Exercício 10.
Nome: Lucas Kou Kinoshita
RM: 2019021557
data: 31/05/2025

1.Implementação:

1.1 Boston Housing

Primeiramente, foram implementadas as funções auxiliares utilizadas para escalonamento dos conjuntos de dados com base na função de normalização proposta no enunciado.

```r
# Função de normalização
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Função de desnormalização
unnormalize <- function(x_scaled, original_min, original_max) {
  return (x_scaled * (original_max - original_min) + original_min)
}
```

Figura 1: Implementação das funções de normalização e desnormalização.

Em seguida, as variáveis globais relacionadas aos experimentos foram definidas, sendo estas o número de experimentos executados para obtenção do erro médio, a proporção entre amostras de treino e teste, o número de neurônios na camada escondida e o número máximo de iterações de treinamento da MLP.

```r
N_EXECUTIONS <- 5
TRAIN_RATIO <- 0.7
HIDDEN_NEURONS <- 5
MAX_ITERATIONS <- 2000
```

Figura 2: Configurações globais para os experimentos

Finalmente, os dados foram tratados (normalizados) e os vetores de armazenamento de erro foram inicializados.

```r
data(Boston) # Carrega do pacote MASS
boston_df <- Boston

# Normaliza o dataframe inteiro
boston_scaled_df <- as.data.frame(lapply(boston_df, normalize))

# Salva os valores mínimo e máximo da variável 'medv' original para desnormalização posterior
min_medv_original_global <- min(boston_df$medv)
max_medv_original_global <- max(boston_df$medv)

# --- Inicialização de Vetores para Armazenar Resultados de RMSE ---
all_rmse_scaled_test <- numeric(N_EXECUTIONS)
all_rmse_original_test <- numeric(N_EXECUTIONS)
```

Figura 3: Tratamento dos dados do *dataset Boston Housing*

Dentro de um loop de N_EXECUTIONS iterações, os dados foram divididos em conjuntos de treino e teste e a rede MLP é treinada com uma arquitetura arbitrária (neste caso inicialmente inspirada na vídeo aula 27).

```
for (exec_num in 1:N_EXECUTIONS) {
  cat(paste("\n--- Execução:", exec_num, "de", N_EXECUTIONS, "---\n"))

  cat("Dividindo dados em conjuntos de treinamento e teste...\n")
  n_obs <- nrow(boston_scaled_df)
  train_indices <- sample(1:n_obs, size = floor(TRAIN_RATIO * n_obs))

  x_train <- boston_scaled_df[train_indices, 1:13]
  y_train_scaled <- boston_scaled_df$medv[train_indices]

  x_test <- boston_scaled_df[-train_indices, 1:13]
  y_test_scaled <- boston_scaled_df$medv[-train_indices]

  # Valores originais (não escalonados) do alvo para o conjunto de teste (para avaliação final)
  y_test_original <- boston_df$medv[-train_indices]

  cat("Treinando a rede MLP...\n")
  rede <- mlp(x_train, y_train_scaled,
              size = HIDDEN_NEURONS,
              maxit = MAX_ITERATIONS,
              initFunc = "Randomize_Weights",
              initFuncParams = c(-0.3, 0.3),
              learnFunc = "Rprop",
              learnFuncParams = c(0.1, 0.1), # Parâmetros como no script original
              updateFunc = "Topological_Order",
              updateFuncParams = c(0),
              hiddenActFunc = "Act_Logistic",
              shufflePatterns = TRUE,
              linOut = TRUE, # Saída linear, apropriado para regressão
              inputsTest = x_test, # Fornece dados de teste para monitorar erro durante o treino
              targetsTest = y_test_scaled)

  cat("Rede treinada. Realizando previsões no conjunto de teste...\n")
```

Figura 4: Definição da arquitetura da rede e do loop experimentos

Por fim, o loop finaliza com os cálculo das previsões realizadas no conjunto de teste e os cálculos de erro (neste caso RMSE).

```
  predicted_values_test_scaled <- predict(rede, x_test)
  predicted_values_test_original <- unnormalize(predicted_values_test_scaled,
                                                min_medv_original_global,
                                                max_medv_original_global)

  current_rmse_scaled <- rmse(y_test_scaled, predicted_values_test_scaled)
  current_rmse_original <- rmse(y_test_original, predicted_values_test_original)

  all_rmse_scaled_test[exec_num] <- current_rmse_scaled
  all_rmse_original_test[exec_num] <- current_rmse_original

  cat(paste("Execução", exec_num, "- RMSE (teste, escalado):", round(current_rmse_scaled, 4), "\n"))
  cat(paste("Execução", exec_num, "- RMSE (teste, original):", round(current_rmse_original, 4), "\n"))

  if (exec_num == N_EXECUTIONS) {
    plotIterativeError(rede, main = paste("Erro Iterativo da MLP (Última Execução -", MAX_ITERATIONS, "iterações)"))
  }
}
```

