# Diferenciación Numérica

# Diferencias Finitas

## 8. Ejercicios Aplicados a Data Science:

### 8.1. Ejercicio 1: Análisis de Crecimiento de Usuarios

Una startup de tecnología registra el número acumulado de usuarios activos mensuales:

| Mes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Usuarios (miles) | 10 | 15 | 23 | 34 | 48 | 65 | 85 |

Código en R:

```
# 8.1

# Datos: meses y usuarios acumulados (miles)

mes  <- 1:7

users <- c(10, 15, 23, 34, 48, 65, 85)

forward_diff <- function(y, x, i, h=1) (y[i + h] - y[i]) / (x[i + h] - x[i])

backward_diff <- function(y, x, i, h=1) (y[i] - y[i - h]) / (x[i] - x[i - h])

central_diff <- function(y, x, i, h=1) (y[i + h] - y[i - h]) / (x[i + h] - x[i - h])

second_diff <- function(y, x, i, h=1) (y[i + h] - 2*y[i] + y[i - h]) / ((x[i + h] - x[i])^2 / (h^2))

# 1) tasa de crecimiento

i <- 4

growth_m4 <- central_diff(users, mes, i, h=1)   # en miles por mes

cat("1) Tasa de crecimiento en mes 4:", growth_m4, "miles/mes\n")

# 2) tasa en mes 1

i <- 1

growth_m1 <- forward_diff(users, mes, i, h=1)

cat("2) Tasa de crecimiento en mes 1:", growth_m1, "miles/mes\n")

# 3) tasa en mes 7

i <- 7

growth_m7 <- backward_diff(users, mes, i, h=1)
```

```
cat("3) Tasa de crecimiento en mes 7:", growth_m7, "miles/mes\n")

# 4) segunda derivada

sec <- rep(NA, length(users))

for (i in 2:(length(users)-1)) {

  sec[i] <- (users[i+1] - 2*users[i] + users[i-1]) / ( (mes[i+1] - mes[i])^2 )

}

cat("4) Segunda derivada por mes (NA en extremos):\n")

print(sec)

# ¿En qué mes fue mayor la aceleración?

idx_max_acc <- which.max(sec, na.rm=TRUE)

cat("Mayor aceleración en mes:", idx_max_acc, "con valor", sec[idx_max_acc],
"miles/mes^2\n")

# 5) ¿acelerada o no?

cat("5) Interpretación: segunda derivada media (omit NA):", mean(sec, na.rm=TRUE),
"\n")

if (mean(sec, na.rm=TRUE) > 0) cat("La startup está creciendo de forma acelerada en
promedio.\n") else cat("Está desacelerándose en promedio.\n")
```

### 8.2. Ejercicio 2: Optimización de Función de Pérdida

Durante el entrenamiento de un modelo de Machine Learning, se registra la función de pérdida (loss) en cada época:

| Época | 0 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| Loss | 2.45 | 1.82 | 1.35 | 1.08 | 0.95 | 0.89 |

Código en R:

```
# Epocas y loss

epoch <- c(0,10,20,30,40,50)

loss  <- c(2.45, 1.82, 1.35, 1.08, 0.95, 0.89)

h <- 10

idx_of <- function(e) which(epoch == e)

# 1) tasa de cambio

i20 <- idx_of(20)
```

```
# centrada: (loss(epoch+h) - loss(epoch-h)) / (2*h)

d_loss_20 <- (loss[i20 + 1] - loss[i20 - 1]) / (2*h)

cat("1) Tasa de cambio (centrada) en epoca 20:", d_loss_20, "loss/epoca\n")

# 2) segunda derivada

i30 <- idx_of(30)

sec_loss_30 <- (loss[i30+1] - 2*loss[i30] + loss[i30-1]) / (h^2)

cat("2) Segunda derivada en epoca 30:", sec_loss_30, " (si >0 => convexidad
positiva)\n")

if (sec_loss_30 < 0) cat("Indica que la curvatura es cóncava hacia abajo: desaceleración
en la caída del loss (puede indicar acercamiento a mínimo).\n")

# 3) ¿En qué epoca la tasa de cambio < 0.01 por epoca?

d_rates <- rep(NA, length(epoch))

for (i in 2:(length(epoch)-1)) {

  d_rates[i] <- (loss[i+1] - loss[i-1]) / (2*(epoch[i+1]-epoch[i-1])/2)

}

d_rates[1] <- (loss[2] - loss[1]) / (epoch[2] - epoch[1])

d_rates[length(epoch)] <- (loss[length(epoch)] - loss[length(epoch)-1]) /
(epoch[length(epoch)] - epoch[length(epoch)-1])

cat("3) Tasa de cambio por epoca (approx):\n")

for (i in seq_along(epoch)) cat(" epoch", epoch[i], " -> rate:", d_rates[i], "\n")

epochs_below_001 <- epoch[which(abs(d_rates) < 0.01)]

cat("Épocas donde |tasa| < 0.01:", if(length(epochs_below_001)>0)
paste(epochs_below_001, collapse=", ") else "ninguna", "\n")

# 4) Estimar loss en epoca 25 usando interpolación lineal basada en derivadas

dL_at_20 <- d_loss_20

L25_by_deriv <- loss[i20] + (25 - 20) * dL_at_20

cat("4) Estimación L(25) usando derivada en 20:", L25_by_deriv, "\n")

L20 <- loss[i20]; L30 <- loss[i20+1]

L25_linear <- L20 + (25-20) * (L30 - L20)/(30-20)

cat("Estimación L(25) por interpolación lineal entre 20 y 30:", L25_linear, "\n")
```

