Lista 3: Otimização em RNA

César A. Galvão - 190011572

1 Questão 1

Considere a função

$$f(x_1, x_2) = x_1^4 + x_2^4 + x_1^2 x_2 + x_1 x_2^2 - 20x_1^2 - 15x_2^2$$

para responder os itens a seguir.

1.1 Item a)

Apresente um gráfico com as curvas de nível de $f(x_1, x_2)$. Quantos pontos críticos a função parece ter? Dica para usuários do R: use a função `geom_contour_filled()`.

A Figura 1 apresenta o gráfico com curvas de nível. A superfície tem 3 pontos de mínimo local e um ponto de mínimo global, que parece ocorrer próximo de $(-3, -3)$. Se forem considerados pontos de sela, mais pontos podem ser considerados, como aqueles entre os mínimos e possívelmente algum ponto próximo da origem, totalizando entre 8 e 9 pontos críticos.

```r
# funcao
f <- function(x1, x2) {
  x1^4 + x2^4 + x1^2*x2 + x1*x2^2 - 20*x1^2 - 15*x2^2
}
# grid dos dados
df <- tibble(
  expand.grid(x1 = seq(-5, 5, length.out = 100),
              x2 = seq(-5, 5, length.out = 100)),
  f = f(x1, x2))

surface <- ggplot() +
  geom_contour_filled(data = df, aes(x = x1, y = x2, z = f)) +
  labs(x = TeX("$x_1$"), y = TeX("$x_2$"))+
  theme_minimal()+
  theme(legend.position = "none",
        panel.grid = element_blank(),
        axis.title.y = element_text(angle = 0, vjust = 0.5))
```

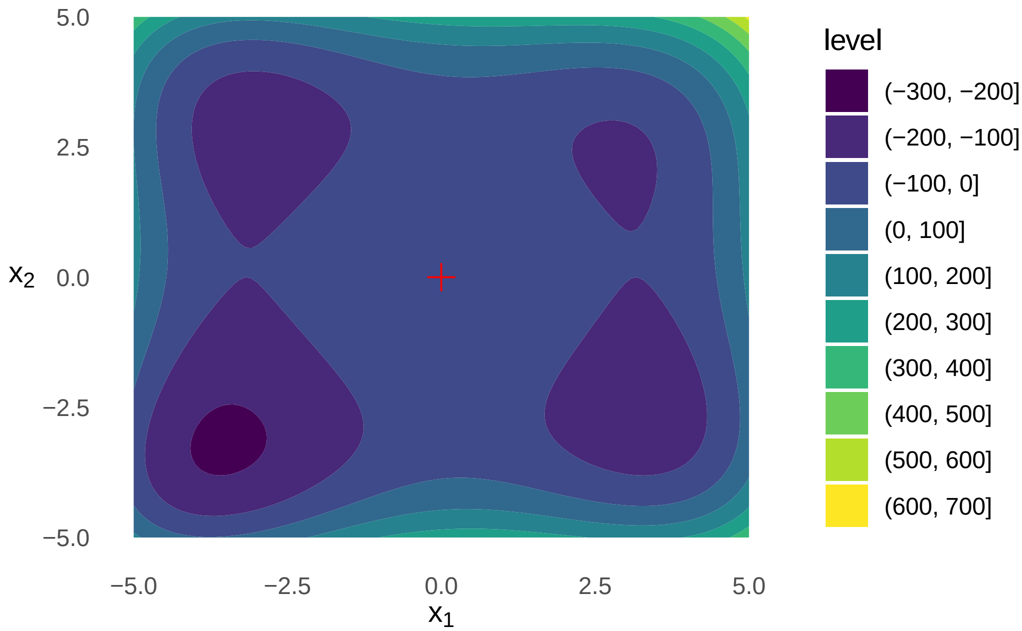


Figura 1: Curvas de nível de $f(x_1, x_2)$

1.2 Item b)

Encontre (algericamente) o gradiente de f em relação ao vetor $x = (x_1, x_2)$. Isso é,

$$\nabla_x f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right).$$

$$\begin{aligned}\nabla_x f(\boldsymbol{x}) &= \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \\ \frac{\partial f}{\partial x_2} \end{bmatrix} \\ &= \begin{bmatrix} 4x_1^3 + 2x_2x_1 + x_2^2 - 40x_1 \\ \\ 4x_2^3 + 2x_1x_2 + x_1^2 - 30x_2 \end{bmatrix} \end{aligned} \tag{1}$$