Figura 5: Finalização do loop de experimento e cálculo do MSE da execução

O script termina com o cálculo da média e desvio padrão do vetor de erros coletados tanto para os dados normalizados quanto sob o conjunto original, além do plot dos gráficos de interesse para diagnóstico da qualidade da previsão obtida pelo quinto e último experimento.

```r
# --- Resultados ---
mean_rmse_scaled_test <- mean(all_rmse_scaled_test)
sd_rmse_scaled_test <- sd(all_rmse_scaled_test)
mean_rmse_original_test <- mean(all_rmse_original_test)
sd_rmse_original_test <- sd(all_rmse_original_test)

cat("Resultados Finais sobre", N_EXECUTIONS, "execuções (média ± desvio padrão):\n")
cat(paste("RMSE (teste, escalado): ", round(mean_rmse_scaled_test, 4), " ± ", round(sd_rmse_scaled_test, 4), "\n"))
cat(paste("RMSE (teste, original): ", round(mean_rmse_original_test, 4), " ± ", round(sd_rmse_original_test, 4), "\n\n"))

# --- Plots de diagnóstico da última execução ---
# Plot dos valores reais vs. previstos (escalados) para a última execução
plot(y_test_scaled, type = 'l', col = 'blue', lwd = 2, # 'l' para linha, lwd para espessura
     ylim = range(c(y_test_scaled, predicted_values_test_scaled)),
     main = "Última Execução: 'medv' Escalado (Teste) - Real vs. Previsto (Linhas)",
     ylab = "medv (escalado)", xlab = "Índice da Observação no Conjunto de Teste")
lines(predicted_values_test_scaled, type = 'l', col = 'red', lwd = 2)
legend("topright", legend = c("Real", "Previsto"), col = c("blue", "red"), lty = 1, lwd = 2) # lty=1 para linha sólida


# Plot dos valores reais vs. previstos (na escala original) para a última execução
plot(y_test_original, predicted_values_test_original,
     main = "Última Execução: 'medv' Original (Teste) - Real vs. Previsto",
     xlab = "Valor Real de medv", ylab = "Valor Previsto de medv",
     pch = 16, col = rgb(0,0,1,0.5)) # Pontos azuis semi-transparentes
abline(a = 0, b = 1, col = "red", lwd = 2) # Linha de referência y=x (ideal)
legend("bottomright", legend = c("Previsões", "Linha Ideal (Real=Previsto)"),
       pch=c(16,NA), lty=c(NA,1), col = c(rgb(0,0,1,0.5), "red"))
```

Figura 6: Cálculo da média da visualização e plot dos gráficos para diagnóstico

1.2 Statlog (Heart)

A implementação dos experimentos para este segundo *dataset* difere do primeiro em função do tipo de problema a ser resolvido, sendo este um problema de classificação.

A começar pelo tratamento dos dados, não é mais necessário normalizar os dados, mas sim mapear as saídas para valores de -1 ou 1, tratar as colunas para que estas possuam valores numéricos e otimizar linhas com valores NA

```r
cat("Carregando e pré-processando os dados...\n")
df <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/heart/heart.dat", sep = " ")
colnames(df) <- c("age", "sex", "chest_pain", "rest_bp", "chol", "fbs", "rest_ecg",
                  "max_hr", "ex_angina", "oldpeak", "slope", "ca", "thal", "target")

df[df == '?'] <- NA
df <- na.omit(df)

feature_cols <- setdiff(colnames(df), "target")
df[, feature_cols] <- lapply(df[, feature_cols], function(x) as.numeric(as.character(x)))

df$target <- ifelse(df$target == 1, -1, 1)
df <- na.omit(df)

X <- as.matrix(df[, feature_cols])
Y <- as.matrix(df$target) # Y é uma matriz de uma coluna com -1 ou 1
```

Figura 7: Tratamento dos dados do *dataset Statlog (Heart)*

Também foram necessárias alterações nos cálculos das métricas no final do loop de execução e mudanças na arquitetura da rede (na função de ativação da camada de saída).

```
cat("Treinando a rede MLP para classificação...\n")
rede <- mlp(x_train, y_train,
            size = HIDDEN_NEURONS,
            maxit = MAX_ITERATIONS,
            initFunc = "Randomize_Weights",
            initFuncParams = c(-0.3, 0.3),
            learnFunc = "Rprop",
            learnFuncParams = c(0.1, 0.1),
            updateFunc = "Topological_Order",
            updateFuncParams = c(0),
            hiddenActFunc = "Act_Bipas",
            outputActFunc = "Act_TanH",      # Função de ativação da camada de SAÍDA
            shufflePatterns = TRUE,
            inputsTest = x_test,
            targetsTest = y_test)

cat("Rede treinada. Realizando previsões no conjunto de teste...\n")

predicted_outputs_test <- predict(rede, x_test)
predicted_classes_test <- ifelse(predicted_outputs_test > 0, 1, -1)

correct_predictions <- sum(predicted_classes_test == y_test)
current_accuracy <- correct_predictions / length(y_test)

all_accuracy_test[exec_num] <- current_accuracy

cat(paste("Execução", exec_num, "- Acurácia (teste):", round(current_accuracy, 4), "\n"))
```

