Homework 7

1. *Logistic regression.* In this problem you will implement a function to fit a linear logistic regression model to data for Boolean classification.

   (a) In `logistic_fit.jl` in the starter code, implement the logistic loss function

   $$\ell(\hat{y}, y) = \log(1 + e^{-y\hat{y}}),$$

   where $y \in \{-1, 1\}$.

   (b) The data file `logistic.json` contains a 300×30 matrix `X_train` and a 300-vector `y_train` consisting of training input and output data, and a 100×30 matrix `X_test` and a 100-vector `y_test` consisting of test input and output data, respectively.

   In `logistic_fit.jl`, the data from `logistic.json` has been loaded, standardized, and a constant feature has been added.

   If you examine the vectors `y_train` and `y_test`, you will notice that their entries are all either 0 or 1. However, the given logistic loss function in part (a) relies on the outcomes being either -1 or 1. In `logistic_fit.jl`, transform the output data so that all the zero entries of the output data are converted to -1.

   (c) In `logistic_fit.jl`, use the function `rerm_lin_reg` from `rerm_lin_reg.jl` to fit a linear model to the given training data with a quadratic regularizer for 10 logarithmically spaced values of λ between 10^{-5} and 10^{5}. Plot the train error rate and test error rate of the linear predictors versus λ.

2. *Multi-class animal classification.* Our task is to create a predictor which identifies the class type of the animal based on 20 traits: HAIR, FEATHERS, EGGS, MILK, AIRBORNE, AQUATIC, PREDATOR, TOOTHED, BACKBONE, BREATHES, VENOMOUS, FINS, 0LEGS, 2LEGS, 4LEGS, 6LEGS, 8LEGS, TAIL, DOMESTIC, CATSIZE. All traits are Boolean.

   There are 7 classes of animals. (Unfortunately the names of the animal classes are not included in the data file, only their numbers.) We want to fit a predictor to this dataset that will achieve classification low error rate on unseen animals.

   In `zoo.json`, you will find an 80×20 matrix `X_train` and a 80-vector `y_train` consisting of training input and output data, and a 21×20 matrix `X_test` and a 21-vector `y_test` consisting of test input and output data, respectively. The output data consists of integers between 1 and 7 corresponding to the class type of the animal.

   We have provided a function

   ```
   nn_multiclass_classification(X, Y, lambda, n_classes)
   ```

   You can find this in `nn_multiclass_classification.jl`. Be aware that in addition to passing a lambda value as you did in `nn_regression`, in this function you must also pass the number of classes of the output.

   (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 there?

   (b) The file `animals.jl` in the starter code already loads the given input and output data. Convert the output data provided in `zoo.json` to one-hot format using the provided `int_to_one_hot` function in `animals.jl`. Using the `nn_multiclass_classification` function, edit `animals.jl` to train a classifier on the training data for $\lambda = 10^{-5}, 10^{-3}, 10^{-1}$. Run `animals.jl`. Report the error rate for the training set and test set for each regularization parameter.

   Hint: There isn't that much data to train with, so don't expect perfect accuracy.

   (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.

   We will first investigate making the network smaller. Open `nn_multiclass_classification.jl` Change the output size of the first layer of the neural net from 10 to 5, and change the input size of the the second (and final) layer to 5 as well.

   Report how many total scalar parameters the model has after this modification. Re-run `animals.jl` and report the train and test error rate for the same three λ values from part (b). Provide a (brief) comment on the results.

(d) Now, revert the changes to the neural net in part (c). Add a new 10 by 10 dense layer with a relu activation to the neural net in between the current first and second layers.

Report how many total scalar parameters the model has after this modification. Re-run `animals.jl` and report the train and test error rate for the same three λ values from part (b). Provide a (brief) comment on the results.

(e) Out of the 9 models you fit (three neural network architectures each with three regularization parameters), which model performed the best? Provide a brief explanation as to why this might be the case.

Remark: In this problem, both the data and the neural net are tiny so that the neural net can be trained quickly. As such some of the typical artifacts associated with very large neural nets (overfitting) or very small neural nets (underfitting) don't fully manifest.