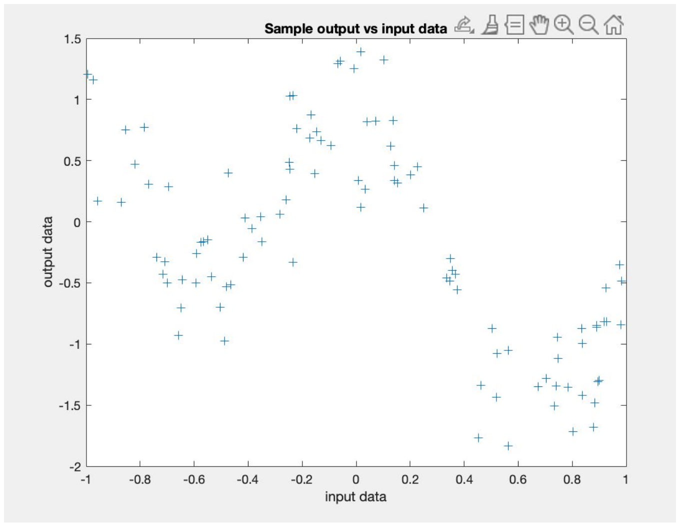
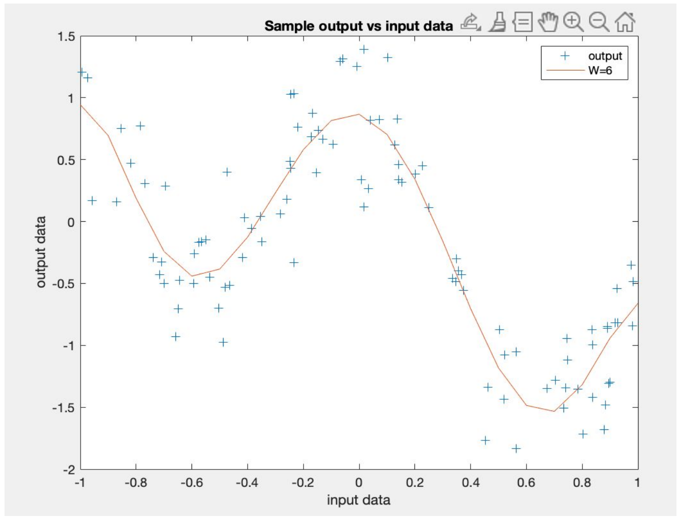
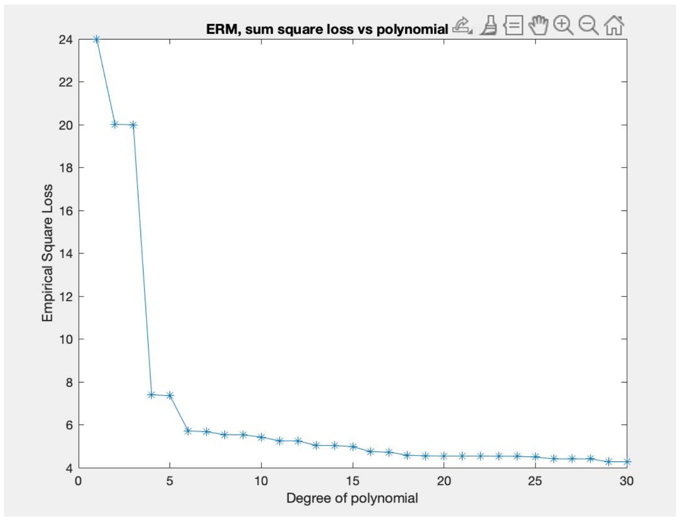
Christopher Boyd
216 869 356
EECS 4404-Fall 2020

Step 1)



