Homework 6: GRNN

Andy Reagan

I code and discuss a Generalized Regression Nueral Network (GRNN) as described in Specht in his 1991 paper.

I. INTRODUCTION

Brought forth by Specht, the GRNN is a powerful interpolating (function approximating) nueral network [2]. One key advantage of this approach over standard regression procedures is that the function does not have to be supplied a priori, and the network can build a wide class of functions. There is a one parameter to control the "smoothness" of the interpolation, labeled σ, and it is used in assigning the output of the pattern layer.

II. METHODS

The network is coded as we drew it in class, see Figure 1. Define the input patterns X as a matrix with the patterns as rows, and Y as the output patterns, with the outputs as rows. In testing, I use a 2-D input pattern and a 1-D output. As Specht notes in his paper, to generate additional output, another A and B unit are added for each output.

The weight matrix P is simply the input patterns, and the non-unity weights S are simply the output patterns. So, we have that $P = X$ and $S = Y$, the other weights all being 1. Tne only other piece is the activation function,

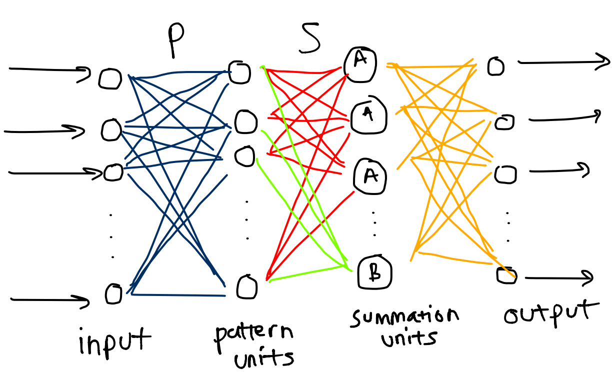

FIG. 1: The network as we drew it up. The weights between the input and pattern units are the matrix P, the weights between the pattern units and the summation units in red are the matrix S. The green weights between the pattern and summation units, and the orange weights between the summation units and output are all set to 1. I was also told that this looked like a bad football play diagram.

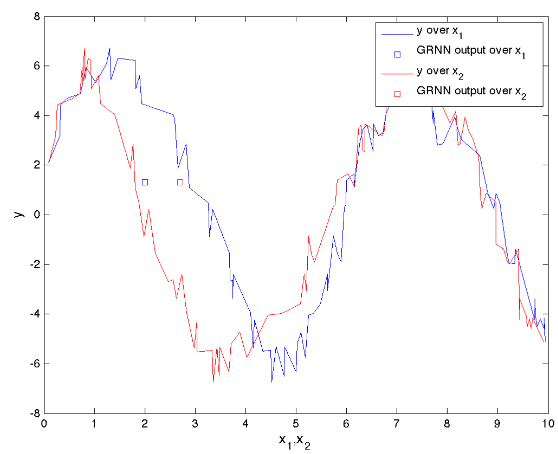

FIG. 2: Interpolation of a single data point from very noisy data, y, from two inputs x_1 and x_2.

which we write as

$$f_\sigma(x) = \exp\left(\frac{x}{2\sigma}\right). \tag{1}$$

I generate some testing data by randomly pulling vectors in 2D for the X input, and using a sin() function and noise for the output, making something that is not linear. The X patterns are both sampled from $[0, 10]$ randomly, and then sorted. The Y output has uniform noise from $[0, 1]$ added, and a in total is the function:

$$Y = 3 * \sin(X_1) + 3 * \sin(X_2 + 1) + \mu. \tag{2}$$

In addition, I test the given testing data which looks like a sigmoidal function, and show the output curves for varying smoothing σ in Figure 4.

III. RESULTS

A simple test of the network, and the coding of the network itself, demonstrate how easy it is to use. In Figure 3, we see that it does a reasonably good job at approximating our nonlinear function, considering the substantial amount of noise that was applied.

To test the network was a little bit tricky, I had to use the trained GRNN to test against a smoothed version of the function, interpolated at the values. In the end,

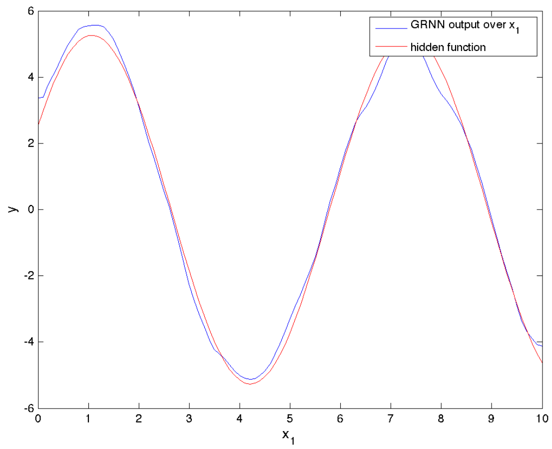

FIG. 3: Interpolation of y using linearly spaced points over x_1 by the GRNN. The true function, hidden by uniform random noise of magnitude 1, is also shown.

