2 General Regression Neural Network (GRNN)

GRNN, as proposed by Donald F. Specht in [Specht 91] falls into the category of probabilistic neural networks as discussed in Chapter one. This neural network like other probabilistic neural networks needs only a fraction of the training samples a backpropagation neural network would need [Specht 91]. The data available from measurements of an operating system is generally never enough for a backpropagation neural network [Specht 90]. Therefore the use of a probabilistic neural network is especially advantageous due to its ability to converge to the underlying function of the data with only few training samples available. The additional knowledge needed to get the fit in a satisfying way is relatively small and can be done without additional input by the user. This makes GRNN a very useful tool to perform predictions and comparisons of system performance in practice.

2.1 Algorithm

The probability density function used in GRNN is the Normal Distribution. Each training sample, X_j, is used as the mean of a Normal Distribution.

$$Y(X) = \frac{\sum_{i=1}^{n} Y_i \exp\left(-D_i^2 / 2\sigma^2\right)}{\sum_{i=1}^{n} \exp\left(-D_i^2 / 2\sigma^2\right)}$$

$$D_i^2 = (X - X_i)^T \times (X - X_i)$$

Eqn. 2.1-1

The distance, D_j, between the training sample and the point of prediction, is used as a measure of how well the each training sample can represent the position of prediction, X. If the Distance, D_j, between the training sample and the point of prediction is small, $exp(-D_j^2/2\sigma^2)$, becomes big. For $D_j=0$, $exp(-D_j^2/2\sigma^2)$ becomes one and the point of evaluation is represented best by this training sample. The distance to all the other training samples is bigger. A bigger distance, D_j , causes the term $exp(-D_j^2/2\sigma^2)$ to become smaller and therefore the contribution of the other training samples to the prediction is relatively small. The term $Y_j*exp(-D_j^2/2\sigma^2)$ for the jth trainng sample is the biggest one and contributes very much to the prediction. The standard deviation or the smoothness parameter, σ, as it is named in [Specht 91], is subject to a search. For a bigger smoothness parameter, the possible representation of the point of evaluation by the training sample is possible for a wideer range of X. For a small

value of the smoothness parameter the representation is limited to a narrow range of *X*, respectively.

With (Eqn. 2.1-1) it is possible to

- predict behavior of systems based on few training samples
- predict smooth multi-dimensional curves
- interpolate between training samples.

In (Fig. 2.1-1) a prediction performed by GRNN is shown. The circles represent the data points or training samples used to predict the solid line going through most of these samples. The bell shaped curves are the individual terms of (Eqn. 2.1-1). Each of these curves is one term, $Y_j * exp(-D_j^2/2\sigma^2)/\sum_{i=1}^{n} exp(-D_i^2/2\sigma^2)$ of the whole equation (Eqn. 2.1-1) used in GRNN for the prediction. These terms are normalized normal distributions. Summing up the values of the individual terms at each position yields the value of the prediction, the solid line going through most of the data points. The smoothness parameter was arbitrarily chosen to σ=0.1.