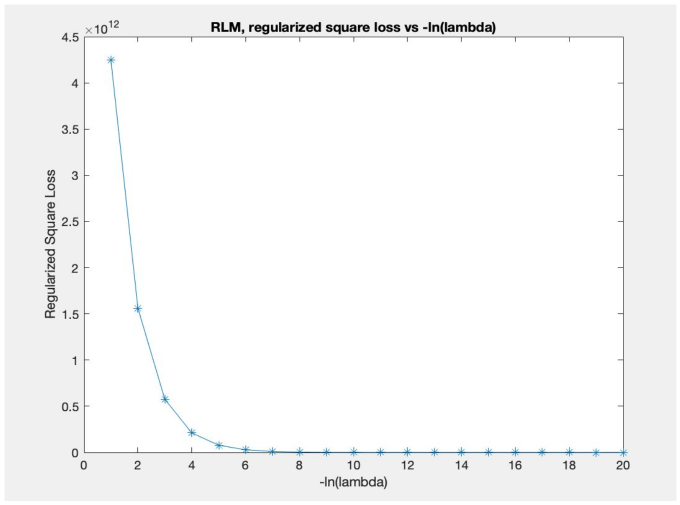


Step 2)



From the above plot, we can see that when the degree of the polynomial is 30 the square loss is minimized. However, this does not produce the most suitable value for W. The resultant polynomial is becoming over fit. In fact, once W is 25 or higher, MATLAB warns of the polynomial being badly conditioned. Taking this into account I believe a polynomial of degree 6 offers the best combination of error minimization and simplicity of model.

Step 3)

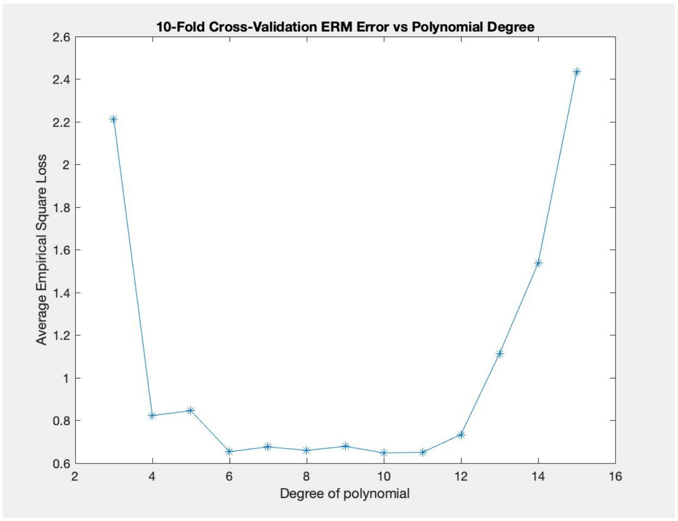


(Note: I completed this before the update to the question to use W=30 and regularization range of [1,30])
From the size of the error one can see that the regularization parameter is not capable of overcoming the large degree of the trained polynomial. Even with the largest regularization factor, $\ln(\lambda) = -20$, the error is still 2.3783e+04.

Compared to the error curve seen in step 2, where the minimal error occurs with a polynomial of degree 30, we can infer that a regularization factor is necessary to prevent a model of this complexity from being selected.

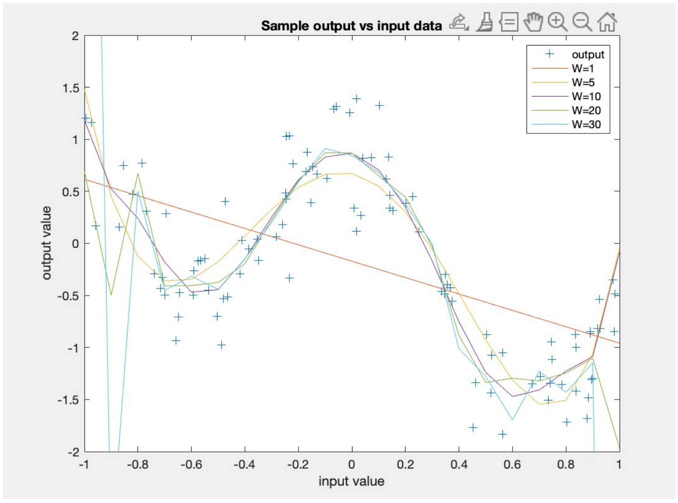
Christopher Boyd
216 869 356
EECS 4404-Fall 2020

Step 4)

