sum_{i=1}^{M} X_1^i & \sum_{i=1}^{M} X_1^i X_1^i & \cdots & \sum_{i=1}^{M} X_1^i X_N^i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{M} X_N^i & \sum_{i=1}^{M} X_1^i X_N^i & \cdots & \sum_{i=1}^{M} X_N^{i\,2} \end{bmatrix}$$

Métodos de Búsqueda en Línea

En general, el método del Descenso del Gradiente pertenece a la familia de los métodos de Búsqueda en Línea. El nombre de estos métodos es porque son iterativos de la siguiente forma:

$$x_{k+1} = x_k + \alpha_k \, p_k$$

En cada iteración se debe determinar:
- La dirección hacia donde se debe mover p_k (p_k debe ser un vector unitario)
- El tamaño de paso, el cual indica que tanto se debe mover α_k (α_k debe ser un escalar)

Otras formas para calcular la dirección de descenso es:

- **Newton**
  Para calcular la dirección Pk se resuelve el siguiente sistema de ecuaciones:
  $$\nabla^2 f(X_k) \; P_k = -\nabla f(X_k)$$
  Y después se calcula el vector unitario.

- **Quasi Newton**
  Parecido a Newton, difiere en que utiliza una aproximación al Hessiano.
  $$B_k \approx \nabla f(X_k)^2$$
  Y al igual que en Newton se resuelve el sistema de ecuaciones y después se calcula el vector unitario.
  $$[B_k] \; P_k = -\nabla f(X_k)$$

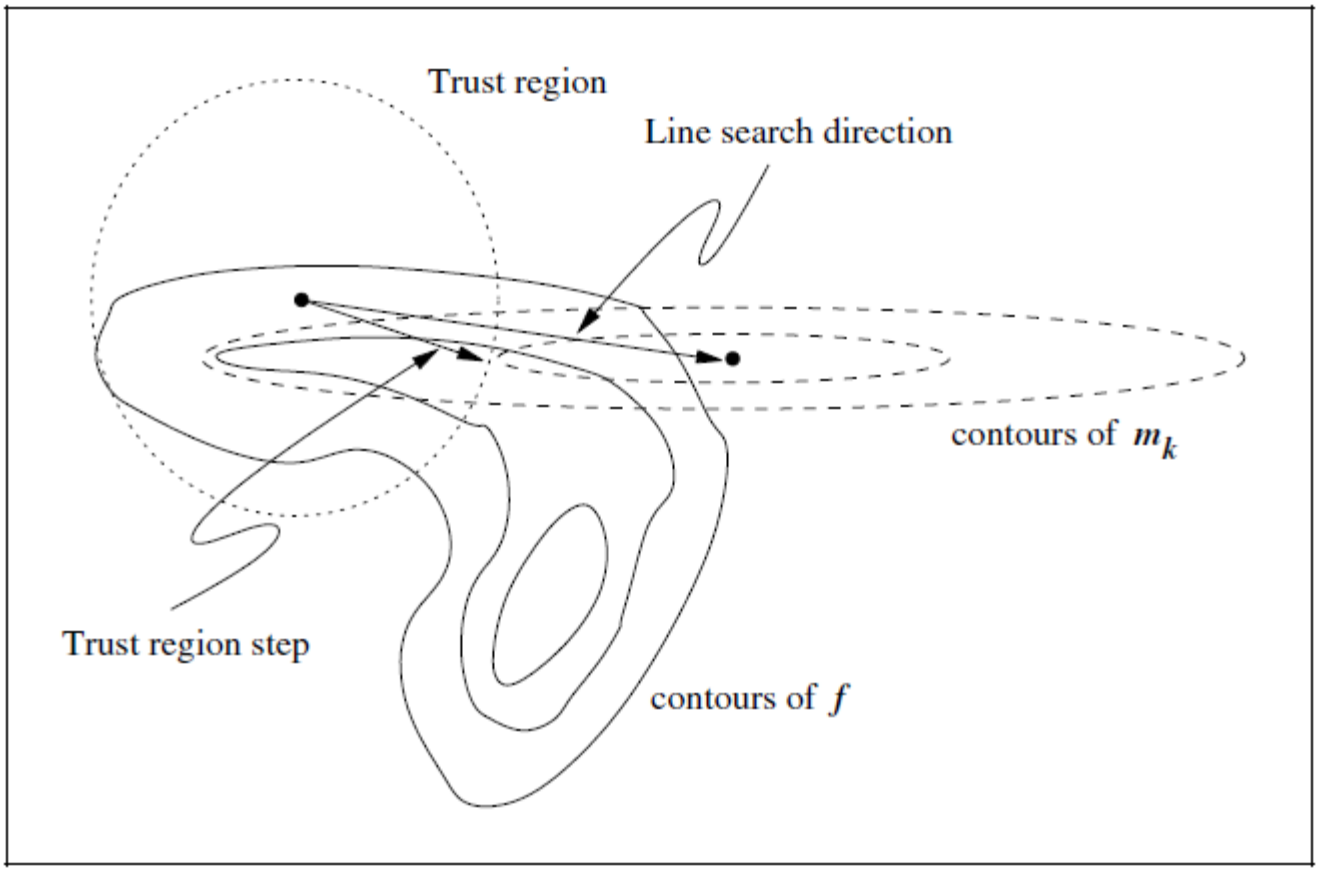
3.4 Métodos de Región de Confianza

Los Métodos de Región de Confianza se utilizan para optimizar funciones; se basan en encontrar el punto óptimo en una serie de iteraciones donde cada punto es mejor al anterior, la estrategia es resolver en cada iteración un modelo que sea más simple de resolver y limitar el paso a la región donde el modelo ajuste bien a la función.

Sea k la iteración actual y x_k el punto actual, entonces el modelo es:

$$m_k(\mathrm{p}) = \mathrm{f_k} + \nabla \mathrm{f_k}^{\mathrm{T}}\mathrm{p} + \frac{1}{2}\mathrm{p}^{\mathrm{T}}\nabla^2\mathrm{f_k}\mathrm{p}$$

Donde: $\mathrm{B_k} = \nabla^2 \mathrm{f_k} + \lambda I$, λ pequeña (probar 0.1)



Lo que se hace en este tipo de métodos es:

Dado un X_k

1. Construir un modelo $m_k(\mathrm{p})$.
2. Resolver el problema $\min_{p \in \mathbb{R}^n} m_k(\mathrm{p})$.
3. Truncar el paso dependiendo de la eficiencia del modelo, y actualizar X_k si es pertinente.

Y esto se itera hasta que converge

La medida o razón de eficiencia del modelo está dada por

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} = \frac{\text{Reducción de la función}}{\text{Reducción del modelo}}$$

Para que la modificación del punto sea "segura", se establece un radio en cada iteración Δ_k donde el modelo es válido. Según el tamaño del radio, el incremento P_k se calcula de la siguiente manera:

$$P_k^B = -\mathrm{B_k}^{-1}\nabla \mathrm{f_k}$$
$$P_k^U = -\frac{\nabla \mathrm{f_k}}{\|\nabla \mathrm{f_k}\|}$$

Si $\left|\left|P_k^B\right|\right| \leq \Delta_k$, entonces $P_k = P_k^B$
Si no, $P_k = \Delta_k\, P_k^U$

El radio Δ_k se ajusta en cada iteración. Además, el punto se actualiza sólo si el modelo "ajusta" bien al problema con el radio dado.

Dado un $\bar{\Delta} > 0$ (radio máximo), $\Delta_0 \in (0, \bar{\Delta}), \eta \in \left[0, {}^1/_4\right]$
Entonces para cada k=0,1,2,3…

1. Obtener P_k resolviendo
   $P_k^B = -\mathrm{B_k}^{-1}\nabla \mathrm{f_k}$
   $P_k^U = -\frac{\nabla \mathrm{f_k}}{\|\nabla \mathrm{f_k}\|}$

   Si $\left|\left|P_k^B\right|\right| \leq \Delta_k$
   $P_k = P_k^B$
   Si no, $P_k = \Delta_k\, P_k^U$

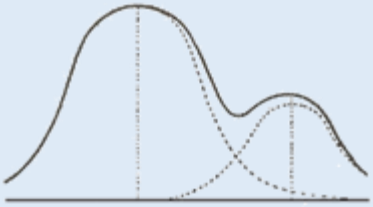
2. Calcular ρ_k

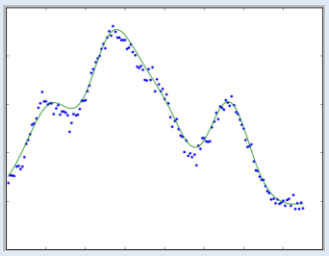
$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

3. Evaluar el modelo y modificar Δ_k
   Si $\rho_k < {}^1/_4$
   $\Delta_{k+1} = \eta\Delta_k$
   Si no
   Si $\rho_k > {}^3/_4$ y $\|P_k\| = \Delta_k$
   $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$
   Si no
   $\Delta_{k+1} = \Delta_k$
4. Actualizar X_k sólo en caso de que el modelo funcione
   Si $\rho_k > {}^1/_4$
   $X_{k+1} = X_k + P_k$
   Si no
   $X_{k+1} = X_k$

Actividad: Aproximación de función con Suma de Gaussianas

La suma de Gaussianas sirve para calcular funciones mediante (como su nombre lo dice) la suma de varias funciones Gaussianas.





