

Salamanca, Guanajuato a 09 de junio del 2023.

**Ingeniería en sistemas computacionales.**

**UNIVERSIDAD DE GUANAJUATO.**

División de ingenierías campus Irapuato

**Aproximación de funciones usando un algoritmo de optimización por enjambre de partículas y funciones de base radial.**

Presentado por:

**Bryan De Jesús Mares Barrientos.**

Docente:

**Dr. Carlos Hugo García C.**

De la materia:

**Inteligencia Artificial.**

Introducción.

El presente informe aborda la implementación del algoritmo Particle Swarm Optimization (PSO) en el lenguaje de programación C, utilizando una combinación lineal de ecuaciones radiales, específicamente la función gaussiana. El PSO es un algoritmo de optimización inspirado en el comportamiento social de los enjambres de partículas, el cual ha demostrado ser eficiente en la resolución de problemas complejos.

La función gaussiana, también conocida como campana de Gauss, es una función matemática que presenta una distribución en forma de campana. Es ampliamente utilizada debido a sus propiedades bien definidas y su capacidad para modelar fenómenos naturales y artificiales. La combinación lineal de ecuaciones radiales con función gaussiana nos permite ajustar los parámetros de estas ecuaciones para obtener resultados óptimos en la resolución de problemas específicos.

La implementación del algoritmo PSO en C nos brinda una herramienta poderosa para buscar los parámetros óptimos de la combinación lineal de ecuaciones radiales. C es un lenguaje de programación ampliamente utilizado y conocido por su eficiencia y flexibilidad. La combinación de PSO y C nos permite aprovechar las capacidades de optimización del algoritmo y la velocidad de ejecución del lenguaje para encontrar soluciones rápidas y precisas.

Este informe presenta la implementación del algoritmo PSO en C para encontrar los parámetros óptimos de una combinación lineal de ecuaciones radiales basadas en la función gaussiana. Se espera que los resultados obtenidos demuestren la efectividad y eficiencia de esta metodología en la resolución de problemas complejos.

Objetivo.

El objetivo principal de este trabajo es utilizar el algoritmo PSO para encontrar los parámetros óptimos de una combinación lineal de ecuaciones radiales basadas en la función gaussiana. Este tipo de combinación lineal es ampliamente utilizado en diversas áreas, como el procesamiento de imágenes, la detección de anomalías y el aprendizaje automático.

Marco Teórico.

Optimización y algoritmo PSO:

La optimización es el proceso de encontrar la mejor solución posible para un problema, dentro de un conjunto de posibles soluciones. El algoritmo PSO (Particle Swarm Optimization) es una técnica de optimización inspirada en el comportamiento social de los enjambres de partículas. En PSO, una población de partículas se mueve por un espacio de búsqueda en busca de la solución óptima. Cada partícula tiene una posición y una velocidad, y se actualiza en función de su mejor experiencia personal y de la mejor experiencia de todo el enjambre.

Combinación lineal de ecuaciones radiales:

Una combinación lineal de ecuaciones radiales es una función que se forma sumando varias funciones radiales, multiplicadas por coeficientes lineales. Las funciones radiales son funciones que dependen únicamente de la distancia entre un punto y un centro, y su valor disminuye a medida que la distancia aumenta. Una función radial comúnmente utilizada es la función gaussiana, que se define como una distribución en forma de campana. La función gaussiana es ampliamente utilizada debido a sus propiedades bien definidas y su capacidad para modelar una variedad de fenómenos.

Implementación del algoritmo PSO en C:

La implementación del algoritmo PSO en el lenguaje de programación C implica la representación de las partículas como estructuras de datos, con atributos como posición, velocidad y experiencia personal. El algoritmo PSO se ejecuta en un bucle iterativo, donde se actualizan las posiciones y velocidades de las partículas utilizando ecuaciones específicas. Estas ecuaciones tienen en cuenta la mejor experiencia personal de cada partícula, así como la mejor experiencia global del enjambre. El proceso continúa hasta que se cumple un criterio de terminación, como alcanzar un número máximo de iteraciones o lograr una convergencia aceptable.

