NBA Combine MLP

Load and Preprocess Data

```
df <- read.csv("combined.csv")

features <- c('WEIGHT', 'HAND_WIDTH', 'LANE_AGILITY_TIME',
              'THREE_QUARTER_SPRINT', 'MAX_VERTICAL_LEAP',
              'MODIFIED_LANE_AGILITY_TIME', 'GAMES_PLAYED')

df <- na.omit(df[, c(features, 'ROOKIE_SCORE', 'PLAYER_NAME')])

X <- as.matrix(df[, features])
y <- as.matrix(df$ROOKIE_SCORE)

X_min <- apply(X, 2, min)
X_max <- apply(X, 2, max)
X <- sweep(X, 2, X_min, `-`) %>% sweep(2, X_max - X_min, `/`)
X <- cbind(1, X)

set.seed(42)
split <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[split, ]; X_test <- X[-split, ]
y_train <- y[split, ]; y_test <- y[-split]
```

Activation Functions

```
tanh_fn <- function(x) tanh(x)
tanh_deriv <- function(x) 1 - tanh(x)^2

leaky_relu <- function(x, leak = 0.01) ifelse(x > 0, x, leak * x)
leaky_relu_deriv <- function(x, leak = 0.01) ifelse(x > 0, 1, leak)
```

MLP Initialization and Forward Pass

```
init_mlp <- function(input_dim, hidden1, hidden2) {
  list(
    W1 = matrix(rnorm(input_dim * hidden1, sd = 0.1), nrow = input_dim),
    W2 = matrix(rnorm(hidden1 * hidden2, sd = 0.1), nrow = hidden1),
    W3 = matrix(rnorm(hidden2, sd = 0.1), nrow = hidden2),
    leak = 0.01
  )
}

forward_pass <- function(model, X) {
  z1 <- X %*% model$W1
  h1 <- tanh_fn(z1)
```

```
  z2 <- h1 %*% model$W2
  h2 <- leaky_relu(z2, model$leak)
  y_hat <- h2 %*% model$W3
  list(z1 = z1, h1 = h1, z2 = z2, h2 = h2, y_hat = y_hat)
}

mse <- function(pred, actual) mean((pred - actual)^2)
```

Training with Manual Gradient Descent

```
train_mlp <- function(model, X, y, lr = 0.01, epochs = 1000) {
  n <- nrow(X)
  errors <- numeric(epochs)

  for (epoch in 1:epochs) {
    fwd <- forward_pass(model, X)
    pred <- fwd$y_hat
    error <- pred - y

    dz3 <- error / n
    dW3 <- t(fwd$h2) %*% dz3

    dh2 <- dz3 %*% t(model$W3)
    dz2 <- dh2 * leaky_relu_deriv(fwd$z2, model$leak)
    dW2 <- t(fwd$h1) %*% dz2

    dh1 <- dz2 %*% t(model$W2)
    dz1 <- dh1 * tanh_deriv(fwd$z1)
    dW1 <- t(X) %*% dz1

    model$W1 <- model$W1 - lr * dW1
    model$W2 <- model$W2 - lr * dW2
    model$W3 <- model$W3 - lr * dW3

    errors[epoch] <- mse(pred, y)
    if (epoch %% 100 == 0) cat("Epoch", epoch, "Loss:", errors[epoch], "\n")
  }
  list(model = model, errors = errors)
}
```

Fit and Evaluate the Model

```
model <- init_mlp(ncol(X_train), 32, 16)
result <- train_mlp(model, X_train, y_train, lr = 0.01, epochs = 5000)
```

```
## Epoch 100 Loss: 15.69097
## Epoch 200 Loss: 14.3793
## Epoch 300 Loss: 14.29947
## Epoch 400 Loss: 14.26896
## Epoch 500 Loss: 14.25391
## Epoch 600 Loss: 14.24378
## Epoch 700 Loss: 14.23504
## Epoch 800 Loss: 14.22643
```

```
## Epoch 900 Loss: 14.21729
## Epoch 1000 Loss: 14.20716
## Epoch 1100 Loss: 14.19554
## Epoch 1200 Loss: 14.18192
## Epoch 1300 Loss: 14.16576
## Epoch 1400 Loss: 14.14657
## Epoch 1500 Loss: 14.12396
## Epoch 1600 Loss: 14.09773
## Epoch 1700 Loss: 14.06781
## Epoch 1800 Loss: 14.03428
## Epoch 1900 Loss: 13.99743
## Epoch 2000 Loss: 13.95776
## Epoch 2100 Loss: 13.91602
## Epoch 2200 Loss: 13.87315
## Epoch 2300 Loss: 13.83014
## Epoch 2400 Loss: 13.78797
## Epoch 2500 Loss: 13.74736
## Epoch 2600 Loss: 13.70893
## Epoch 2700 Loss: 13.6728
## Epoch 2800 Loss: 13.63887
## Epoch 2900 Loss: 13.60689
## Epoch 3000 Loss: 13.57654
## Epoch 3100 Loss: 13.5474
## Epoch 3200 Loss: 13.51838
## Epoch 3300 Loss: 13.49041
## Epoch 3400 Loss: 13.46336
## Epoch 3500 Loss: 13.52694
## Epoch 3600 Loss: 13.44092
## Epoch 3700 Loss: 13.49618
## Epoch 3800 Loss: 13.42516
## Epoch 3900 Loss: 13.47361
## Epoch 4000 Loss: 13.41831
## Epoch 4100 Loss: 13.43662
## Epoch 4200 Loss: 13.3816
## Epoch 4300 Loss: 13.40356
## Epoch 4400 Loss: 13.33683
## Epoch 4500 Loss: 13.36302
## Epoch 4600 Loss: 13.31834
## Epoch 4700 Loss: 13.30671
## Epoch 4800 Loss: 13.3104
## Epoch 4900 Loss: 13.27347
## Epoch 5000 Loss: 13.27959
```

```
model <- result$model
y_pred <- forward_pass(model, X_test)$y_hat

cat("Test MSE:", mse(y_pred, y_test), "\n")
```

```
## Test MSE: 10.76321
```

```
baseline_pred <- mean(y_train)
baseline_mse <- mse(rep(baseline_pred, length(y_test)), y_test)
cat("Baseline MSE:", baseline_mse, "\n")
```

```
## Baseline MSE: 25.19003
```

```
r2 <- 1 - sum((y_test - y_pred)^2) / sum((y_test - mean(y_test))^2)
cat("R^2 Score:", r2, "\n")
```

```
## R^2 Score: 0.5724595
```

Visualization

```
results <- tibble(Actual = y_test, Predicted = y_pred)

ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.7) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Predicted vs Actual ROOKIE_SCORE",
       x = "Actual",
       y = "Predicted") +
  theme_minimal()
```