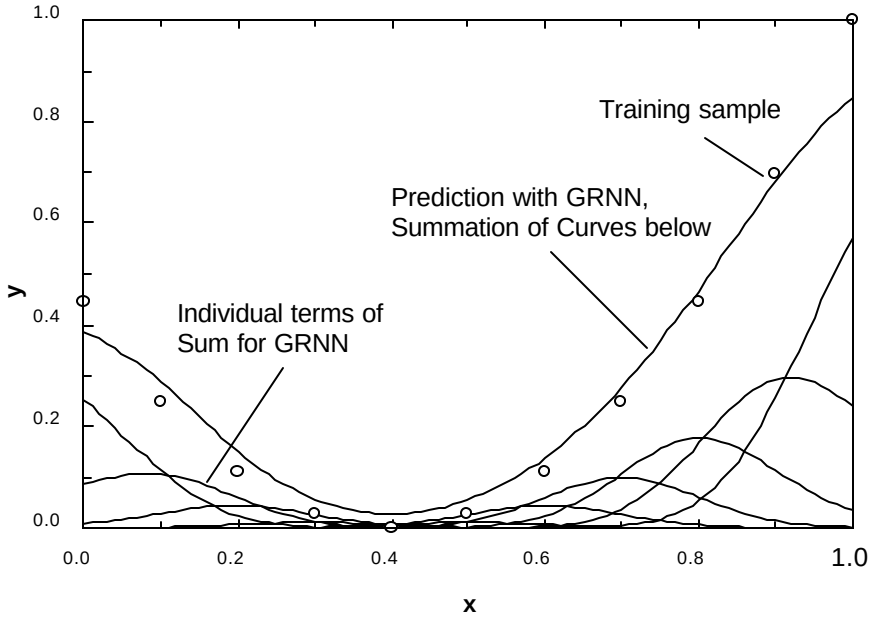

Fig. 2.1-1 GRNN with individual terms contributing to prediction, s=0.1

In Chapter one it was discussed how Neural Networks weight the individual signals of each neuron differently. In (Fig. 2.1-2) GRNN is shown in a familiar representation, a Backpropagation Neural Network was shown before (Chapter 1). The calculations performed in each pattern neuron of GRNN are $exp(-D_j^2/2\sigma^2)$, the normal distribution centered at each training sample. The signals of the pattern neuron i, going into the Denominator neuron are weighted with the corresponding values of the training samples, Y_i. The weights on the signals going into the Numerator Neuron are one. Each sample from the training data influences every point that is being predicted by GRNN.

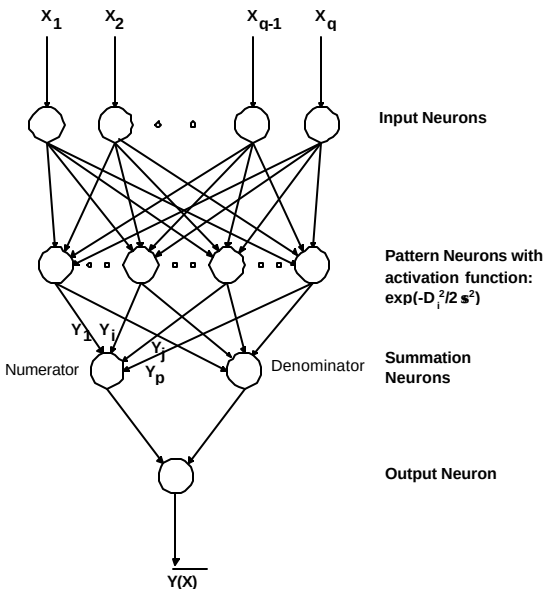

Fig. 2.1-2 GRNN built up in a way such that it can be used as a parallel Neural Network

2.2 How to choose σ

Any prediction tool can potentially be used for feed forward control. Controllers using a derivative algorithm need values in their algorithm that also include a derivative. This means that the prediction tools have to submit a prediction that does not only represent the reality in the precision of the value of the prediction but as well in the slope of the prediction. The slope of the prediction can even be more important than the actual value of the prediction, depending on the algorithm a controller uses. Depending on the use of the prediction tool the emphasis has to be put on one of the two, precision or smoothness, or even both. Fault detection and diagnosis can include several methods, they can include as well algorithms using a slope in their approach. But again, depending on the use of the prediction tool, many aspects are important.

The smoothness parameter is the only parameter of the procedure. The search for the smoothness parameter has to take several aspects into account depending on the application the predicted output is used for.

A bad characteristic that GRNN shows is that it can have wiggles (Fig. 2.2-1). A wiggle is defined as an inflection point at a position where no inflection should be. Wiggles can be as severe that such sudden changes in the values of the prediction happen that these changes almost appear to be steps. In (Fig. 2.2-1) again the individual terms of the prediction using GRNN are shown. The individual terms have a very narrow range of influence compared

to (Fig. 2.1-1). The individual terms almost solely influence the prediction in the close vicinity of the training sample.