La función de la suma de Gaussianas es:

$$f(x) = \sum_{k=1}^{M} B_k \exp\left(-\frac{1}{2\sigma^2}(x - C_k)^2\right)$$

Donde M es el número de Gaussianas
B_k la altura de la Gaussiana k
C_k representa el centro de la Gaussiana k

Dada una serie de puntos X,Y (obtenidos de simulaciones de señales) se deben encontrar las M Gaussianas que al sumarlas aproximen mejor la función. Para encontrar las Gaussianas óptimas se debe resolver el siguiente problema:

$$\min_{B_k's,\ C_k's} \sum_{x=1}^{N}\left[y - \sum_{k=1}^{M} B_k \exp\left(-\frac{1}{2\sigma^2}(x - C_k)^2\right)\right]^2$$

Donde N.- número de puntos M.- número de Gaussianas
B_k.- la altura de la Gaussiana k C_k.- representa el centro de la Gaussiana k
En este caso $\sigma = 10$ constante

Gradiente

$$\begin{bmatrix} \dfrac{d}{dB_k} = \displaystyle\sum_{x=1}^{N}\left[-2Exp(C_k)\right][Fun] \\ \dfrac{d}{dC_k} = \displaystyle\sum_{x=1}^{N}\left[-\frac{2}{\sigma^2}(x - C_k)\right]\left[B_k Exp(C_k)\right][Fun] \end{bmatrix}$$

Hessiano

$$\begin{bmatrix} \dfrac{d}{dB_kB_j} = \displaystyle\sum_{x=1}^{N}[2Ex(C_k)][Ex(C_j)] \\ \dfrac{d}{dB_kC_k} = \dfrac{d}{dC_kB_k} = \displaystyle\sum_{x=1}^{N}\left[2B_k\frac{(x - C_k)}{\sigma^2}[Ex(C_k)]^2 - 2\frac{(x - C_k)}{\sigma^2}[Ex(C_k)][Fu]\right] \\ \dfrac{d}{dB_kC_j} = \dfrac{d}{dC_jB_k} = \displaystyle\sum_{x=1}^{N}[2Ex(C_k)][Ex(C_j)] \\ \dfrac{d}{dC_kC_j} = \displaystyle\sum_{x=1}^{N}\left[\frac{2}{\sigma^4}(x - C_k)(x - C_j)\right][B_jEx(C_j)][B_kEx(C_k)] \\ \dfrac{d}{dC_kC_k} = \displaystyle\sum_{x=1}^{N}\left[\left[\frac{2}{\sigma^2}\right][B_kEx(C_k)][Fu] - \left[\frac{2}{\sigma^4}(x - C_k)^2\right][B_kEx(C_k)][Fu] + \left[\frac{2}{\sigma^4}(x - C_k)^2\right][B_kEx(C_k)]^2\right] \end{bmatrix}$$

Donde:

$$Ex(C_l) = \exp\left(-\tfrac{1}{2\sigma^2}(x - C_l)^2\right) \quad Fu = \left[y - \textstyle\sum_{k=1}^{M} B_kEx(C_k)\right]$$

Parámetros:

$\bar{\Delta} = \Delta_0 = 0.1$
$\eta = 0.05$

3.5 Gradiente Conjugado

El Gradiente Conjugado es un método iterativo para minimizar funciones cuadráticas convexas que tienen la siguiente forma:

$$\min_x f(x) = \frac{1}{2}x^T Ax - x^T b$$

donde $A \in R^{nxn}$, $\mathrm{x}, \mathrm{b} \in R^n$. A es una matriz simétrica positiva definida.

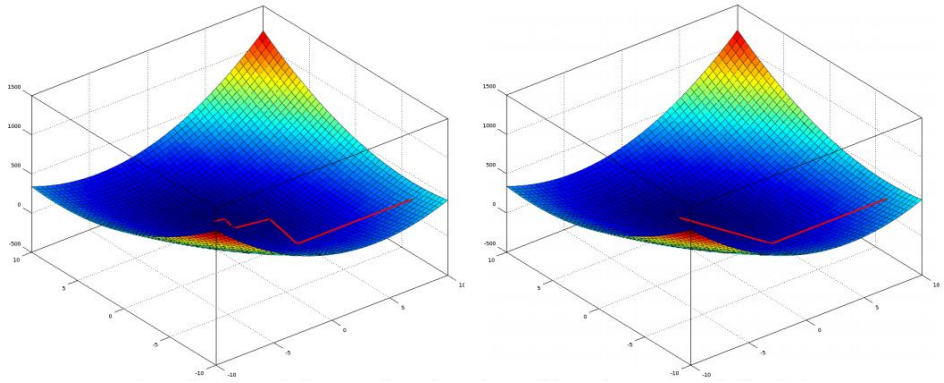


Figura 1 Descenso del Gradiente a la izquierda, Gradiente Conjugado a la derecha. (www.cimat.mx/~miguelvargas)

Como x es nuestra variable, podemos derivar e igualar a 0 para encontrar la solución:

$$\nabla f(x) = Ax - b = 0$$
$$Ax = b$$

Como se puede ver en la ecuación anterior, el método del Gradiente Conjugado se puede utilizar también como un método numérico para resolver sistemas de ecuaciones lineales cuando A es no invertible.

El Gradiente y el Hessiano se pueden calcular fácilmente:

$$\nabla f(x) = Ax - b$$
$$\nabla^2 f(x) = A$$

Al igual que el método Descenso del Gradiente es un método iterativo de la siguiente forma:

$$X_{k+1} = X_k + \alpha_k P_k$$

en cada iteración se debe determinar:

- La dirección hacia donde se debe mover P_k (P_k debe ser un vector unitario)
- El tamaño de paso, el cual indica que tanto se debe mover α_k

Cálculo del tamaño de paso

$\alpha_k = \min_{\alpha_k} f(x_k + \alpha_k P_k)$

Si $f(x) = \frac{1}{2}x^T Ax - x^T b$ entonces

$$f(x_k + \alpha_k P_k) = \frac{1}{2}(x_k + \alpha_k P_k)^T A(x_k + \alpha_k P_k) - (x_k + \alpha_k P_k)^T b$$
$$= \frac{1}{2}x_k^T Ax_k + \frac{1}{2}\alpha_k x_k^T AP_k + \frac{1}{2}\alpha_k P_k^T Ax_k + \frac{1}{2}\alpha_k^2 P_k^T AP_k - x_k^T b - \alpha_k P_k^T b$$
$$= \frac{1}{2}x_k^T Ax_k + \alpha_k P_k^T Ax_k + \frac{1}{2}\alpha_k^2 P_k^T AP_k - x_k^T b - \alpha_k P_k^T b$$

Al derivar e igualar a 0

$$\frac{\partial f(x_k + \alpha_k P_k)}{\partial \alpha_k} = P_k^T A x_k + \alpha_k P_k^T A P_k - P_k^T b = 0$$

$$\alpha_k P_k^T A P_k = P_k^T b - P_k^T A x_k$$

$$\alpha_k = \frac{P_k^T b - P_k^T A x_k}{P_k^T A P_k}$$

$$\alpha_k = \frac{-(P_k^T A x_k - P_k^T b)}{P_k^T A P_k}$$

$$\alpha_k = \frac{-P_k^T (A x_k - b)}{P_k^T A P_k}$$

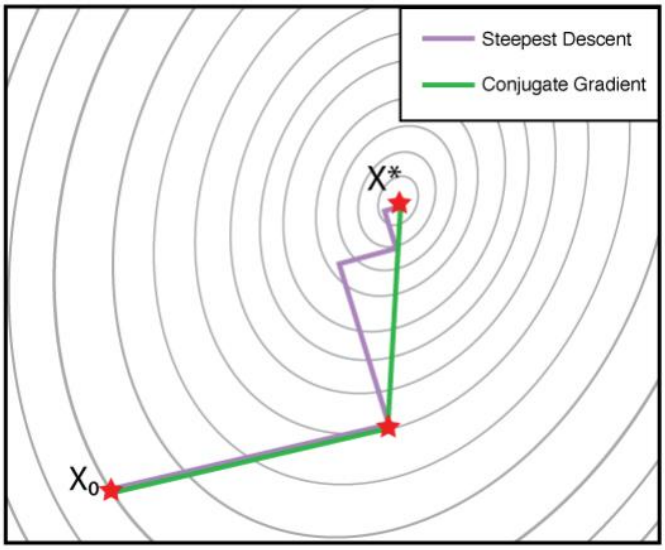
$$\boldsymbol{\alpha_k = \frac{-P_k^T g_k}{P_k^T A P_k}}$$

Cálculo de la dirección

La idea es utilizar direcciones conjugadas como direcciones de descenso para **llegar a lo más en n pasos** al punto óptimo (n es la dimensión del vector de soluciones).

