The dual representation and its application to seismic reservoir analysis

Brian Russell[1]

[1] GeoSoftware, Calgary, Alberta, brian.russell@geosoftware.com

ABSTRACT

Geophysical and machine learning algorithms are based on finding the best weights that predict a set of observations using a set of measured attributes. A good example of this is the prediction of seismic reservoir parameters, such as density or porosity, from a set of seismic attributes. Many problems can be written as a linear regression model in which the weights can be determined using a generalized inverse. Although this involves a linearly weighted combination of the input attributes, we can apply a nonlinear function to the attributes themselves, which allows us to get a much better fit on the output. In this study I will consider two different forms of the generalized inverse, the primal and dual, and show the dual form leads to several very powerful analysis techniques, called kernel regression methods, that go well beyond the multi-linear regression techniques used in many applications. I will illustrate these techniques using two simple datasets, one with only three points and the other with ten points, and then apply them to seismic reservoir analysis.

INTRODUCTION

In geophysical inversion and machine learning, we often try to find a weighted sum of a set of D attributes that predict some type of observation such as a well log curve. Mathematically, the attributes can be represented as N D-dimensional vectors. The two fundamental problems in machine learning are regression and classification. In regression, we try to predict the observations in a point-by-point way using the D attributes (note that there is usually a bias weight, meaning there are $D+1$ weights). In classification, we divide the N points into K classes, and try to find linear or nonlinear boundaries between the classes. In this study, I will focus only on the regression problem.

Regression methods can be broken into two categories: linear and nonlinear. There is only one linear approach, and that is where we are going to start today. However, there are a variety of nonlinear regression methods, such as polynomial regression and the radial basis function approach, also called kernel regression. Linear regression is also the simplest polynomial regression approach and uses a first order polynomial. In this study I will discuss three types of polynomial regression methods: linear, quadratic, and cubic, applied to a simple three-point problem and a more complex ten-point noisy sine wave. I will start with the primal solution and then discuss the dual solution, which will lead directly to kernel regression. I will finish by applying all these methods to a seismic reservoir prediction problem.

PRIMAL AND DUAL REGRESSION SOLUTIONS

In both geophysics and machine learning, the simplest linear data model is written:

$$\boldsymbol{y} = X\boldsymbol{w}, \tag{1}$$

where $\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$, $X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1D} \\ 1 & x_{21} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{bmatrix}$, and $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}$.

In the model given in equation 1, $\boldsymbol{y}$ is a vector of N observations, X is matrix consisting of D columns of attributes, or features, each with N samples, plus a column of N ones, and $\boldsymbol{w}$ is a vector consisting of D+1 weights, where w_0 is called the bias. Examples of this model include AVO analysis, predicting a reservoir parameter from multiple attributes, seismic tomography, and so on. In most cases, N is much greater than D, so we use the least-squares solution given by:

$$\boldsymbol{w} = \left(X^T X + \lambda I_{M+1} \right)^{-1} X^T \boldsymbol{y} = X^\dagger \boldsymbol{y}, \tag{2}$$

where X^TX is the inner product, or autocorrelation, of X, an D+1 x D+1 square matrix, $X^T\boldsymbol{y}$ is the cross-correlation of X and $\boldsymbol{y}$, an D+1 length vector, and λI_{D+1}, where λ is a pre-whitening term and I_{M+1} is the D+1 x D+1 identity matrix, is called pre-whitening in geophysics and ridge regression in statistics. Equation 2 is called the primal solution. As also shown in equation 2, another way to think of computing $\boldsymbol{w}$ is with the D+1 x N size matrix $X^\dagger$, called the generalized inverse. The generalized inverse interpretation shows us that each of the D+1 weights is created by a weighted sum of the N observations.

The pre-whitening, or ridge regression, form of the inverse also allows us to derive the dual form of the solution (see Appendix) which is

$$\boldsymbol{w} = X^T \left(XX^T + \lambda I_N \right)^{-1} \boldsymbol{y} = X^\dagger \boldsymbol{y}. \tag{3}$$

In equation 3, XX^T is the outer product of X, an N x N square matrix, and λI_N is a pre-whitening term that now uses the N x N identity matrix. With the proper amount of pre-whitening the dual generalized inverse, $X^\dagger$, is identical to the primal generalized matrix.

Although equation 3 gives us a second way of computing the weights, it usually doesn't help us because we must invert an N x N matrix rather than a smaller D+1 x D+1 matrix. But by using what is called the kernel trick, or kernel substitution, this outer product matrix leads to several nonlinear regression methods. In the next section the concept of polynomial regression will be introduced to explain these concepts, as well as a simple numerical example.

POLYNOMIAL REGRESSION

Suppose we have measured N points for a single attribute, or feature, $\boldsymbol{x}$ (D = 1) and N observations in $\boldsymbol{y}$, where:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \text{ and } \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{4}$$

In polynomial regression, we want to find a set of weights which will transform $\boldsymbol{x}$ into an estimate of $\boldsymbol{y}$ (which I will write as $\boldsymbol{y}$ with a hat), or:

$$\hat{\boldsymbol{y}} = w_0 + w_1\boldsymbol{x} + w_2\boldsymbol{x}^2 + \ldots + w_p\boldsymbol{x}^p \,. \tag{5}$$

Equation 5 is called a p^{th} order polynomial fit to the points.