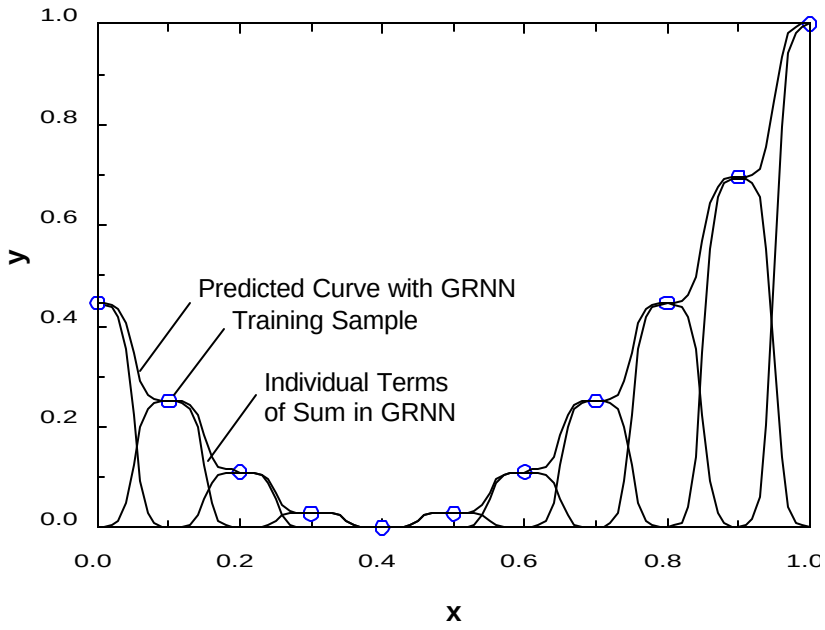

Fig. 2.2-1 GRNN Prediction with extreme wiggles, including the individual signals, $s=0.027$

The appearance of wiggles is solely connected to the value of the smoothness parameter σ. For a prediction that is close to one of the training samples and a sufficiently small smoothness parameter the influence of the neighboring training samples is minor. The contribution to the prediction, $exp(-D_k^2/2\sigma^2)$, is a lot smaller than the contribution of the training sample that the prediction is close to, $exp(-D_j^2/2\sigma^2)$. The influence of the training samples that are further away from the point of prediction can be neglected. Due to the normalization the prediction therefore yields the value of the training sample in the vicinity of

the each training sample (Eqn. 2.2-1). For a bigger smoothnes parameter the influence of the neighboring training samples cannot be neglected. The prediction then is influenced by more points and the prediction is getting smoother.

$$\lim_{\sigma \to 0} \left(\frac{\sum_{i=1}^{n} Y_i \exp\left(-D_i^2 / 2\sigma^2 \right)}{\sum_{i=1}^{n} \exp\left(-D_i^2 / 2\sigma^2 \right)} \right)$$

X close to X_j

$$= \frac{Y_j \lim_{\sigma \to 0} \exp\left(-D_j^2 / 2\sigma^2 \right) + \sum_{\substack{i=1 \\ i \neq j}}^{n} Y_i \lim_{\sigma \to 0} \exp\left(-D_i^2 / 2\sigma^2 \right)}{\lim_{\sigma \to 0} \exp\left(-D_j^2 / 2\sigma^2 \right) + \sum_{\substack{i=1 \\ i \neq j}}^{n} \lim_{\sigma \to 0} \exp\left(-D_i^2 / 2\sigma^2 \right)}$$

$$= \frac{Y_j \lim_{\sigma \to 0} \exp\left(-D_j^2 / 2\sigma^2 \right)}{\lim_{\sigma \to 0} \exp\left(-D_j^2 / 2\sigma^2 \right)} = Y_j$$

Eqn. 2.2-1

With even larger σ the predicted curve will get flatter more smooth as well. In some cases this is desirable. For example when the available data include a lot of noise, then the prediction has to interpolate the data whereas if the data are correct, GRNN has to fit the data more precisely and has to follow each little trend the data makes. If σ approaches infinity the predicted value is simply the average of all the sample points (Eqn. 2.2-2). The effect of a big

smoothness parameter can be seen very early.starts very early. In (Fig. 2.2-2) an example for a smoothenss parameter of one is shown.