1.3 Item c)

Crie uma função computacional que implemente o método do gradiente para minimizar a função emestudo. Permita ao usuário definir a taxa de aprendizado, o número de passos e o ponto de partida.

A função **gradiente()** implementa o método do gradiente para minimizar a função $f(x_1, x_2)$. Os argumentos padrão são:

- x_0: o ponto de partida, tomando a origem como padrão;
- `l.rate`: a taxa de aprendizado, com valor padrão de 0.01;
- **steps**: o número de passos, com valor padrão de 1000;
- **keep**: indica se os pontos intermediários devem ser mantidos.

A função retorna o ponto de mínimo encontrado depois de **steps** passos e única função que a função minimiza é a que foi indicada na lista.

```r
# funcao do gradiente
grad <- function(x) {
    c(4*x[1]^3 + 2*x[2]*x[1] + x[2]^2 - 40*x[1],
      4*x[2]^3 + 2*x[1]*x[2] + x[1]^2 - 30*x[2])
  }

gradiente <- function(x0 = c(0,0), l.rate = 0.01, steps = 100, keep = FALSE) {

  if(keep == FALSE){
    x <- x0
    # loop padrão
    for (i in 1:steps) {
      x <- x - l.rate * grad(x)
    }
  } else {
      # loop mantendo pontos intermediários
      x <- matrix(NA, nrow = steps+1, ncol = 2)
      x[1,] <- x0
      for (i in 1:steps) {
        x[i+1,] <- x[i,] - l.rate * grad(x[i,])
      }
  }

  return(x)
}
```

1.4 Item d)

Use a função criada no item c) para encontrar o valor obtido pelo método do gradiente partindo-se do ponto inicial $\left(x_1^{(0)}, x_2^{(0)}\right) = (0, 5)$. Use taxa de aprendizado igual a 0.01 e execute 100 passos.

```r
gradiente(x0 = c(0,5), l.rate = 0.01, steps = 100)
```

```
[1] -3.040141  2.865981
```