A simple example, and one which I will use to illustrate all our methods, uses N = 3 and contains the numbers 1, 2, and 3 in a slightly different order in the two vectors:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{ and } \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}.$$

The cross-plot of these two vectors, shown in Figure 1, raises several questions:

1. Is there a straight line (i.e.: p = 1) that will give an exact fit to these points?
2. Is there a polynomial fit (i.e.: p > 1) that will give an exact fit to these points?
3. Will we get a better fit by increasing the order of the polynomial to larger values?

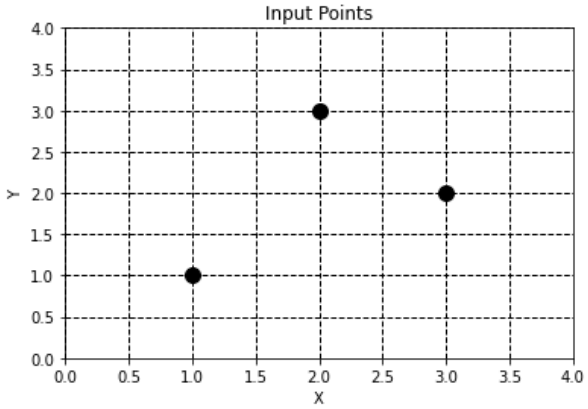


Figure 1: A cross-plot of the x and y vectors given above.

To answer these questions, let's expand the previous equation for the three points in our example:

$$\begin{aligned} y_1 &= w_0 + w_1 x_1 + w_2 x_1^2 + \ldots + w_P x_1^P \\ y_2 &= w_0 + w_1 x_2 + w_2 x_2^2 + \ldots + w_P x_2^P \\ y_3 &= w_0 + w_1 x_3 + w_2 x_3^2 + \ldots + w_P x_3^P \end{aligned} \tag{6}$$

We now have $N = 3$ equations with p+1 terms, where there are three different possible situations:

1. If $p + 1 < N$, this is called the over-determined case.
2. If $p + 1 = N$, this is called the even-determined case.
3. If $p + 1 > N$, this is called the under-determined case.

I will illustrate these three cases using the three-point example, which leads to linear, quadratic, and cubic curve fitting. First, if we let $p = 1$ this leads to the following set of three linear equations with two unknowns, which is an over-determined case:

$$\begin{aligned} y_1 &= w_0 + w_1 x_1 \\ y_2 &= w_0 + w_1 x_2 \\ y_3 &= w_0 + w_1 x_3 \end{aligned} \tag{7}$$

These equations can be put into the same form as equation 1 using $X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$ and $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$. Note that X is a non-square 3x2 matrix so must be inverted using either the primal or dual solution given in equation 2 and 3. The first form of the primal solution can be written as follows with zero pre-whitening:

$$\boldsymbol{w} = \left(X^T X \right)^{-1} X^T \boldsymbol{y} = A^{-1} \boldsymbol{c} \,, \tag{8}$$

where $A = X^T X$ is the called autocorrelation matrix and $\boldsymbol{c} = X^T \boldsymbol{y}$ is called the cross-correlation vector. The solution for our simple numerical example is:

$$\begin{aligned} \boldsymbol{w} = A^{-1} \boldsymbol{c} &= \left\{ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \right\}^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \\ &= \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 13 \end{bmatrix} = \begin{bmatrix} 7/3 & -1 \\ -1 & 1/2 \end{bmatrix} \begin{bmatrix} 6 \\ 13 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix} \end{aligned} \,.$$

Figure 2 shows the linear solution for this problem, where the blue regression line is:

$$\hat{\boldsymbol{y}} = w_0 + w_1 \boldsymbol{x} = 1 + \frac{\boldsymbol{x}}{2}. \tag{9}$$

For each point x on the x-axis in Figure 2, the predicted y value can be found by:

$$\hat{y}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = w_0 + w_1 x = 1 + \frac{x}{2}. \tag{10}$$

For example, the point $\hat{y}(2.5) = \begin{bmatrix} 1 & 2.5 \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = 2.25$ is shown as the open circle in figure 2.

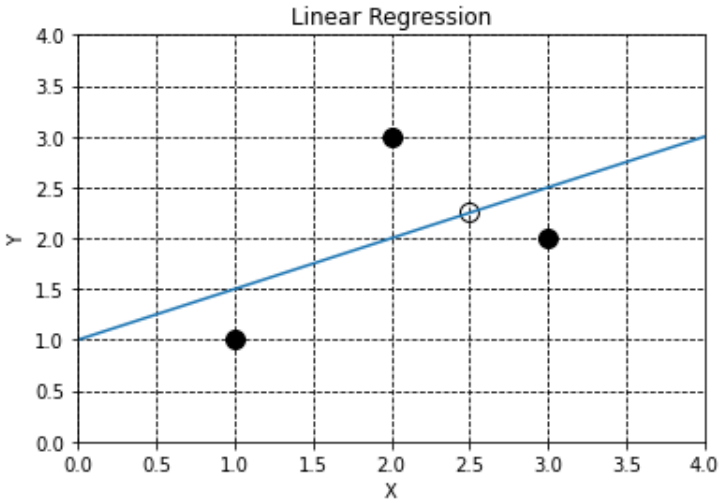


Figure 2: The linear regression fit to the three points in Figure 1, with a predicted point at x = 2.5.