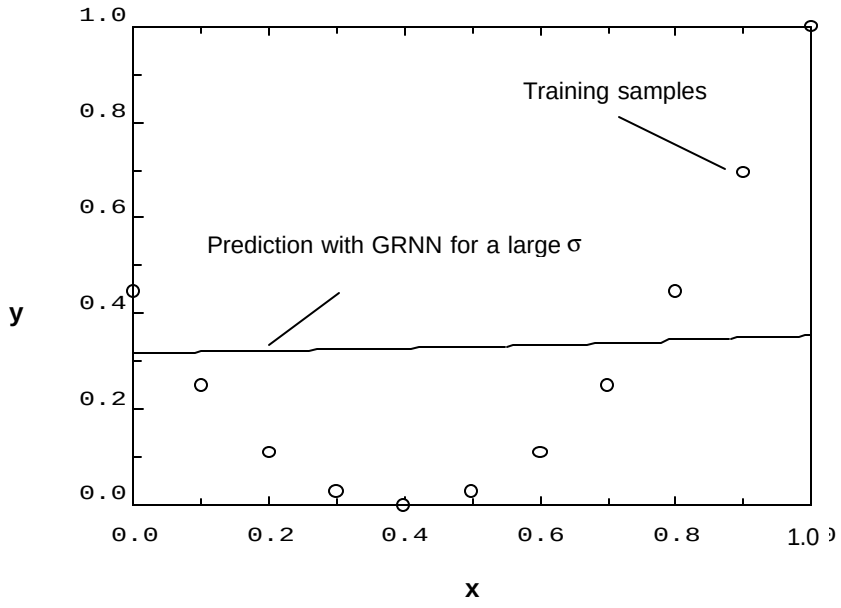

Fig. 2.2-2 Prediction with GRNN for a large smoothness parameter, s =1.0

$$\lim_{s\to\infty}\left(\frac{\sum_{i=1}^{n} Y_i \exp\left(-D_i^2/2s^2\right)}{\sum_{i=1}^{n} \exp\left(-D_i^2/2s^2\right)}\right) = \frac{\sum_{i=1}^{n} Y_i \lim_{s\to\infty} \exp\left(-D_i^2/2s^2\right)}{\sum_{i=1}^{n} \lim_{s\to\infty} \exp\left(-D_i^2/2s^2\right)}$$

$$= \frac{\sum_{i=1}^{n} Y_i}{n}$$

Eqn. 2.2-2

Due to the fact that data are not generally without measurement errors and that the circumstances change from application to application, there cannot be a right or a wrong way

to chose σ. The application and the required features of the prediction highly determine which σ should be finally chosen. A tradeoff between smoothness and minimal error has to be made depending on the data and the later use of the prediction.

2.2.1 The Holdout Method

Specht suggests in [Specht 91] the use of the holdout method to select a good value of σ. In the holdout method, one sample of the entire set is removed and for a fixed σ GRNN is used again to predict this sample with the reduced set of training samples. The squared difference between the predicted value of the removed training sample and the training sample itself is then calculated and stored. The removing of samples and prediction of them again for this chosen σ is repeated for each sample-vector. After finishing this process the mean of the squared differences is calculated for each run. Then the process of reducing the set of training samples and predicting the value for these samples is repeated for many different values of σ. The σ for which the sum of the mean squared difference is the minimum of all the mean squared differences is the σ that should be used for the predictions using this set of training samples. According to Specht there are no restrictions to this process, but unfortunately it turned out that for certain conditions this process does not show the desired results.

The holdout method works with smoothness parameters that are very small. The evaluation of the exponential function therefore often causes numerical problems; even for 64 bit data storage. Given that there are no numerical problems, another problem is that the minimum of the Sum of Squares, Specht describes already as wide is simply so wide (Fig.

2.2-3) that the final choosing of σ is very imprecise. σ chosen in the range of the minimum can show one of the above mentioned problems of either extreme: Wiggles or an unacceptable interpolation, but usually wiggles were observed. In (Fig. 2.2-3) an example of the result of the holdout method is shown. The Sum of Sqaures is plotted versus the smoothness parameter. The holdout method suggests to use a smoothness parameter of σ=0.01. With the naked eye the minimum, the holdout method suggests expands up to a value of the smoothness parameter of σ=0.04.