Se utilizan direcciones A conjugadas:

$$v_1^T A v_2 = 0$$



Entonces, si las direcciones son conjugadas:

$$P_k^T A P_{k+1} = 0$$

Luego, cada dirección puede ser una combinación lineal del gradiente y la dirección anterior:

$$\boldsymbol{P_{K+1} = -g_{k+1} + \beta P_k}$$

Entonces:

$$P_k^T A P_{k+1} = 0$$

$$P_k^T A(-g_{k+1} + \beta P_k) = 0$$

$$-P_k^T A g_{k+1} + \beta P_k^T A P_k = 0$$

$$\beta P_k^T A P_k = P_k^T A g_{k+1}$$

$$\boldsymbol{\beta = \frac{P_k^T A g_{k+1}}{P_k^T A P_k}}$$

Algoritmo de Gradiente Conjugado (Versión inicial)

Recibe como parámetro: X_0 y G_0
$k = 0, P_0 = -G_0$
Mientras $(k < \text{MaxIteraciones} \;\&\&||g_k|| > \varepsilon)$

$$\alpha_k = \frac{-P_k^T g_k}{P_k^T A P_k}$$
$$X_{k+1} = X_k + \alpha_k P_k$$
$$g_{k+1} = AX_{k+1} - b$$
$$\beta_{k+1} = \frac{P_k^T A g_{k+1}}{P_k^T A P_k}$$
$$P_{k+1} = -g_{k+1} + \beta_{k+1} P_k$$
$$k = k + 1$$

Fin mientras
Regresar el valor de X_k

Optimizando el cálculo del gradiente

$$g_{k+1} = Ax_{k+1} - b$$
$$g_{k+1} = A(x_k + \alpha_k P_k) - b$$
$$g_{k+1} = Ax_k + \alpha_k A P_k - b$$
$$\mathbf{g_{k+1} = g_k + \alpha_k A P_k}$$

Optimizando el cálculo de β

$$g_{k+1} = g_k + \alpha_k A P_k \text{ entonces } AP_k = \frac{g_{k+1} - g_k}{\alpha_k}$$

Luego

$$\beta_{k+1} = \frac{P_k^T A g_{k+1}}{P_k^T A P_k} = \frac{g_{k+1}^T A P_k}{P_k^T A P_k} = \frac{g_{k+1}^T \left(\frac{g_{k+1} - g_k}{\alpha_k}\right)}{P_k^T \left(\frac{g_{k+1} - g_k}{\alpha_k}\right)}$$
$$\beta_{k+1} = \frac{g_{k+1}^T (g_{k+1} - g_k)}{P_k^T (g_{k+1} - g_k)} = \frac{g_{k+1}^T g_{k+1} - g_{k+1}^T g_k}{P_k^T g_{k+1} - P_k^T g_k}$$

Si las direcciones son conjugadas $P_{k+1}^T P_k = 0, P_k \cong g_k$

$$\beta_{k+1} = \frac{g_{k+1}^T g_{k+1} - \cancel{g_{k+1}^T g_k}}{\cancel{P_k^T g_{k+1}} - P_k^T g_k}$$

$$\boldsymbol{\beta_{k+1} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}}$$

Algoritmo de Gradiente Conjugado (Versión práctica)

Recibe como parámetro: $X_0, g_0 = \nabla f(X_0), \quad A = \nabla^2 f(X_0)$
$k = 0, \; P_0 = -g_0$
Mientras $(k < MaxIteraciones \;\&\&||g_k|| > \varepsilon)$

$$\alpha_k = \frac{-P_k^T g_k}{P_k^T A P_k}$$
$$X_{k+1} = X_k + \alpha_k P_k$$
$$g_{k+1} = g_k + \alpha_k A P_k$$
$$\beta_{k+1} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$$
$$P_{k+1} = -g_{k+1} + \beta_{k+1} P_k$$
$$k = k + 1$$

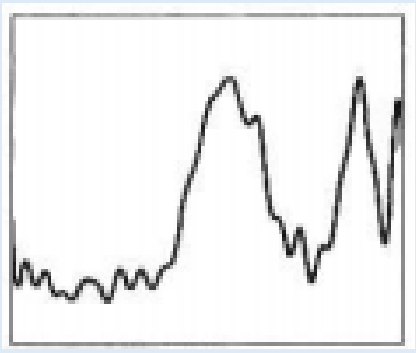
Fin mientras
Regresar el valor de X_k

Actividad: Suavizado de señales utilizando Gradiente Conjugado

Si definimos:
Y : es un vector con la señal de entrada con ruido
Y^c: es un vector con la señal de salida suavizada (debe ser calculada por el algoritmo)

Y: señal con ruido

Y^c: señal suavizada



La función a optimizar es:

$$\min_{Y^c \in \mathbb{R}} F(X) = \frac{1}{2}\left[\sum_{i=1}^{n}(Y_i^c - Y_i)^2 + \lambda \sum_{i=2}^{n}(Y_i^c - Y_{i-1}^c)^2\right]$$

Se recomienda utilizar un valor de $\lambda = 4$. Se pueden utilizar las derivadas analíticas:

$$\nabla F(X) = \left[\frac{\delta F}{\delta Y_i^c}\right] = [\, Y_i^c - Y_i + \lambda(Y_i^c - Y_{i-1}^c)|_{i>1} + \lambda(Y_i^c - Y_{i-1}^c)|_{i<n}]$$

$$\nabla^2 F(X) = \begin{bmatrix} 1+\lambda & -\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda & 1+2\lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 1+2\lambda & -\lambda & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & -\lambda & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1+2\lambda & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda & 1+2\lambda & -\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 1+\lambda \end{bmatrix}$$

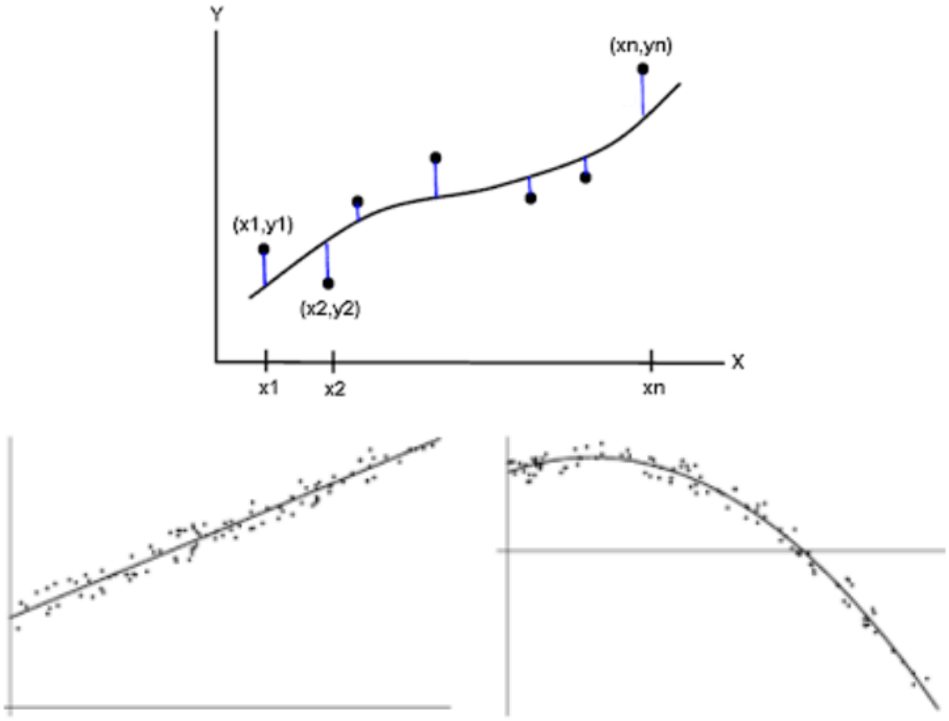
$$\nabla^2 F(X) * Z = \{\; Z_i + \lambda(Z_i - Z_{i-1})|_{i>1} + \lambda(Z_i - Z_{i+1})|_{i<n} \;\}_{i=1,2,\ldots}$$

Se deben calcular los valores para suavizar cuatro señales y entregar un reporte de resultados donde contenga:

- La gráfica de las señales originales
- Las gráficas de las señales suavizadas (utilizando Gradiente Conjugado)

3.6 Mínimos Cuadrados (Least squares)

En algunas ocasiones es necesario encontrar una función NO LINEAL que modele la respuesta de un fenómeno observado, por ejemplo:



Para resolver este problema primero se debe fijar un modelo, por ejemplo una parábola, un polinomio, etc., y después ajustar los N parámetros de dicho modelo utilizando optimización. Por lo tanto el problema a optimizar es:

$$\min_B \theta(B) = \frac{1}{2} \sum_{i=1}^{M} \left(f(X^i) - Y^i\right)^2, \qquad r_i(B) = f(X^i) - Y^i$$

Sabemos que existen M residuos, uno por cada muestra, por lo tanto se define el vector de residuos $r(B)$ como:

$$r(B) = \begin{bmatrix} r_1(B) \\ r_2(B) \\ \vdots \\ r_M(B) \end{bmatrix}$$

Así como el vector Jacobiano $J(B)$, con las derivadas parciales de los residuos respecto a cada parámetro:

$$J(B) = \left[\frac{\partial r_j}{\partial B_i}\right]_{\substack{j=1,2,\dots,M \\ i=1,2,\dots,N}} = \begin{bmatrix} \nabla r_1(B)^T \\ \nabla r_2(B)^T \\ \vdots \\ \nabla r_M(B)^T \end{bmatrix}_{MxN} = \begin{bmatrix} \frac{\partial r_1}{\partial B_1} & \frac{\partial r_1}{\partial B_2} & \cdots & \frac{\partial r_1}{\partial B_N} \\ \frac{\partial r_2}{\partial B_1} & \frac{\partial r_2}{\partial B_2} & \cdots & \frac{\partial r_2}{\partial B_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial B_1} & \frac{\partial r_M}{\partial B_2} & \cdots & \frac{\partial r_M}{\partial B_N} \end{bmatrix}_{MxN}$$