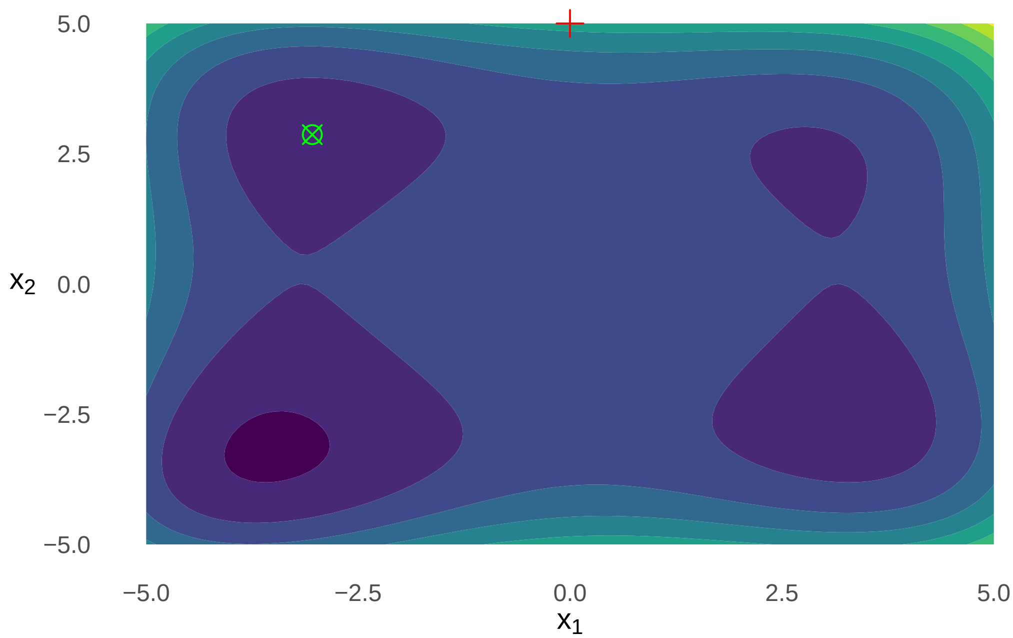


Figura 2: Ponto final do método do gradiente partindo de (0,5)

1.5 Item e)

Repita o item d), agora com as seguintes taxas de aprendizado: 1, 0.1, 0.01, 0.001, 0.0001. Qual dessas opções lhe parece mais apropriada nesse caso? Justifique sua resposta.

A Tabela 1 apresenta os resultados obtidos para diferentes taxas de aprendizado. A taxa de aprendizado de 0.01 parece ser a mais apropriada, pois é a única que converge para o mínimo local da função. A partir da iteração 22 não é mais possível observar mudanças na direção do gradiente considerando 6 casas decimentis. As taxas de aprendizado 0.001 e 0.0001 possivelmente convergiriam, mas exigiriam muito mais passos.

As taxas de aprendizado 1 e 0.1 divergem rapidamente. Suas primeiras iterações já direcionam o próximo ponto para fora do mapa considerado, rapidamente indo para infinito. Isso provavelmente ocorre devido à magnitude do vetor gradiente nos primeiros pontos do algoritmo do gradiente.

```
q1e_table <- bind_rows(
  as_tibble(gradiente(x0 = c(0,5), l.rate = 1, steps = 100, keep = TRUE)),
  as_tibble(gradiente(x0 = c(0,5), l.rate = 0.1, steps = 100, keep = TRUE)),
  as_tibble(gradiente(x0 = c(0,5), l.rate = 0.01, steps = 100, keep = TRUE)),
  as_tibble(gradiente(x0 = c(0,5), l.rate = 0.001, steps = 100, keep = TRUE)),
  as_tibble(gradiente(x0 = c(0,5), l.rate = 0.0001, steps = 100, keep = TRUE))
) %>%
```

```r
  mutate(l.rate = rep(c('1', '0.1', '0.01', '0.001', '0.0001'), each = 101),
         step = rep(0:100, times = 5),
         across(everything(), ~ if_else(is.nan(.), NA, .)))

tibble(
  l.rate = c('1', '0.1', '0.01', '0.001', '0.0001'),
  conv = c("não", "não", "sim", "sim*", "sim*")
) %>%
  knitr::kable(
    align = "lc",
    col.names = c("Learning rate", "Converge?")
  )
```

Tabela 1: Resultados do método do gradiente para diferentes taxas de aprendizado

Learning rate	Converge?
1	não
0.1	não
0.01	sim
0.001	sim*
0.0001	sim*

1.6 Item f)

Fixe a semente do gerador de números aleatórios no valor 123 (se estiver usando o R, basta executar o código `set.seed(123)`). Repita novamente o item d), agora partindo de 20 pontos escolidos aleatoriamente (uniformemente) no quadrado $5 < x_1, x_2 < 5$. Refaça o gráfico do item a) e adicione uma linha representando o caminho percorrido por cada uma das 20 otimizações. Qual foi o percentual de vezes em que o algoritmo encontrou o mínimo global da função (despresando um eventual desvio de menor importância)?

