4. Cross-Validation and Bias-Variance decomposition

4.1 Goals

The goal of this exercise is to

- Implement 4-fold cross-validation.
- Understand bias-variance decomposition.

4.2 Data and sample code

This exercise is partly based on materials from last week's Exercise 3. If you don't have it already, please download the code and data that was given to you for Exercise 3. You might also want to use some of the functions you wrote yourself during previous exercises (e.g. `leastSquares.m`, grid search, etc.). Additionally, please download `cvDemo.m` from the course website.

4.3 Cross-validation

The simple 50%-50% split of test and training data from last week's exercise corresponds to the first phase of 2-fold cross-validation. It may not give an unbiased test error. A better way to do this is to use full K-fold cross-validation.

Exercise 4.1 Implementing 4-fold cross-validation

- Using your code from last week and the file `cvDemo.m` do cross-validation for polynomial degree 7. Plot the train and test RMSE as a function of λ. The resulting figure should look like Figure 4.1: Similar to what you had last week, but based on 4-fold cross-validation.
- How will you use 4-fold cross-validation to select the best model among various degrees, say from 2 to 10? Write code to do it.
- **BONUS:** Using cross-validation, one can also compute variance of RMSE. This tells us how confident we could be of our *model selection*. You can use boxplots to do this.

■

4.4 Visualizing bias-variance decomposition

In the lecture, we learned about the expected test and train error. In this exercise, we will reproduce the figure 7.1 from HTF (reproduced on page 3 of the "bias-variance" lecture notes). Read the lecture notes and go through the figure again to understand which quantities you need to plot.

You can generate the training and test set using the code below. Setting the `setSeed()` to a different number will generate different random sequence, e.g. `setSeed(2)` and `setSeed(998)` will generate two independent training and test data splits. This way you can generate many training and test splits. The `setSeed.m` was given to you in the last exercise session.

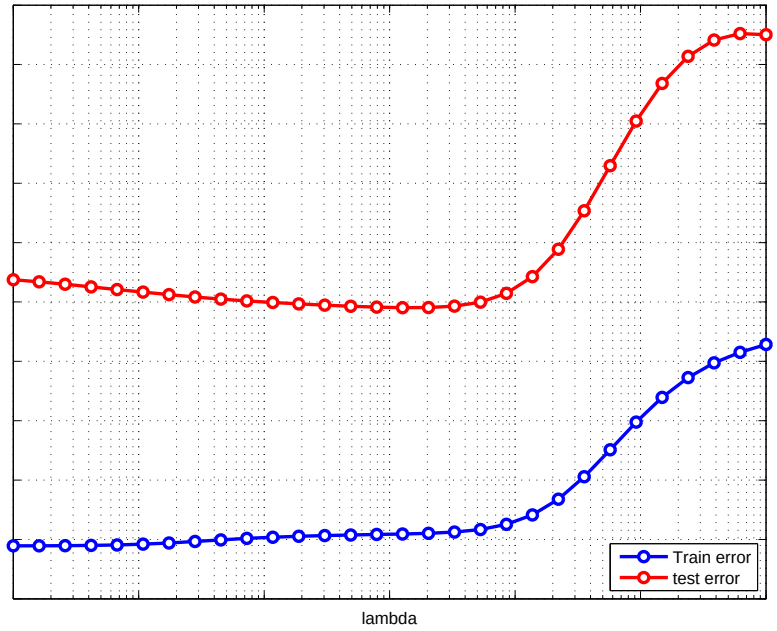

Figure 4.1: Effect of λ on training and test errors, calculated using 4-fold cross-validation

```
setSeed(1);
% generate data
N = 50;
X = linspace(0.1,2*pi,N)';
y = sin(X(:)) + 0.3*randn(N,1);
% randomly permute data
idx = randperm(N);
y = y(idx);
X = X(idx);
% split data
[XTr, yTr, XTe, yTe] = split(y,X,0.9);
```