Entonces, se pueden calcular el Gradiente y el Hessiano de la siguiente manera:

$$\min_B \theta(B) = \frac{1}{2} \sum_{i=1}^{M} (r_i(B))^2$$

- Gradiente

$$\nabla\theta = \sum_{i=1}^{M} \nabla r_i(B) r_i(B) = J(B)^T r(B) = \begin{bmatrix} \frac{\partial r_1}{\partial B_1} & \frac{\partial r_1}{\partial B_2} & \cdots & \frac{\partial r_1}{\partial B_N} \\ \frac{\partial r_2}{\partial B_1} & \frac{\partial r_2}{\partial B_2} & \cdots & \frac{\partial r_2}{\partial B_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial B_1} & \frac{\partial r_M}{\partial B_2} & \cdots & \frac{\partial r_M}{\partial B_N} \end{bmatrix}^T_{NxM} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_M \end{bmatrix}_{Mx1}$$

- Hessiano

$$\nabla^2\theta = \sum_{i=1}^{M} \nabla r_i(B) \nabla r_i(B) + \sum_{i=1}^{M} r_i(B) \nabla^2 r_i(B) = J(B)^T J(B) + \sum_{i=1}^{M} r_i(B) \nabla^2 r_i(B)$$

$$J(B)^T J(B) = \begin{bmatrix} \frac{\partial r_1}{\partial B_1} & \frac{\partial r_1}{\partial B_2} & \cdots & \frac{\partial r_1}{\partial B_N} \\ \frac{\partial r_2}{\partial B_1} & \frac{\partial r_2}{\partial B_2} & \cdots & \frac{\partial r_2}{\partial B_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial B_1} & \frac{\partial r_M}{\partial B_2} & \cdots & \frac{\partial r_M}{\partial B_N} \end{bmatrix}^T_{NxM} \begin{bmatrix} \frac{\partial r_1}{\partial B_1} & \frac{\partial r_1}{\partial B_2} & \cdots & \frac{\partial r_1}{\partial B_N} \\ \frac{\partial r_2}{\partial B_1} & \frac{\partial r_2}{\partial B_2} & \cdots & \frac{\partial r_2}{\partial B_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial B_1} & \frac{\partial r_M}{\partial B_2} & \cdots & \frac{\partial r_M}{\partial B_N} \end{bmatrix}_{MxN}$$

$$\sum_{i=1}^{M} r_i(B) \nabla^2 r_i(B) = \sum_{i=1}^{M} r_i(B) \begin{bmatrix} \frac{\partial^2 r_i}{\partial B_1^2} & \frac{\partial^2 r_i}{\partial B_1 \partial B_2} & \cdots & \frac{\partial^2 r_i}{\partial B_1 \partial B_N} \\ \frac{\partial^2 r_2}{\partial B_1 \partial B_2} & \frac{\partial^2 r_2}{\partial B_2^2} & \cdots & \frac{\partial^2 r_2}{\partial B_N \partial B_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 r_M}{\partial B_N \partial B_1} & \frac{\partial^2 r_M}{\partial B_N \partial B_2} & \cdots & \frac{\partial^2 r_M}{\partial B_N^2} \end{bmatrix}_{NxN}$$

Método de Gauss – Newton para resolver Mínimos Cuadrados NO Lineales

En Gauss – Newton se calcula el Hessiano como una aproximación utilizando sólo:

$$\nabla^2\theta \approx J(B)^T J(B)$$

En base al ahorro computacional de calcular los hessianos de cada uno de los residuos: $\nabla^2 r_i(B)$ en el segundo término:$\nabla^2\theta = J(B)^T J(B) + \sum_{i=1}^{M} r_i(B) \nabla^2 r_i(B)$, además de que el término de primer orden generalmente domina al término de segundo orden.

Algoritmo General de Mínimos Cuadrados

Recibe como parámetro: B_0,
$k = 0$
Mientras $(k < \text{MaxIteraciones} \;\&\&||g_k|| > \varepsilon)$
Calcular el Hessiano H_k
$H_k = J(B)^T J(B)$ ó $H_k = J(B)^T J(B) + \sum_{i=1}^{M} r_i(B)\nabla^2 r_i(B)$
Calcular la dirección P_k resolviendo el siguiente sistema de ecuaciones

$$H_k\ P_k = -G_k$$
$$P_k = \frac{P_k}{||P_k||}$$

Calcular α_k con alguno de los métodos propuestos en Descenso del Gradiente
$X_{k+1} = X_k + \alpha_k P_k$
$k = k + 1$
Fin mientras
Regresar el valor de X_k

Actividad: Regresión NO LINEAL
Se tiene la siguiente información

Hydrogen	N-Pentane	Isopentane	Reaction Rate
470	300	10	8.55
285	80	10	3.79
470	300	120	4.82
470	80	120	0.02
470	80	10	2.75
100	190	10	14.39
100	80	65	2.54
470	190	65	4.35
100	300	54	13.00
100	300	120	8.50
100	80	120	0.05
285	300	10	11.32
285	190	120	3.12

Y se cree que se puede crear una función como la siguiente para modelar la Taza de Reacción de acuerdo a los parámetros Hydrogen, N-Pentane, Isopentane:

$$R = \frac{B_1 Hydrogen - Isopentane/B_5}{1 + B_2 Hydrogen + B_3 NPentane + B_4 Isopentane}$$

3.7 Programación Lineal

La Programación lineal es una técnica para resolver problemas de optimización sobre una función lineal con restricciones lineales.

En la programación lineal hay tres elementos clave:

- **Función Objetivo**. Que es el modelo del problema a resolver, por ejemplo: ¿cómo se calcula la utilidad (maximización)? ¿cómo se calculan los costos (minimización)?
- **Variables**. Representan los valores que pueden modificarse, y a partir de los cuales podemos encontrar una solución óptima.
- **Restricciones**. Es todo aquello que limita la libertad de los valores, por ejemplo: ¿Cuántos empleados como máximo puedo tener? ¿Cuánta materia prima tengo?

Modelado

La fábrica de Hilados y Tejidos "X" requiere fabricar dos tejidos de calidad diferente M y N; se dispone de 500 Kg de hilo a, 300 Kg de hilo b y 108 Kg de hilo c. Para obtener un metro de M diariamente se necesitan 125 gr de a, 150 gr de b y 72 gr de c; para producir un metro de N por día se necesitan 200 gr de a, 100 gr de b y 27 gr de c. El M se vende a $4000 el metro y el N se vende a $5000 el metro. Si se debe obtener el máximo beneficio, ¿cuántos metros de M y N se deben fabricar?

Variables:
Xm -> metros de M que se deben fabricar
Xn -> metros de N que se deben fabricar

Restricciones:
Límite de a, 125Xm + 200Xn <= 500,000
Límite de b, 150Xm + 100Xn <= 300,000
Límite de c, 72Xm + 27Xn <= 108,000
No puedo hacer metros negativos, Xn, Xm>0

Función objetivo:
¿Minimizar o maximizar? ¿Cómo obtengo la ganancia?
Maximizar G = 4000Xm + 5000Xn

Por lo tanto, el problema es:

Maximizar G = 4000Xm + 5000Xn
Sujeto a
125Xm + 200Xn <= 500,000
150Xm + 100Xn <= 300,000
72Xm + 27Xn <= 108,000

Actividad: Modelado de problemas de programación lineal

1. Un herrero con 80 Kg. de acero y 120 Kg. de aluminio quiere hacer bicicletas de paseo y de montaña que quiere vender, respectivamente a 20.000 y 15.000 pesos cada una para sacar el máximo beneficio. Para la de paseo empleará 1 kg de acero y 3 kg de aluminio, y para la de montaña 2 kg de ambos metales. ¿Cuántas bicicletas de paseo y de montaña deberá fabricar para maximizar las utilidades?

2. Una compañía fabrica y venden dos modelos de lámpara L1 y L2. Para su fabricación se necesita un trabajo manual de 20 minutos para el modelo L1 y de 30 minutos para el L2; y un trabajo de máquina de 10 minutos para L1 y de 10 minutos para L2. Se dispone para el trabajo manual de 100 horas al mes y para la máquina 80 horas al mes. Sabiendo que el beneficio por unidad es de 15 y 10 euros para L1 y L2, respectivamente, planificar la producción para obtener el máximo beneficio.

3. En una granja de pollos se da una dieta, para engordar, con una composición mínima de 15 unidades de una sustancia A y otras 15 de una sustancia B. En el mercado sólo se encuentra dos clases de compuestos: el tipo X con una composición de una unidad de A y 5 de B, y el otro tipo, Y, con una composición de cinco unidades de A y una de B. El precio del tipo X es de 10 euros y del tipo Y es de 30 €. ¿Qué cantidades se han de comprar de cada tipo para cubrir las necesidades con un coste mínimo?