Figura 8: Fim do loop de execução dos experimentos sobre o *dataset Statlog (Heart)*

2. Resultados

2.1 Boston Housing

As tabelas à seguir mostram as métricas obtidas para os experimentos realizados:

Número de neurônios	Função de ativação da camada oculta	Função de ativação da camada de saída	RMSE (conjunto normalizado)	RMSE (conjunto original)
2	Act_Logistic	Linear	0.0972 ± 0.0179	4.3748 ± 0.8035
5	Act_Logistic	Linear	0.0929 ± 0.0186	4.1798 ± 0.8356
10	Act_Logistic	Linear	0.0982 ± 0.0317	4.4212 ± 1.4258

Tabela 1: Variação dos resultados para diferentes número de neurônios na camada escondida (Boston Housing)

Número de neurônios	Função de ativação da camada oculta	Função de ativação da camada de saída	RMSE (conjunto normalizado)	RMSE (conjunto original)
5	Act_Bipas	Linear	0.0747 ± 0.0137	3.3601 ± 0.6179
5	Act_TanH	Linear	0.082 ± 0.0102	3.6897 ± 0.4592
5	Act_Elliott	Linear	0.0787 ± 0.0072	3.5404 ± 0.3236

Tabela 2: Variação dos resultados para diferentes funções de ativação na camada escondida (Boston Housing)

À seguir, a visualização da predição realizada sob o conjunto de testes da última execução do experimento com menor MSE (função de ativação por uma sigmoide bipolar na camada intermediária e 5 neurônios):

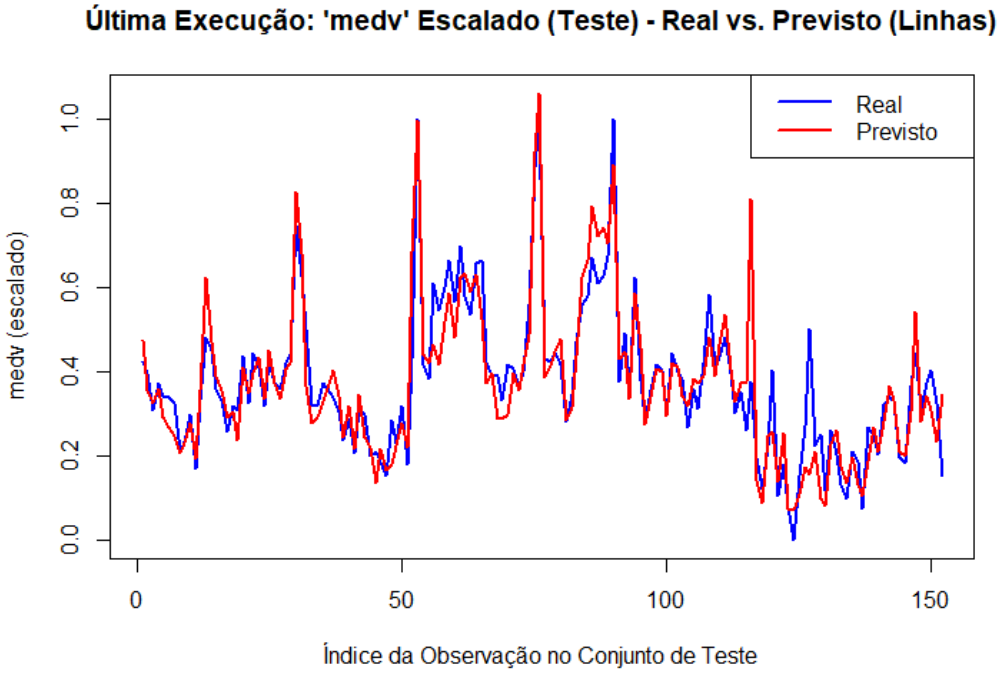

Figura 9: Predições obtidas sobre o conjunto de testes normalizado.