As shown in equation 2, an alternate way to arrange the primal solution is:

$$\boldsymbol{w} = \left[\left(X^T X \right)^{-1} X^T \right] \boldsymbol{y} = X^{\dagger} \boldsymbol{y} \tag{11}$$

where $X^{\dagger}$ is called the generalized inverse. The generalized inverse gives the same solution as we got in the previous approach but notice that the solution to the weights is now a weighted sum of the inputs given by

$$\boldsymbol{w} = \left\{ \begin{bmatrix} \frac{7}{3} & -1 \\ -1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \right\} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{4}{3} & \frac{1}{3} & -\frac{2}{3} \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{4}{3} + 1 - \frac{4}{3} \\ -\frac{1}{2} + 0 + 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}.$$

Let's now go back to the linear least-squares problem and see its dual form. For the linear problem, the N x N matrix XX^T looks like this:

$$XX^T = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 5 & 7 \\ 4 & 7 & 10 \end{bmatrix}.$$

If we take the determinant of XX^T, we get:

$$\det\left(XX^T\right)=\begin{vmatrix}2 & 3 & 4\\ 3 & 5 & 7\\ 4 & 7 & 10\end{vmatrix}=2(50-49)-3(30-28)+4(21-20)=2-6+4=0\,.$$

Remember that a determinant of 0 means that XX^T is not invertible. To solve this problem, all we need to do is add some pre-whitening, in this case 10^{-6}:

$$\boldsymbol{w}=\begin{bmatrix}w_0\\ w_1\end{bmatrix}=\left[X^T\left(XX^T\right)^{-1}+10^{-6}I_3\right]\boldsymbol{y}\,. \tag{12}$$

Now the dual solution now gives us the same solution as the primal solution, or:

$$\boldsymbol{w}=\left\{\begin{bmatrix}1 & 1\\ 1 & 2\\ 1 & 3\end{bmatrix}\frac{10^6}{3}\begin{bmatrix}0.5000061 & -0.9999987 & 0.4999966\\ -0.9999987 & 2.0000003 & -1.0000007\\ 0.4999966 & -1.0000007 & 0.5000021\end{bmatrix}\right\}\begin{bmatrix}1\\ 3\\ 2\end{bmatrix}$$

$$=\begin{bmatrix}\frac{4}{3} & \frac{1}{3} & -\frac{2}{3}\\ -\frac{1}{2} & 0 & \frac{1}{2}\end{bmatrix}\begin{bmatrix}1\\ 3\\ 2\end{bmatrix}=\begin{bmatrix}1\\ \frac{1}{2}\end{bmatrix}$$

(By the way, don't expect to get the same answer as above if you enter those numbers into a computer program, since they are truncated from full precision values). It is obvious that inverting the 2 x 2 matrix in the primal form is more efficient than inverting the 3 x 3 matrix in the dual form. But if we write the outer product matrix in its general form, we find that:

$$XX^T=\begin{bmatrix}1 & x_1\\ 1 & x_2\\ 1 & x_3\end{bmatrix}\begin{bmatrix}1 & 1 & 1\\ x_1 & x_2 & x_3\end{bmatrix}=\begin{bmatrix}1+x_1^2 & 1+x_1x_2 & 1+x_1x_3\\ 1+x_2x_1 & 1+x_2^2 & 1+x_2x_3\\ 1+x_3x_1 & 1+x_3x_2 & 1+x_3^2\end{bmatrix}.$$

Notice that this means that the matrix multiplication can be written in an alternate way, where each term is computed analytically by the equation

$$K_{ij}=1+x_ix_j\,. \tag{13}$$

This is called the kernel matrix and is a fundamental idea in this talk. But before looking at the general case of the kernel matrix, let's move to quadratic polynomial regression. Letting $p = 2$ leads to the following set of three equations with three unknowns, which is the even-determined case:

$$\begin{aligned}y_1&=w_0+w_1x_1+w_2x_1^2\\ y_2&=w_0+w_1x_2+w_2x_2^2\\ y_3&=w_0+w_1x_3+w_2x_3^2\end{aligned} \tag{14}$$

These three quadratic equations can be put into the same form as equation 1 using $X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$ and $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$. Since X is now square, it can be inverted directly, giving $\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} -4 \\ 6.5 \\ -1.5 \end{bmatrix}$. The resulting fit is shown in Figure 3, where the blue line fit is given as follows:

$$\hat{\boldsymbol{y}} = -4 + 6.5\boldsymbol{x} - 1.5\boldsymbol{x}^2, \tag{15}$$

Note that the point $\hat{y}(2.5) = \begin{bmatrix} 1 & 2.5 & 6.25 \end{bmatrix} \begin{bmatrix} -4 \\ 6.5 \\ -1.5 \end{bmatrix} = 2.875$ is shown as the open circle.

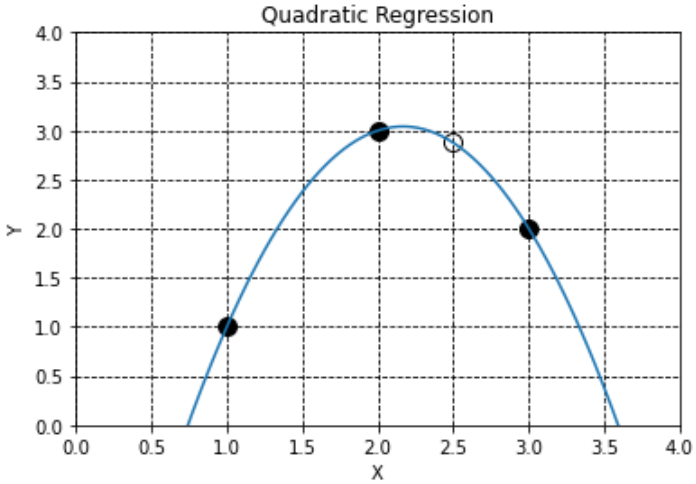


Figure 3: The best quadratic regression fit to the three points in Figure 1.

