Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

20 de outubro de 2025

Lista 3: Avaliando a rede neural.

Solução original: William Edward Rappel de Amorim

Prof. Guilherme Rodrigues

Redes Neurais Profundas

Tópicos em Estatística 2

(A) **As questões deverão ser respondidas em um único relatório *PDF* ou *html*, produzido usando as funcionalidades do *Rmarkdown* ou outra ferramenta equivalente.**

(B) **O aluno poderá consultar materiais relevantes disponíveis na internet, tais como livros, *blogs* e artigos.**

(C) **O trabalho é individual. Suspeitas de plágio e compartilhamento de soluções serão tratadas com rigor.**

(D) **Os códigos *R* utilizados devem ser disponibilizados na integra, seja no corpo do texto ou como anexo.**

(E) **O aluno deverá enviar o trabalho até a data especificada na plataforma *Microsoft Teams*.**

(F) **O trabalho será avaliado considerando o nível de qualidade do relatório, o que inclui a precisão das respostas, a pertinência das soluções encontradas, a formatação adotada, dentre outros aspectos correlatos.**

(G) **Escreva seu código com esmero, evitando operações redundantes, comentando os resultados e usando as melhores práticas em programação.**

(H) **O uso de Modelos de Linguagem de Grande Escala (LLMs), como ChatGPT, Gemini ou equivalentes, é permitido exclusivamente como ferramenta de apoio para organização de ideias, revisão textual, esclarecimento de conceitos e sugestões de escrita. O uso indiscriminado pode ser caracterizado como plágio acadêmico.**

Considere os dados e os modelos descritos na Lista 2 para responder os itens a seguir. Caso deseje, use os códigos disponibilizados no arquivo de solução da Lista 2 e no final deste documento.

a) Calcule os valores previstos ($\hat{y}_i$) e os resíduos ($y_i - \hat{y}_i$) da rede no conjunto de teste e represente-os graficamente em função de X_1 e X_2. Dica: tome como base o código usado para a visualização da superfície $(E(Y|X_1, X_2), X_1, X_2)$. Altere o gradiente de cores e, se necessário, use pontos semi-transparentes. Analise o desempenho da rede nas diferentes regiões do plano. Há locais onde o modelo é claramente viesado ou menos acurado?

b) Faça um gráfico do valor observado (y_i) em função do valor esperado ($\hat{y}_i = E(Y_i|x_{1i}, x_{2i})$) para cada observação do conjunto de teste. Interprete o resultado.

c) Para cada $k = 2, \ldots, 300$, recalcule o gradiente obtido no item d) usando apenas as k-primeiras observações do banco de treinamento. Novamente, use $\mathbf{\Phi} = \mathbf{\Phi}^\star$. Apresente um gráfico com o valor do primeiro elemento do gradiente (isso é, a derivada parcial $\frac{\partial J}{\partial w_1}$) em função do número de amostras k. Como referência, adicione uma linha horizontal vermelha indicando o valor obtido na Lista 2, item d). Em seguida, use a função `microbenchmark` para comparar o tempo de cálculo do gradiente para $k = 300$ e $k = 80000$. Explique de que maneira os resultados dessa análise podem ser usados para acelerar a execução do item e) da Lista 2.

d) Ajuste sobre o conjunto de treinamento um modelo linear normal (**modelo linear 1**)

$$Y_i \sim N(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}, \sigma^2)$$

usando a função `lm` do pacote `R` (ou outra equivalente). Em seguida, inclua na lista de covariáveis termos quadráticos e de interação linear. Isso é, assuma que no **modelo linear 2**,

$$E(Y|x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2.$$

Compare o erro quadrático médio no conjunto de teste dos dois modelos lineares acima com o da rede neural ajustada anteriormente. Qual dos 3 modelos você usaria para previsão? Justifique sua resposta.

e) Para cada modelo ajustado (os dois lineares e a rede neural), descreva o efeito no valor esperado da variável resposta causado por um aumento de uma unidade da covariável x_1?

f) Novamente, para cada um dos 3 modelos em estudo, calcule o percentual de vezes que o intervalo de confiança de 95% (para uma nova observação!) capturou o valor de y_i. Considere apenas os dados do conjunto de teste. No caso da rede neural, assuma que, aproximadamente, $\frac{y_i - \hat{y}}{\hat{\sigma}} \sim N(0, 1)$, onde $\hat{\sigma}$ representa a raiz do erro quadrático médio da rede no conjunto de validação. Comente os resultados. Dica: para os modelos lineares, use a função `predict(mod, interval="prediction")`.

