

## Resumen del Artículo

### "Comparing the Moore-Penrose Pseudoinverse and Gradient Descent"

- La regresión lineal es uno de los métodos más clásicos en estadística y aprendizaje automático. Busca ajustar un modelo lineal  $\mathbf{y} = \mathbf{X}\beta + \epsilon$  minimizando el error cuadrático (mínimos cuadrados o OLS)

- Una vez formulado ese problema existen principalmente dos enfoques a resolverlo:

1. Solución Analítica mediante la Pseudoinversa de Moore-Penrose.
2. Método Iterativo como el gradiente descendente.

Cada enfoque tiene ventajas y desventajas. Complejidad, estabilidad numérica, escalabilidad. El artículo tiene como objetivo comparar estos métodos tanto teóricamente como con experimentos, para delinear en qué situaciones uno es preferible sobre el otro.

¿Bajo qué condiciones (dimensionalidad, tamaño de muestra, condición del sistema, etc.) la pseudoinversa es más eficiente o más estable y cuándo conviene usar la gradiente descendente?

### Formulación de la Regresión Lineal

- Se tienen datos  $(\mathbf{X}, \mathbf{y})$  con  $\mathbf{X} \in \mathbb{R}^{n \times d}$  ( $n$  muestras de características) y  $\mathbf{y} \in \mathbb{R}^n$ .
- Se modela  $\mathbf{y} = \mathbf{X}\beta + \epsilon$ , con  $\epsilon$  ruido medido.
- El objetivo es minimizar la función perdida

$$L(\beta) = \| \mathbf{X}\beta - \mathbf{y} \|^2 = (\mathbf{X}\beta - \mathbf{y})^T (\mathbf{X}\beta - \mathbf{y}).$$

como  $L(\beta)$  es convexa, cualquier mínimo local también es global

### Solución con la Pseudoinversa

el método clásico usado es:

$\mathbf{X}^T \mathbf{X} \beta = \mathbf{X}^T \mathbf{y}$  y así obtener  $\hat{\beta}_{\text{pseud}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ ,  
cuando  $\mathbf{X}$  es invertible.  
• en general, si la matriz  $\mathbf{X}$  no es de rango completo se usa la pseudoinversa de Moore-Penrose  $\hat{\beta}_{\text{pseud}} = \mathbf{X}^+ \mathbf{y}$ ,

### Fortalezas

- usa una sola operación (una inversión / descomposición)
- Resultados exactos (no iterativos)

### Limitaciones

- computacionalmente costosa para matrices grandes
- problemas numéricos si  $\mathbf{X}^T \mathbf{X}$  es mal condicionada
- no escala bien cuando el número de muestras o características es muy grande

### Gradiente descendente aplicado a regresión lineal

se minimiza iterativamente

$$\beta_{k+1} = \beta_k - n \nabla L(\beta_k)$$
 donde  $n$  es la tasa de aprendizaje

$$\nabla L(\beta) = 2 \mathbf{X}^T (\mathbf{X}\beta - \mathbf{y})$$

$$\beta_{k+1} = \beta_k - 2n \mathbf{X}^T (\mathbf{X}\beta_k - \mathbf{y})$$

- el método requiere una **initialización**, elegir criterios de parada (número máximo de iteraciones)
- no da solución exacta en un solo paso, sino converge (idealmente) al óptimo

### Ventajas

- escalable no necesita invertir matrices enormes
- flexible puede adaptarse a datasets (gradiente estocástico)
- menos memoria.

- convergencia lenta si la tasa de aprendizaje es pequeña
- sensible a la condición numérica

### Metodología experimental

#### Experimentos con datos sintéticos

Se generan matrices  $\mathbf{X}$  con distintos tamaños  
se implementan ambos métodos  
tiempo de computo - errores de predicción  
- estabilidad numérica

#### Experimento con datos reales

además de datos reales generados con  $10$  (Número de variables, muestras)  
escaladas

Para problemas pequeños medianos con matrices bien condicionadas la pseudoinversa ofrece una solución rápida.

con datos muy grandes el gradiente resulta más escalable y práctico requiere cuidados.

