Homework 7

1. Predicting Covid-19 infection based on symptoms.

In this problem we would like to build a binary classifier that given a list of symptoms, predicts whether or not a patient has Covid-19.

In covid_symptoms.json you will find a matrix $U$ with 20 binary features (answers to yes/no questions) and a vector $v$ with the binary classification output: 1 if the patient was diagnosed with Covid-19 and -1 if they were not.

We have also provided a function in classification_regression_fit.jl.

classification_regression_fit(X, Y, l, r, lambda, kappa; numiters=40)

This function returns $\theta^*$, the parameter vector for a linear classifier:

$$
\theta^* \leftarrow \text{minimize } l(\hat{y}, y) + \lambda r(\theta)
$$

This function takes in input/output data $X$ and $Y$, a loss function $l(yhat, y)$, a regularizer $r(\text{theta})$, $\lambda$, $\kappa$, and optionally, the number of epochs Flux should run. You will have to import LinearAlgebra and Flux to run the function.

(a) In binary classification we can use the Neyman-Pearson metric with parameter $\kappa$:

$$
\kappa E_{fn} + E_{fp}
$$

$\kappa$ expresses our relative dislike for mistaking a positive example for a negative one. In this context (predicting Covid-19 infection), which is more dangerous: a false positive or a false negative? Should $\kappa$ be less than 1 or greater than 1? Provide a short justification explaining your answer.

(b) First, add a column of ones to $U$ (as in previous homeworks). We will use a random 90/10 train/test split. Partition your data into $U_{\text{train}}$, $v_{\text{train}}$, $U_{\text{test}}$ and $v_{\text{test}}$.

We will use SVM. Define the hinge loss:

$$
\ell_{\text{hinge}}(\hat{y}, y) = \begin{cases} (1 + \hat{y})_+ & \text{if } y = -1 \\ \kappa(1 - \hat{y})_+ & \text{if } y = 1 \end{cases}
$$

and L2 regularizer (you can exclude the $\theta_i$ corresponding to your column of ones, although it’s not required).

$$
r(\theta) = \|\theta\|_2^2
$$

You can set $\lambda = 0.01$ or experiment with different values; the focus of this problem is understanding how $\kappa$ changes the false negative and false positive rate of our SVM classifier.
(c) For $\kappa$ between 0.01 and 10 (see the hint), use `classification_regression_fit` to compute the predictor $\theta$. Find the accuracy and number of false negatives and positives for the test set. Plot the false negatives and false positives on one graph (with $\kappa$ on the x axis).

Plot the accuracy on a separate graph (with $\kappa$ on the x axis).

(d) Evaluate your results from part (c) and choose the “best” classifier (for example, the one with highest accuracy or fewest false negatives). Report the accuracy and confusion matrix of the “best” classifier you chose.

**Hint:** Use `kappas = 10 .^ range(-2,1,length=10)` to compute your vector of $\kappa$ values.

**Hint:** `sign.(y_hat)` will convert a vector `y_hat` of floating-point values to a prediction of -1 or +1.

**Hint:** Using a logarithmic x axis for these graphs will help you see the trend. To plot $x$ and $y$ with a log-space x axis, use `plot(x, y, xaxis=:log)`.

2. **Multi-class animal classification.** Our task is to create a predictor which identifies the class type of the animal based on 16 traits: hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, legs, tail, domestic, catsize. All traits except legs (Numeric) are Boolean.

There are 7 classes of animals. (Unfortunately the names of the animals are not included in the data file, only their numbers.)

In `zoo.json`, you will find an $n \times d$ matrix $U$ of data, with rows $(u^i)^T$, with $u^i \in \mathbb{R}^{16}$, and $n$-vector $v$ with the class type of the animal.

We have provided a function

\[
nn\text{multiclass\_classification}(X, Y, n\_classes)
\]

You can find this in `nn_multiclass_classification.jl`.

(a) Inspect `nn_multiclass_classification.jl`. What is the form of the neural network model? Specify the number of layers: for all layers, specify the activation function and model parameters (including dimensions).

In total, how many scalar model parameters are three?

(b) Convert the $v$ vector provided in `zoo.json` to one-hot format. You do not need to do any processing on the $U$ matrix (although you are welcome to experiment).

Using the provided function, train a classifier on a randomly selected 90% of the data. Report the accuracy for the training set and test set. Compute and report the confusion matrix for the training and test set.

**Hint:** To convert a scalar $vi$ with 7 possible values to one-hot format, use `onehot(vi, 1:7)`. To convert a vector $v$ to one-hot format, use `onehotbatch(y, 1:7)`. Run this in the Julia REPL and make sure you understand what dimensions the result takes.
Hint: There isn’t that much data to train with, so don’t expect perfect accuracy. 75% is fine. You may see varied results due to the random train-test split: if this worries you, use `Random.seed!(0)` to get reproducible results.

(c) In this problem, we provided your neural network architecture. However, when solving a problem on your own you will need to decide how many layers to use, and how many parameters to have in each layer. If you have too many parameters and not enough data, it will not be possible to train the network with your dataset. However, if you have too few layers or parameters, it will not be possible to learn complex features of your dataset. We’ll investigate two changes to this network. First, try adding another Dense layer. (You can just insert one in the middle of the `Chain`: the size and activation function is up to you.) Describe the change(s) you made. Retrain your network and report the train and test accuracy. How would you explain this performance?

Hint: If you use Jupyter notebooks, don’t forget to reload your notebook after modifying `nn_multiclass_classification.jl`.

(d) Remove the layer you added in Part (c). Now we will try making the network smaller. You can change the number of parameters, delete a layer, or both. Describe the change(s) you made. Retrain your network and report the train and test accuracy. Provide a (brief) comment on the results.