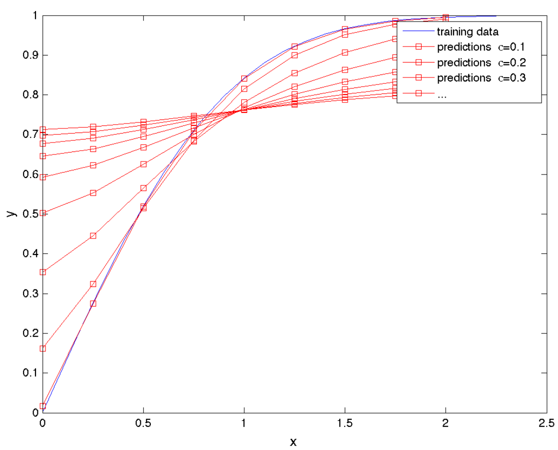

FIG. 4: Interpolation of sigmoid y using linearly spaced points over x_1 by the GRNN. Both the training data (blue, no squares) and the GRNN interpolation for various σ are plotted.

I wrote my own Y data, so I could generate a better, exact, interpolated dataset to test against. The error that I showed you after class was me plotting the data wrong, I was plotting the transpose, oops! Anyway, as σ decreases we find a better and better approximation in Figure 4.

When σ is sufficiently cranked down, the GRNN performs well. It seems to me that σ must be chosen to be a "good bit" smaller than the gaps between the training dataaset, to properly smooth. When it is too big, the GRNN finds the mean of the data. However, for very small σ, we do find an increase in the error as the GRNN overfits the data. An example of what the overfit looks like is shown in Figure 5.

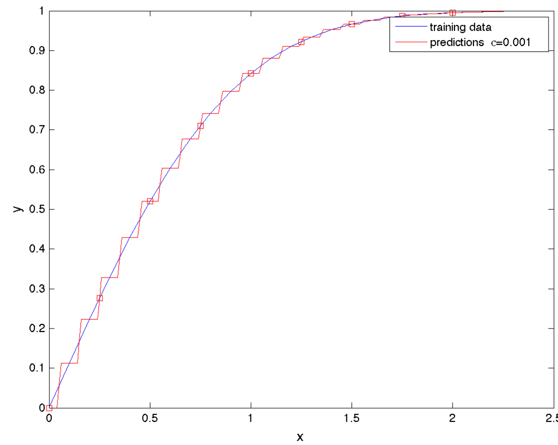

FIG. 5: Overfit of the GRNN.

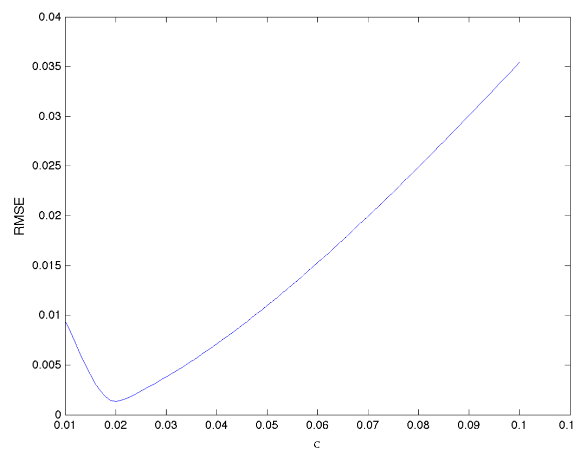

FIG. 6: RMSE versus the σ parameter.

Finally, I plotted the RMSE over different σ in Figure 6, and the GRNN does have an optimal value, as I hoped! Without knowing the true function, I do think that the "smaller than training gap" rule of thumb would do okay, and just looking at the output for flat spots.

I wonder if symbolic regression could find this function from the given data. Symbolic regression feels like a completely different solution to a very related problem, when the model is not known. Fun!

[1] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE 78*(9), 1464–1480.
[2] Specht, D. F. (1991). A general regression neural network. *Neural Networks, IEEE Transactions on 2*(6), 568–576.