Notice in Figure 3 how rapidly the quadratic fit becomes negative to the left and right of the points, which might not be the best fit if we observe more points. Also, note that both the primal and dual solutions will give the same answer without needing any pre-whitening:

$$\boldsymbol{w} = \left(X^T X\right)^{-1} X^T \boldsymbol{y} = X^T \left(XX^T\right)^{-1} \boldsymbol{y} = X^{-1} \boldsymbol{y}. \tag{16}$$

If we apply the less efficient, but equally valid, dual solution to the quadratic fit and write the outer product matrix in its general form, we get:

$$XX^T = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} = \begin{bmatrix} 1 + x_1^2 + x_1^4 & 1 + x_1 x_2 + x_1^2 x_2^2 & 1 + x_1 x_3 + x_1^2 x_3^2 \\ 1 + x_2 x_1 + x_2^2 x_1^2 & 1 + x_2^2 + x_2^4 & 1 + x_2 x_3 + x_2^2 x_3^2 \\ 1 + x_3 x_1 + x_3^2 x_1^2 & 1 + x_3 x_2 + x_3^2 x_2^2 & 1 + x_3^2 + x_3^4 \end{bmatrix} \tag{17}$$

This suggests that the kernel matrix be written as follows, which is very close to the correct answer except for two extra $x_i x_j$ terms:

$$K_{ij} = \left(1 + x_i x_j\right)^2 \approx XX^T, \tag{18}$$

since $\left(1 + x_i x_j\right)^2 = 1 + 2x_i x_j + x_i^2 x_j^2$.

Before leaving this simple problem, let's look at the cubic polynomial, since it will teach us the important concept of over-fitting. If we let $p = 3$, we get the following set of three cubic polynomial equations with four unknowns, or the cubic polynomial:

$$\begin{aligned} y_1 &= w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 \\ y_2 &= w_0 + w_1 x_2 + w_2 x_2^2 + w_3 x_2^3 \\ y_3 &= w_0 + w_1 x_3 + w_2 x_3^2 + w_3 x_3^3 \end{aligned} . \tag{19}$$

This is now an under-determined problem, since the number of unknowns exceeds the number of input points. These three equations can be put into the same form as equation 1 using $X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix}$, and $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$. The cubic fit can then be computed using the primal generalized inverse as follows:

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_3 \\ w_4 \end{bmatrix} = \left[\left(X^T X\right)^{-1} X^T\right] \boldsymbol{y}, \tag{20}$$

However, there is a big problem lurking in equation 20. The problem is again that the determinant of $X^T X$ is equal to zero:

$$\det\left(X^T X\right) = \det \begin{bmatrix} 3 & 6 & 14 & 44 \\ 6 & 14 & 36 & 114 \\ 14 & 36 & 98 & 308 \\ 44 & 114 & 308 & 986 \end{bmatrix} = 0 .$$

This means that the covariance matrix cannot be inverted. If you code up equation 20 on a computer, some packages will tell you there is problem, but others will do the calculation and come up with spurious results. To solve this problem, we add a small amount of pre-whitening to the main diagonal of the matrix before inversion, giving:

$$\boldsymbol{w} = \left[\left(X^T X + \lambda I_{p+1}\right)^{-1} X^T\right] \boldsymbol{y}, \tag{21}$$

Using $\lambda = 10^{-6}$, we get the following result:

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_3 \\ w_4 \end{bmatrix} = \left[\left(X^T X + 10^{-6} I_4 \right)^{-1} X^T \right] \boldsymbol{y} = \begin{bmatrix} -0.768 \\ 0.575 \\ 1.732 \\ -0.539 \end{bmatrix}.$$

The resulting fit is shown in Figure 4 and is as follows:

$$\hat{\boldsymbol{y}} = -0.768 + 0.575\boldsymbol{x} + 1.732\boldsymbol{x}^2 - 0.539\boldsymbol{x}^3. \tag{22}$$

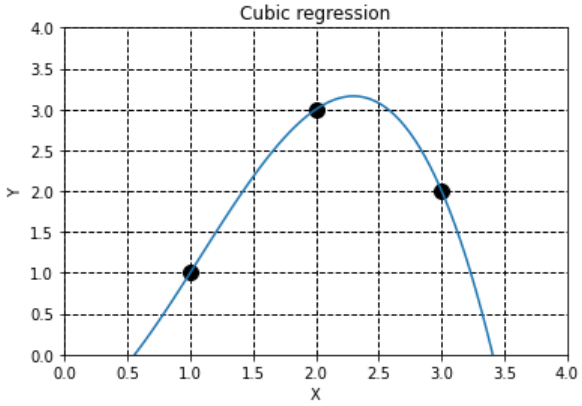


Figure 4: The cubic regression fit to the three points of Figure 1 using a pre-whitening term.

