medium

Akanksha Koshti

2025-03-07

```r
# URL of the dataset
spam_url = "https://hastie.su.domains/ElemStatLearn/datasets/spam.data"

# Load data
spam_data = read.table(spam_url, header = FALSE)

# View dataset structure
print(dim(spam_data))
```

```
## [1] 4601   58
```

```r
print(head(spam_data))
```

```
##     V1   V2   V3 V4   V5   V6   V7   V8   V9  V10  V11  V12  V13  V14  V15  V16
## 1 0.00 0.64 0.64  0 0.32 0.00 0.00 0.00 0.00 0.00 0.00 0.64 0.00 0.00 0.00 0.32
## 2 0.21 0.28 0.50  0 0.14 0.28 0.21 0.07 0.00 0.94 0.21 0.79 0.65 0.21 0.14 0.14
## 3 0.06 0.00 0.71  0 1.23 0.19 0.19 0.12 0.64 0.25 0.38 0.45 0.12 0.00 1.75 0.06
## 4 0.00 0.00 0.00  0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00 0.31
## 5 0.00 0.00 0.00  0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00 0.31
## 6 0.00 0.00 0.00  0 1.85 0.00 0.00 1.85 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##    V17  V18  V19  V20  V21 V22  V23  V24 V25 V26 V27 V28 V29 V30 V31 V32 V33
## 1 0.00 1.29 1.93 0.00 0.96   0 0.00 0.00   0   0   0   0   0   0   0   0   0
## 2 0.07 0.28 3.47 0.00 1.59   0 0.43 0.43   0   0   0   0   0   0   0   0   0
## 3 0.06 1.03 1.36 0.32 0.51   0 1.16 0.06   0   0   0   0   0   0   0   0   0
## 4 0.00 0.00 3.18 0.00 0.31   0 0.00 0.00   0   0   0   0   0   0   0   0   0
## 5 0.00 0.00 3.18 0.00 0.31   0 0.00 0.00   0   0   0   0   0   0   0   0   0
## 6 0.00 0.00 0.00 0.00 0.00   0 0.00 0.00   0   0   0   0   0   0   0   0   0
##   V34 V35 V36  V37 V38 V39  V40 V41 V42  V43 V44  V45  V46 V47 V48  V49   V50
## 1   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0 0.00 0.000
## 2   0   0   0 0.07   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0 0.00 0.132
## 3   0   0   0 0.00   0   0 0.06   0   0 0.12   0 0.06 0.06   0   0 0.01 0.143
## 4   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0 0.00 0.137
## 5   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0 0.00 0.135
## 6   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0 0.00 0.223
##   V51   V52   V53   V54   V55 V56  V57 V58
## 1   0 0.778 0.000 0.000 3.756  61  278   1
## 2   0 0.372 0.180 0.048 5.114 101 1028   1
## 3   0 0.276 0.184 0.010 9.821 485 2259   1
## 4   0 0.137 0.000 0.000 3.537  40  191   1
## 5   0 0.135 0.000 0.000 3.537  40  191   1
## 6   0 0.000 0.000 0.000 3.000  15   54   1
```

```r
# Convert to matrix
data_matrix = as.matrix(spam_data[, -ncol(spam_data)])
labels = as.numeric(spam_data[, ncol(spam_data)]) + 1  # Shift all labels up by 1

# Normalize data (zero mean, unit variance)
data_mean = colMeans(data_matrix)
data_sd = apply(data_matrix, 2, sd)
data_matrix = scale(data_matrix, center = data_mean, scale = data_sd)

# Convert to torch tensors
data_tensor = torch_tensor(data_matrix, dtype = torch_float())
labels_tensor = torch_tensor(labels, dtype = torch_long())
```

```r
# Define a dataset class for Spam Data
spam_dataset = dataset(
  name = "SpamDataset",

  initialize = function(x, y) {
    self$data = x  # Ensure it's a tensor
    self$labels = y  # Ensure labels are tensors
  },

  .getitem = function(index) {
    list(
      x = self$data[index, ],  # Ensure features remain a tensor
      y = self$labels[index]  # Labels remain a tensor
    )
  },

  .length = function() {
    self$data$shape[1]  # Returns number of samples
  }
)

# Initialize the dataset with torch tensors
dataset = spam_dataset(data_tensor, labels_tensor)