A seguir são exibidos os 20 pontos gerados aleatoriamente e seus pontos finais. A Figura 3 apresenta o caminho percorrido por cada uma das 20 otimizações. O percentual de vezes em que o algoritmo encontrou o mínimo global da função foi de 20%.

```r
#semente aleatoria
set.seed(123)
# 20 pontos iniciais
x0 <- tibble(x1 = runif(20, -5, 5),
             x2 = runif(20, -5, 5))
# 20 pontos finais
end_points <- x0 %$%
  map2(x1, x2, ~gradiente(c(.x, .y), l.rate = 0.01, steps = 100)) %>%
```

```r
  map(., ~ t(as.matrix(.))) %>%
  map2(., seq_along(.), ~ as_tibble(.x) %>% mutate(id = .y)) %>%
  bind_rows() %>%
  mutate(id = as.factor(id)) %>%
  rlang::set_names(c("x1", "x2", "id"))

# resultados
paths <- x0 %$%
  map2(x1, x2, ~gradiente(c(.x, .y), l.rate = 0.01, steps = 100, keep = TRUE)) %>%
  map2(., seq_along(.), ~ as_tibble(.x) %>% mutate(id = .y)) %>%
  bind_rows() %>%
  mutate(id = as.factor(id)) %>%
  rlang::set_names(c("x1", "x2", "id"))

# gráfico
# Assuming x0 has columns x1, x2, and y for labels
# Add the labels to x0 if not already present
x0 <- x0 %>% mutate(y = paste0("(", round(x1, 2), ", ", round(x2, 2), ")"))

ggplot() +
  geom_contour_filled(data = df, aes(x = x1, y = x2, z = f)) +
  geom_path(data = paths, aes(x = x1, y = x2, group = id), color = "yellow",
  ↪  linetype = "dotted") +
  geom_label(data = x0, aes(x = x1, y = x2, label = c(1:20)), color = "red", size =
  ↪  2) +
  geom_point(data = end_points, aes(x = x1, y = x2), color = "green", size = 2,
  ↪  shape = 13) +
  labs(x = TeX("$x_1$"), y = TeX("$x_2$")) +
  theme_minimal() +
  theme(legend.position = "none",
        panel.grid = element_blank(),
        axis.title.y = element_text(angle = 0, vjust = 0.5))
```

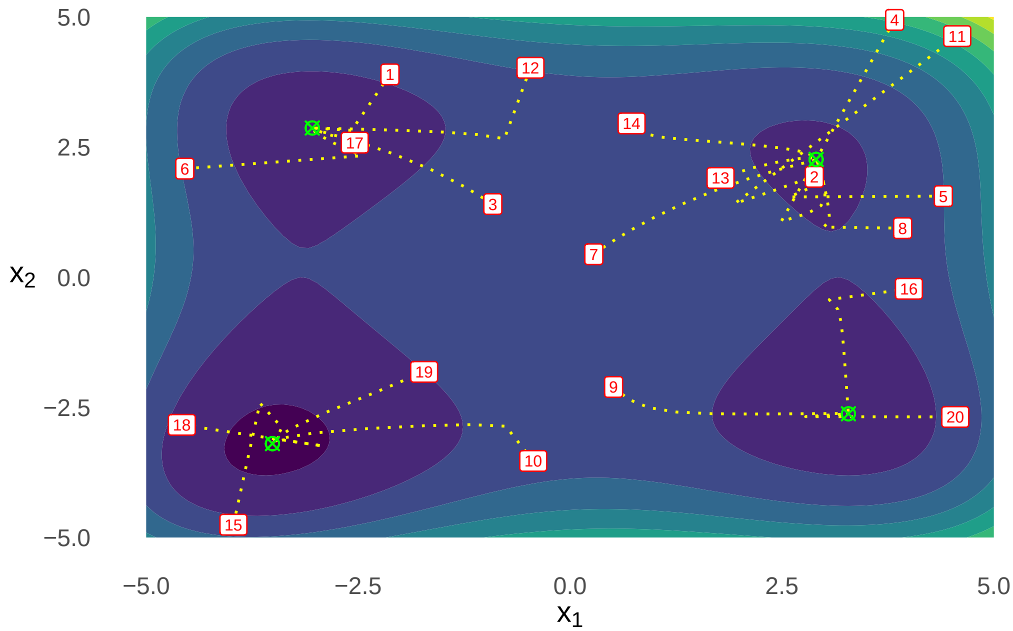


Figura 3: Caminho percorrido por 20 otimizações

g) Repita o item d), substituindo o método do gradiente pelo método do gradiente com momento (veja a Seção 8.3.2 do livro *Deep Learning*). Use taxa de aprendizado $\epsilon = 0.01$, parâmetro de momento $\alpha = 0.9$ e velocidade inicial $v = 0$.

