Lab 3 - Machine Learning for Social Science

To be handed in no later than September 17th 10:00. The submission should include code, relevant output, as well as answers to questions. We recommend the use of RMarkdown to create the report.

Part 1: Salary prediction

In the first part of this lab, we will work with a dataset (`adults`) containing information about a sample of US individuals' salaries as well as a handful sociodemographic variables. We will also consider an augmented version of this dataset (`adults_aug`), combining the original data with (simulated) highly nuanced lifecourse information. With this, the objective is to explore how accurately we can predict the salaries of individuals, comparing neural networks with other methods from previous weeks.

1. Begin by importing `adults.rds` and partition the data into a *training* and *test* set. We will use the training set to fit our model, and the test set to assess accuracy and compare across models. The dataset contains cirka 50,000 individuals, and for each we have information about five traditional variables (age, education, hours worked per week, capital gain and capital loss). We are reasonably sure there are no complex interactions or non-linearities. With this as the background, do you believe a neural network model is likely to excel on this dataset? Why/why not?

2. Begin with estimating a standard logistic regression model to the training set you just created. Hint: you may use the glm(. . . , family='binomial') function to estimate a standard logistic regression model. When estimation has finished, use the predict() function to predict the outcome on the test dataset, and calculate (report) the accuracy.

3. Estimate a neural network with 1 *hidden layer* and 5 *hidden units*. In the `compile()` function, set the optimizer to `optimizer_adam()`, the loss function equal to `"binary_crossentropy"` (as the outcome is binary), and metrics to `"accuracy"`. Report the accuracy. Did the result match your expectations formulated in #1?

4. Next, you shall estimate a considerably more complex neural network, containing 4 *hidden layers*. The first hidden layer should have 256 *hidden units*, the second hidden layer 128 hidden units, the third hidden layer 64 hidden units, and the fourth hidden layer 32 hidden units. Use the same settings for `compile()` as in #3. Report the *total number of parameters* and then estimate the model. Do you find that it outperforms #3 meaningfully? Why do you think this is (or is not) the case?

5. Suppose now that we retrieve a dataset (`adults_aug`) that expands upon the original '`adults` dataset. This dataset contains 6 extra variables; which capture complex aspects of the individuals life courses, with various interdependencies between them and possible non-linearities. Could this expanded data make a difference in terms of more clearly outperforming the standard logit? Investigate this by repeating steps 2-4 on the adults_aug dataset. Does your conclusion about the relevance of more complex network structure differ between the two datasets? Why?

6. Given the results, you may find it unnecessary to to try further revisions. But for learning purposes, suppose we want to re-estimate the best-performing model on the adults_aug data using *dropout learning*. Update the model to perform dropout for the first three hidden layers. Estimate one model

where you inactivate 10% for each of the hidden layers, and a second model where you inactivate 90% per layer, and report the results. In what direction does the results change? Do you attribute this change to a change in bias or in variance and why?

Part 2: image prediction

In the second part of the lab, we'll work with *Fashion MNIST data* (https://www.kaggle.com/datasets/zalando-research/fashionmnist) containing images of 10 different types of fashion items — including t-shirts, coats, bags, and sneakers. All images are 28 pixels in width and 28 pixels in height (784 in total). Each pixel has a single value associated with it; its gray-scale value ranging between 0 and 255.

For the purpose of this lab, we are imagining a fictive scenario in which each image correspond to a consumed good, and that we have (simulated) *basic socioeconomic information* about the *customer* who bought the fahion item, as well as the *year* it was consumed. Our social scientific interest is to examine whether there are any trends in the association between types of fashion goods and the socioeconomic status of the customer. For example, are sneakers more or less associated with low/high economic status, and has this changed over time?

The practical problem that we face, is that we only have labeled data of the images (i.e., identifying which fashion item is in the image) for part of the time-period (in 2016 but not in 2017). Our objective, thus, is to use *deep learning* to estimate a model on the part of the data where we do have labeled image data, and then use the trained model to predict on the yet to be labeled images.

1. Begin by importing the datafiles *"fashion_2016_train.rds"* and *"fashion_2016_test.rds"*.

2. Next, estimate a simple convolutional neural network with K=1 convolutional layers and M=1 regular type of (fully connected) hidden layers. Specify the *number of filters* (in the convolutional layer) and the *number of hidden units* (in the fully connected / regular hidden slides) to be 8. Remember also that included after each added convolutional layer, *pooling* (max-pooling, $2 * 2$) should to be applied. Finally, include `layer_dense(units = 10, activation = "softmax")` at the end. When *compiling* the model using the compile() function, set loss = "sparse_categorical_crossentroy". Report the results, and reflect on whether you think improvement can be made by increasing either M, K, or the number of filters or hidden units in the regular hidden layer.

3. Next, you shall estimate a slightly more complex convolutional neural network, increasing the number of filters to 32 in the first layer, and 64 in the second convolutional layer. Similarly, increase the number of hidden units in the first regular/fully-connected hidden layer to 64, and the second to 32. Report the results. Does this slightly more complicated model improve/degrade upon the simple one in #2? Speculate why in terms of the bias/variance trade-off. Report and contrast also the number of parameters between the two.

4. In #2-3, we used max-pooling of 2*2. Why is it generally not a good idea to increase the dimension of the pooling to become large relative to the images? Would this increase bias or variance?

5. Because we are interested in using the predictions from these models for downstream analysis (of trends in consumer polarization), it is extra important to validate the measure. This is what you shall do now, focusing on the CNN model you deemed the best thus far. In particular you should break down the accuracy per item category. Is there meaninful difference in predictability beween item classes? Does the ordering make substantive sense? Hint: to compute accuracy by group, you may use the following code:

```r
predictions <- model_fashion %>% predict(x_test)
predicted_classes <- apply(predictions, 1, which.max) - 1
accuracy_by_class_dt <- data.table(actual=y_test,predicted=predicted_classes)
accuracy_by_class_dt[,correct := fifelse(actual==predicted, yes = 1, no = 0)]
```

```r
fashion_labels <- c("T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
                    "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot")
fashion_labels_dt <- data.table(label=fashion_labels,number=0:9)
accuracy_by_class_dt <- merge(x=accuracy_by_class_dt,y=fashion_labels_dt,by.x='actual',by.y='number')
accuracy_by_class_dt[,.(accuracy=mean(correct)),by=label][order(accuracy,decreasing = T)]
```

6. Next, we shall finally adress the question we set out to answer: how patterns in consumer behavior changed between 2016 to 2017. To answer this question, please follow these steps:

- Import the `fashion2016_2017_unlabeled.rds` file, which contains all images and the sociodemographic info of the individual associated with the image. Note: the data has already been preprocced (normalized pixel values, etc.).
- Use the predict() function to predict for this dataset based on the CNN model you thought were the best. Note that, when you use the predict() function for a model where you have multiple-category outcome variables, like here, each observation get a probability over the items. To assign the prediction to the maximum value, you may use this code:

```r
apply(preds_2016_2017, 1, which.max) - 1
```

- Then you can simply merge the predicted category with the demographic variable data.frame, and calculate various associations by classical statistical means.
- Report your findings.