4. Con el comienzo del curso se va a lanzar unas ofertas de material escolar. Unos almacenes quieren ofrecer 600 cuadernos, 500 carpetas y 400 bolígrafos para la oferta, empaquetándolo de dos formas distintas; en el primer bloque pondrá 2 cuadernos, 1 carpeta y 2 bolígrafos; en el segundo, pondrán 3 cuadernos, 1 carpeta y 1 bolígrafo. Los precios de cada paquete serán 6.5 y 7 €, respectivamente. ¿Cuántos paquetes le convienen poner de cada tipo para obtener el máximo beneficio?

5. Una escuela prepara una excursión para 400 alumnos. La empresa de transporte tiene 8 autobuses de 40 plazas y 10 de 50 plazas, pero sólo dispone de 9 conductores. El alquiler de un autocar grande cuesta 800 € y el de uno pequeño 600 €. Calcular cuántos autobuses de cada tipo hay que utilizar para que la excursión resulte lo más económica posible para la escuela.

4. Computo Evolutivo

Computación Evolutiva es un área de investigación dentro de las Ciencias Computacionales que busca resolver problemas, y como su nombre lo indica se inspira en el proceso natural de evolución propuesto por Charles Darwin [1].

Los elementos de la computación evolutiva son:

- Entorno: El problema que se pretende resolver.
- Individuo: Representa una posible solución al problema.
- Aptitud: Es la calidad de la solución de un individuo específico.

Evolución	Problema
Entorno	↔ Problema
Individuo	↔ Solución
Aptitud	↔ Calidad

La idea en común de todos los algoritmos evolutivos es la misma: primero se genera una población de individuos o soluciones de forma aleatoria y se calcula la aptitud o calidad de cada uno de los individuos. Después de forma repetitiva, donde cada repetición representa una generación: (1) se seleccionan los individuos con mayor aptitud con el objetivo de mejorar en cada generación la aptitud en promedio de la población; (2) se recombinan dos o más individuos (también llamados padres) para formar otros nuevos individuos (hijos) con el objetivo de diversificar a la población; (3) se mutan algunos de los individuos con el objetivo de explorar nuevas soluciones y (4) se calculan las aptitudes de los nuevos individuos. Este proceso es iterado hasta que exista un individuo con suficiente calidad o que los limites computacionales se hayan alcanzado.

En la siguiente figura se muestra un esquema general de un algoritmo evolutivo.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Los componentes de un algoritmo evolutivo son:

- Representación (definición de los individuos)
- Evaluación de la función (*fitness*)
- Población
- Selección de los padres
- Operadores de variación: reproducción y mutación
- Selección de los sobrevivientes
- Inicialización
- Condición de paro

Representación (definición de los individuos)

El primer paso para definir un algoritmo evolutivo es ligar el "mundo real" con el "mundo evolutivo", es decir, encontrar la forma de representar las soluciones. Definiciones:
Fenotipo (*Phenotype*): Una posible solución en el contexto del problema original
Genotipo (*Genotypes*): Un individuo del algoritmo evolutivo

Ejemplo:

Si tenemos un problema de optimización donde se quiere optimizar una variable entera x, y la representación será en binario con 5 bits entonces tenemos que:
Fenotipo: 4 Genotipo: 00100

La representación consiste en mapear los fenotipos a genotipos.

Evaluación de la función (*fitness*)

El rol de la evaluación de la función es medir que tan buena es cada solución, o dicho de otra forma, que tan apto es un individuo. Para esto se debe plantear una función objetivo que reciba como parámetro el fenotipo y dé como resultado un valor numérico con la aptitud (*fitness*) de la solución.

Ejemplo:

Suponga que se quiere optimizar el problema $\min_x x^2$, y nuestro individuo es fenotipo: 4, genotipo: 00100.
Entonces su aptitud es: 4*4 = 16

La función de evaluación es comúnmente llamada: *fitness function*.

Población

El rol de la población es guardar (la representación de) las soluciones encontradas hasta el momento. Es decir, la población es un conjunto de genotipos.

Ejemplo:

Suponga que se quiere optimizar el problema $\min_x x^2$, y la representación de cada individuo se define con 5 bits, una posible población podría ser:

Población
10010
11111
00100
00001
01010

Selección de los padres

Consiste en elegir individuos de la población en base a su aptitud para con ellos crear uno o varios individuos nuevos. La selección de los padres es típicamente probabilística, donde los individuos con mayor *fitness* tienen mayor probabilidad de ser elegidos.

Una de las formas más comunes para la selección es torneo binario, el cual consiste en seleccionar aleatoriamente dos individuos y quedarse con el individuo con mejor *fitness*.

Operadores de variación: reproducción y mutación

El objetivo de los operadores de variación es crear individuos nuevos a partir de los individuos anteriores.

Reproducción (*recombination* or *crossover*): Este operador toma el genotipo de dos individuos padres, los combina, y genera uno o más para individuos hijos. El principio detrás de la reproducción es simple: mediante el apareamiento de dos individuos con diferentes características se puede producir un hijo que combina las características de los individuos padres, dicho principio tiene un soporte fuerte: por milenios se ha aplicado satisfactoriamente a plantas y animales.

Mutación (*mutation*): Este operador es aplicado a un solo individuo, y lo que hace es modificarlo un poco con la esperanza de mejorar su aptitud.

Selección de los sobrevivientes

La selección de los sobrevivientes consiste en seleccionar a los individuos que pasan a la siguiente generación. Es similar a la selección de los padres, pero es usada en una etapa distinta del ciclo evolutivo. La selección debe basarse en la aptitud de los individuos favoreciendo a aquellos con mayor aptitud. La selección de los individuos es comúnmente llamada remplazamiento (*replacement or replacement strategy*).

Inicialización

Consiste en generar la población inicial. En principio se puede generar una población inicial de forma aleatoria o utilizando algunas heurísticas.

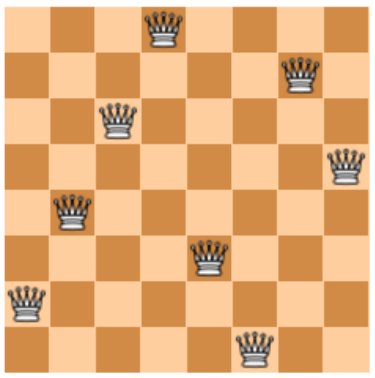
Condición de paro

Se pueden distinguir dos casos para definir la condición de paro. Si el problema tiene un nivel de aptitud óptima conocida, entonces se puede definir como condición de paro el encontrar a un individuo con dicha aptitud, o al menos con una aptitud cercana. Sin embargo, generalmente no se conoce dicho valor, o no se encuentra un individuo tan apto, por lo cual se suelen utilizar otras condiciones como:

- Número de evaluaciones de aptitud
- Número de generaciones
- La mejora de la aptitud en un cierto periodo de tiempo

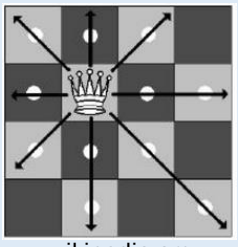
Actividad: **Resolver el problema de 8 reinas utilizando un Algoritmo Evolutivo**

El problema consiste en colocar ocho reinas en un tablero de ajedrez de 8x8 sin que se amenacen entre ellas:



wikipedia.org

Una reina puede matar a cualquier ficha que este colocada en la misma fila, columna o en diagonal.



wikipedia.org

Representación (definición de los individuos)

Fenotipo

	1	2	3	4	5	6	7	8
1			■					
2	■							
3						■		
4		■						
5					■			
6				■				
7							■	
8								■

Genotipo

[2,4,1,6,5,3,7,8]

La solución se representa como un arreglo donde la posición representa la columna y el valor la fila, el arreglo sólo puede tener los números del 1 a 8 y no debe contener valores repetidos. Es decir, el espacio de los genotipos son las permutaciones de los números del 1 al 8.

Nota: Con esta representación se garantiza que ninguna ficha va a estar en la misma fila o en la misma columna. ¡Se resuelve la mitad del problema!

Evaluación de la función (fitness)

La aptitud de cada genotipo se puede definir como el número de pares de reinas que se amenazan entre ellas. Para la solución mostrada anteriormente el *fitness* es 5 porque se amenazan entre sí las reinas (2,4), (4,5), (5,7), (5,8), (7,8).

El objetivo sería encontrar una solución con fitness 0.