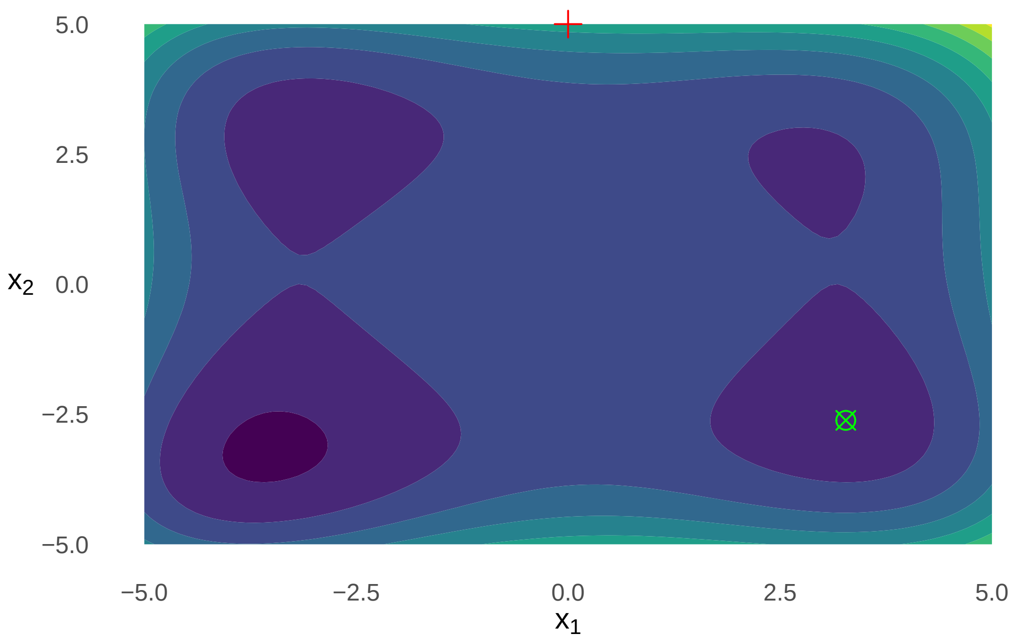
```
gradiente_momento <- function(x0 = c(0,0), l.rate = 0.01, alpha = 0.9, steps = 100,
↪  keep = FALSE) {

  if(keep == FALSE){
    x <- x0
    v <- 0
    # loop padrão
    for (i in 1:steps) {
      #update velocity
      v <- alpha * v - l.rate * grad(x)
      x <- x + v
    }
  } else {
    # loop mantendo pontos intermediários
    x <- matrix(NA, nrow = steps+1, ncol = 2)
    x[1,] <- x0
    v <- 0
    for (i in 1:steps) {
      #update velocity
```

```r
      v <- alpha * v - l.rate * grad(x[i,])
      x[i+1,] <- x[i,] + v
    }
  }
  return(x)
}

q1g <- gradiente_momento(x0 = c(0,5))

surface+
  geom_point(aes(x = 0, y = 5), color = "red", size = 3, shape = 3)+
  geom_point(aes(x = q1g[1], y = q1g[2]), color = "green", size = 3, shape = 13)
```



1.7 Item h)

Repita o item d), substituindo o método do gradiente pelo método RMSProp (veja a Seção 8.5.2 do livro Deep Learning). Use taxa de aprendizado $\epsilon = 0.001$, taxa de decaimento $\rho = 0.9$ e constante $\delta = 10^6$.