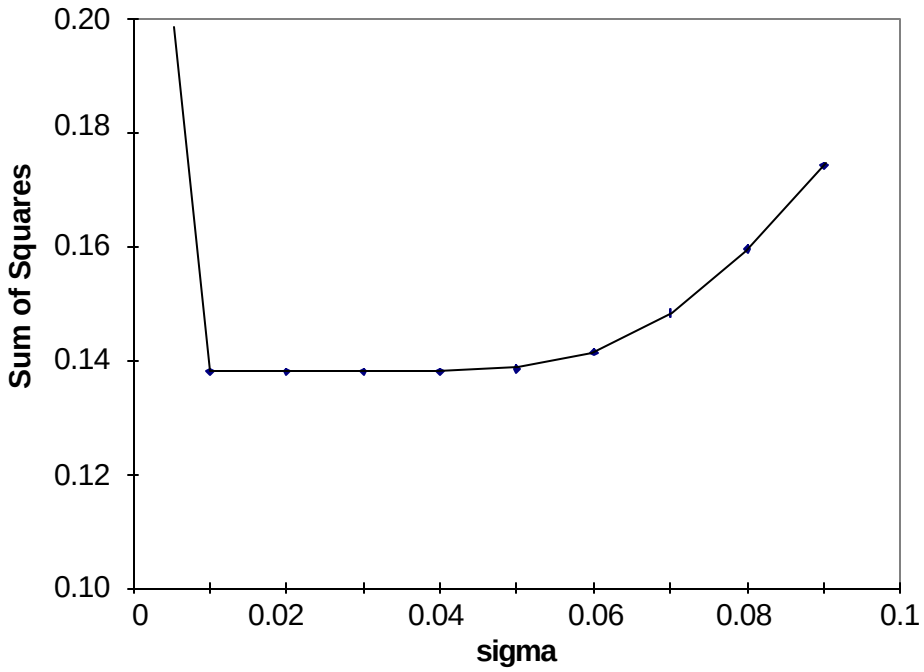

Fig. 2.2-3 Result from Holdout Method

In (Fig. 2.2-4) the results for these previously chosen smoothenss parameters are plotted. Both of the plots yield a curve that includes wiggles. The wiggles are step like for a

smoothenss parameter of σ=0.01, the wiggles are more gentle for a smoothness parameter of σ=0.04. The precision of the prediction is very good at the training points but the changes in the slope of this curve are unacceptable for several applications. As a comparison a prediction with a smoothness parameter of σ=0.07 is included too. This prediction does not include wiggles but is therefore less precise at the training samples. A decision has to be made what curve is better. This decision depends on the application the prediction is finally used for.

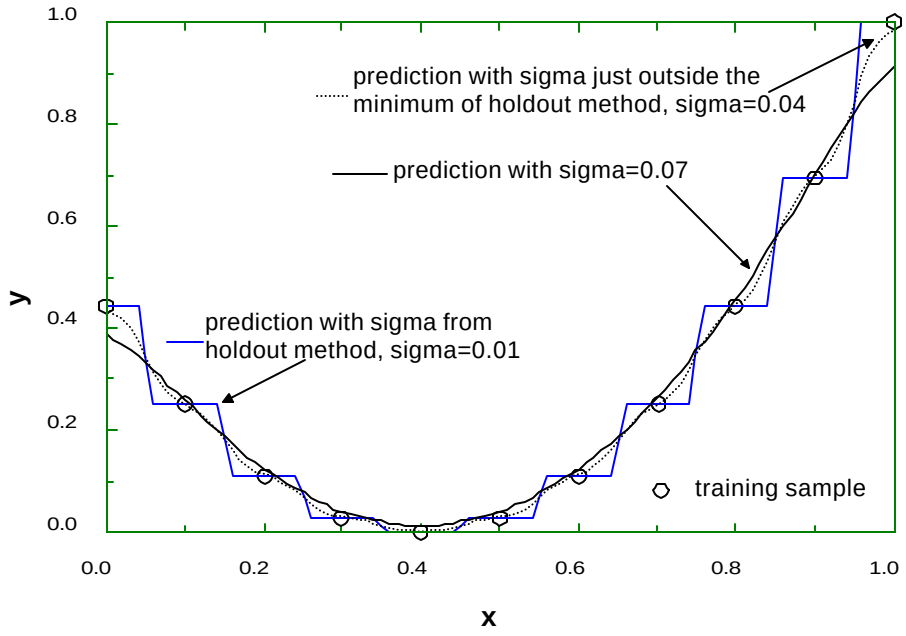

Fig. 2.2-4 Using s as suggested by the Holdout Method and other s to compare

2.2.2 The Wiggle-Method

All possible procedures for the choosing of σ are empirical procedures and cannot yield an exact or correct value for σ. The holdout method can work very well for some examples but the results of the holdout method cannot be influenced such that they

