IMDB_dataset

Z.CAI

2021 11 24

Classifiying Movie Reviews: A Binary Classification Example - the IMDB dataset

loading the IMDB dataset

```
library(keras)

imdb <- dataset_imdb(num_words = 10000)
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\CAI\anaconda3\envs\rstudio_conda/python.exe":

## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\CAI\anaconda3\envs\tensorflow_2.6/python.exe":

## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\CAI\anaconda3\envs\rstudio_conda/python.exe":

## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\CAI\anaconda3\envs\tensorflow_2.6/python.exe":

## Loaded Tensorflow version 2.6.1
```

```
c(c(train_data, train_labels), c(test_data, test_labels)) %<-% imdb
```

the multi_assignment (%<-%) operator equally:

```
imdb <- dataset_imdb(num_words = 10000)

train_data <- imdb$train$x
train_labels <- imdb$train$y
test_data <- imdb$test$x
test_labels <- imdb$test$y
```

```
str(train_data[[1]])
```

```
##  int [1:218] 1 14 22 16 43 530 973 1622 1385 65 ...
```

```
train_labels[[1]]
```

```
## [1] 1
```

```
max(sapply(train_data, max))
```

```
## [1] 9999
```

decodes into English words

```
word_index <- dataset_imdb_word_index()
reverse_word_index <- names(word_index)
names(reverse_word_index) <- word_index

decoded_review <- sapply(train_data[[1]], function(index) {
  word <- if (index >= 3) reverse_word_index[[as.character(index - 3)]]
  if (!is.null(word)) word else "?"
})

decoded_review
```

```
##   [1] "?"          "this"       "film"       "was"        "just"
##   [6] "brilliant"  "casting"    "location"   "scenery"    "story"
##  [11] "direction"  "everyone's" "really"     "suited"     "the"
##  [16] "part"       "they"       "played"     "and"        "you"
##  [21] "could"      "just"       "imagine"    "being"      "there"
##  [26] "robert"     "?"          "is"         "an"         "amazing"
##  [31] "actor"      "and"        "now"        "the"        "same"
##  [36] "being"      "director"   "?"          "father"     "came"
##  [41] "from"       "the"        "same"       "scottish"   "island"
##  [46] "as"         "myself"     "so"         "i"          "loved"
##  [51] "the"        "fact"       "there"      "was"        "a"
##  [56] "real"       "connection" "with"       "this"       "film"
##  [61] "the"        "witty"      "remarks"    "throughout" "the"
##  [66] "film"       "were"       "great"      "it"         "was"
##  [71] "just"       "brilliant"  "so"         "much"       "that"
##  [76] "i"          "bought"     "the"        "film"       "as"
##  [81] "soon"       "as"         "it"         "was"        "released"
##  [86] "for"        "?"          "and"        "would"      "recommend"
##  [91] "it"         "to"         "everyone"   "to"         "watch"
##  [96] "and"        "the"        "fly"        "fishing"    "was"
## [101] "amazing"    "really"     "cried"      "at"         "the"
## [106] "end"        "it"         "was"        "so"         "sad"
## [111] "and"        "you"        "know"       "what"       "they"
## [116] "say"        "if"         "you"        "cry"        "at"
## [121] "a"          "film"       "it"         "must"       "have"
## [126] "been"       "good"       "and"        "this"       "definitely"
## [131] "was"        "also"       "?"          "to"         "the"
## [136] "two"        "little"     "boy's"      "that"       "played"
## [141] "the"        "?"          "of"         "norman"     "and"
## [146] "paul"       "they"       "were"       "just"       "brilliant"
```

```
## [151] "children"   "are"        "often"      "left"       "out"
## [156] "of"         "the"        "?"          "list"       "i"
## [161] "think"      "because"    "the"        "stars"      "that"
## [166] "play"       "them"       "all"        "grown"      "up"
## [171] "are"        "such"       "a"          "big"        "profile"
## [176] "for"        "the"        "whole"      "film"       "but"
## [181] "these"      "children"   "are"        "amazing"    "and"
## [186] "should"     "be"         "praised"    "for"        "what"
## [191] "they"       "have"       "done"       "don't"      "you"
## [196] "think"      "the"        "whole"      "story"      "was"
## [201] "so"         "lovely"     "because"    "it"         "was"
## [206] "true"       "and"        "was"        "someone's"  "life"
## [211] "after"      "all"        "that"       "was"        "shared"
## [216] "with"       "us"         "all"
```

encoding the integer sequenes into a binary matrix

```
vectorise_sequences <- function(sequences, dimension = 10000) {
  results <- matrix(0, nrow = length(sequences), ncol = dimension)
  for (i in 1:length(sequences))
    results[i, sequences[[i]]] <- 1
  results
}

x_train <- vectorise_sequences(train_data)
x_test <- vectorise_sequences(test_data)
```

```
str(x_train[1,])
```

```
##  num [1:10000] 1 1 0 1 1 1 1 1 1 0 ...
```

```
y_train <- as.numeric(train_labels)
y_test <- as.numeric(test_labels)
```

the model definition

```
library(keras)

model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
```

compiling the model

```
model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

alternatively: configure the optimizer

```
model %>% compile(
  optimizer = optimizer_rmsprop(lr=0.001),
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

```
## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argument has
## been renamed to `learning_rate`.
```

alternatively: using custom losses and metrics

```
model %>% compile(
  optimizer = optimizer_rmsprop(lr=0.001),
  loss = "loss_binary_crossentropy",
  metrics = metric_binary_accuracy
)
```

```
## Warning in backcompat_fix_rename_lr_to_learning_rate(...): the `lr` argument has
## been renamed to `learning_rate`.
```

setting aside a validation set

```
val_indices <- 1:10000

x_val <- x_train[val_indices,]
partial_x_train <- x_train[-val_indices,]
y_val <- y_train[val_indices]
partial_y_train <- y_train[-val_indices]
```

training the model

```
model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

history <- model %>% fit(
  partial_x_train,
  partial_y_train,
  epochs = 20,
  batch_size = 512,
  validation_data = list(x_val, y_val)
)
```