### 8.3. Ejercicio 3: Análisis de Series Temporales de Ventas

Una empresa de e-commerce registra sus ventas diarias (en miles de dólares) durante una semana de campaña:

| Día | Lun | Mar | Mié | Jue | Vie | Sáb | Dom |
|---|---|---|---|---|---|---|---|
| Ventas ($k) | 45 | 52 | 61 | 58 | 73 | 89 | 95 |

Código en R:

```
# 8.3
dias <- c("Lun","Mar","Mie","Jue","Vie","Sab","Dom")
x <- 1:7
ventas <- c(45, 52, 61, 58, 73, 89, 95)
# funciones
forward_diff_v <- function(y, i, h=1) (y[i+h] - y[i]) / h
backward_diff_v <- function(y, i, h=1) (y[i] - y[i-h]) / h
central_diff_v <- function(y, i, h=1) (y[i+h] - y[i-h]) / (2*h)
second_diff_v <- function(y, i, h=1) (y[i+h] - 2*y[i] + y[i-h]) / (h^2)
# 1) velocidad para cada día
vel <- rep(NA, length(ventas))
for (i in 1:length(ventas)) {
  if (i == 1) vel[i] <- forward_diff_v(ventas, i)
  else if (i == length(ventas)) vel[i] <- backward_diff_v(ventas, i)
  else vel[i] <- central_diff_v(ventas, i)
}
cat("1) Velocidad (ventas) por día (miles $/día):\n")
print(data.frame(dia=dias, velocidad=vel))
# 2) día con mayor aceleración
sec <- rep(NA, length(ventas))
for (i in 2:(length(ventas)-1)) sec[i] <- second_diff_v(ventas, i)
idx_max_sec <- which.max(sec, na.rm=TRUE)
cat("2) Mayor aceleración en:", dias[idx_max_sec], "con segunda derivada",
sec[idx_max_sec], "\n")
# 3) El jueves hubo caída: magnitud de desaceleración
i <- 4
```

```
mag_desacel <- sec[i]

cat("3) Magnitud de la desaceleración (segunda derivada) en Jue:", mag_desacel, "miles $/día^2\n")

# 4) Extrapolación lineal:

slope_dom <- vel[length(vel)]

pred_lun <- ventas[length(ventas)] + slope_dom * 1  # 1 día ahead

cat("4) Predicción ventas Lunes siguiente (miles $):", pred_lun, "\n")
```

### 8.4. Ejercicio 4: Gradiente de Función de Activación

Enredes neuronales, necesitas calcular el gradiente de la función sigmoide $\sigma(x) = 1/(1+e^{-x})$ en varios puntos para el backpropagation. No tienes la derivada anal´ıtica programada.

Datos evaluados de la función:

| x | -3.0 | -2.0 | -1.0 | 0.0 | 1.0 | 2.0 | 3.0 |
|---|---|---|---|---|---|---|---|
| σ(x) | 0.0474 | 0.1192 | 0.2689 | 0.5000 | 0.7311 | 0.8808 | 0.9526 |