accommodate for the different possible desired characteristics of the prediction. Some effort was undertaken to find a different, more flexible way of choosing σ. The result is the ***Wiggle-Method***. The wiggle-method is a purely empirical method that turns out to be very easy to work with. The results using the wiggle-method can be influenced very easily. It leaves room to adjust the algorithm to pay closer attention to smoothness or to precision in the prediction.

The wiggle-method (Fig. 2.2-5) works for any number of dimensions. Using a two dimensional example the wiggle-method will be explained. The wiggle-method needs to have the allowable number of inflection points specified. This information is usually known or can easily be assessed. The judgment that is needed though is to allow the right number of additional inflections to count for unequally spaced data or measurement errors, problems that will be discussed later. For the wiggle-method, GRNN predicts the curve over the entire range of the data. First σ is chosen very small, too small to predict the curve without wiggles. The value for σ is increased by a specifiable amount constantly until the number of inflection points is equal to the allowable number of inflection points. Increasing σ will sooner or later yield a curve that is too smooth, that has not enough inflections, then σ has to be decreased again. This iteration will go on until a stop criteria is satisfied. The stop criteria can be a maximum number of iterations around the maximal allowable number of wiggles, or it can be a maximum tolerable change in σ or a combination of these two.

The number of wiggles is calculated with a numerical approximation. Many predictions for equally spaced points are calculated. The spacing of the predicted points needs to be

sufficiently more dense than the spacing of the training samples, otherwise the method would not notice the inflection points. Using about five times or more points of predictions than training samples is suggested. The second derivative using three consecutive points is then calculated numerically (Eqn. 2.2-3) over the entire range. Inflection points show a sign change in the second derivative. The number of sign changes over the range of the calculated predictions is equal to the number of inflection points.

