TRABAJO NUMERO_PROGRAMACION NUMERICA

Job Edward Apaza Curtihuanca

TRABAJO NUMERO_ 1

El trabajo numero es realizar un código en “R” donde hallemos la raíz de una función a travez de método de la gradiente, con la cantidad de iteraciones realizadas, en un cuadro.

El método de la gradiente consiste en busca el valor óptimo de una función ajustando sus variables paso a paso, siguiendo la dirección contraria al gradiente, hasta que la pendiente sea casi cero.

CODIGO EN “R”:

```
# Método del Gradiente en R
# Función simple: f(x, y) = x^2 + y^2

# Definimos la función
f <- function(x, y) {
  return(x^2 + y^2)
}

# Gradiente de f(x, y)
gradiente <- function(x, y) {
  gx <- 2 * x
  gy <- 2 * y
  return(c(gx, gy))
}

# Parámetros iniciales
x <- 3       # valor inicial de x
y <- 4       # valor inicial de y
alpha <- 0.1  # tasa de aprendizaje
tol <- 1e-6   # tolerancia
max_iter <- 1000  # máximo de iteraciones

# Iteraciones
```

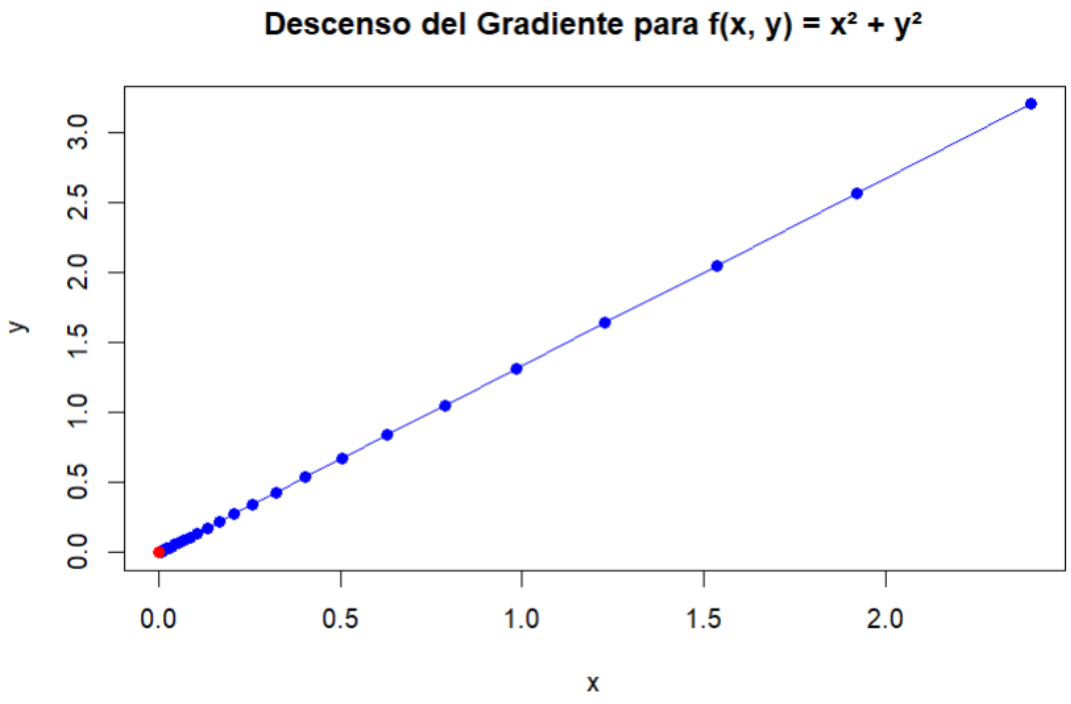
```
iter <- 0
historial <- data.frame(iteracion = numeric(), x = numeric(), y = numeric(), fxy = numeric())

repeat {
  iter <- iter + 1
  g <- gradiente(x, y)

  # Actualizamos los valores
  x_new <- x - alpha * g[1]
  y_new <- y - alpha * g[2]

  # Guardamos el historial
  historial <- rbind(historial, data.frame(iteracion = iter, x = x_new, y = y_new, fxy = f(x_new, y_new)))

  # Condición de parada
  if (sqrt(sum(g^2)) < tol || iter >= max_iter) {
    break
  }

  # Actualizamos variables
  x <- x_new
  y <- y_new
}

# Mostramos resultados
cat("Número de iteraciones:", iter, "\n")
cat("Mínimo aproximado en: (", round(x_new, 6), ",", round(y_new, 6), ")\n")
cat("Valor de f(x, y):", round(f(x_new, y_new), 6), "\n")

# Mostrar las primeras iteraciones
print(head(historial, 10))
```

```
# Graficar la trayectoria
plot(historial$x, historial$y, type = "o", col = "blue", pch = 19,
     main = "Descenso del Gradiente para f(x, y) = x² + y²",
     xlab = "x", ylab = "y")
points(0, 0, col = "red", pch = 19)
```