Classic

**comparación de rendimiento  
entre el modelo Lineal  
y el metodo Gradiente Desendente.**

**Objetivo** evaluar la eficiencia computacional del método de regresión lineal lm-model frente al algoritmo de Gradiente Desendente en términos de tiempo de ejecución (segundos) y uso de memoria (bytes) para diferentes tamaños de muestras.

**1. Resultados Tabulares:** a continuación se muestran los resultados del rendimiento con los valores de tiempo.

Tamaños (N)	Método	Tiempo (s)	Memoria (bytes)
400	lm-model	0	64,680
400	Gradiente	0	1,830,776
1000	lm-model	0	619,048
4000	Gradiente	0	17,619,200
10,000	lm-model	0	6,213,928
10,000	Gradiente	0	176,019,200
100,000	lm-model	0	64,851,432
100,000	Gradiente	2	1,760,019,200
500,000	lm-model	0	308,197,460
500,000	Gradiente	7	8,800,019,200
1,000,000	lm-model	1	646,391,464
1,000,000	Gradiente	17	17,678,009,744
2M	lm-model	2	1,232,780,072
2M	Gradiente	34	35,200,039,024

análisis de los resultados que claramente se evidencian una ventaja clara del método lm-model frente al gradiente Desendente

- Velocidad (Tiempo) el lm-model escala mucho mejor para 2M de observaciones el Gradiente Desendente requiere 34 segundos lo que lo hace aproximadamente 17 veces más lento que lm-model (2 segundos)
- USO de recursos (Memoria) la diferencia es abrumadora. Para la muestra grande el gradiente consume más de 35 millones de bytes mientras que lm-model se limita aproximadamente a 2 millones de bytes demostrando una eficiencia de recursos significativamente superior.

en consecuencia para las tareas de regresión lineal con volúmenes de datos medianos grandes, el lm-model ofrece un rendimiento superior y un uso de recursos superior y un uso de más recursos más eficientes que el gradiente Desendente.

		Memory	Time
1	profvis (f)		
(profvis)	2	model_lm <- lm (y ~ x)	198.4 380
	3	gradiente_desc (x, y)	-20538.4 26046.8 8030
	4	3)	

# Universidad Nacional del Altiplano de Puno

## Facultad de Estadística e Informática

Alumno: Henry Ccoarite Dueñas

### Método del Gradiente Descendente en R

```
# Definimos la función y su derivada
f <- function(x) { x^2 } # f(x) = x^2
f_deriv <- function(x) { 2*x } # f'(x) = 2x

# Parámetros
x0 <- 5; eta <- 0.1; n_iter <- 15
x_vals <- numeric(n_iter); f_vals <- numeric(n_iter); deriv_vals <- numeric(n_iter)

# Inicialización
x_vals[1] <- x0; f_vals[1] <- f(x0); deriv_vals[1] <- f_deriv(x0)

# Ciclo de iteraciones
for (i in 2:n_iter) {
  x_new <- x_vals[i-1] - eta * f_deriv(x_vals[i-1])
  x_vals[i] <- x_new
  f_vals[i] <- f(x_new)
  deriv_vals[i] <- f_deriv(x_new)
}

# Gráfica de resultados
plot(x_vals, f_vals, type="l", col="blue", lwd=2,
      xlab="Valores de x (actualizaciones del gradiente)",
      ylab="Valor",
      main="Función, Derivada y Gradiente Descendente (eje X = valores de x)")
lines(x_vals, deriv_vals, col="red", lwd=2)
lines(x_vals, x_vals, col="green", lwd=2)
legend("topright",
       legend=c("f(x)", "f'(x)", "x_i (valores del gradiente)"),
       col=c("blue", "red", "green"), lwd=2)
```

### Convergencia del Gradiente Descendente en $f(x) = x^2$