Full code

```
clear all
close all

fprintf('time␣for␣GRNN\n');

% [X,Y] = makeTrainingData;

% disp(size(X));
% disp(size(Y));
% num_patterns = size(X,1);

% % figure(111);
% % plot(X(:,1),Y)
% % saveas(111,'figures/training_output.png')

% % no training, but let's set the weight matrices explicitly
% % first layer:
% P = X';
% % that was easy...
% % second layer (summation units):
% num_A_units = size(Y,2);
% num_B_units = size(Y,2);
% S = Y';

% % ...that's it!

% % now let's test it with an input
% x = [2,2.7];

% sigma = 0.5;

% % in a couple steps...
% % pattern_output = sigmf(sum(abs(P-repmat(x,num_patterns,1)'),1),[10 0.5]);
% pattern_output = exp(-sum(abs(P-repmat(x,num_patterns,1)'),1)./(2*sigma^2));
% summation_a_units = pattern_output*S';
% summation_b_units = pattern_output*ones(num_patterns,num_B_units);
% output = summation_a_units/summation_b_units;

% figure(112)
% plot(X(:,1),Y,'b')
% hold on;
% plot(x(1),output,'bs')
% plot(X(:,2),Y,'r')
% plot(x(2),output,'rs')
% xlabel('x_1,x_2','FontSize',12)
% ylabel('y','FontSize',12)
% legend('y over x_1','GRNN output over x_1','y over x_2','GRNN output over x_2','FontSize',12)
% saveas(112,'figures/GRNN.png')

% allx = 0:.1:10
% output = zeros(size(allx))
% for i=1:length(allx)
%     x = [allx(i),allx(i)];
%     pattern_output = exp(-sum(abs(P-repmat(x,num_patterns,1)'),1)./(2*sigma^2));
%     summation_a_units = pattern_output*S';
%     summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%     output(i) = summation_a_units/summation_b_units;
% end

% figure(113)
% plot(allx,output,'b')
% hold on;
% plot(allx,f(allx)+f(allx+1),'r')
% xlabel('x_1','FontSize',12)
% ylabel('y','FontSize',12)
% legend('GRNN output over x_1','hidden function')
% saveas(113,'figures/GRNN-allx.png')

raw = csvread('TrainData.csv');
X = raw(:,1);
% make an exact Y that I can test against!
Y = sigmf(X,[2,0])*2-1;
% Y = raw(:,2);
num_patterns = size(X,1);

% first layer:
P = X'./max(X); % and scale it to [0,1]
% second layer (summation units):
num_A_units = size(Y,2);
```

```
num_B_units = size(Y,2);
S = Y';

% sigma = 0.5;
% predict = csvread('PredictData.csv');

% output = zeros(size(predict))
% for i=1:length(predict)
%     x = predict(i)/max(X);
%     pattern_output = exp(-sum(abs(P-repmat(x,num_patterns,1)'),1)./(2*sigma^2));
%     summation_a_units = pattern_output*S';
%     summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%     output(i) = summation_a_units/summation_b_units;
% end

% figure(114)
% plot(X,Y,'b')
% hold on;
% plot(predict,output,'rs')
% xlabel('x','FontSize',12)
% ylabel('y','FontSize',12)
% legend('training data','predictions \sigma = 0.5')
% saveas(114,'figures/GRNN-givenData.png')

% sigmas = 0.1:0.1:0.9;
% % sigmas = 0.01:.01:0.1;
% predict = csvread('PredictData.csv');

% output = zeros(size(predict,1),length(sigmas));
% for j=1:length(sigmas)
%     sigma = sigmas(j);
%     for i=1:length(predict)
%         x = predict(i)/max(X);
%         pattern_output = exp(-sum(abs(P-repmat(x,num_patterns,1)'),1)./(2*sigma^2));
%         summation_a_units = pattern_output*S';
%         summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%         output(j,i) = summation_a_units/summation_b_units;
%     end
% end

% figure(115)
% plot(X,Y,'b')
% hold on;
% for j=1:length(sigmas)
%     plot(predict,output(j,:),'rs-')
% end
% % plot(predict,output,'s-')
% xlabel('x','FontSize',12)
% ylabel('y','FontSize',12)
% legend('training data','predictions \sigma=0.1',['predictions ' ...
%                     '\sigma=0.2'],'predictions \sigma=0.3','...')
% saveas(115,'figures/GRNN-givenData-allsigma.png')

% output = zeros(size(predict))
% for i=1:length(predict)
%     x = predict(i)/max(X);
%     pattern_output = exp(-sum((P-repmat(x,num_patterns,1)').^2,1)./(2*sigma^2));
%     summation_a_units = pattern_output*S';
%     summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%     output(i) = summation_a_units/summation_b_units;