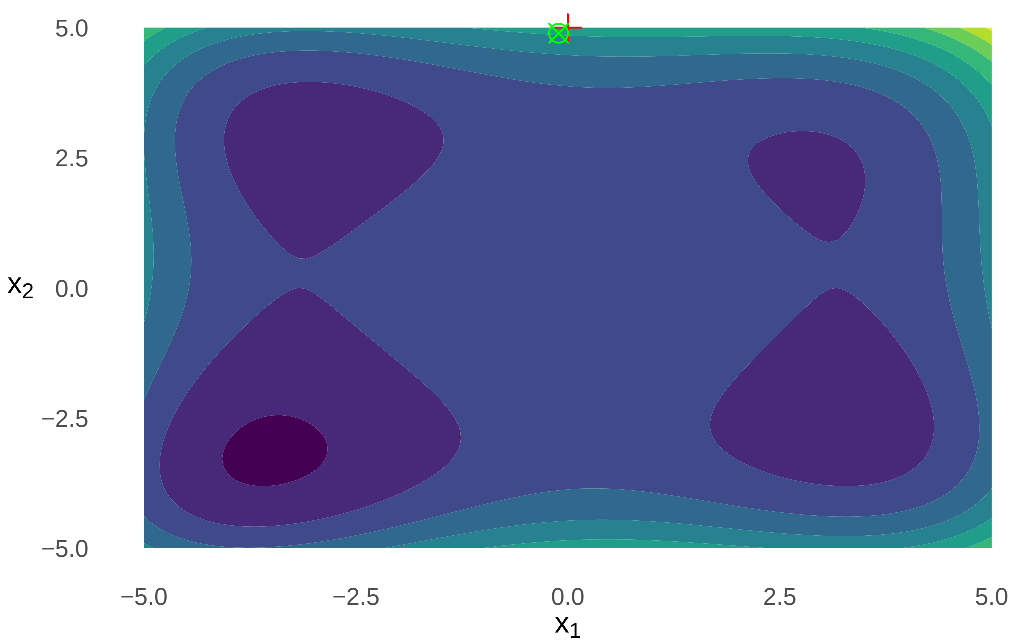
```r
gradiente_RMSProp <- function(x0 = c(0,0), l.rate = 0.001, rho = 0.9, delta =
↪  10^(-6), steps = 100, keep = FALSE) {

  if(keep == FALSE){
    x <- x0
    s <- 0
```

```r
    # loop padrão
    for (i in 1:steps) {
      #update s
      s <- rho * s + (1 - rho) * grad(x)^2
      #parameter update (theta + delta*theta)
      # theta = theta - l.rate / sqrt(s + delta) * grad
      x <- x - l.rate* grad(x) / sqrt(s + delta)
    }
  } else {
    # loop mantendo pontos intermediários
    x <- matrix(NA, nrow = steps+1, ncol = 2)
    x[1,] <- x0
    s <- 0
    for (i in 1:steps) {
      #update s
      s <- rho * s + (1 - rho) * grad(x[i,])^2
      x[i+1,] <- x[i,] - l.rate* grad(x[i,])/sqrt(s + delta)
    }
  }
  return(x)
}

q1h <- gradiente_RMSProp(x0 = c(0,5), keep = FALSE)

surface+
  geom_point(aes(x = 0, y = 5), color = "red", size = 3, shape = 3)+
  geom_point(aes(x = q1h[1], y = q1h[2]), color = "green", size = 3, shape = 13)
```



1.8 Item i)

Repita o item d), substituindo o método do gradiente pelo método ADAM (veja a Seção 8.5.3 do livro Deep Learning). Use taxa de aprendizado $\epsilon = 0.001$ e taxas de decaimento $\rho 1 = 0.9$ e $\rho_2 = 0.999$.