$$\mathbf{sec} = \frac{\left(y_{i+1} - 2y_i + y_{i+1}\right)}{2\Delta x^2}$$

Eqn. 2.2-3

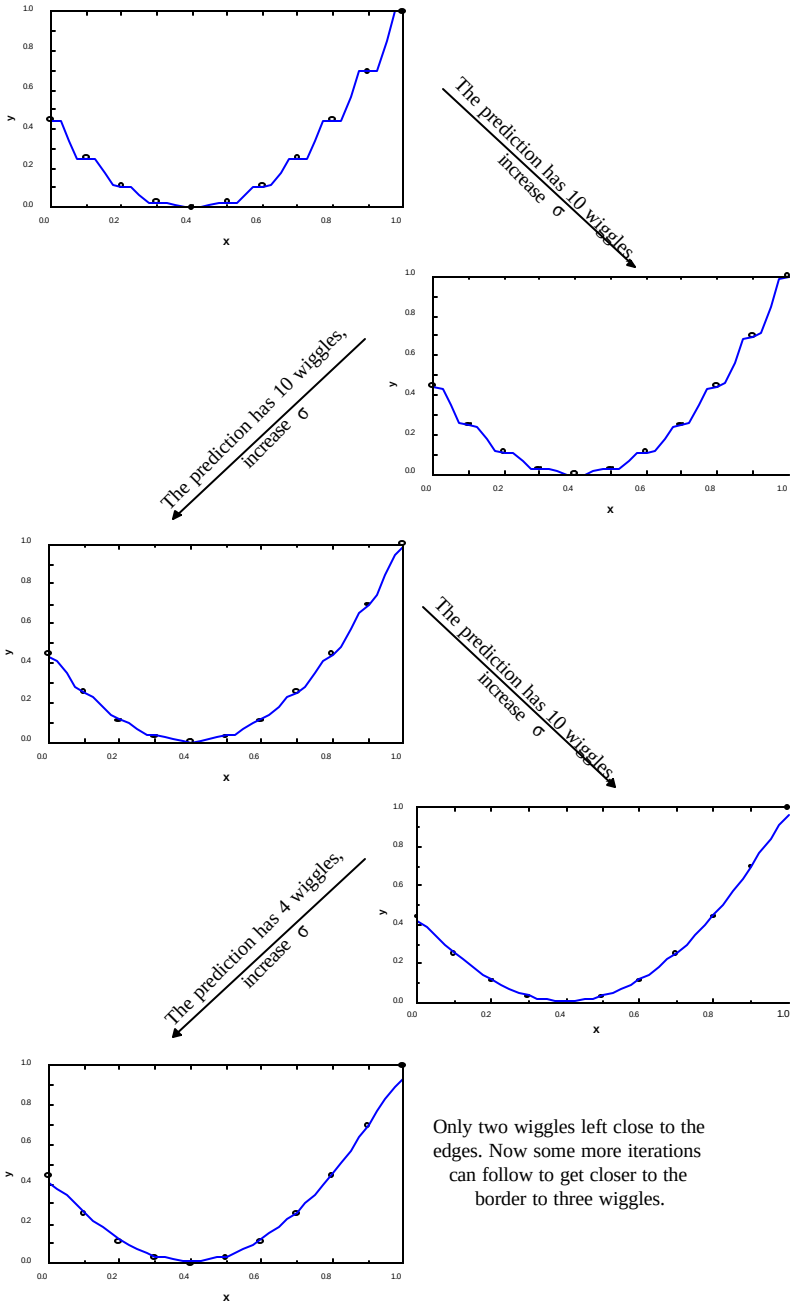

Fig. 2.2-5 Wiggle-method

The precision of last curve in (Fig. 2.2-5) is not as good as it could possibly be but it is very a very smooth curve. This curve has the desired characteristics for many applications, it is smooth and it is precise.

The number of inflections remain constant over a range of σ. But the precision at the training samples changes continuously with σ. The precision declines for bigger σ. σ should be chosen at the lower bound of the range σ for the desired number since the quality of the prediction of the training samples increases as σ gets smaller (Fig. 2.2-6).

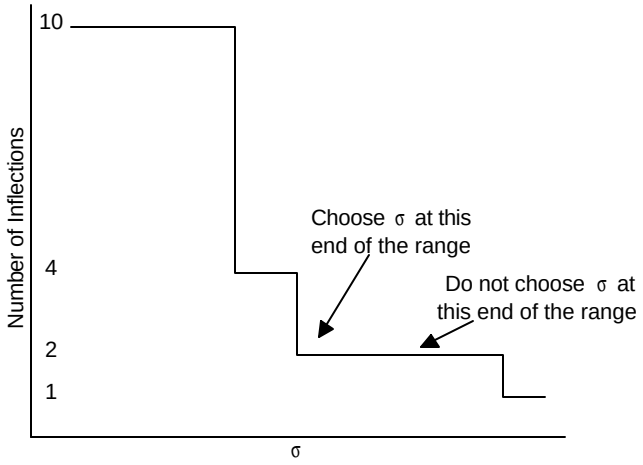

Fig. 2.2-6 Selection of s

The quality of the prediction is not going to be bad, supposing that the number of inflections given is close to the true number of inflections that the system really has. It was proven by Parzen in [Parzen] that the probabilistic approach will converge to the underlying function as more training samples are availbale. Counting the inflection points suggested by the user is a measure of how close the prediction is to the underlying function. This is simply a

measure of quality as the sum of squares is as well a measure of quality. The wiggle-method also allows handling of measurement errors and handling of the influence of unequally spaced data. These problems will be discussed later. Using the wiggle-method the curve will not be influenced by the scatter of the data such that extreme deviations occur as with Spline functions. The curve will have the desired shape and the proof of convergence by Parzen make sure that the curve will interpolate the data. A procedure can be included that shows the development of the mean squared error over the entire process. Adjustments to the search method can then be made. These adjustments are nothing else than increasing or decreasing the tolerable number of wiggles for the prediction in order to get a more precise or smooth curve. Later (Section 3.6) a way to increase the precision without loosing the smoothness of the curve will be discussed.

The wiggle-method will converge to a value of the smoothness parameter σ. For a very small σ the prediction includes too many wiggles. For a σ that is aiming towards infinity, the value of the prediction will be the average of the values of the training samples without any wiggle. According to [Specht 91] there is a value of σ for which the prediction converges to the underlying function of the samples. The wiggle-method will find a value that lies between these two boundaries, many wiggles or average value and no wiggles.