SALIDA:

```
>
> source("~/.active-rstudio-document")
Número de iteraciones: 74
Mínimo aproximado en: ( 0 , 0 )
Valor de f(x, y): 0
   iteracion         x         y        fxy
1          1 2.4000000 3.2000000 16.0000000
2          2 1.9200000 2.5600000 10.2400000
3          3 1.5360000 2.0480000  6.5536000
4          4 1.2288000 1.6384000  4.1943040
5          5 0.9830400 1.3107200  2.6843546
6          6 0.7864320 1.0485760  1.7179869
7          7 0.6291456 0.8388608  1.0995116
8          8 0.5033165 0.6710886  0.7036874
9          9 0.4026532 0.5368709  0.4503600
10        10 0.3221225 0.4294967  0.2882304
>
```

GRAFICO:



TRABAJO_2:

El trabajo numero consuste en realizar una comparación de métodos de OLS y GD en “R”, de tal manera que muestre el tiempo de procesamiento con la librería “profvis”.

CODIGO:

```
library(profvis)
set.seed(123)
n <- 10000
x <- runif(n, 0, 10)
y <- 5 + 3*x + rnorm(n, 0, 1)
X <- cbind(1, x)

# Función OLS (solución cerrada)
ols_solver <- function(X, y) {
  solve(t(X) %*% X) %*% t(X) %*% y
}

#  Función GD (descenso de gradiente)
gd_solver <- function(X, y, lr = 0.000001, iter = 1000) {
  m <- nrow(X)
  beta <- matrix(0, ncol(X), 1)
  for (i in 1:iter) {
    grad <- (t(X) %*% (X %*% beta - y)) / m
    beta <- beta - lr * grad
  }
  return(beta)
}

#  Comparación de tiempos con profvis
profvis({
  cat("=== MÉTODO OLS ===\n")
  tiempo_ols <- system.time({
    beta_ols <- ols_solver(X, y)
```

```
})
print(beta_ols)
print(tiempo_ols)


cat("\n=== MÉTODO GD ===\n")
tiempo_gd <- system.time({
  beta_gd <- gd_solver(X, y, lr = 0.000001, iter = 1000)
})
print(beta_gd)
print(tiempo_gd)
})
```

SALIDA:

```
> source("~/.active-rstudio-document")
=== MÉTODO OLS ===
      [,1]
  5.022224
x 2.995935
   user  system elapsed
      0       0       0

=== MÉTODO GD ===
        [,1]
  0.01961444
x 0.12171119
   user  system elapsed
   0.15    0.01    0.17
```

CONCLUCION:

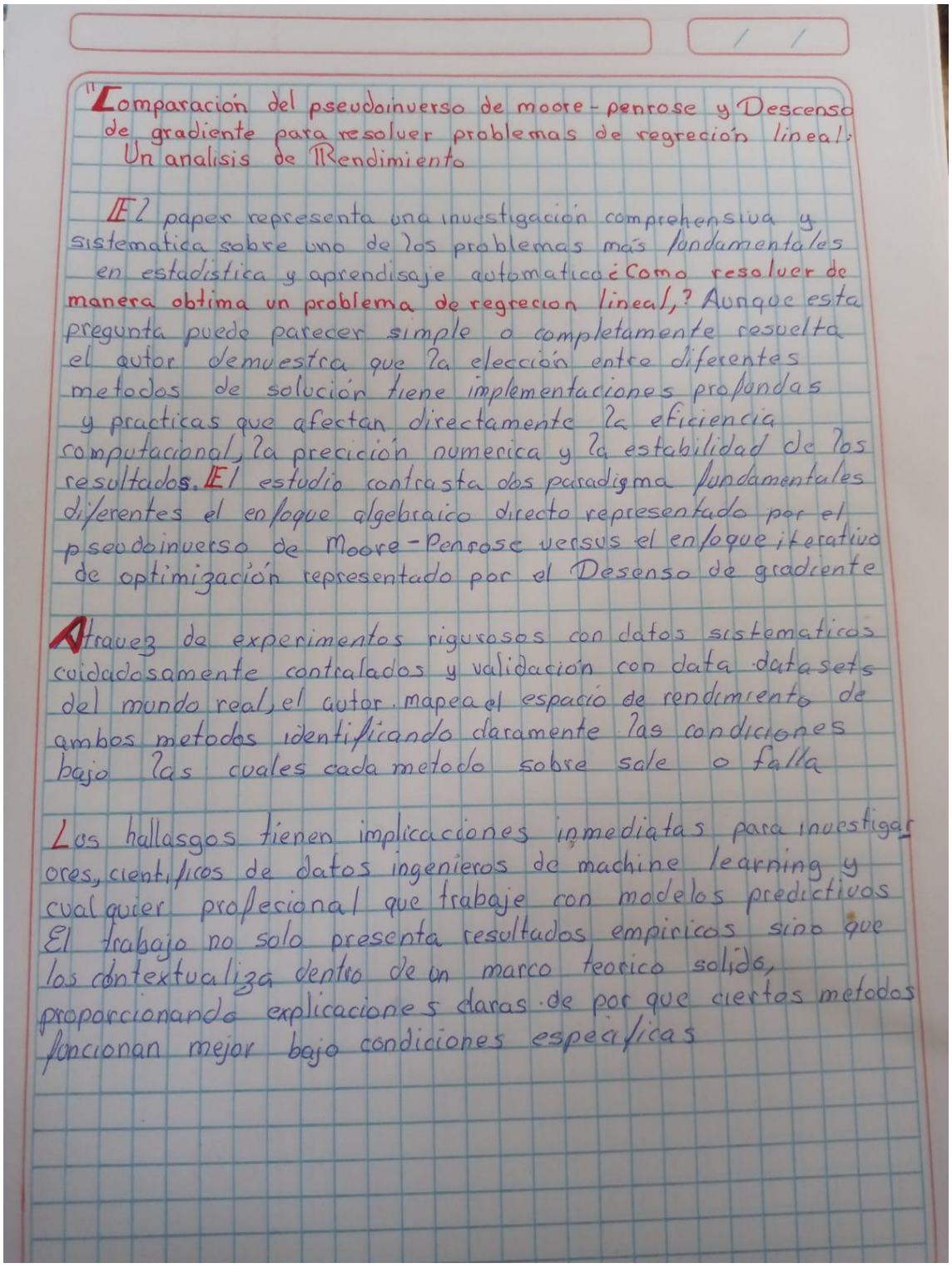
Mi conclusión seria de que en cuestiones de velocidad de calculo seria de que OLS es mas rápida para conjunto de datos pequeños o medianos por que obtiene la solución directamente mediante una formula cerrada, sin embargo, con datos mas grandes se vuelve más lento o hasta incluso imposible cuando con muy grandes y lo ideal seria para uso de datos pequeños y medianos.

