Homework 5

1. *Polynomial fit.* You are given a dataset `polyfit.json` with raw data \( u_i \in \mathbb{R} \) represented in a \( n \)-vector \( u \) and raw output \( v_i \in \mathbb{R} \) represented in a \( n \)-vector \( v \).

   (a) Split the given dataset using 80-20 train-test split. Apply embeddings \( \phi(u) = u \) and \( \psi(v) = v \) to the raw input and raw output, respectively. Fit a least squares model to the train part of the dataset. Perform out of sample validation for the introduced data split and report root-mean-square error (RMSE) for the training set and test set.

   (b) Now, we will introduce polynomial embedding which maps the raw input to the polynomial features of the degree \( d - 1 \) as shown below.

   \[
   \phi(u) = (1, u, u^2, ..., u^{d-1})
   \]  

   Write a Julia function \( X=\text{polynomial\_features}(u,d) \) which performs a polynomial embedding of degree \( d - 1 \) described above for \( n \)-vector \( u \) with raw input data and returns a \( n \times d \) matrix \( X \), where \( x_i^T \) row represents the polynomial mapping for a scalar raw data input \( u_i \). Embedding for raw output remains the same as in part (a).

   (c) Find the optimal parameters \( \theta^* \) that minimizes the square loss for the polynomial fits up to degree 20 for the train dataset. Plot the train and test RMSE versus the degree. Select the degree which enables the best fit for the given dataset. Compare and plot the fittings for one example of underfitting, one example of overfitting, and the best fit. Comment the results.

   (d) Implement a Julia function \( \text{avg\_RMSE, std = cross\_validation}(X,y,k) \) for the \( n \times d \) matrix \( X \) of features, the \( n \)-vector \( y \) of corresponding labels, and the \( k \) number of folds. The output should contain the average of root-mean-square errors for the test set and its standard deviation for the \( k \) fits.

   Plot the average RMSE versus the degree for the test set with standard deviation error bars for \( k = 5 \). Select the best degree choice for the given dataset.

   *Hint.* Explore the Plots.jl documentation to plot the error bars.

2. *Sequential outlier removal.* We consider the problem of fitting data corrupted with outliers, using a simple sequential outlier removal method. In `outlier\_rem.json`, you will find a 250 \( \times \) 10 matrix \( U_{\text{train}} \) and a 250-vector \( v_{\text{train}} \) consisting of raw training input and output data, and a 250 \( \times \) 10 matrix \( U_{\text{test}} \) and a 250-vector \( v_{\text{test}} \) consisting of raw test input and output data, respectively. We will work with input and output embeddings \( x = \phi(u) = u \) and \( y = \psi(v) = v \), and you will use a simple linear predictor (without a constant feature) with square loss to fit the model. We will judge model performance using the RMS error on the test set.
A number of the output data entries in the training set have been corrupted (but in a non-obvious way). You do not know which data points have been corrupted, or how many, but you can assume no more than 50. You will explore a simple sequential method to remove the corrupted data points and form a prediction model.

Repeat the following for 50 iterations:

- Create a linear predictor from the training data set.
- Find the data point in the training data set with the largest prediction error.
- Remove the data point from the training data set.

This results in 50 predictors. Plot the test RMS error for them, versus the number of points removed. Give a guess as to how many of the data points were corrupted, with justification.

*Julia hint.* To remove row $i$ from a matrix $X$ and a vector $y$, use $X[\text{setdiff}(1:\text{end}, i), :]$ and $y[\text{setdiff}(1:\text{end}, i)]$, respectively.

3. *Augmenting features with the average.* You are fitting a regression model $\hat{y} = x^T \theta$ to data, computing the model coefficients $\theta \in \mathbb{R}^d$ using least squares. A friend suggests adding a new feature, which is the average of the original features, *i.e.*, he suggests using the new feature vector $\tilde{x} = (x, \text{avg}(x))$. He explains that by adding this new feature, you might end up with a better model. (Of course, you would test the new model using validation.) Is this a good idea? How probable is it that this idea results in a better model with smaller error?