Notice that the curve in Figure 4 fits the points well and looks reasonable. But is it? Let's expand the previous plot by extending both the *x* and *y* axes to -2, as shown in Figure 5. Notice that the cubic fit takes a sharp turn upwards just as the *x*-axis goes negative. The reason for this is that a cubic polynomial will always intersect the *x*-axis at three points, unlike a quadratic that intersects at two points and a linear fit that intersects at one point. This is called over-fitting, and it means that we have used too many parameters to do the fitting, in this case four parameters for three points.

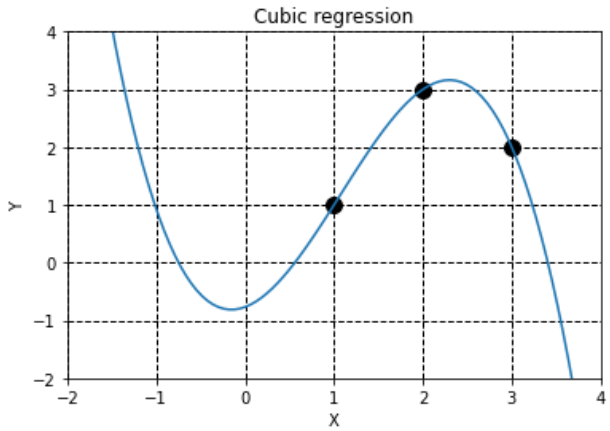


Figure 5: An expanded view of figure 4, where we now see that we have over-fitted the three points.

If we use a small value of λ, there is never any harm in always applying ridge regression, even when the matrix being inverted is well conditioned. For example, if we go back to the linear regression equation with two parameters and apply ridge regression with the same pre-whitening value of 10^{-1}, we get an equally good fit:

$$\boldsymbol{w} = \left(X^T X + \lambda I_2 \right)^{-1} X^T \boldsymbol{y} = \begin{bmatrix} 3.000001 & 6 \\ 6 & 14.000001 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 13 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}. \tag{23}$$

However, there is an even more important reason to apply a pre-whitening term. As mentioned in the first section, adding a pre-whitening term allows us to derive what is called the dual solution to the regression problem (see the Appendix).

Note, however, that the cubic solution the dual form gives the correct answer without pre-whitening

$$\boldsymbol{w} = X^T \left(X^T X \right)^{-1} \boldsymbol{y} = \begin{bmatrix} -0.768 \\ 0.575 \\ 1.732 \\ -0.539 \end{bmatrix}. \tag{24}$$

Now let's write the outer product matrix for the cubic regression problem in its general form (note that it is invertible since it is a 3 x 3 matrix):

$$XX^T = \begin{bmatrix} 1 + x_1^2 + x_1^4 + x_1^6 & 1 + x_1 x_2 + x_1^2 x_2^2 + x_1^3 x_2^3 & 1 + x_1 x_3 + x_1^2 x_3^2 + x_1^3 x_3^3 \\ 1 + x_2 x_1 + x_2^2 x_1^2 + x_2^3 x_1^3 & 1 + x_2^2 + x_2^4 + x_2^6 & 1 + x_2 x_3 + x_2^2 x_3^2 + x_2^3 x_3^3 \\ 1 + x_3 x_1 + x_3^2 x_1^2 + x_3^3 x_1^3 & 1 + x_3 x_2 + x_3^2 x_2^2 + x_3^3 x_2^3 & 1 + x_3^2 + x_3^4 + x_3^6 \end{bmatrix}. \tag{25}$$

This suggests that the kernel matrix be written as follows, which has two extra $x_i x_j$ and $x_i^2 x_j^2$ terms:

$$K_{ij} \left(1 + x_i x_j \right)^3 \approx XX^T, \tag{26}$$

since $\left(1 + x_i x_j \right)^3 = 1 + 3x_i x_j + 3x_i^2 x_j^2 + x_i^3 x_j^3$. I think we can see a pattern here as we move to higher polynomials, so let's next look at the general theory of linear basis function models and kernels.

LINEAR BASIS FUNCTION MODELS

The linear basis function model (Bishop, 2006) is an extension of a linear model where x is replaced with the nonlinear basis function $f(x)$, or:

$$y_i = w_0 \phi_0(x_i) + w_1 \phi_1(x_i) + \ldots + w_M \phi_M(x_i) = \sum_{j=0}^{M} w_j \phi_j(x_i), i = 1, \ldots, N, \tag{27}$$

Note that polynomial regression is a basis function that can be written:

$$\phi_j(x_i) = x_i^j \,, \tag{28}$$

A second basis function is the Gaussian:

$$\phi_j(x_i) = \exp\left[-\frac{(x_i - \mu_j)^2}{2\sigma^2}\right], \tag{29}$$

where μ = mean, and σ^2 = variance .

A third basis function is the hyperbolic tangent:

$$\phi_j(x_i) = \tanh[b + cx_i]\,. \tag{30}$$

To solve for the weights, we set up the problem as follows, where Φ is an N x M+1 dimensional matrix containing the basis functions:

$$\boldsymbol{y} = \Phi \boldsymbol{w}\,, \tag{31}$$

$$\text{where } \Phi = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_M(x_N) \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \text{ and } \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}.$$

