STAT 4710: Homework 5

Name

Due: December 6, 2022 at 9:00pm

Contents

Instructions

Materials

The allowed materials are as stated on the Syllabus:

> "Students may consult all course materials, including course textbooks, for all assignments and assessments. For programming-based assignments (homeworks and exams), students may also consult the internet (e.g. Stack Overflow) for help with general programming tasks (e.g. how to add a dashed line to a plot). Students may not search the internet for help with specific questions or specific datasets on any homework or exam. In particular, students may not use solutions to problems that may be available online and/or from past iterations of the course."

Collaboration

The collaboration policy is as stated on the Syllabus:

> "Students are permitted to work together on homework assignments, but must write up and submit solutions individually. In particular, students may not copy each others' solutions. Furthermore, students must disclose all classmates with whom they collaborated on a given homework assignment."

In accordance with this policy,

Please list anyone you discussed this homework with:

Writeup

Use this document as a starting point for your writeup, adding your R code using code chunks and adding your text answers using **bold text**. Consult the preparing reports guide for guidance on compilation, creation

of figures and tables, and presentation quality. In particular, if the instructions ask you to "print a table", you should use `kable`. If the instructions ask you to "print a tibble", you should not use `kable` and instead print the tibble directly.

Programming

The `tidyverse` paradigm for data visualization, manipulation, and wrangling is required. No points will be awarded for code written in base R.

We'll need to use the following `R` and `python` packages:

```
library(keras)          # to train neural networks
install_keras()         # install keras Python package and its dependencies
                        # (need to run just once; delete line after installation)
library(kableExtra)     # to print tables
library(cowplot)        # to print side-by-side plots
library(stat471)        # for deep learning helper functions
library(tidyverse)      # tidyverse
```

Grading

The point value for each problem sub-part is indicated. Additionally, the presentation quality of the solution for each problem (as exemplified by the guidelines in Section 4 of the preparing reports guide will be evaluated on a per-problem basis. There are 100 points possible on this homework, 85 of which are for correctness and 15 of which are for presentation.

Fashion MNIST Data

In this homework, we will analyze the Fashion MNIST data, which is like MNIST but with clothing items rather than handwritten digits. There are ten classes, as listed in Table 1.

Table 1: The ten classes in the Fashion MNIST data.

Index	Name
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

The code provided below loads the data, and prepares it for modeling with `keras`.

```
# load the data
fashion_mnist <- dataset_fashion_mnist()

# extract information about the images
num_classes <- nrow(class_names)                    # number of image classes
```

```
num_train_images <- dim(fashion_mnist$train$x)[1]    # number of training images
num_test_images <- dim(fashion_mnist$test$x)[1]      # number of test images
img_rows <- dim(fashion_mnist$train$x)[2]            # rows per image
img_cols <- dim(fashion_mnist$train$x)[3]            # columns per image
num_pixels <- img_rows*img_cols                      # pixels per image
max_intensity <- 255                                 # max pixel intensity

# normalize and reshape the images
x_train <- array_reshape(fashion_mnist$train$x/max_intensity,
                         c(num_train_images, img_rows, img_cols, 1))
x_test <- array_reshape(fashion_mnist$test$x/max_intensity,
                        c(num_test_images, img_rows, img_cols, 1))

# extract the responses from the training and test data
g_train <- fashion_mnist$train$y
g_test <- fashion_mnist$test$y

# recode response labels using "one-hot" representation
y_train <- to_categorical(g_train, num_classes)
y_test <- to_categorical(g_test, num_classes)
```

1 Data exploration

1. How many observations in each class are there in the training data? (Kable output optional.) [Hint: Try the `table()` function.]
2. Plot the first six training images in a 2×3 grid, each image titled with its class name from the second column of Table 1.
3. Comment on the extent to which you (a human) would have been able to successfully classify the observations plotted in part ii. Would you have had any trouble? If so, with which observations?
4. What is the human performance on this classification task? You can find it at the Fashion MNIST webpage linked above by searching for "human performance."

2 Model training

2.1 Multi-class logistic regression

1. Define a `keras_model_sequential` object called `model_lr` for multi-class logistic regression, and compile it using the `categorical_crossentropy` loss, the `adam` optimizer, and the `accuracy` metric.
2. Print the summary of the model (no need to use kable). How many total parameters are there? How does this number follow from the architecture of this simple neural network?
3. Train the model for 10 epochs, using a batch size of 128, and a validation split of 20%. Save the model to `model_lr.h5` and its history to `model_lr_hist.RDS`, and then set this code chunk to `eval = FALSE` to avoid recomputation. How many total stochastic gradient steps were taken while training this model, and how did you arrive at this number? Based on the output printed during training, roughly how many milliseconds did each stochastic gradient step take?
4. Load the model and its history from the files saved in part iii. Create a plot of the training history. Based on the shape of the validation loss curve, has any overfitting occurred during the first 10 epochs?

2.2 Fully connected neural network

1. Define a `keras_model_sequential` object called `model_nn` for a fully connected neural network with three hidden layers with 256, 128, and 64 units, `relu` activations, and dropout proportions 0.4, 0.3, and 0.2, respectively. Compile it using the `categorical_crossentropy` loss, the `rmsprop` optimizer, and the `accuracy` metric.
2. Print the summary of the model. How many total parameters are there? How many parameters correspond to the connections between the second and third hidden layers? How does this number follow from the architecture of the neural network?
3. Train the model using 15 epochs, a batch size of 128, and a validation split of 0.2. Save the model to "model_nn.h5" and its history to "model_nn_hist.RDS", and then set this code chunk to `eval = FALSE` to avoid recomputation. Based on the output printed during training, roughly how many milliseconds did each stochastic gradient step take?
4. Load the model and its history from the files saved in part 3. Create a plot of the training history.

2.3 Convolutional neural network

1. Define a `keras_model_sequential` object called `model_cnn` for a convolutional neural network with a convolutional layer with 32 3×3 filters, followed by a convolutional layer with 64 3×3 filters, followed by a max-pooling step with 2×2 pool size with 25% dropout, followed by a fully-connected layer with 128 units and 50% dropout, followed by a softmax output layer. All layers except the output layer should have `relu` activations. Compile the model using the `categorical_crossentropy` loss, the `adadelta` optimizer, and the `accuracy` metric.
2. Print the summary of the model. How many total parameters are there? How many parameters correspond to the connections between the first and second convolutional layers? How does this number follow from the architecture of the neural network?
3. Train the model using 5 epochs, a batch size of 128, and a validation split of 0.2. Save the model to `model_cnn.h5` and its history to `model_cnn_hist.RDS`, and then set this code chunk to `eval = FALSE` to avoid recomputation. Based on the output printed during training, roughly how many milliseconds did each stochastic gradient step take?
4. Load the model and its history from the files saved in part 3. Create a plot of the training history.

3 Evaluation

1. Evaluate the test accuracy for each of the three trained neural network models. Output this information in a table, along with the number of layers, number of parameters, and milliseconds per stochastic gradient descent step. Also include a row in the table for human performance. Compare and contrast the three neural networks and human performance based on this table.
2. Plot confusion matrices for each of the three methods. For each method, what class gets misclassified most frequently? What is most frequent wrong label for this class?
3. Consider CNN's most frequently misclassified class. What are the three most common incorrect classifications for this class? Extract one image representing each of these three type of misclassifications, and plot these side by side (titled with their predicted labels). Would you have gotten these right?