Matriz de amenaza entre reinas

	1	2	3	4	5	6	7	8
1								
2	0							
3	0	0						
4	0	1	0					
5	0	0	0	1				
6	0	0	0	0	0			
7	0	0	0	0	1	0		
8	0	0	0	0	1	0	1	

Población

La población contendrá los genotipos de varias soluciones, por ejemplo:

Individuo 1	[1,4,7,6,2,3,5,8]
Individuo 2	[7,4,2,8,1,5,3,6]
Individuo 3	[6,3,4,7,8,1,2,5]
…	…
Individuo N	[2,4,1,6,5,3,7,8]

Selección de los padres

En cada generación se seleccionaran por torneo binario dos padres

Operadores de variación

Reproducción

Información de los padres:

Padre 1	[1,4,7,6,2,3,5,8]
Padre 2	[7,4,2,8,1,5,3,6]

Seleccionar un punto de cruza aleatorio entre (1 y 7): 4

Padre 1	[1,4,7,6 | 2,3,5,8]
Padre 2	[7,4,2,8 | 1,5,3,6]

Crear a los hijos a partir del swap de la información de los padres:

Hijo 1	[1,4,7,6,1,5,3,6]
Hijo 2	[7,4,2,8, 2,3,5,8]

Validar la solución y ajustar los valores de forma aleatoria:

Hijo 1	[1,4,7,6,1,5,3,6]	[1,4,7,6,2,5,3,8]
Hijo 2	[7,4,2,8,2,3,5,8]	[7,4,2,8,6,3,5,1]

Mutación

Individuo:

Individuo	[1,4,7,6,2,3,5,8]

Seleccionar dos posiciones de forma aleatoria: 1,3

Individuo	[1,4,7,6,2,3,5,8]

Hacer un swap en esas dos posiciones:

Individuo	[7,4,1,6,2,3,5,8]

Condición de paro

- Número de iteraciones mayor a un máximo de iteraciones
- Obtener un individuo con *fitness* 0

Inicialización

Crear aleatoriamente N permutaciones de los números del 1 al 8

Algoritmo Evolutivo

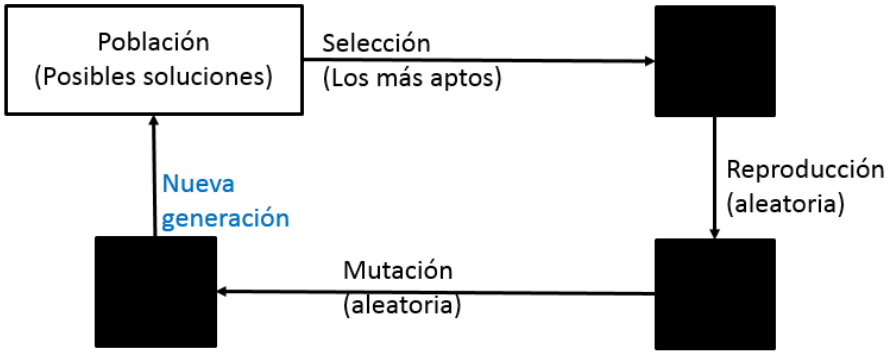
```
Parámetros:
N: Número de individuos en la población
Pm: Probabilidad de mutación, número entre 0 y 1

Algoritmo:
Inicializar la población con N individuos
Evaluar la aptitud de los N invididuos
Mientras (no se haya encontrado un individuo con fitness 0  Y el # de iteraciones no sea mayor al máximo de iteraciones):
    Seleccionar al padre1 con torneo binario
    Seleccionar al padre2 con torneo binario
    Reproducir al padre1 y al padre2 para crear al hijo1 y al hijo2
    Si Pm<= a un número aleatorio entre 0 y 1:
        Mutar al hijo 1
    Si Pm <= número aleatorio entre 0 y 1:
        Mutar al hijo2
    posición1 = número aleatorio entre 1 y N
    posición2 = número aleatorio entre 1 y N
    Si el hijo1 tiene mejor fitness que el individuo en la posición1
      Reemplazar al individuo en la posición1 por el hijo1
    Si el hijo2 tiene mejor fitness que el individuo en la posición2
      Reemplazar al individuo en la posición2 por el hijo2
Regresar el fenotipo de la mejor solución
```

4.1 Algoritmos Genéticos

Son algoritmos evolutivos que pueden utilizarse para resolver problemas de búsqueda y optimización. Fueron propuestos en los años 1970 por John Henry Holland y desde entonces han sido una de las líneas más prometedoras de la inteligencia artificial.

En un algoritmo genético, una población está formada por N individuos, cada uno de ellos representa una posible solución. Esos individuos son sometidos a las operaciones: selección, reproducción y mutación en cada generación. Después de cierto número de generaciones los individuos evolucionan proporcionando mejores soluciones.



Lo que hace especial a los algoritmos genéticos es su representación ya que las soluciones son un arreglo de valores, y la representación más clásica es que los individuos son arreglos binarios.

Representación

Cada individuo en la población (posible solución) se representa con 0's y 1's, por ejemplo:

Individuos

X1 o X	X2 o Y	...	Función de aptitud
23.9	0.7	...	145
-1.7	2.4	...	10
36.8	52.9	...	296

X1 o X	X2 o Y	...	Función de aptitud
1101	0001	...	145
0110	0010	...	10
1101	1101	...	296

- Representación de números enteros:
  Valores posibles: $2^{\#\text{bits}}$
  Se puede seguir la representación clásica:

Signo	2^6	2^5	2^4	2^3	2^2	2^1	2^0

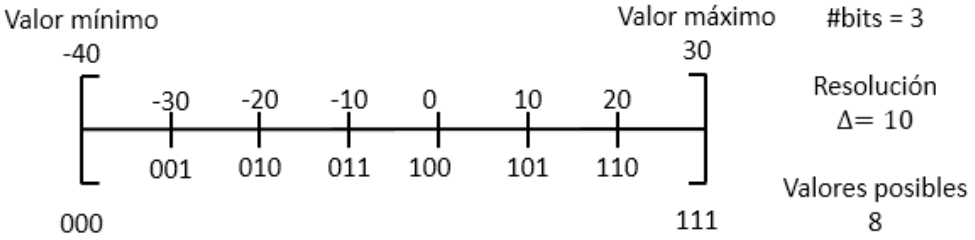
- Representación de números reales:

$$Resolución\ \Delta = \frac{Valor\ máximo\ - Valor\ mínimo}{2^{\#bits} - 1}$$

$$\text{Valores posibles} = 2^{\#\text{bits}}$$

$$Valor\ real = Valor\ mínimo + (\Delta)(Valor\ del\ número\ binario)$$

Por ejemplo:



Actividad: Representación binaria

1. Si se representa un número real en el rango [-20,20] utilizando 6 bits:
   a) ¿Qué resolución tendrá?
   b) ¿Cuántos valores diferentes puede haber?
   c) ¿Qué valor representará 011101?

2. Si se representa un número real en el rango [50,100] utilizando sólo 3 bits:
   a) ¿Cuántos valores diferentes puede haber? y ¿Cuáles son?
   b) ¿Qué resolución tendrá?
   c) ¿Qué valor representará el 110?

3. Si se requiere representar un número real en el rango [-100,100] con una resolución máxima de 5, ¿Cuántos bits se deben utilizar?

Ejemplo:

Problema: Una escuela prepara una excursión para 400 alumnos. La empresa de transporte tiene 8 autobuses de 40 plazas y 10 de 50 plazas, pero sólo dispone de 9 conductores. El alquiler de un autocar grande cuesta $8,000 y el de uno pequeño $6,000. Calcular cuántos autobuses de cada tipo hay que utilizar para que la excursión resulte lo más económica posible para la escuela.

Formulación matemática

$$Min \quad 6000p + 8000g$$
$$s.a.$$
$$p + g \leq 9$$
$$p \leq 8$$
$$g \leq 9$$
$$40p + 50g \geq 400$$
$$p, g \geq 0$$

Representación

Podemos representar un individuo con 8 bits de la siguiente manera:

Número de autobuses pequeños (p)				Número de autobuses grandes (g)			
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0

Representación								Aptitud	
0	0	1	0	0	1	1	1	2 pequeños y 7 grandes	68,000
0	0	1	1	1	0	0	1	3 pequeños y 9 grandes -> NO Válido	
0	0	1	1	0	1	1	0	3 pequeños y 6 grandes	66,000
0	1	1	1	0	1	1	1	7 pequeños y 7 grandes -> NO Válido	
0	1	0	0	0	1	0	1	4 pequeños y 5 grandes	64,000

Selección

El objetivo en esta etapa es seleccionar a los mejores individuos. Al seleccionar a los mejores individuos puede darse el caso de seleccionar varias veces al mismo individuo.

Población									Función de aptitud
0	0	1	0	0	0	1	1	1	68
0	1	0	1	0	1	0	1	1	62
0	1	0	0	0	0	1	0	1	64
0	0	0	0	1	1	1	1	0	64
1	0	1	0	0	0	1	1	0	66

Selección →

Población después de la selección									Función de aptitud
0	1	0	1	0	1	0	1	1	62
0	1	0	1	0	1	0	1	1	62
1	0	1	0	0	0	1	1	0	66
0	1	0	0	0	0	1	0	1	64
0	1	0	1	0	1	0	1	1	62

Dos formas de realizar la selección son:

1. Torneo binario

   Algoritmo

   ```
   Repetir N veces // Para seleccionar a cada uno de los individuos
           Se eligen al azar dos individuos de la población
           Se selecciona el individuo con mejor aptitud
   ```

2. Ruleta:

   Algoritmo

   ```
   Se asigna un porcentaje (en número decimal) a cada individuo según su aptitud
       % = Aptitud del individuo / Suma de todas las aptitudes
   Se calcula el rango de aceptación de cada individuo
       El inicio es igual al fin del rango del individuo anterior (0 para el primer individuo)
       El fin es igual al inicio más el porcentaje del individuo

   Repetir N veces          // Para seleccionar a cada uno de los individuos
     Se calcula un número al azar entre 0 y 1
     Se selecciona el individuo que en su rango incluya al número anterior
   ```

   Nota: Aplica cuando el fitness solo tiene valores positivos y es mejor cuando tiene un valor más grande

   Ejemplo:

Individuos	Aptitud	%	Rango
011001	20	0.10	0.00 – 0.10
110100	60	0.30	0.10 – 0.40
010101	10	0.05	0.40 – 0.45
110110	40	0.20	0.45 – 0.65
010110	70	0.35	0.65 – 1.00

Actividad: Selección

1. Suponga que se tienen los siguientes individuos con su nombre y aptitud:
(A,15), (B,30), (C,5), (D,20), (E,15), y (F,50).
   a) ¿Cuáles individuos se seleccionan utilizando torneo binario?
   b) ¿Cuáles individuos se seleccionan utilizando ruleta?