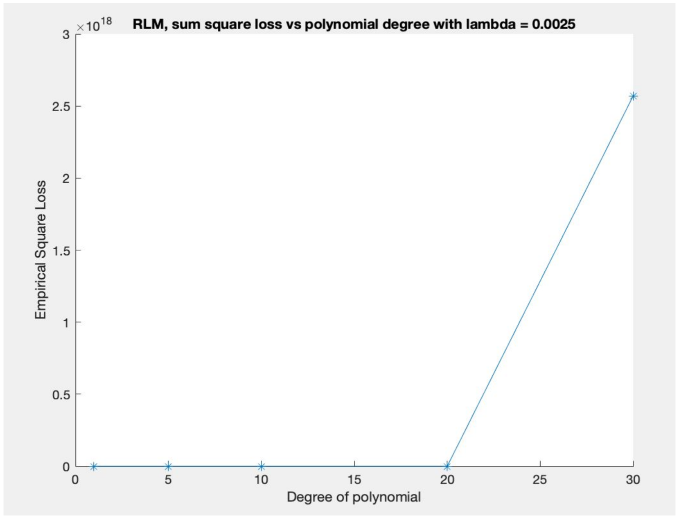


Using a W range of [3,15] shows a drop in error rate going from W=5 to W=6 and staying fairly consistent until error rates start to rise after W=11. From this I would state that W=6 is most suitable as the decreased error rate in the higher degree polynomials is not worth the added complexity.

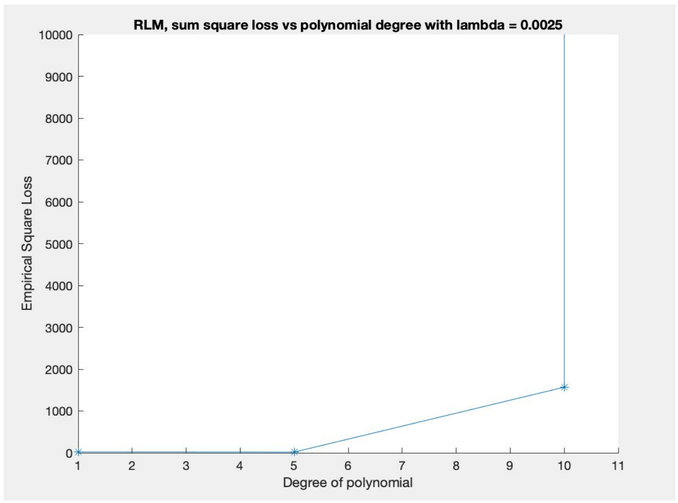
Step 5)



Comparison of W=1,5,10,20,30



Using W=[1,5,10,20,30] with a regularization parameter of λ = 0.0025 the error rate quickly gets astronomical.

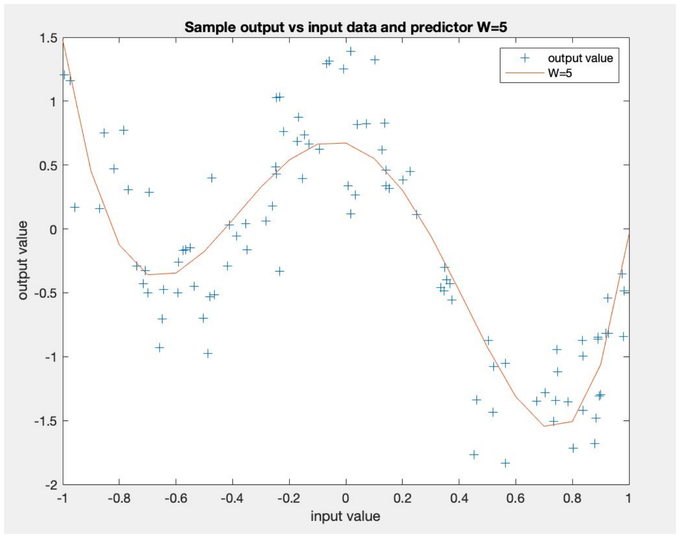


If we limit the max error shown on the y-axis to 10,000, the graph now shows the error climbing almost vertically at W=10. Once W>5 the error begins to climb rapidly. Closer analysis, using W=6 as a test value shows that going from W=5 to W=6 does introduce a significant increase (x3), just not as significant as this graph indicates due to the scale.

Taking everything shown within all of the graphs of prediction error and regularization, I would now update my recommendation for the degree of polynomial used from W=6 to W=5. The regularization parameter begins to negatively influence a choice of W=6 much more than the

Christopher Boyd
216 869 356
EECS 4404-Fall 2020

slight decrease of prediction error in selecting W=6 over W=5 shown in the 10-fold cross validation.