Por otro lado GD es mas lento ya que necesita de mas iteraciones para converger y la velocidad depende mucho de learning rate y del número de iteraciones, lo bueno de la GD es que puede trabajar con métodos muy grandes sin calcular inversas y es perfecto para modelos grandes,

OLS= trabaja con datos pequeños y medianos.

GD = excelente para datos grandes.

RESUMEN: DEL PE



"Comparación del pseudoinverso de moore-penrose y Descenso de gradiente para resolver problemas de regresión lineal: Un analisis de Rendimiento

El paper representa una investigación comprehensiva y sistematica sobre uno de los problemas más fundamentales en estadistica y aprendisaje automatico ¿Como resolver de manera optima un problema de regresion lineal,? Aunque esta pregunta puede parecer simple o completamente resuelta el autor demuestra que la elección entre diferentes metodos de solución tiene implementaciones profundas y practicas que afectan directamente la eficiencia computacional, la precición numerica y la estabilidad de los resultados. El estudio contrasta dos paradigma fundamentales diferentes el enfoque algebraico directo representado por el pseudoinverso de Moore-Penrose versus el enfoque iterativo de optimización representado por el Desenso de gradiente

Atravez de experimentos rigurosos con datos sistematicos cuidadosamente contralados y validación con data datasets del mundo real, el autor mapea el espacio de rendimiento de ambos metodos identificando claramente las condiciones bajo las cuales cada metodo sobre sale o falla

Los hallasgos tienen implicaciones inmediatas para investigadores, cientificos de datos ingenieros de machine learning y cualquier profesional que trabaje con modelos predictivos El trabajo no solo presenta resultados empiricos sino que los contextualiza dentro de un marco teorico solido, proporcionando explicaciones claras de por que ciertos metodos funcionan mejor bajo condiciones especificas

TAREA NUMERO_4

En los modelos de regresión lineal, el objetivo es encontrar los coeficientes que mejor ajustan una relación entre una variable dependiente (y) y una o más variables independientes (x).

el ols es un método analítico y exacto, mientras que el gradiente es numérico e iterativo.
en términos de precisión, ols suele ser más exacto porque calcula los coeficientes directamente, mientras que gd depende del número de iteraciones y de la tasa de aprendizaje elegida.
en cuanto a velocidad, ols es más rápido cuando se trabaja con pocos datos o modelos simples, pero se vuelve más lento y consume más memoria a medida que el tamaño de los datos crece, ya que requiere invertir matrices grandes.
el gradiente, aunque más lento en converger, puede manejar volúmenes de datos mucho mayores porque no necesita almacenar ni invertir matrices completas.
el costo computacional de ols crece de manera cúbica con el tamaño del conjunto de datos, mientras que el gradiente tiene un costo proporcional al número de iteraciones y la cantidad de datos procesados.
la principal ventaja de ols es su simplicidad y exactitud, mientras que la del gradiente es su escalabilidad y flexibilidad para aplicarse en modelos grandes o complejos, como los que se usan en el aprendizaje automático.
sin embargo, ols se vuelve ineficiente cuando los datos son muy numerosos o dispersos, y el gradiente puede no converger si se escoge mal la tasa de aprendizaje.
en la práctica, ols se usa comúnmente en la regresión lineal tradicional, mientras que el gradiente es la base de muchos algoritmos modernos de inteligencia artificial y redes neuronales.

En conclusión el método de los cuadrados ordinarios resulta más adecuado para conjuntos de datos pequeños o medianos, donde se busca rapidez y una solución exacta.

el método del gradiente es preferible en problemas con grandes volúmenes de información o modelos complejos, donde la inversión de matrices sería muy costosa o imposible.

en pruebas con datos simulados, ols tiende a ser más rápido en obtener resultados, pero el gradiente, aunque tarda más, se adapta mejor a sistemas de gran escala y con limitaciones de memoria.