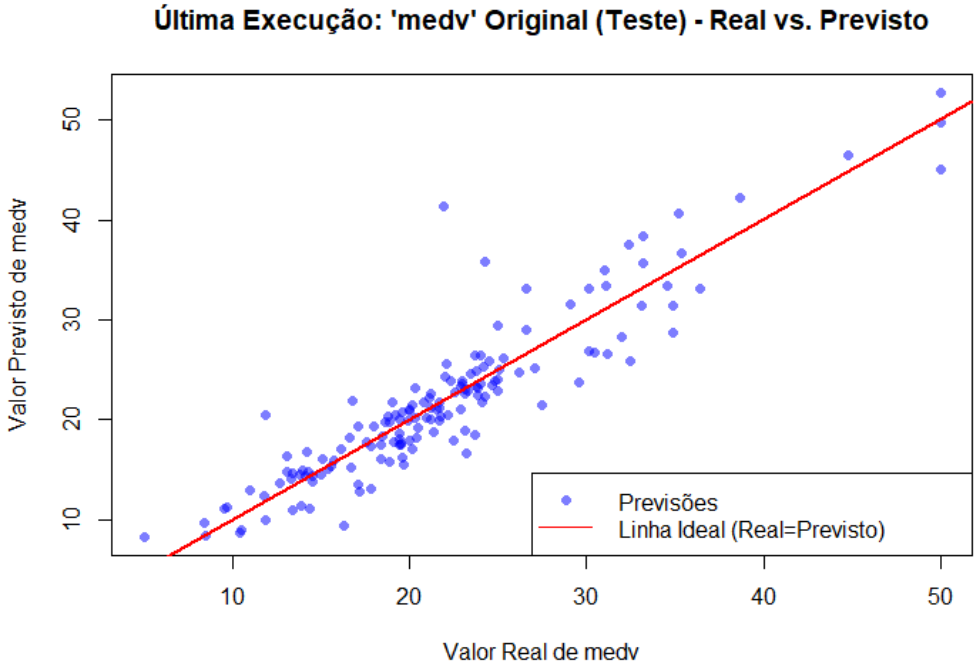

Figura 10: Predições obtidas sobre o conjunto de testes original

2.2 Statlog (Heart)

As tabelas à seguir mostram as métricas obtidas para os experimentos realizados:

Número de neurônios	Função de ativação da camada oculta	Função de ativação da camada de saída	Acurácia
2	Act_Logistic	Act_TanH	0.6519 ± 0.1596
5	Act_Logistic	Act_TanH	0.6938 ± 0.1299
10	Act_Logistic	Act_TanH	0.8395 ± 0.0151

Tabela 3: Variação dos resultados para diferentes número de neurônios na camada escondida (Statlog)

Número de neurônios	Função de ativação da camada oculta	Função de ativação da camada de saída	Acurácia
10	Act_Bipas	Act_TanH	0.7901 ± 0.0409
10	Act_TanH	Act_Elliott	0.5062 ± 0.1121
10	Act_Elliott	Act_Logistic	0.4247 ± 0.0334
10	Act_Logistic	Act_Elliott	0.7778 ± 0.0175

Tabela 4: Variação dos resultados para diferentes funções de ativação (Statlog)

3. Discussões

Primeiramente é importante observar que o problema de regressão foi estudado sob a ótica de uma função de ativação linear em sua saída para evitar problemas de *squashing*, já que uma saída linear permite que a rede neural alcance previsões em qualquer escala numérica. De forma diametralmente oposta, o problema de classificação foi tratado sob a ótica de funções de ativação não lineares na camada de saída.

Para os experimentos realizados para ambos os problemas, nota-se que a função de ativação por meio da sigmoide bipolar (Act_Bipas) parece alcançar os melhores resultados em termos de RMSE e acurácia, assim como evidenciado nas Tabelas 4 e 2. É possível notar que, para o problema de classificação, a tangente hiperbólica e a função de Elliot na camada intermediária resultaram nos piores desempenhos, mesmo que tenham alcançado bons resultados quando utilizadas na camada de saída em união com a sigmoide logística ou bipolar nas camadas intermediárias.

Com relação ao número de neurônios, o valor ótimo dentre os testados foram 5 neurônios para o problema de regressão e 10 neurônios para o problema de classificação. No

entanto, é importante evidenciar que, pela baixa robustez estatística (poucas execuções e poucos experimentos), os resultados obtidos durante os experimentos para o problema de regressão não aparentam melhoria ou piora expressiva em seus desempenhos. Diferentemente, os experimentos sob o problema de classificação apresentaram melhoria significativa de desempenho durante o salto de cinco para dez neurônios na camada de ativação.

Por fim, pela inspeção visual das Figuras 8 e 9 e das métricas nas Tabelas 1,2,3,4, podemos confirmar que, para ao menos uma das arquiteturas, ambos os problemas foram bem resolvidos pelas redes implementadas. Apresentando RMSE de 3.3601 ± 0.6179 (Tabela 2) e acurácia de 0.8395 ± 0.0151 (Tabela 3).