Código en R:

```
# Dato:

x <- c(-3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0)

sigma <- c(0.0474, 0.1192, 0.2689, 0.5000, 0.7311, 0.8808, 0.9526)

# funciones

central_diff_vals <- function(y, i, h=1) {

  .if (i <= 1 || i >= length(y)) return(NA)

  (y[i + h] - y[i - h]) / (2*h)

}

i0 <- which(x == 0.0)

sigma_prime_0_num <- central_diff_vals(sigma, i0, h=1)

cat("1) σ'(0) (num, centrada h=1):", sigma_prime_0_num, "\n")

i_m2 <- which(x == -2.0)

i_p2 <- which(x == 2.0)

sigma_prime_m2 <- central_diff_vals(sigma, i_m2, h=1)

sigma_prime_p2 <- central_diff_vals(sigma, i_p2, h=1)
```

```
cat("2) σ'(-2) num:", sigma_prime_m2, "  σ'(2) num:", sigma_prime_p2, "\n")

sigma_prime_analytic <- sigma * (1 - sigma)

cat("3) Comparación numérica vs analítica :\n")

comp <- data.frame(x=x, sigma=sigma, num_deriv=c(NA, sigma_prime_m2, NA,
sigma_prime_0_num, NA, sigma_prime_p2, NA),

                   analytic = sigma_prime_analytic)

print(comp)

cat("Comparación en -2:\n num=", sigma_prime_m2, " analytic=",
sigma_prime_analytic[i_m2], "\n")

cat("Comparación en 0:\n num=", sigma_prime_0_num, " analytic=",
sigma_prime_analytic[i0], "\n")

cat("Comparación en 2:\n num=", sigma_prime_p2, " analytic=",
sigma_prime_analytic[i_p2], "\n")

cat("4) Recomendación: usar h pequeño (ej. 0.1 o 0.01) si puedes evaluar σ(x) con
mayor resolución.\n")

cat("Razon: centrada reduce error O(h^2); con h=1 la aproximación es cruda porque los
puntos están separados.\n")

# 5) ¿Por qué derivada es simétrica alrededor de 0?

cat("5) La sigmoide es simétrica (anti-impar) respecto a x=0 en su pendiente: σ(-x)=1-
σ(x),\n")

cat("    lo que hace que σ'(x) sea par (simétrica): σ'(-x) = σ'(x).\n")
```

### 8.5. Ejercicio 5: Detección de Anomalías en Métricas de Sistema

Un sistema de monitoreo registra el tiempo de respuesta (en ms) de una API cada hora durante un incidente:

| Hora | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Latencia (ms) | 120 | 125 | 128 | 135 | 280 | 290 | 275 | 155 |

Código en R:

```
hora <- 0:7

latency <- c(120,125,128,135,280,290,275,155)

# diferencias

forward <- function(y,i) if(i < length(y)) y[i+1] - y[i] else NA

backward <- function(y,i) if(i > 1) y[i] - y[i-1] else NA
```

```
central <- function(y,i) if(i>1 && i<length(y)) (y[i+1] - y[i-1]) / 2 else NA

second <- function(y,i) if(i>1 && i<length(y)) y[i+1] - 2*y[i] + y[i-1] else NA

# 1) primera derivada por hora

first_deriv <- sapply(seq_along(latency), function(i) {

  if (i==1) forward(latency,i) else if (i==length(latency)) backward(latency,i) else
central(latency,i)

})

cat("1) Primera derivada (ms/h) por hora:\n")

print(data.frame(hora=hora, latency=latency, d1=first_deriv))

# 2) identificar pico de anomalía

sec <- sapply(seq_along(latency), function(i) second(latency,i))

# buscar indices i where sec[i-1] > 0 and sec[i+1] < 0  (approx)

change_points <- c()

for (i in 3:(length(latency)-2)) {

  if (!is.na(sec[i-1]) && !is.na(sec[i+1])) {

    if (sec[i-1] > 0 && sec[i+1] < 0) change_points <- c(change_points, i)

  }

}

cat("2) Segunda derivada por hora (NA en extremos):\n"); print(sec)

if (length(change_points)>0) cat("Cambio de signo +->- alrededor de horas (indices):",
paste(change_points, collapse=", "), "\n") else cat("No se detectó un claro +->- en
vecinos inmediatos.\n")

# 3) magnitud salto brusco entre horas 3 y 4

jump_3_4 <- latency[5] - latency[4]

cat("3) Salto entre hora 3 y 4: lat(h4)-lat(h3) =", jump_3_4, "ms\n")

# 4) A partir de hora 6, recuperación: tasa de recuperación

i6 <- which(hora == 6)

recovery_rate <- if(i6 < length(latency)) (latency[i6+1] - latency[i6]) else NA

cat("4) Tasa de recuperación a partir de hora 6 (ms/h):", recovery_rate, "\n")

# 5) anomalía

threshold <- 50
```

```
anoms <- which(abs(c(NA, diff(latency))) > threshold)  # diff gives change from i to i+1 at index i

# convert to hours where change occurs (between hora[i] and hora[i+1])

if (length(anoms) > 0) {

  cat("5) Anomalías detectadas entre horas:\n")

  for (a in anoms) cat(" entre", hora[a-1+1], "y", hora[a+1], " (cambio:", diff(latency)[a], "ms )\n")

} else cat("5) No se detectaron anomalías > 50 ms/hora\n")
```