# Print dataset details
cat("Dataset length:", dataset$.length(), "\n")
```

```
## Dataset length: 4601
```

```r
# Print a sample to verify structure
print(dataset$.getitem(1))
```

```
## $x
## torch_tensor
## -0.3424
##  0.3308
##  0.7128
## -0.0469
##  0.0116
```

```
## -0.3502
## -0.2918
## -0.2625
## -0.3233
## -0.3713
## -0.2968
##  0.1141
## -0.3120
## -0.1749
## -0.1901
##  0.0862
## -0.3211
##  2.0810
##  0.1509
## -0.1679
##  0.1251
## -0.1182
## -0.2902
## -0.2130
## -0.3288
## -0.2992
## -0.2279
## -0.2318
## -0.1667
## -0.2252
## ... [the output was truncated (use n=-1 to disable)]
## [ CPUFloatType{57} ]
##
## $y
## torch_tensor
## 2
## [ CPULongType{} ]
```

```r
# Split into training (80%) and testing (20%)
# Define batch size and dataloader
batch_size = 32
dataloader = dataloader(dataset, batch_size = batch_size, shuffle = TRUE)
```

```r
net = nn_module(
  "SpamNet",
  initialize = function() {
    self$fc1 = nn_linear(ncol(data_matrix), 128)  # More neurons
    self$fc2 = nn_linear(128, 64)
    self$fc3 = nn_linear(64, 2)  # Output layer (2 classes)
    self$dropout = nn_dropout(p = 0.3)  # Dropout for regularization
  },

  forward = function(x) {
    x %>%
      self$fc1() %>%
      nnf_relu() %>%
      self$dropout() %>%
      self$fc2() %>%
      nnf_relu() %>%
```

```r
      self$dropout() %>%
      self$fc3() %>%
      nnf_log_softmax(dim = 1)  # Apply log-softmax for classification
  }
)

# Initialize model
model = net()

# Define optimizer (SGD with learning rate)
optimizer = optim_adam(model$parameters, lr = 0.001)
```

```r
num_epochs = 20  # More epochs for better learning

for (epoch in 1:num_epochs) {
  losses = c()

  coro::loop(for (batch in dataloader) {
    optimizer$zero_grad()  # Reset gradients

    output = model(batch$x)  # Forward pass
    loss = nnf_nll_loss(output, batch$y)  # Compute loss

    loss$backward()  # Backpropagation
    optimizer$step()  # Update model parameters

    losses = c(losses, loss$item())  # Store loss for reporting
  })

  # Print loss for each epoch
  cat(sprintf("Epoch %d: Loss = %.4f\n", epoch, mean(losses)))
}
```

```
## Epoch 1: Loss = 3.1979
## Epoch 2: Loss = 3.0416
## Epoch 3: Loss = 3.0045
## Epoch 4: Loss = 2.9870
## Epoch 5: Loss = 2.9756
## Epoch 6: Loss = 2.9696
## Epoch 7: Loss = 2.9640
## Epoch 8: Loss = 2.9511
## Epoch 9: Loss = 2.9512
## Epoch 10: Loss = 2.9479
## Epoch 11: Loss = 2.9405
## Epoch 12: Loss = 2.9398
## Epoch 13: Loss = 2.9339
## Epoch 14: Loss = 2.9359
## Epoch 15: Loss = 2.9239
## Epoch 16: Loss = 2.9241
## Epoch 17: Loss = 2.9152
## Epoch 18: Loss = 2.9161
## Epoch 19: Loss = 2.9115
## Epoch 20: Loss = 2.9202
```

R Markdown

Steps Involved

1. Dataset Loading: Downloaded and processed the Spam dataset from the UCI ML Repository.
2. Data Preprocessing:

- Converted it into a matrix.
- Applied normalization (zero mean, unit variance).
- Converted it into torch tensors.

3. Dataset Handling: Implemented a custom dataset class (spam_dataset) for structured data processing.
4. Batch Processing: Used torch dataloaders to efficiently load batches.
5. Model Training:

- Implemented a 3-layer neural network with:
- ReLU activations.
- Dropout for regularization.
- Log-softmax for classification.
- Optimized with Adam optimizer.
- Trained for 20 epochs, tracking the loss.
