Homework 5

Before running any code for this week's problems, run the script `setup.jl` included with this week's starter code. You only need to run the script once. It will install the necessary Julia packages for this week's problems.

1. *Non-quadratic losses and regularizers.* In `rerm_lin_reg.jl` we have provided you with a function

   `rerm_lin_reg(X, Y, l, r, lambda)`.

   This function takes in input/output data `X` and `Y`, a loss function `l(yhat, y)`, a local regularizer function `r(theta)`, and a local positive regularization hyper-parameter `lambda`. It outputs the model parameters `theta` for the associated RERM linear predictor. The function handles four different loss functions:

   - Quadratic loss: $\ell(\hat{y}, y) = (\hat{y} - y)^2$.
   - Absolute loss: $\ell(\hat{y}, y) = |\hat{y} - y|$.
   - Huber loss, with $\alpha = 1$: $\ell(\hat{y}, y) = p_\alpha^{\text{hub}}(\hat{y} - y)$, where

   $$p_\alpha^{\text{hub}}(r) = \begin{cases} r^2 & |r| \leq \alpha \\ \alpha(2|r| - \alpha) & |r| > \alpha. \end{cases}$$

   - Log Huber loss, with $\alpha = 1$: $\ell(\hat{y}, y) = p_\alpha^{\text{dh}}(\hat{y} - y)$, where

   $$p_\alpha^{\text{dh}}(r) = \begin{cases} r^2 & |r| \leq \alpha \\ \alpha^2(1 - 2\log(\alpha) + \log(r^2)) & |r| > \alpha. \end{cases}$$

   The function handles two types of regularization:

   - Quadratic or ridge regularization: $r(\theta) = \|\theta_{2:k}\|_2^2$.
   - Lasso regularization: $r(\theta) = \|\theta_{2:k}\|_1$.

   (a) First, implement the penalty, loss, and regularization functions in the the file `regression_fit.jl` in the provided starter code. You only need to edit the code wherever there is a comment of the form `#TODO`. Do not make other edits. Attach your completed `regression_fit.jl`.

   (b) In `non_quadratic.json`, you will find a 500×300 matrix `X_train` and a 500-vector `y_train` consisting of raw training input and output data, and a 500×300 matrix `X_test` and a 500-vector `y_test` consisting of raw test input and output data, respectively.

   Load the data in `non_quadratic.json`. Standardize the training set, and standardize the test set using the mean and standard deviation from the corresponding feature columns in the training set. Add a constant feature.

(c) Use the loss functions you implemented in part (a) and `rerm_lin_reg` to fit linear models with each combination of quadratic, Huber, and log Huber loss; quadratic and lasso regularization; and regularization parameter λ values $10^{-2}, 10^{0}, 10^{2}$. In total, you should fit 18 linear models. Report a training and test average absolute error for each model. Attach your code.

(d) Which loss / regularizer / λ combination performs best? Create a one-sentence conjecture about why the particular model was the best one.

2. *How often does your predictor over-estimate?* In this problem, you will identify how often linear predictors with tilted absolute losses over-estimate. In `residual_props.json`, you will find a 500×10 matrix `X_train` and a 500-vector `y_train` consisting of raw training input and output data, and a 500×10 matrix `X_test` and a 500-vector `y_test` consisting of test input and output data, respectively.

   Recall that the tilted absolute penalty is

$$p_\tau(u) = \begin{cases} -\tau u & u < 0 \\ (1-\tau)u & u \geq 0, \end{cases}$$

   where $\tau \in [0, 1]$.

   Your task is to implement the tilted absolute penalty and tilted absolute loss in the file `residual_props.jl` in the provided starter code. After implementing the necessary functions, run `residual_props.jl` and report how frequently this predictor over-estimates (% of $\hat{y} > y$) on both the training and testing set for $\tau = 0.15, 0.5, 0.85$. Attach both the completed `residual_props.jl` and the generated plot of the empirical CDFs (Cumulative Distribution Function) of the residuals.