Notice that Φ can be thought of in two ways, as a combination of N M+1- dimensional row vectors, or as a combination of M+1 N-dimensional column vectors, or

$$\Phi = \begin{bmatrix} \boldsymbol{\phi}_{row1} \\ \boldsymbol{\phi}_{row2} \\ \vdots \\ \boldsymbol{\phi}_{rowN} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_{col0} & \boldsymbol{\phi}_{col1} & \dots & \boldsymbol{\phi}_{colM} \end{bmatrix}, \tag{32}$$

$$\text{where } \boldsymbol{\phi}_{rowi}^T = \begin{bmatrix} \phi_0(x_i) \\ \phi_1(x_i) \\ \vdots \\ \phi_M(x_i) \end{bmatrix}, \text{ and } \boldsymbol{\phi}_{colj} = \begin{bmatrix} \phi_j(x_1) \\ \phi_j(x_2) \\ \vdots \\ \phi_j(x_N) \end{bmatrix}.$$

To solve for the weights, the ridge-regression ***primal*** solution is:

$$\boldsymbol{w} = \left(\Phi^T\Phi + \lambda I_{M+1}\right)^{-1}\Phi^T\boldsymbol{y}\,, \tag{33}$$

and the ridge-regression ***dual*** solution is:

$$\boldsymbol{w} = \Phi^T\left(\Phi\Phi^T + \lambda I_N\right)^{-1}\boldsymbol{y}\,. \tag{34}$$

Going back to the row and column notation of equation 32, notice that the elements of the M+1 x M+1-dimensional matrix product $\mathbf{\Phi}^T\mathbf{\Phi}$ can be written:

$$\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_{col0}^T\boldsymbol{\phi}_{col0} & \cdots & \boldsymbol{\phi}_{col0}^T\boldsymbol{\phi}_{colM} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_{colM}^T\boldsymbol{\phi}_{col1} & \cdots & \boldsymbol{\phi}_{colM}^T\boldsymbol{\phi}_{colM} \end{bmatrix}, \tag{35}$$

and the elements of the N x N-dimensional matrix product $\mathbf{\Phi}\mathbf{\Phi}^T$ can be written:

$$\mathbf{\Phi}\mathbf{\Phi}^T = \begin{bmatrix} \boldsymbol{\phi}_{row1}\boldsymbol{\phi}_{row1}^T & \cdots & \boldsymbol{\phi}_{row1}\boldsymbol{\phi}_{rowN}^T \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_{rowN}\boldsymbol{\phi}_{row1}^T & \cdots & \boldsymbol{\phi}_{rowN}\boldsymbol{\phi}_{rowN}^T \end{bmatrix}. \tag{36}$$

Since all this theory has been quite abstract, let me illustrate it with the linear regression of the simple three-point problem discussed in the last section, which is given by equation 28 written as $\phi_j(x_i) = x_i^j$, where $j = 0,1$ and $i = 1,2,3$. This leads to:

$$\mathbf{\Phi} = \begin{bmatrix} x_1^0 & x_1^1 \\ x_2^0 & x_2^1 \\ x_3^0 & x_3^1 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}.$$

In the column interpretation, this gives:

$$\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_{col0}^T\boldsymbol{\phi}_{col0} & \boldsymbol{\phi}_{col0}^T\boldsymbol{\phi}_{col1} \\ \boldsymbol{\phi}_{col1}^T\boldsymbol{\phi}_{col0} & \boldsymbol{\phi}_{col1}^T\boldsymbol{\phi}_{col1} \end{bmatrix} = \begin{bmatrix} 3 & \sum_{i=1}^{3} x_i \\ \sum_{i=1}^{3} x_i & \sum_{i=1}^{3} x_i^2 \end{bmatrix}, \tag{37}$$

where $\boldsymbol{\phi}_{col0} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, and $\boldsymbol{\phi}_{col1} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$. In the row interpretation, this gives:

$$\mathbf{\Phi}\mathbf{\Phi}^T = \begin{bmatrix} \boldsymbol{\phi}_{row1}\boldsymbol{\phi}_{row1}^T & \boldsymbol{\phi}_{row1}\boldsymbol{\phi}_{row2}^T & \boldsymbol{\phi}_{row1}\boldsymbol{\phi}_{row3}^T \\ \boldsymbol{\phi}_{row2}\boldsymbol{\phi}_{row1}^T & \boldsymbol{\phi}_{row2}\boldsymbol{\phi}_{row2}^T & \boldsymbol{\phi}_{row2}\boldsymbol{\phi}_{row3}^T \\ \boldsymbol{\phi}_{row3}\boldsymbol{\phi}_{row1}^T & \boldsymbol{\phi}_{row3}\boldsymbol{\phi}_{row2}^T & \boldsymbol{\phi}_{row3}\boldsymbol{\phi}_{row3}^T \end{bmatrix} = \begin{bmatrix} 1+x_1^2 & 1+x_1x_2 & 1+x_1x_3 \\ 1+x_2x_1 & 1+x_2^2 & 1+x_2x_3 \\ 1+x_3x_1 & 1+x_3x_2 & 1+x_3^2 \end{bmatrix}, \tag{38}$$

where $\boldsymbol{\phi}_{row1}^T = \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$, $\boldsymbol{\phi}_{row2}^T = \begin{bmatrix} 1 \\ x_2 \end{bmatrix}$, and $\boldsymbol{\phi}_{row3}^T = \begin{bmatrix} 1 \\ x_3 \end{bmatrix}$. This new notation allows us to re-write equation 10 for the computation of a point on the regression line in a new way:

$$\hat{y}(x) = \boldsymbol{\phi}_{row}^T(x)\boldsymbol{w}, \tag{39}$$

where $\boldsymbol{\phi}_{row}^{T}(x) = \begin{bmatrix} 1 \\ x \end{bmatrix}$. As before, for x = 2.5 this gives $\hat{y}(2.5) = \begin{bmatrix} 1 & 2.5 \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = 2.25$, which is the identical expression. But, as I will show in the next section, this will lead us to a completely different interpretation of regression, called kernel substitution.