Final model predictor polynomial with degree 5 shown compared to the actual output values.

MATLAB Code

ERM

```
x = load('dataset1_inputs.txt');
t = load('dataset1_outputs.txt');
w = 30;
errors = zeros(1,w);
for i=1:w
    %polynomial curve fitting
    p = polyfit(x,t,i);
    results = polyval(p,x);
    errors(i) = sum(0.5 .* (results - t).^2);
end
figure;
plot([1:1:w],errors,'-*');
title('ERM, sum square loss vs polynomial degree');
xlabel('Degree of polynomial');
ylabel('Empirical Square Loss');

%step 5 visualization
%plot of learned polynomials
figure;
x2 = [-1:.1:1];
plot(x,t,'+');
hold on;
plot(x2,polyval(polyfit(x,t,1),x2));
plot(x2,polyval(polyfit(x,t,5),x2));
plot(x2,polyval(polyfit(x,t,10),x2));
plot(x2,polyval(polyfit(x,t,20),x2));
plot(x2,polyval(polyfit(x,t,30),x2));
ylim([-2 2]);
title('Sample output vs input data');
xlabel('input value');
ylabel('output value');
legend('output','W=1','W=5','W=10','W=20','W=30');
hold off;

%plot of errors for W=1,5,10,20,30
data = [errors(1),errors(5),errors(10),errors(20),errors(30)];
figure;
hold on;
plot([1,5,10,20,30],data,'-*');
title('ERM, sum square loss vs polynomial degree');
xlabel('Degree of polynomial');
ylabel('Empirical Square Loss');
hold off;
```

RLM

```
x = load('dataset1_inputs.txt');
t = load('dataset1_outputs.txt');

p = polyfit(x,t,20);
results = polyval(p,x);
errors = zeros(1,20);
n = norm(p)^2;
for i=1:20
    errors(i) = sum(0.5 .* (results - t).^2 + (exp(-i)/2 * n));
end

figure;
plot([1:1:20],errors,'-*');
title('RLM, regularized square loss vs ln(lambda)');
xlabel('ln(lambda) * -1');
ylabel('Regularized Square Loss');

%step 5 visualization
w = [1,5,6,10,20,30]
disp_errors = zeros(1,6);
for j=1:6
    p = polyfit(x,t,w(j));
    disp_data = polyval(p,x);
    disp_norm = norm(p)^2;
    disp_errors(j) = sum(0.5 .* (disp_data - t).^2 + (0.0025/2 * disp_norm));
end

%plot of errors for W=1,5,10,20,30
figure;
hold on;
plot(w,disp_errors,'-*');
ylim([0,100]);
title('RLM, sum square loss vs polynomial degree with lambda = 0.0025');
xlabel('Degree of polynomial');
ylabel('Empirical Square Loss');
```

Christopher Boyd
216 869 356
EECS 4404-Fall 2020

Cross-Validation

```
x = load('dataset1_inputs.txt');
t = load('dataset1_outputs.txt');

m = [x t];
errors = zeros(1,15);
table = array2table(m,'VariableNames',{'input','output'});
% Nonstratified partition
hpartition = cvpartition(100,'KFold',10);

%loop through each degree of polynomial for model
for w=3:15
    %loop through using each of 10 subsets as testing data, while
    %remaining 9 subsets are all used as training data
    for i=1:10
        %get a column vector, 1 indicates include row in training
        %idxTrain is the ith training set
        idxTrain = training(hpartition,i);
        %get rows identified for training from table
        tblTrain = table(idxTrain,:);
        %use remaining rows of table for test partition
        idxNew = test(hpartition,i);
        tblTest = table(idxNew,:);

        %fit polynomial
        p = polyfit(tblTrain.input,tblTrain.output,w);
        %get predicted output for training data
        pred = polyval(p, tblTest.input);
        errors(w) =  errors(w) + sum(0.5 .* (pred - tblTest.output).^2);
    end
    errors(w) = errors(w) / 10;
end
%plot average errors for each of w=[3,8] against w=[3,8]

figure;
plot([3:1:15],errors(3:15),'-*');
title('10-Fold Cross-Validation ERM Error vs Polynomial Degree');
xlabel('Degree of polynomial');
ylabel('Average Empirical Square Loss');
```