g) Para o **modelo linear 2**, faça um gráfico de dispersão entre x_1 e x_2, onde cada ponto corresponde a uma observação do conjunto de teste. Identifique os pontos que estavam contidos nos respectivos intervalos de confianças utilizando a cor verde. Para os demais pontos, use vermelho. Comente o resultado.

```r
### Funções ----
# Função Sigmoide
sigmoide <- function(x) {
  return(1/(1+exp(-x)))
}

# Função para realizar o forward propagation dados Phi e x
forward_prop <- function(Phi, x, retornar_tudo = T) {
  # Organizando objetos
  f <- vector("list", 2) ; names(f) <- str_c("f", 0:1)
  h <- vector("list", 3) ; names(h) <- str_c("h", 0:2)
  a <- list(\(z) sigmoide(z), \(z) z) ; names(a) <- c("sigmoide", "Identidade")
  ifelse(is.matrix(x), h[[1]] <- t(x), h[[1]] <- as.matrix(x))

  # Propagando pela rede
  for(camada in 1:length(a)) {
    f[[camada]] <- Phi$b[[camada]] + Phi$W[[camada]] %*% h[[camada]]
    h[[camada+1]] <- a[[camada]](f[[camada]])  # função de ativação
  }

  # Previsão
  y_hat <- as.double(h[[length(h)]])

  # Output
  if(retornar_tudo) return(list(y_hat = y_hat, f = f, h = h))
  else return(y_hat)
}

# Função para calcular a função de perda
mse_cost <- function(y_true, y_hat) {
  return(mean((y_true - y_hat)^2))
}

# Função Derivada da Sigmoide
derivada_sigmoide <- function(x) {
  sig <- sigmoide(x)
  return(sig * (1 - sig))
}

# Função para realizar o back-propagation para um vetor theta, uma dada matriz design
# e as observações
back_prop <- function(Phi, x, y){
  # Organizando os objetos
  aux <- forward_prop(Phi, x)
  a_linha <- list(\(z) derivada_sigmoide(z), \(z) rep(1, length(z)))
  Phi_grad <- list(W = vector(mode="list", 2),
                   b = vector(mode="list", 2))

  ### Em seguida, passamos para a implementação do back propagation
  g <- -2*(y - aux$y_hat)/length(y)
  for (camada in 2:1) {
    g <- g * a_linha[[camada]](aux$f[[camada]])
    if(!is.matrix(g)) g <- matrix(g, ncol=nrow(x))
    Phi_grad$b[[camada]] <- rowSums(g)
    Phi_grad$W[[camada]] <- g %*% t(aux$h[[camada]])
    g <- t(Phi$W[[camada]]) %*% g
  }

  ### Final
```

```r
  # Criamos um vetor com os gradientes de cada parâmetro
  return(Phi_grad)
}

### Gerando dados "observados"
set.seed(22025)
m.obs <- 100000
dados <- tibble(x1.obs=runif(m.obs, -3, 3),
                x2.obs=runif(m.obs, -3, 3)) %>%
  mutate(mu=abs(x1.obs^3 - 30*sin(x2.obs) + 10),
         y=rnorm(m.obs, mean=mu, sd=1))

# Taxa de aprendizagem
epsilon <- .1
# Número de iterações
M <- 100

# Divisão dos dados em treinamento, validação e teste
treino   <- dados[1:80000, ]
val      <- dados[80001:90000, ]
teste    <- dados[90001:nrow(dados), ]
x_treino <- treino %>%
  select(x1.obs, x2.obs) %>%
  as.matrix()
x_val <- val %>%
  select(x1.obs, x2.obs) %>%
  as.matrix()
x_teste  <- teste %>%
  select(x1.obs, x2.obs) %>%
  as.matrix()
y_treino <- treino$y
y_val <- val$y
y_teste <- teste$y

# Lista para receber os parâmetros estimados em cada iteração
Phi_est <- list()

# Theta inicial
Phi_0 <- list(W = list(W0 = matrix(rep(0, 4), nrow=2),
                       W1 = matrix(rep(0, 2), nrow=1)),
              b = list(b0 = rep(0, 2), b1 = 0))

Phi_est[[1]] <- Phi_0

# Vetores para receber as perdas de, respectivamente, treino e validação em cada iteração
perda_treino <- perda_val <- numeric(M)

# Execução
for(i in 1:M) {
  # Cálculo dos gradientes dos parâmetros
  grad <- back_prop(Phi = Phi_est[[i]], x = x_treino, y = y_treino)

  # Cálculo do custo de treino
  perda_treino[i] <- mse_cost(y_treino,
                              forward_prop(Phi_est[[i]], x_treino)$y_hat)

  # Cálculo do custo de validação
  perda_val[i] <- mse_cost(y_val,
                           forward_prop(Phi_est[[i]], x_val)$y_hat)
```

```
  # Atualização dos parâmetros
  Phi_est[[i+1]] <- map2(Phi_est[[i]], grad,
                         \(x, y) map2(x, y, \(atual, passo) atual - epsilon * passo))
}

min_perda_treino <- min(perda_treino)
min_perda_val <- min(perda_val)
Phi_til <- Phi_est[which.min(perda_val)][[1]]
```