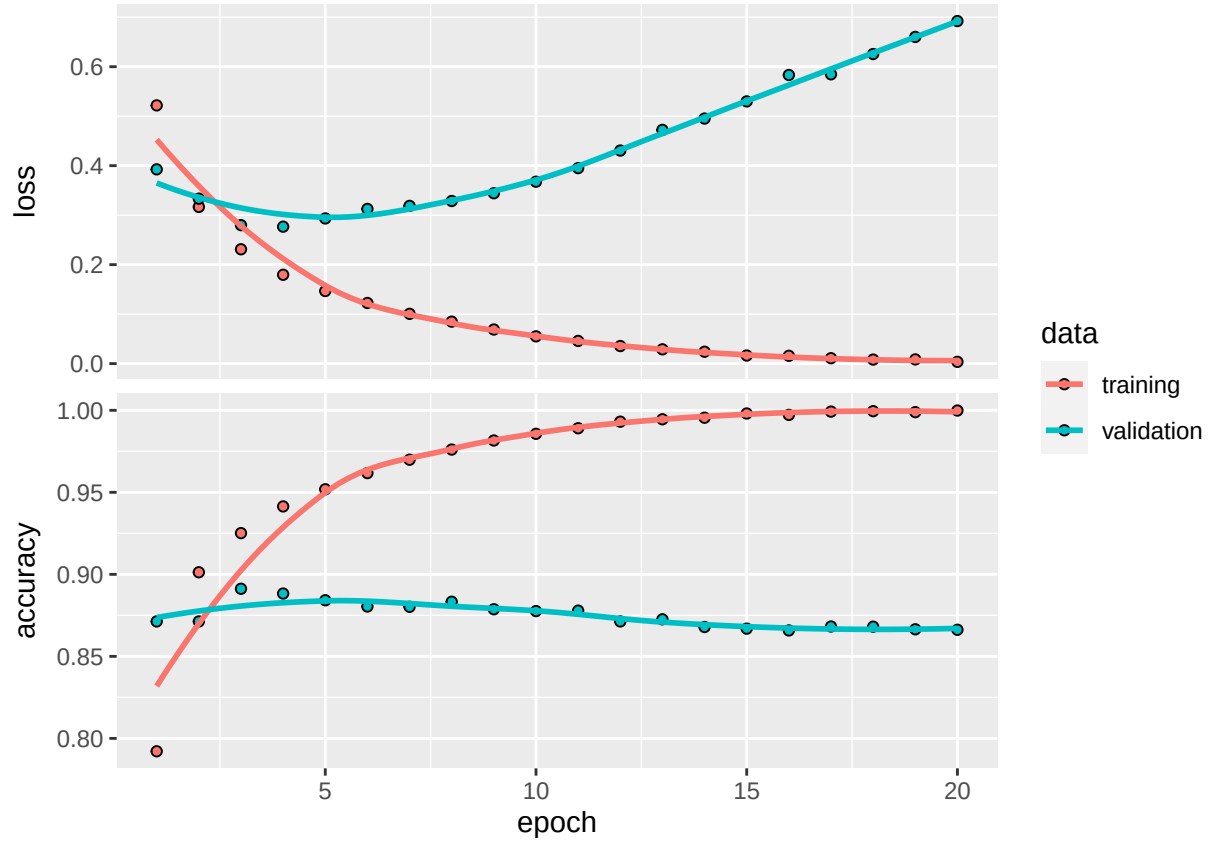
```
str(history)
```

```
## List of 2
##  $ params :List of 3
##   ..$ verbose: int 1
##   ..$ epochs : int 20
```

```
##   ..$ steps  : int 30
##  $ metrics:List of 4
##   ..$ loss        : num [1:20] 0.522 0.317 0.231 0.18 0.147 ...
##   ..$ accuracy    : num [1:20] 0.792 0.901 0.925 0.941 0.952 ...
##   ..$ val_loss    : num [1:20] 0.393 0.333 0.28 0.277 0.293 ...
##   ..$ val_accuracy: num [1:20] 0.871 0.871 0.891 0.888 0.884 ...
##  - attr(*, "class")= chr "keras_training_history"
```

```
plot(history)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
history_df <- as.data.frame(history)
str(history_df)
```

```
## 'data.frame':    80 obs. of  4 variables:
##  $ epoch : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ value : num  0.522 0.317 0.231 0.18 0.147 ...
##  $ metric: Factor w/ 2 levels "loss","accuracy": 1 1 1 1 1 1 1 1 1 1 ...
##  $ data  : Factor w/ 2 levels "training","validation": 1 1 1 1 1 1 1 1 1 1 ...
```

retraining a model from scratch

```
model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

model %>% fit(x_train, y_train, epochs = 4, batch_size = 512)
results <- model %>% evaluate(x_test, y_test)

results
```

```
##      loss  accuracy
## 0.2898905 0.8844800
```

generate predictions on new data

```
model %>% predict(x_test[1:10,])
```

```
##             [,1]
##  [1,] 0.26662400
##  [2,] 0.99988472
##  [3,] 0.93186337
##  [4,] 0.83929491
##  [5,] 0.95941567
##  [6,] 0.85135883
##  [7,] 0.99969268
##  [8,] 0.01342796
##  [9,] 0.97243911
## [10,] 0.99525166
```