README.md

2024-02-14

```
knitr::opts_chunk$set(echo = TRUE)

relu <- function(x) {
  # Rectified linear unit activation function
  # Replaces negative values with zero
  x[x < 0] <- 0
  x
}

softmax <- function(x) {
  # Softmax activation function
  # Normalizes the input to a probability distribution
  exp_x <- exp(x)
  row_sums <- apply(exp_x, 1, sum)
  exp_x / row_sums
}

one_hot_encode <- function(x) {
  # One-hot encoding function
  # Converts a vector of labels into a matrix of binary indicators
  n <- length(x)
  k <- max(x)
  mat <- matrix(0, nrow = n, ncol = k)
  mat[cbind(1:n, x)] <- 1
  mat
}

    # Initialize the parameters with random values
initialize <- function(n_hidden, n_features = 2, n_class = 2) {
      return(list(
        W1 = matrix(rnorm(n_features * n_hidden), nrow = n_features, ncol = n_hidden),
        b1 = rnorm(n_hidden),
        W2 = matrix(rnorm(n_hidden * n_class), nrow = n_hidden, ncol = n_class),
        b2 = rnorm(n_class)
      ))
}


    # forwardPropogate pass of the neural network
forwardPropogate <-  function( input_data,params) {
      W1 <- params$W1
      W2 <- params$W2
      b1 <- params$b1
      b2 <- params$b2
    a1 <-  (input_data %*% W1) + b1
    params$z1 <- relu(a1)
    a2 <- (params$z1 %*% W2) + b2
    params$z2 <- softmax(a2)

    return(list(params$z1, params$z2))
  }

  # Fit the neural network to the input data and labels
fit = function(input_data, label, batch_size, iter_num, params) {
    for (epoch in 1:iter_num) {
      p <- sample(1:length(label))
      input_data <- input_data[p, ]
      label <- label[p]
      for (i in seq(1, length(label), by = batch_size)) {
        batch_data <- input_data[i:(i + batch_size - 1), ]
        batch_label <- label[i:(i + batch_size - 1)]
        params <- sgd(batch_data, batch_label, params)
      }
    }
  return(params)
  }

  # Stochastic gradient descent update of the parameters
  sgd <- function(data, label, params, alpha = 1e-4) {
    grad <- backward(data, label, params)
    for (layer in names(grad)) {
      params[[layer]] <- params[[layer]] + (alpha * grad[[layer]])
    }
    return(params)
  }

  # Backward pass of the neural network
backward <- function(data, label, params) {
    W1 <- params$W1
    W2 <- params$W2
    b1 <- params$b1
    b2 <- params$b2
    z1 <- forwardPropogate(data, params)[[1]]
    z2 <- forwardPropogate(data, params)[[2]]

    #label <- one_hot_encode(label)
    db2_temp <- label - z2
    db2 <- colSums(db2_temp)
    dW2 <- t(z1) %*% db2_temp
    db1_temp <- db2_temp %*% t(W2)
    db1_temp[z1 <= 0] <- 0
    db1 <- colSums(db1_temp)
    dW1 <- t(data) %*% db1_temp

    return(list(W1 = dW1, b1 = db1, W2 = dW2, b2 = db2))
  }
```

```r
    # Test the accuracy of the neural network on the test data and labels
  test = function(train_data, train_label, test_data, test_label, batch_size, iter_num, params) {
      pred_label <- forwardPropogate(test_data, params)[[2]]
      pred_label <- apply(pred_label, 1, which.max)
      acc <- mean(pred_label == test_label)
      return(acc)
    }





set.seed(123)
n_train <- 100
n_test <- 20
n_features <- 2
n_class <- 2
train_data <- matrix(rnorm(n_train * n_features), nrow = n_train, ncol = n_features)
train_label <- sample(1:n_class, n_train, replace = TRUE)
test_data <- matrix(rnorm(n_test * n_features), nrow = n_test, ncol = n_features)
test_label <- sample(1:n_class, n_test, replace = TRUE)

# Instantiate the NeuralNet class with 10 hidden units
nn <- initialize(n_hidden = 10)

# Train the neural network with batch size of 10 and 50 iterations
fit(train_data, train_label, batch_size = 10, iter_num = 400, nn)
```

```
## $W1
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   NaN
## [2,]  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN   NaN
##
## $b1
##  [1] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
##
## $W2
##       [,1] [,2]
##  [1,]  NaN  NaN
##  [2,]  NaN  NaN
##  [3,]  NaN  NaN
##  [4,]  NaN  NaN
##  [5,]  NaN  NaN
##  [6,]  NaN  NaN
##  [7,]  NaN  NaN
##  [8,]  NaN  NaN
##  [9,]  NaN  NaN
## [10,]  NaN  NaN
##
## $b2
## [1] NaN NaN
```

```r
# Test the accuracy of the neural network on the test data
acc <- test(train_data, train_label, test_data, test_label, batch_size = 10, iter_num = 50, nn)
cat("Accuracy:", acc, "\n")
```

```
## Accuracy: 0.5
```