Suponga que se obtienen los siguientes valores al azar
- Enteros (entre 1 y 6): 3,6,4,2,1,5,6,3,1,4,2,4,3,1,6,4,5,3,1,5,3,2,4,6,1,2,4,1,5,6
- Reales (entre 0 y 1): 0.76, 0.32, 0.95, 0.12, 0.56, 0.67, 0.54.

2. Suponga que se tienen los siguientes individuos con su nombre y aptitud:
(A,100), (B,50), (C,150), (D,200), (E,80)
   a) ¿Cuáles individuos se seleccionan utilizando torneo binario?
   b) ¿Cuáles individuos se seleccionan utilizando ruleta?

Suponga que se obtienen los siguientes valores al azar
- Enteros (entre 1 y 5): 3,5,4,2,1,5,3,1,4,2,4,3,1,4,5,3,1,5,3,2,4,1,2,4,1,5
- Reales (entre 0 y 1): 0.76, 0.32, 0.95, 0.12, 0.56, 0.67, 0.54.

Reproducción o cruza

La reproducción se realiza con el objetivo de explorar el espacio de búsqueda.

Algoritmo

Se establece una probabilidad de reproducción Pr
Repetir N/2 veces // Para obtener los nuevos individuos, de dos padres se generan dos hijos
  Se eligen al azar dos individuos de la población para ser los padres
  Se elige un número al azar entre 0 y 1
  Si el número anterior es menor o igual a Pr
    Se elige al azar un punto de cruce
    En base al punto de cruce se crean dos hijos intercambiando
    los cromosomas de los padres

Padre 1	1	1	0	1	1	1	0	1
Padre 2	0	0	0	1	1	1	1	0
Hijo 1	1	1	0	1	1	1	1	0
Hijo 2	0	0	0	1	1	1	0	1

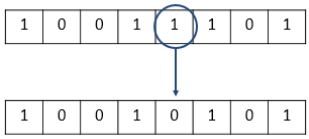
Punto de cruce

  Si no
    Se generan dos hijos como copias de sus padres

Mutación

La mutación se realiza con el objetivo de salir de mínimos locales.
Algoritmo

Se establece una probabilidad de mutación Pm
Repetir N // Para obtener los nuevos individuos
Se elige un número al azar entre 0 y 1
Si el número anterior es menor o igual a Pm
Repetir
Se intercambia un cromosoma al azar



Hasta que el individuo tenga un valor válido
Si no
Se deja el individuo como estaba

Individuo elite

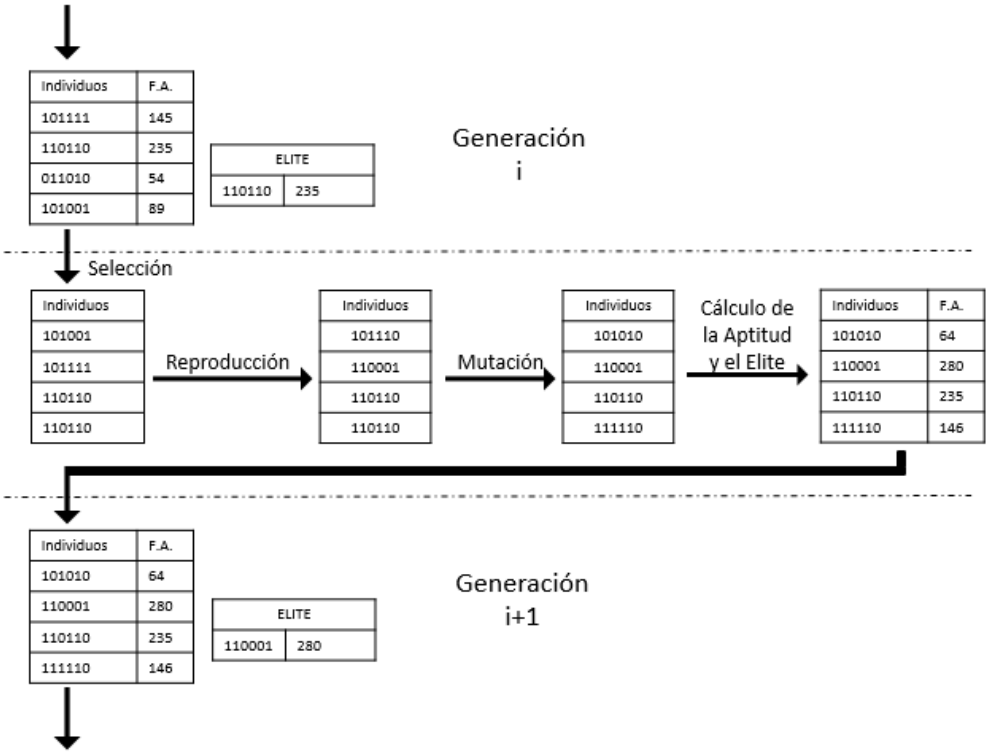
Para garantizar que el algoritmo converja se debe conservar la **Elite**. La Elite es el individuo mejor adaptado hasta el momento en la población, y por lo tanto se debe evitar que se pierda en la selección, o reproducción o mutación.

Para conservarlo se debe verificar lo siguiente en cada generación:

- Si en la nueva generación existe un individuo con mejor función de aptitud que la elite, entonces la elite se reemplaza por dicho individuo.
- Si en la nueva generación NO existe un individuo mejor que la elite, se reemplaza un individuo al azar por la elite, con el objetivo que la elite entre nuevamente al proceso evolutivo.

Proceso evolutivo

El proceso evolutivo de un Algoritmo Genético está representado en la siguiente imagen:



Algoritmo

```
Parámetros:
    N, número de individuos
    G, número de generaciones
    Pr, probabilidad de reproducción
    Pm, probabilidad de mutación
    FuncionAptitud, función que recibe un individuo y calcula su aptitud

Inicio
  Generar la población inicial de forma aleatoria validando que todos los individuos
  sean soluciones válidas
  Calcular las aptitudes de todos los individuos
  Guardar la elite
  Repetir G veces
      Seleccionar
      Reproducir
      Mutar
      Calcular aptitudes
      Si hay un mejor individuo que la elite
          Reemplazar la elite
      Si no
          Incluir la elite en la población
  Fin repetir
Fin
```

Actividad: Programación lineal

Resolver los problemas de Programación Lineal utilizando Algoritmos Genéticos.

Algoritmo genético con representación flotante

Existe otra implementación de los Algoritmos Genéticos donde cada individuo se representa con valores flotantes. Las únicas operaciones que cambian del Algoritmo Genético Binario son:

Reproducción

Se establece una probabilidad de reproducción Pr
Repetir N/2 veces // Para obtener los nuevos individuos, se crean de 2 en 2
Se eligen al azar dos individuos de la población para ser los padres
Se elige un número al azar entre 0 y 1
Si el número anterior es menor o igual a Pr
Se obtienen dos hijos con las siguientes fórmulas

$$H1 = \alpha P1 + (1-\alpha)P2$$
$$H2 = \alpha P2 + (1-\alpha)P1$$
$$donde\ \alpha\ es\ un\ número\ aleatorio\ entre\ [-0.5\ y\ 1.5]$$

Si no
Se generan dos hijos como copias de sus padres

Mutación

Se establece una probabilidad de mutación Pm
Repetir N // Para obtener los nuevos individuos
Se elige un número al azar entre 0 y 1
Si el número anterior es menor o igual a Pm
Se muta el individuo con la fórmula

$$I = c_1 r_1 (Elite - I) + C_2 r_2 (Elite - I)$$
$$donde$$
$$c_1 y\ c_2\ son\ constantes\ de\ aceleración$$
$$generalemente\ se\ usa\ c_1 = 5\ ,\ c_2 = 10$$
$$r_1 y\ r_2\ son\ aleatorios\ entre\ 0\ y\ 1$$

Si no
Se deja el individuo como estaba

Actividad: Ajuste de imágenes

El problema del ajuste de imágenes consiste en definir los parámetros traslación - rotación que hace que dos imágenes muy similares se ajusten. En este caso, los parámetros que debemos calcular son:

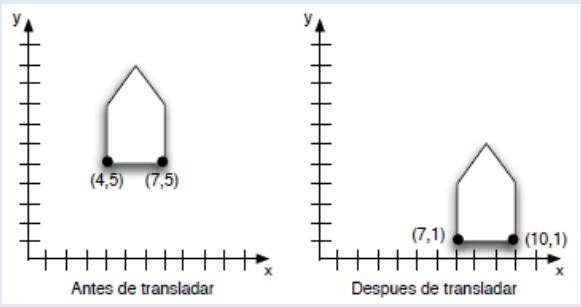
- Grados para girar la imagen
- Incremento/decremento de la posición en X
- Incremento/decremento de la posición en Y