As mentioned before a trade-off has to be made between precision and smoothness. In (Fig. 2.2-7) the same example that was previously used is shown again. Previously the prediction did not yield very precise values. The prediction is very smooth but the prediction

does not represent the training samples very precisely. In (Fig. 2.2-8) the prediction represents the training samples very well but the prediction has many wiggles. Depending on the use of the results one or the other prediction can be better. In (Fig. 2.2-7) and (Fig. 2.2-8) the prediction itself and the slope of the prediction is shown for the examples used in (Fig. 2.2-7) and (Fig. 2.2-8). The slope of the prediction for using a σ=0.1 is very smooth too. A controller using the two signals, value and slope of the prediction will perform very different for the prediction using a σ=0.1 than the same controller using the prediction and the slope for a σ=0.027.

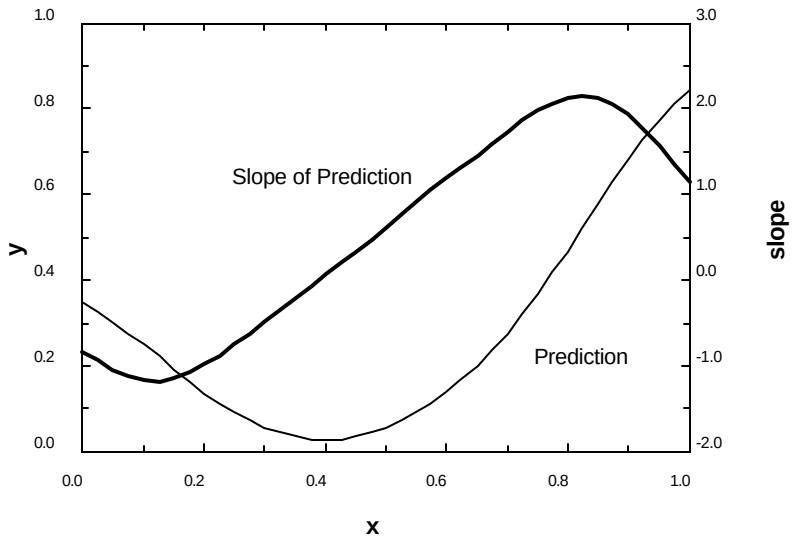

Fig. 2.2-7 Prediction and Slope of Prediction for a very smooth curve, s =0.1

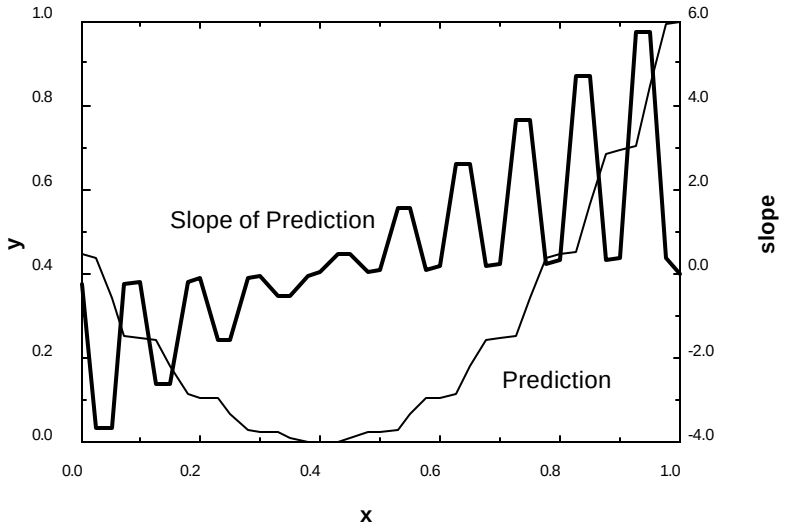

Fig. 2.2-8 Prediction and Slope of Prediction for a curve with many wiggles, s =0.027

It cannot be assumed that the wiggle-method is the best way of choosing σ, but the results turned out to be very satisfying. The holdout method could work without additional information provided by the user; the wiggle-method needs additional information. This makes the method vulnerable to errors but as well more flexible.