Note that our linear regression example is simply a re-expression of what we saw in the previous section, but with a more general interpretation given by the basis function approach. But let's now expand it by using the Gaussian basis function of equation 29. Using the values $\mu_1 = 1$, $\mu_2 = 2$, and $\sigma = 0.1$ gives the following matrix solution for the 3-point problem (since $x_1 = 1$ and $x_2 = 2$):

$$\Phi = \begin{bmatrix} \exp\left[-\frac{(x_1-\mu_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_1-\mu_2)^2}{2\sigma^2}\right] \\ \exp\left[-\frac{(x_2-\mu_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_2-\mu_2)^2}{2\sigma^2}\right] \\ \exp\left[-\frac{(x_3-\mu_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_3-\mu_2)^2}{2\sigma^2}\right] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{40}$$

This then leads to the following weights using the primal solution:

$$\boldsymbol{w} = \left(\Phi^T\Phi\right)^{-1}\Phi^T\boldsymbol{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}. \tag{41}$$

The full solution is shown in Figure 6 and is a double Gaussian that fits the first two points but ignores the third point

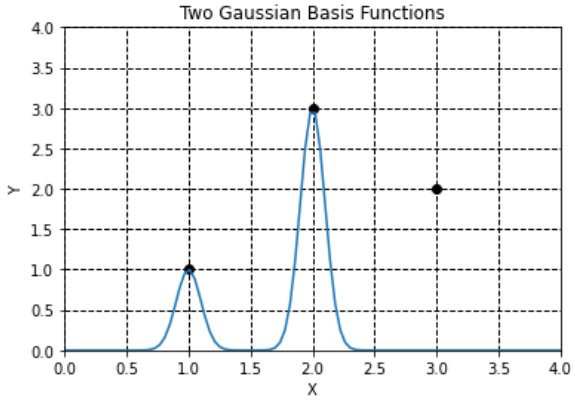


Figure 6: The Gaussian basis function fit using means $\mu_1 = 1$, $\mu_2 = 2$ and $\sigma = 0.1$.

Because we chose mean values equal to the first two x values note that:

$$y(1) = w_0\phi_1(1) + w_1\phi_2(1) = \exp[0] + 3\exp[-50] = 1 + 0 = 1, \text{ and}$$

$$y(2) = w_0\phi_1(2) + w_1\phi_2(2) = \exp[-50] + 3\exp[0] = 0 + 3 = 3.$$

Now let's add a third mean, giving the following square matrix:

$$\Phi = \begin{bmatrix} \exp\left[-\frac{(x_1-\mu_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_1-\mu_2)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_1-\mu_3)^2}{2\sigma^2}\right] \\ \exp\left[-\frac{(x_2-\mu_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_2-\mu_2)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_2-\mu_3)^2}{2\sigma^2}\right] \\ \exp\left[-\frac{(x_3-\mu_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_3-\mu_2)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_3-\mu_3)^2}{2\sigma^2}\right] \end{bmatrix}$$

Using values of $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$, and $\sigma = 0.1$ gives the same solution for the dual, primal, and full inverse, since $M = N$:

$$\boldsymbol{w} = \Phi^{-1}\boldsymbol{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}. \tag{42}$$

The full solution is shown in Figure 7 and is a triple Gaussian that fits all three points.

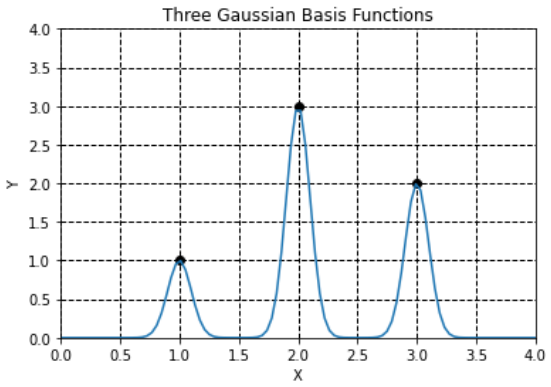
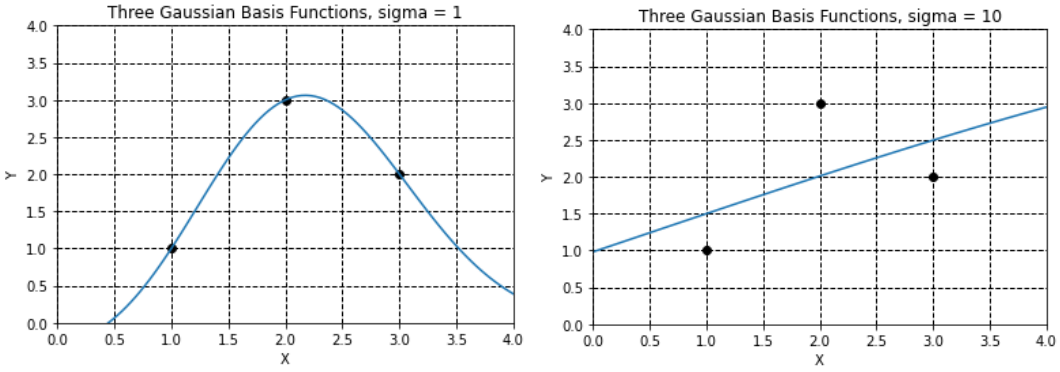