Ventajas y aplicaciones de la combinación lineal de ecuaciones radiales:

La combinación lineal de ecuaciones radiales con función gaussiana ofrece varias ventajas en la resolución de problemas. La función gaussiana permite modelar relaciones complejas y capturar patrones en conjuntos de datos. La combinación lineal proporciona flexibilidad al ajustar los coeficientes para obtener el mejor ajuste a los datos o la solución óptima. Esta técnica se utiliza en diversas áreas, como el procesamiento de imágenes, la detección de anomalías, el aprendizaje automático y la optimización de funciones.

Desarrollo.

Para iniciar la codificación y solución de nuestro problema, es importante entender este mismo, saber que es lo que se quiere hacer y cómo poder hacerlo.

La problemática de esta implementación radica más en el raciocinio de la lógica antes que la implementación del código como tal, es por esto, que lo primero es realizar la escritura de nuestras formulas y poder entender que es lo que queremos hacer y que se debe de calcular. Si bien, las formulas ya están propuestas, la sustitución de valores dentro de la misma supone un problema, dado la nula experiencia de trabajo sobre este tipo de ecuaciones.

Texto, Carta

Descripción generada automáticamentePara el entendimiento de la implementación de este código, se tomo como referencia el siguiente desarrollo matemático:

Una vez fundamentado esto, la implementación del código es relativamente sencilla, ya que es sólo modificar la *funciónObjetivo()* de nuestro código, quedando de la siguiente manera:

double FuncionObjetivo(double\* Xi, double\* Px, double\* Py, int m, int n){

    double fit=0, rbf=0;

    for(int i=0; i<n; i++){

        for(int k=0; k<m; k++){

            // for

            rbf += Xi[3\*k] \* exp(-pow(Px[i]-Xi[(3\*k)+1], 2)/pow(Xi[(3\*k)+2], 2));

        }

        fit += pow(rbf-Py[i], 2);

        rbf=0;

    }

    return (double)-fit/n;

}

Si bien, los valores a ingresar serán arbitrarios, es importantes mencionar que el número de veces a ejecutar (n y m) están implementados en los parámetros de las demás funciones, por lo cual ahora mismo no se le dará importancia.

Para evitar que nuestra implementación explote, hemos decidido hacer un “clamping” a esta misma, para mantener nuestros valores de búsqueda en un rango óptimo para la realización de esta.

Como tal, se nos proporcionaron los valores de x y y, siendo estos los puntos de las funciones sobre las cuales se hará la búsqueda y se comparará contra los obtenido, para lograr esto, creamos una función la cual recupera estos valores mediante la inspección de la carpeta en la cual se encuentra alojado nuestro código, esto para agilizar las cosas y no tener que ingresar toda la matriz de datos.

Por último, la parte importante de esto es la de los valores para nuestros parámetros de cada una de las funciones gaussianas, ya que cada una de ellas debe de tener algún valor en especifico para que estas puedan llevar a cabo su búsqueda, para esto, dentro de nuestras funciones, implementamos que cada uno de estos sea generado automáticamente dentro de los espacios de limites superiores e inferiores definidos al inicio de nuestro código, agilizando el proceso de ejecución de nuestro código.

Pruebas y Resultados.

Imagen que contiene Calendario

Descripción generada automáticamentePara la muestra de los resultados, y a manera de agilizar la representación de estos, decidí ordenarlos directamente en la función de impresión de consola, mostrándose como se ve a continuación.

En la columna de **i**, se muestra el identificador de la particula.

Las columnas de **Xi** y **Vi** corresponden al vector de posición y su velocidad, respectivamente.

La imagen de la izquierda es el resultado de uno de los mejores resultados para los valores siguientes:

60 funciones, 1500 iteraciones y un espacio para amplitud, centro y radio de:

const float limSup[DIM\_] = { 1.0, 1.2, 1.0};

const float limInf[DIM\_] = {-1.0,-0.2, 0.0};

Texto

Descripción generada automáticamenteLos valores a continuación muestran, de la columna izquierda, el valor que debió de salir, en el centro el obtenido, y a la derecha el error.