Imagen patrón

Imagen a ajustar

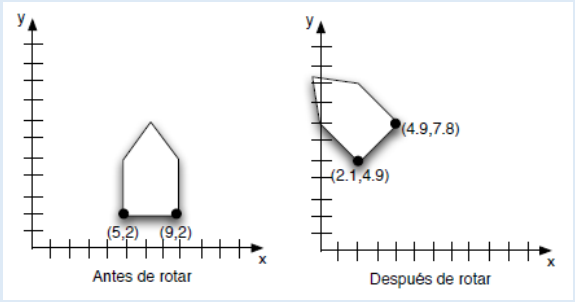
Traslación: Para mover un punto P(x,y) Δx unidades en el eje x y Δy unidades en el eje y.

$$x' = x + \Delta x \qquad y' = y + \Delta y$$



Rotación: Los puntos se rotan por un ángulo Θ alrededor del origen.
Una rotación se define:

$$x' = xcos\theta - ysin\theta \qquad y' = xsin\theta + ycos\theta$$



4.2 Programación Genética

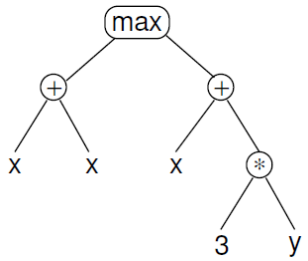
La Programación Genética (PG) es un algoritmo evolutivo que ha recibido mucha atención últimamente gracias a que resuelve problemas difíciles de la vida real de forma satisfactoria sin el requerimiento de que el usuario conozca o especifique la forma o estructura de la solución, básicamente lo que hace es buscar en un espacio de programas a aquel programa que mejor resuelva el problema.

Siguiendo la estructura de los algoritmos evolutivos, en programación genética una población de programas computacionales es evolucionada de generación en generación de forma estocástica con el objetivo de encontrar al individuo que mejor resuelva el problema.

Los elementos de la Programación Genética son los siguientes:

a) Representación

En PG, los individuos (o programas) usualmente son representados como árboles de sintaxis, como se muestra en la siguiente figura:



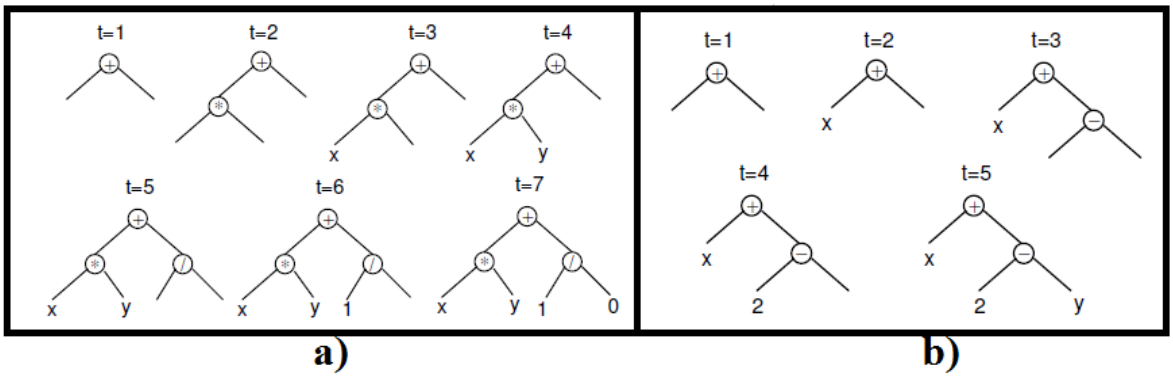
. Los elementos de un árbol de sintaxis son:

- Las **terminales**, que representan a las variables o constantes, y que son nodos hoja en los árboles de sintaxis. Las terminales consisten en: las entradas externas del programa, típicamente denominadas variables (ejemplo x,y); funciones sin argumentos, por ejemplo numero_aleatorio(); y finalmente las constantes, las cuales pueden ser predefinidas o generadas aleatoriamente en alguna parte de la creación de los árboles.

- Las **funciones**, que representan a las operaciones que se pueden hacer, son nodos internos en el árbol de sintaxis. El conjunto de funciones se define por la naturaleza del problema, por ejemplo, para problemas numéricos las funciones podrían ser los operadores aritméticos, funciones trigonométricas, etc.; y para problemas booleanos podrían ser los operadores AND, OR y NOT.

b) Inicialización de la población

Como en otros algoritmos evolutivos, en PG los individuos en la población inicial se generan aleatoriamente. Existen muchos métodos para la inicialización, dos de ellos son los métodos *full* y *grow*, y una técnica llamada *Ramped half-and-half* que hace una combinación de ellos. En ambos métodos, *full* y *grow,* los individuos iniciales son generados de tal forma que no excedan una profundidad máxima. El método *full* genera árboles aleatorios llenos, es decir, todas sus hojas tienen la misma profundidad. En la creación de los árboles iniciales, los nodos en niveles intermedios sólo toman funciones, y los nodos en el último nivel tienen que ser terminales, un ejemplo de este método es mostrado en la siguiente figura a. El método *grow* genera árboles de diferentes tamaños, ya que los nodos en niveles intermedios seleccionan aleatoriamente funciones o terminales, y los nodos en el último nivel seleccionan terminales forzosamente, un ejemplo de este método se muestra en la figura b. Finalmente, utilizando la técnica *Ramped half-and-half* la mitad de la población inicial es construida con el método *full* y la otra mitad con el método *grow*.



Creación de los individuos iniciales a) Método full b) Método grow (t=tiempo) *[6]*

c) Función de aptitud

La función de aptitud debe medir la calidad de cada individuo en la población. Existen diferentes formas de medirla, por ejemplo:

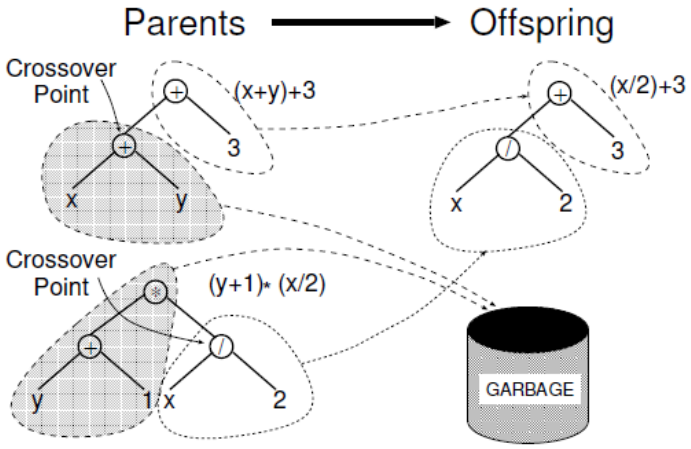
- Calcular la diferencia entre la salida del programa del individuo y la salida esperada
- El tiempo necesario para resolver un problema
- La precisión del programa en un problema clasificación

d) Selección

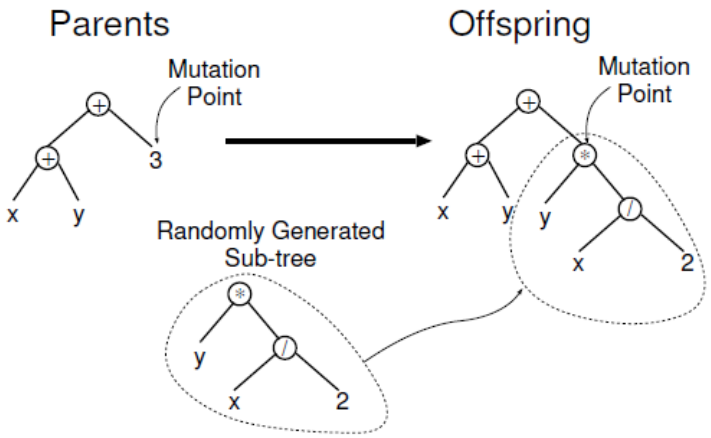
Como en la mayoría de los algoritmos evolutivos, la selección de los individuos en PG se realiza en base a su aptitud. Los mejores individuos (los programas que generan mejores resultados) tienen mayor probabilidad de sembrar la siguiente generación. Selección por torneo aleatorio es el método más utilizado por PG para la selección, consiste en seleccionar aleatoriamente un cierto número de individuos de la población, compararlos y seleccionar al mejor de ellos.

e) Recombinación y mutación

El operador clásico de **recombinación** consiste en que, dados dos padres, se selecciona un punto de cruza (un nodo) en cada padre, y después se crea el hijo a partir del primer padre pero remplazando el subárbol que se encuentra en el punto de cruza por una copia del subárbol en el punto de cruza del segundo padre, como se muestra en la siguiente figura.



El operador clásico de **mutación** consiste en seleccionar aleatoriamente un punto de mutación en el árbol y substituirlo por un nuevo árbol generado aleatoriamente, como se muestra en la siguiente figura:



5. Metaheurísticas

5.1 Hill Climbing (Ascenso de colinas)

Es una técnica de optimización basada en búsqueda local. Comienza con una solución inicial, después en cada iteración esa solución se reemplaza por el vecino con el máximo o mínimo valor según sea el caso. El algoritmo termina cuando un punto es mejor que todos sus vecinos.
(+) Simple de implementar
(-) Se atora en mínimos locales

La heurística tiene que ver en la forma en como determinamos los puntos vecinos.

Ejemplo:

Bibliografía

[1] A. Eiben y J. Smith, Introduction to Evolutionary Computing, Amsterdam: Springer, 2007.