% end

% figure(116)
% plot(X,Y,'b')
% hold on;
% plot(predict,output,'rs')
% xlabel('x','FontSize',12)
% ylabel('y','FontSize',12)
% legend('training data','predictions')
% saveas(116,'figures/GRNN-givenData-sqdiff.png')

% sigmas = 0.1:0.1:0.9;
% predict = csvread('PredictData.csv');

% output = zeros(size(predict,1),length(sigmas));
% for j=1:length(sigmas)
%     sigma = sigmas(j);
%     for i=1:length(predict)
%         x = predict(i)/max(X);
%         pattern_output = exp(-sum((P-repmat(x,num_patterns,1)').^2,1)./(2*sigma^2));
%         summation_a_units = pattern_output*S';
%         summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%         output(j,i) = summation_a_units/summation_b_units;
```

```
%     end
% end

% figure(117)
% plot(X,Y,'b')
% hold on;
% for j=1:length(sigmas)
%     plot(predict,output(j,:),'rs-')
% end
% xlabel('x','FontSize',12)
% ylabel('y','FontSize',12)
% legend('training data','predictions \sigma=0.1',['predictions ' ...
%                       '\sigma=0.2'],'predictions \sigma=0.3','...')
% saveas(117,'figures/GRNN-givenData-allsigma-sqdiff.png')

% exact = Y(1:5:21)';

% sigmas = 0.001:0.001:0.01;
% output = zeros(length(sigmas),size(predict,1));
% for j=1:length(sigmas)
%     sigma = sigmas(j);
%     for i=1:length(predict)
%         x = predict(i)/max(X);
%         pattern_output = exp(-sum((P-repmat(x,num_patterns,1)').^2,1)./(2*sigma^2));
%         summation_a_units = pattern_output*S';
%         summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%         output(j,i) = summation_a_units/summation_b_units;
%     end
% end

% figure(118)
% errors = zeros(size(sigmas));
% for j=1:length(sigmas)
%     errors(j) = rmse(output(j,1:2:9),exact);
% end
% plot(sigmas,errors)
% xlabel('\sigma','FontSize',12)
% ylabel('RMSE','FontSize',12)
% legend('training data','predictions \sigma=0.1',['predictions ' ...
%                       '\sigma=0.2'],'predictions \sigma=0.3','...')
% saveas(118,'figures/GRNN-givenData-allsigma-sqdiff-RMSE.png')

% prevoutput = output;

% sigma = 0.001;
% allx = 0:.01:2.5
% output = zeros(size(allx))
% for i=1:length(allx)
%     x = allx(i)/max(X);
%     pattern_output = exp(-sum((P-repmat(x,num_patterns,1)').^2,1)./(2*sigma^2));
%     summation_a_units = pattern_output*S';
%     summation_b_units = pattern_output*ones(num_patterns,num_B_units);
%     output(i) = summation_a_units/summation_b_units;
% end

% figure(119)
% plot(X,Y,'b')
% hold on;
% plot(allx,output,'r-')
% plot(predict,prevoutput(1,:),'rs')
% xlabel('x','FontSize',12)
% ylabel('y','FontSize',12)
% legend('training data','predictions \sigma=0.001')
% saveas(119,'figures/GRNN-givenData-sigma0.001-sqdiff.png')

predict = 0:0.01:2.5;
exact = sigmf(predict,[2,0])*2-1;
sigmas = 0.01:0.001:0.1;
output = zeros(length(sigmas),size(predict,1));
for j=1:length(sigmas)
    sigma = sigmas(j);
    for i=1:length(predict)
        x = predict(i)/max(X);
        pattern_output = exp(-sum((P-repmat(x,num_patterns,1)').^2,1)./(2*sigma^2));
        summation_a_units = pattern_output*S';
        summation_b_units = pattern_output*ones(num_patterns,num_B_units);
        output(j,i) = summation_a_units/summation_b_units;
    end
end

errors = zeros(size(sigmas));
for j=1:length(sigmas)
    errors(j) = rmse(output(j,:),exact);
```

```
end

figure(120)
plot(sigmas,errors)
xlabel('\sigma','FontSize',12)
ylabel('RMSE','FontSize',12)
saveas(120,'figures/GRNN-givenData-allsigma-sqdiff-RMSE-correct.png')
```

```
function [x,y] = makeTrainingData()
% sample points, 1D, randomly
% x = linspace(0,10,100);
x1 = rand(100,1)*10;
x1 = sort(x1);

x2 = rand(100,1)*10;
x2 = sort(x2);

x = [x1,x2];

% sample those points, with error
y_error = 1;
y = f(x1)+f(x2+1)+rand(size(x1))*2*y_error-y_error;

end
```

```
function [y] = f(x)
% y = x^3/1000 + 3*sin(x);
y = 3*sin(x);
end
```