Al igual que el valor para la mejor partícula.Resultados para:

#define DIM\_ 3

//Tamaño del enjambre

const unsigned int NParticulas = 100;

//Numero de variables del problema,

const unsigned int Dimension = DIM\_;

const float limSup[DIM\_] = { 1.0, 1.2, 1.0};

const float limInf[DIM\_] = {-2.0,-1.2, 0.1};

const unsigned int IteracionesMaximas = 1000;

const unsigned int NoFunciones = 15;

Imagen que contiene Calendario

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Resultados para:

#define DIM\_ 3

//Tamaño del enjambre

const unsigned int NParticulas = 200;

//Numero de variables del problema,

const unsigned int Dimension = DIM\_;

const float limSup[DIM\_] = { 2.0, 4.0, 16.0};

const float limInf[DIM\_] = {-2.0,-1.2,  0.1};

const unsigned int IteracionesMaximas = 2000;

Calendario

Descripción generada automáticamente con confianza bajaImagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamenteEscala de tiempo

Descripción generada automáticamenteconst unsigned int NoFunciones = 30;

Conclusiones.

En este informe se ha presentado la implementación del algoritmo Particle Swarm Optimization (PSO) en el lenguaje de programación C, utilizando una combinación lineal de ecuaciones radiales basadas en la función gaussiana. A lo largo del trabajo, se exploraron los conceptos fundamentales de la optimización, el algoritmo PSO y la combinación lineal de ecuaciones radiales.

Los resultados obtenidos durante la implementación del algoritmo PSO en C fueron prometedores en términos de la capacidad de encontrar los parámetros óptimos de la combinación lineal de ecuaciones radiales. Sin embargo, es importante mencionar que el nivel de ejecución en nuestra computadora presentó limitaciones, lo que restringió la cantidad de pruebas que se pudieron realizar. Esto impidió explorar en profundidad el rendimiento y la eficiencia del algoritmo en diferentes escenarios y conjuntos de datos.

A pesar de estas limitaciones, se reconoce el potencial de la combinación lineal de ecuaciones radiales basadas en la función gaussiana y el algoritmo PSO para la resolución de problemas complejos. Sería beneficioso en futuras investigaciones explorar la implementación de algoritmos jerárquicos o técnicas de ajuste automático de parámetros. Estos enfoques permitirían encontrar los parámetros adecuados de manera automática, sin necesidad de ajustes manuales. Esto podría mejorar aún más la eficiencia y la capacidad de adaptación del algoritmo a diferentes problemas y conjuntos de datos.

A pesar de las limitaciones en la ejecución y la falta de pruebas exhaustivas, la implementación del algoritmo PSO en C y la combinación lineal de ecuaciones radiales con función gaussiana ofrecen un enfoque prometedor para la resolución de problemas complejos. Se sugiere explorar en futuras investigaciones la aplicación de algoritmos jerárquicos y técnicas de ajuste automático de parámetros para mejorar aún más el rendimiento y la adaptabilidad del algoritmo. Estas mejoras podrían abrir nuevas oportunidades en diversas áreas donde se requiere la optimización de funciones y la modelización de relaciones complejas.

**LINK DEL VIDEO:**

https://drive.google.com/drive/folders/1oyMEuTHtTFGl1lgB\_n7MRcXl-pEBw6W\_?usp=sharing

**LINK PARA EL CÓDIGO UTILIZADO**:

<https://github.com/BryanMaresDev/pso_proyecto01>

Referencias:

Hindawi. (2015). Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. Mathematical Problems in Engineering, 2015, 731207. Recuperado de https://www.hindawi.com/journals/mpe/2015/731207/

Smith, J. A., & Johnson, L. K. (2018). Particle Swarm Optimization for Solving Nonlinear Optimization Problems. International Journal of Swarm Intelligence Research, 9(2), 1-18.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks (pp. 1942-1948). Recuperado de https://ieeexplore.ieee.org/document/488968/