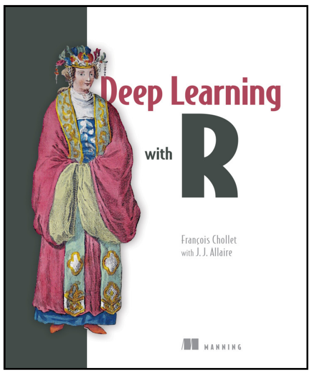
MANNING PUBLICATIONS



Errata: *January 14, 2019*

Thank you for purchasing *Deep Learning with R*. Please post any errors, other than those listed below, in the book's Author Online Forum. We'll update this list as necessary. Thank you!

Page 35, section 2.3.1

The code for the `naive_relu()` and `naive_add()` functions needs to use the full range of row and column indexes for iteration. The correct code is:

```
naive_relu <- function(x) {
  for (i in 1:nrow(x))
    for (j in 1:ncol(x))
      x[i, j] <- max(x[i, j], 0)
  x
}

naive_add <- function(x, y) {
  for (i in 1:nrow(x))
    for (j in 1:ncol(x))
      x[i, j] = x[i, j] + y[i, j]
  x
}
```

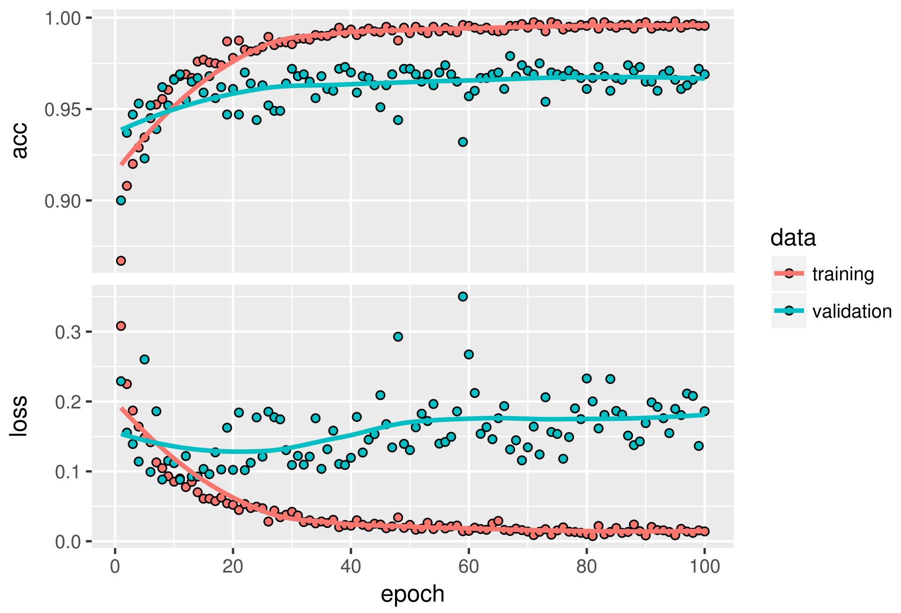
Page 79, listing 3.24
Page 90, listing 4.2

The code that demonstrates k-fold validation has an error in the call to `cut()`. The code should read as follows:

```
folds <- cut(1:length(indices), breaks = k, labels =
  FALSE)
```

Page 145, figure 5.16

The PDF/eBook version of the book has an incorrect Figure 5.16: Training and validation metrics for fine-tuning. This figure was modified during editing and only the black and white version used in the print book was updated. The corrected figure is below:



Page 169, listing 6.4

In the word level one-hot encoding example, the invocation of the hash function should use index j rather than index 1:

```
index <- abs(spooky.32(words[[j]])) %% dimensionality
```

Page 195, listing 6.33

The generator function has an issue with selection of rows (some observations at boundaries are excluded). The generator code should be updated to:

```
generator <- function(data, lookback, delay,
  min_index, max_index,
                      shuffle = FALSE, batch_size =
  128, step = 6) {
  if (is.null(max_index))
    max_index <- nrow(data) - delay - 1
  i <- min_index + lookback
  function() {
    if (shuffle) {
      rows <-
  sample(c((min_index+lookback):max_index), size =
  batch_size)
    } else {
      if (i + batch_size >= max_index)
        i <<- min_index + lookback
      rows <- c(i:min(i+batch_size-1, max_index))
      i <<- i + length(rows)
    }

    samples <- array(0, dim = c(length(rows),
                                lookback / step,
                                dim(data)[[-1]]))
    targets <- array(0, dim = c(length(rows)))

    for (j in 1:length(rows)) {
      indices <- seq(rows[[j]] - lookback,
  rows[[j]]-1,
                     length.out = dim(samples)[[2]])
      samples[j,,] <- data[indices,]
      targets[[j]] <- data[rows[[j]] + delay,2]
    }
    list(samples, targets)
  }
}
```

Page 223, listing 7.1

The code which creates the embedding layers should be modified as follows:

```
text_input <- layer_input(shape = list(NULL), dtype =
  "int32", name = "text")
encoded_text <- text_input %>%
  layer_embedding(input_dim = text_vocabulary_size+1,
  output_dim = 32) %>%
  layer_lstm(units = 32)
```

```r
question_input <- layer_input(shape = list(NULL),
  dtype = "int32", name = "question")
encoded_question <- question_input %>%
  layer_embedding(input_dim = ques_vocabulary_size+1,
  output_dim = 16) %>%
  layer_lstm(units = 16)
```

Page 225, listing 7.3

The code which defines `embedded_posts` should read:

```r
embedded_posts <- posts_input %>%
  layer_embedding(input_dim = vocabulary_size+1,
  output_dim = 256)
```

Page 238, listing 7.9

The tensorboard example should be modified to remove the `embeddings_freq` parameter:

```r
callbacks = list(
  callback_tensorboard(
    log_dir = "my_log_dir",
    histogram_freq = 1
))
```

Page 262, listing 8.10

The calculation of the loss should be updated as follows:

```r
# This code is no longer required since we don't use
  layer_dict below
# layer_dict <- model$layers
# names(layer_dict) <- lapply(layer_dict,
  function(layer) layer$name)


loss <- k_variable(0)
for (layer_name in names(layer_contributions)) {
  coeff <- layer_contributions[[layer_name]]
  activation <- get_layer(model, layer_name)$output
  scaling <- k_prod(k_cast(k_shape(activation),
  "float32"))
  loss <- loss + (coeff * k_sum(k_square(activation))
  / scaling)
}
```

Page 319, section A.4

The shell command for monitoring GPU utilization should be:

```
$ watch -n 5 nvidia-smi -a --display=utilization
```