In the above example, we are generating 50 data points and assigning 90% to training. Below is the pseudo-code with which you can plot the Figure from page 3 of the "Bias-Variance" lecture notes.

```
for s = 1:10 % # of seeds
    setSeed(s);
    % generate data as shown in above matlab code
    .....
    ....
    [XTr, yTr, XTe, yTe] = split(y,X,0.005);

    % for degree k
    degrees = [1:10]
    for k = 1:length(degrees)
        % get beta using least squares

        % compute train and test RMSE
        rmseTr(s,k) = ...
```

```
        rmseTe(s,k) = ...
    end
end
% compute expected train and test error
rmseTr_mean = mean(rmseTr);
rmseTe_mean = mean(rmseTe);
% plot
plot(degrees, rmseTe,'r-','color',[1 0.7 0.7]);
hold on
plot(degrees, rmseTr,'b-','color',[0.7 0.7 1]);
plot(degrees, rmseTe_mean,'r-','linewidth', 3);
hold on
plot(degrees, rmseTr_mean,'b-','linewidth', 3);
xlabel('degree');
ylabel('error');
```

Exercise 4.2 Visualizing bias-variance trade-off.

- Experiment with the number of seeds, number of training and testing examples to get a plot that looks similar to Fig. 4.1, i.e. on average the train error should go down and test error should go up for higher degrees.
- Look at the variance of test errors. Does it increase with the degree of polynomial?
- Another good visualization is to use the boxplot (use `ylim()` to get an appropriate visualization). You can clearly see the distribution of test error.

  ```
  boxplot(rmseTe, 'boxstyle', 'filled');
  ```

- What would you expect to happen if you replace least-squares with Ridge regression? Go through the lecture notes to understand that.
- BONUS: Produce the same plot with Ridge regression. You will have to automatically find the best λ for each degree. You do this using K-fold cross-validation (CV) only on the training set. So you need to insert the CV code inside. You also have to make sure that you chose an appropriate range of λ, so that you achieve the minimum test error.
- BONUS: In the lecture, we discussed that CV produces an estimate of expected train and test error. Check if this is the case for this data. You can do this using the CV error that you get for Ridge regression and then comparing it against the expected test error (repeating this for all the training datasets). See Fig. 7.14 of HTF book.

■

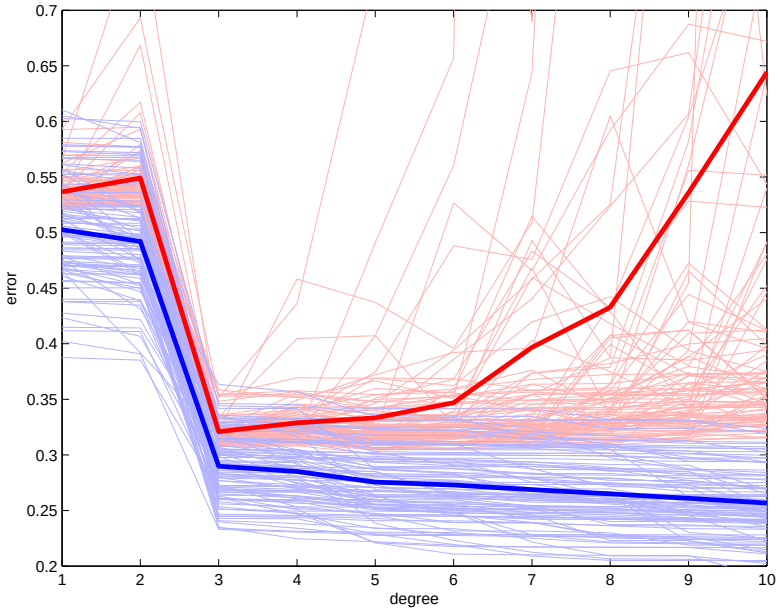

Figure 4.2: Bias-variance decomposition