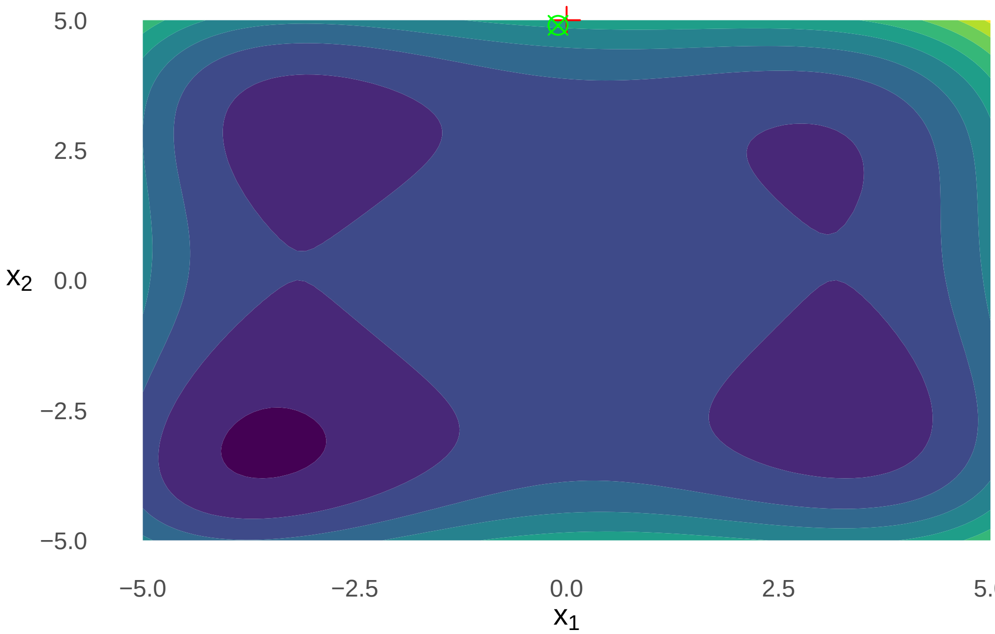
```
gradiente_ADAM <- function(x0 = c(0,0), l.rate = 0.001, rho1 = 0.9, rho2 = 0.999,
↪  delta = 10^(-6), steps = 100, keep = FALSE) {

  if(keep == FALSE){
    x <- x0
    # initialize moment variables
    s <- 0
    r <- 0
    # loop padrão
    for (i in 1:steps) {
      #update first moment estimate
      s <- rho1 * s + (1 - rho1) * grad(x)
      # update second moment estimate
      r <- rho2 * r + (1 - rho2) * grad(x)^2
      #correct bias in first moment
      s_hat <- s / (1 - rho1^i)
      #correct bias in second moment
      r_hat <- r / (1 - rho2^i)
      #apply update (theta = theta + delta*theta)
      # delta*theta = -l.rate*(s_hat) *(sqrt(r_hat) + delta)
      x <- x - l.rate*s_hat / ((sqrt(r_hat) + delta))
    }
  } else {
    # loop mantendo pontos intermediários
    x <- matrix(NA, nrow = steps+1, ncol = 2)
    x[1,] <- x0
    s <- 0
    r <- 0
    for (i in 1:steps) {
      #update m and v
      s <- rho1 * s + (1 - rho1) * grad(x[i,])
      r <- rho2 * r + (1 - rho2) * grad(x[i,])^2
      #bias correction
      s_hat <- s / (1 - rho1^i)
      r_hat <- r / (1 - rho2^i)
      #parameter update
      x[i+1,] <- x[i,] - l.rate*s_hat / ((sqrt(r_hat) + delta))
    }
  }

  return(x)
}

q1j <- gradiente_ADAM(x0 = c(0,5), keep = FALSE)
```

```r
surface+
  geom_point(aes(x = 0, y = 5), color = "red", size = 3, shape = 3)+
  geom_point(aes(x = q1j[1], y = q1j[2]), color = "green", size = 3, shape = 13)
```



