


UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

REPORTE DE ALGORITMOS

REGRESIÓN POLINOMIAL

Nombre	Expediente
Zuñiga Fragoso Diego Joel	317684

Asignatura: Método Numéricos 2023-2

Docente: Vargas Vázquez Damián

I. Antecedentes teóricos

La regresión polinómica es una técnica utilizada en estadísticas y análisis de datos para modelar la relación entre una variable independiente ‘x’ y una variable dependiente ‘y’ mediante un polinomio.

1. Selección del Grado del Polinomio:

La elección del grado del polinomio es crucial y debe basarse en la naturaleza de los datos y el conocimiento del problema. Grados más altos pueden ajustarse mejor a los datos, pero también pueden resultar en sobreajuste (overfitting) si no se justifican.

2. Matriz de Diseño:

La formulación matricial es común en regresión polinómica. Se construye una matriz de diseño que contiene las potencias sucesivas de la variable independiente. La solución se obtiene resolviendo un sistema de ecuaciones normales utilizando álgebra lineal.

II. Algoritmos y sus resultados

Cada algoritmo esta seccionado e incluye descripciones de lo que sucede. Además de contar con capturas de sus resultados

Código

```
#include <iostream>
#include <cmath>

using namespace std;

// Método de eliminación de Gauss: eliminación hacia adelante.
void eliminacionGauss(double** A, double B[], int n)
{
	double inv;
	for (int k = 0; k < n; k++)
		for (int i = k + 1; i < n; i++)
		{
			inv = A[i][k] / A[k][k];
			for (int j = k; j < n; j++) {
				A[i][j] = A[i][j] - inv * A[k][j];
			}
			B[i] = B[i] - inv * B[k];
		}
}
```

```
//Método de eliminación de Gauss: sustitución inversa.
void sustitucionAtras(double** A, double B[], int n, double C[])
{
	double suma;

	C[n - 1] = B[n - 1] / A[n - 1][n - 1];
	for (int i = n - 2; i >= 0; i--)
	{
		suma = 0;
		for (int j = i + 1; j < n; j++)
			suma = suma + A[i][j] * C[j];
		C[i] = (B[i] - suma) / A[i][i];
	}
}

// Regresión polinomial
void regresionPolinomial(double x[], double y[], int n, int m)
{
	double sum_x = 0, sum_xy = 0;
	int length = m + 1;
	double* solucion = new double[length];
	double** ecuaciones;

	ecuaciones = new double* [length];
	//Inicialización del arreglo bidimensional.
	for (int i = 0; i < length; i++)
		ecuaciones[i] = new double[length];

	//Cálculo de las sumatorias y armado del sistema.
	for (int i = 0; i < length; i++)
	{
		sum_xy = 0;

		for (int j = 0; j < n; j++)
			sum_xy += pow(x[j], i) * y[j];
		solucion[i] = sum_xy;

		for (int j = 0; j < length; j++)
		{
			sum_x = 0;
			if (i == 0 && j == 0)
				ecuaciones[i][j] = n;
			else
			{
				for (int h = 0; h < n; h++)
					sum_x += pow(x[h], (j + i));
				ecuaciones[i][j] = sum_x;
			}
		}
	}

	//Resolucion de sistemas de ecuaciones.
	eliminacionGauss(ecuaciones, solucion, length);

	double* x_vector = new double[length];
```

```
	sustitucionAtras(ecuaciones, solucion, length, x_vector);

	//Construcción de la ecuación final.
	cout << "\n\nLa ecuacion es: ";
	for (int i = 0; i < length; i++)
	{
		cout << x_vector[i];
		if (i != 0)
			cout << "x^" << i;
		if (i != length - 1)
			cout << " + ";
	}
	cout << endl;

	//Cálculo de los errores
	double* e = new double[n];
	for (int i = 0; i < n; i++)
	{
		double y_calculada = 0;
		for (int j = length - 1; j >= 1; j--)
			y_calculada += x_vector[j] * (pow(x[i], j));
		y_calculada += x_vector[0];
		e[i] = pow(y[i] - y_calculada, 2);
	}

	double sum_y = solucion[0];

	double sr = 0;
	double st = 0;
	for (int i = 0; i < n; i++)
	{
		sr += e[i];
		st += pow(y[i] - (sum_y / n), 2);
	}

	double err = sqrt(sr / (n - length));

	double r2 = (st - sr) / st;
	double r = sqrt(r2);

	//Desplegado de errores.
	cout << "Error estandar de la estimacion: " << err << endl;
	cout << "Coeficiente de determinacion: " << r2 << endl;
	cout << "Coeficiente de correlacion: " << r << endl;
	cout << endl;

}

int main()
{
	Start:
	system("cls");

	int points, minpoints, degree, flag;
```

```
	cout << "Este programa realiza una regresion polinomial, dependiendo el numero de puntos aumenta la presicion del polinomio" << endl;

	cout << "\n\nIngrese el numero de puntos:\t";
	cin >> points;

	// Creamos arreglos de memoria dinamica
	double* x = new double[points];
	double* y = new double[points];

	// Pedimos al usuario que ingrese los puntos
	for (int i = 0; i < points; i++)
	{
		cout << "\n\nIngrese x_" << i << " =\t\t";		cin >> x[i];
		cout << "Ingrese f(x_" << i << ") =\t";		cin >> y[i];
	}

	Degree:
	cout << "\n\nIngrese el grado del polinomio (maximo " << points - 1 << "):\t";
	cin >> degree;

	if (degree > (points - 1))
		goto Degree;

	regresionPolinomial(x, y, points, degree);

	// Liberamos memoria dinamica
	delete[] x;
	delete[] y;

	cout << "\n\nSi desea volver a hacerlo ingrese 1, de lo contrario ingrese cualquier tecla:\t";
	cin >> flag;

	if (flag == 1)
		goto Start;

	return 0;
}
```

Resultado

```
Este programa realiza una regresion polinomial, dependiendo el numero de puntos aumenta la presicion del polinomio


Ingrese el numero de puntos:    6


Ingrese x_0 =           0
Ingrese f(x_0) =        2.1


Ingrese x_1 =           1
Ingrese f(x_1) =        7.7


Ingrese x_2 =           2
Ingrese f(x_2) =        13.6


Ingrese x_3 =           3
Ingrese f(x_3) =        27.2


Ingrese x_4 =           4
Ingrese f(x_4) =        40.9


Ingrese x_5 =           5
Ingrese f(x_5) =        61.1


Ingrese el grado del polinomio (maximo 5):      2


La ecuacion es: 2.47857 + 2.35929x^1 + 1.86071x^2
Error estandar de la estimacion: 1.11752
Coeficiente de determinacion: 0.998509
Coeficiente de correlacion: 0.999254



Si desea volver a hacerlo ingrese 1, de lo contrario ingrese cualquier tecla:
```

III. Conclusiones

En conclusión, la regresión polinómica es una técnica versátil y ampliamente utilizada para modelar relaciones no lineales entre variables. Su capacidad para ajustarse a datos complejos mediante polinomios permite una mayor flexibilidad en el modelado, especialmente cuando las relaciones subyacentes no pueden describirse de manera lineal.