### 8.6. Ejercicio 6: Análisis de Tasa de Conversión

Una campaña de marketing muestra la siguiente tasa de conversión (porcentaje) en función del gasto en publicidad (en miles de dólares):

| Gasto ($k) | 0 | 5 | 10 | 15 | 20 | 25 |
|---|---|---|---|---|---|---|
| Conversión (%) | 2.1 | 3.8 | 5.2 | 6.1 | 6.7 | 7.0 |

Código en R:

```
gasto <- c(0,5,10,15,20,25)

conv  <- c(2.1, 3.8, 5.2, 6.1, 6.7, 7.0)

h <- 5

roi_marginal_per5k <- rep(NA, length(conv))

for (i in 2:(length(conv)-1)) {

  roi_marginal_per5k[i] <- (conv[i+1] - conv[i-1]) / (gasto[i+1] - gasto[i-1])  # % per k$

}

roi_marginal_per5k[1] <- (conv[2] - conv[1]) / (gasto[2] - gasto[1])

roi_marginal_per5k[length(conv)] <- (conv[length(conv)] - conv[length(conv)-1]) / (gasto[length(gasto)] - gasto[length(gasto)-1])

cat("1) ROI marginal (% por $1k) en cada punto:\n")

print(data.frame(gasto=gasto, conversion=conv,
roi_marginal_pct_per_k=roi_marginal_per5k))

idx_gt_02 <- which(roi_marginal_per5k > 0.2)

cat("2) Rangos donde ROI marginal > 0.2%/k$ ocurren en índices:", idx_gt_02, "\n")

if (length(idx_gt_02)>0) print(data.frame(gasto=gasto[idx_gt_02],
roi=roi_marginal_per5k[idx_gt_02]))
```

```
i15 <- which(gasto == 15)

sec_at_15 <- (conv[i15+1] - 2*conv[i15] + conv[i15-1]) / ((gasto[i15+1] - gasto[i15])^2)

cat("3) Segunda derivada en $15k:", sec_at_15, " (% per (k$)^2 )\n")

cat("4) Interpretación: si segunda derivada negativa => rendimientos decrecientes.\n")

cat("En nuestros datos la ROI marginal en 25k es:", roi_marginal_per5k[length(roi_marginal_per5k)], "\n")

cat("Como la ROI marginal cae (y segunda derivada tiende a ser negativa), matemáticamente no es rentable aumentar indefinidamente más allá de 25k salvo que haya objetivos no marginales.\n")
```

### 8.7. Ejercicio 7: Feature Engineering con Derivadas

Tienes un dataset con la señal de un sensor de temperatura en un proceso industrial medida cada segundo:

| Tiempo (s) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Temp (°C) | 20.1 | 20.3 | 20.8 | 21.5 | 22.6 | 24.2 | 26.1 | 28.5 |

Código en R:

```
time <- 0:7

temp <- c(20.1,20.3,20.8,21.5,22.6,24.2,26.1,28.5)  # °C, cada segundo

n <- length(temp)

vel <- rep(NA, n)

acc <- rep(NA, n)

for (i in 1:n) {

  if (i==1) vel[i] <- (temp[i+1] - temp[i]) / (time[i+1] - time[i])

  else if (i==n) vel[i] <- (temp[i] - temp[i-1]) / (time[i] - time[i-1])

  else vel[i] <- (temp[i+1] - temp[i-1]) / (time[i+1] - time[i-1])

}

for (i in 2:(n-1)) {

  acc[i] <- (temp[i+1] - 2*temp[i] + temp[i-1]) / ((time[i+1]-time[i])^2)

}

features <- data.frame(time=time, temp=temp, vel=vel, acc=acc)

cat("1-2) Features (velocidad y aceleración):\n")
```

```
print(features)

alerts <- which(vel > 0.8)

cat("3) Alertas por vel > 0.8 °C/s en tiempos:\n")

if(length(alerts)>0) print(features[alerts,]) else cat("No hay alertas.\n")

min_max <- function(v) {

  if(all(is.na(v))) return(rep(NA,length(v)))

  (v - min(v, na.rm=TRUE)) / (max(v, na.rm=TRUE) - min(v, na.rm=TRUE))

}

features$vel_norm <- min_max(features$vel)

features$acc_norm <- min_max(features$acc)

cat("4) Features normalizadas (min-max):\n")

print(features)

cat("5) Justificación:\n")

cat(" - Velocidad y aceleración capturan cambios locales y patrones temporales que no aparecen en la señal cruda.\n")

cat(" - Para clasificación de anomalías, estas derivadas resaltan picos repentinos y cambios de tendencia, facilitando que modelos detecten eventos anómalos.\n")
```