1.9 Item j)

Apresente graficamente, em uma única figura, os caminhos percorridos pelas otimizações executadas nos itens d), g), h) e i).

```r
x0 <- c(0,5)

paths <- bind_rows(
  as_tibble(gradiente(x0, keep = TRUE)),
  as_tibble(gradiente_momento(x0, keep = TRUE)),
  as_tibble(gradiente_RMSProp(x0, keep = TRUE)),
  as_tibble(gradiente_ADAM(x0, keep = TRUE))
) %>%
  mutate(
    metodo = rep(c("GD", "Momento", "RSMProp", "Adam"), each = 101)
  ) %>%
  rlang::set_names("x1", "x2", "id")

endpoints <- paths[c(101, 202, 303, 404),]

ggplot() +
  geom_contour_filled(data = df, aes(x = x1, y = x2, z = f), show.legend = FALSE) +
  geom_path(data = paths, aes(x = x1, y = x2, group = id, color = id)) +
  geom_point(data = endpoints, aes(x = x1, y = x2), color = "green", size = 2, shape
  ↪  = 13) +
```

```
labs(x = TeX("$x_1$"), y = TeX("$x_2$"), color = "Otimização") +
theme_minimal() +
theme(panel.grid = element_blank(),
      axis.title.y = element_text(angle = 0, vjust = 0.5),
      legend.position = "bottom")
```

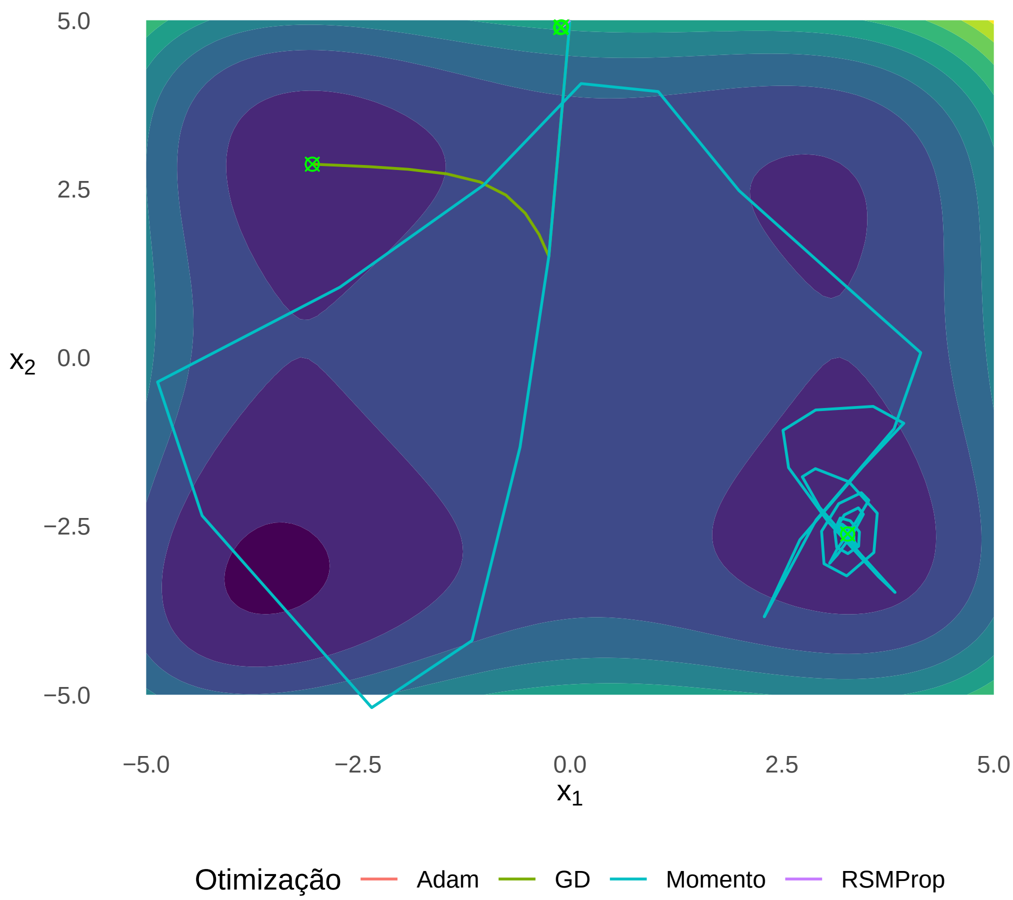


Figura 4: Caminhos percorridos pelas otimizações executadas nos itens d), g), h) e i)