Figure 7: The Gaussian basis function fit using means $\mu_1 = 1$, $\mu_2 = 2$ and $\mu_3 = 3$, and $\sigma = 0.1$

To see that this fits at the two x values of 1, 2, and 3, note that:

$$y(1) = \exp[0] + 3\exp[-50] + 2\exp[-200] = 1,$$
$$y(2) = \exp[-50] + 3\exp[0] + 2\exp[-50] = 3, \text{ and}$$
$$y(3) = \exp[-200] + 3\exp[-50] + 2\exp[0] = 2.$$

To simplify the mathematics, I used $\sigma = 0.1$ in our previous examples, but the fit was too "spiky". Figure 8(a) shows the fit with $\sigma = 1.0$, which is much smoother, but fits the points. Figure 8(b) shows the fit with $\sigma = 10$, which is almost the same as the linear fit and does not fit the points.



(a) (b)

Figure 8: The three mean Gaussian basis function fit using (a) $\sigma = 1$, and (b) $\sigma = 10$.

Before leaving this section, it is important to note that all the examples used in this section assumed that we only had one seismic attribute, or $D = 1$ in the notation of the first section. We can easily extend this theory to multiple attributes by assuming that each of the N samples x_i is a D dimensional vector given by $\boldsymbol{x}_i^T = \begin{bmatrix} x_{i1} & \dots & x_{iD} \end{bmatrix}$. Note that this vector is a single row vector from the matrix X defined in equation 1.

Having looked at linear basis function models in this section, let us now look at the related subject of kernel functions.

KERNEL FUNCTIONS

In the previous section we found that the ridge-regression ***dual*** solution for the linear basis function model was given by:

$$\boldsymbol{w} = \Phi^T \left(\Phi\Phi^T + \lambda I_N \right)^{-1} \boldsymbol{y} . \tag{43}$$

$$\text{where } \Phi = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_M(x_N) \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \text{ and } \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}.$$

An alternate way to write equation 43 is as follows:

$$\boldsymbol{w} = \Phi^T \left(K + \lambda I_N \right)^{-1} \boldsymbol{y} , \tag{44}$$

where $K = \Phi\Phi^T$ is called the Gram matrix. In some cases, the Gram matrix can be created by the matrix multiplication $\Phi\Phi^T$. However, it can also be created using what is called a kernel function. This is sometimes called kernel substitution, or the kernel "trick", and allows us to build N x N matrix representations of the kernel matrix without having to use a matrix multiplication. Note the strong resemblance of the kernel function to the basis functions discussed in the last section except for one important fact: the kernel function is symmetric. That is, is involves the inversion of a full N x N matrix, where N is equal to the number of input points or observations.

Thus, we can think of basis functions as a subset of kernel functions, where we can use fewer basis functions than the number of points (that is $M < N$). The simplest example of a kernel function is the polynomial kernel, which is written:

$$K_{ij} = k(x_i, x_j) = (1 + x_i x_j)^p , \tag{45}$$

where p is the order of the polynomial. We have already seen this kernel in equations 13 and 38 for the linear polynomial ($p = 1$) where we found that this function gives the same solution as the matrix multiplication. However, in equations 18 and 26 we found that this function does not give the same solution as matrix multiplication for the quadratic ($p = 2$) and cubic ($p = 3$) cases.

Yet another way to write equation 44 is:

$$\boldsymbol{w} = \Phi^T \boldsymbol{a} , \tag{46}$$

where $\boldsymbol{a} = \left(K + \lambda I_N\right)^{-1} \boldsymbol{y} = \begin{bmatrix} a_1 & \cdots & a_N \end{bmatrix}^T$ is a vector of what are called the dual variables. Recall that in equation 29 we saw that we can apply the weights to an unknown point x using the basis vector associated with the point:

$$\hat{y}(x) = \boldsymbol{\phi}_{row}^T(x)\boldsymbol{w} , \tag{47}$$

where, although we illustrated $\boldsymbol{\phi}_{row}^T(x)$ using linear regression, it could be implemented by any basis function. But what if we only know the functional form of the kernel, and not the basis function matrix Φ? This is where the kernel "trick" shows its true usefulness, since we can rewrite equation 47 using the dual variables as

$$\hat{y}(x) = k(x, x_i)\boldsymbol{a} , \tag{48}$$

where $k(x, x_i)$ is now written in functional form like equation 45. If we expand equation 48, we see that it is the sum of the N products between the dual variables and the kernel functions between x and x_i, or

$$\hat{y}(x) = a_1 k(x, x_1) + \ldots + a_N k(x, x_N) . \tag{49}$$

For example, if we use the polynomial kernel function, we can write

$$k(x, x_i) = (1 + x x_i)^p . \tag{50}$$

Let me again illustrate this with our simple three-point linear regression problem shown in Figure 2. Since $p = 1$, we can write

$$K_{i,j} = 1 + x_i x_j = \begin{bmatrix} 1 + x_1^2 & 1 + x_1 x_2 & 1 + x_1 x_3 \\ 1 + x_2 x_1 & 1 + x_2^2 & 1 + x_2 x_3 \\ 1 + x_3 x_1 & 1 + x_3 x_2 & 1 + x_3^2 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 5 & 7 \\ 4 & 7 & 10 \end{bmatrix}, \tag{51}$$

which we have seen before and know is uninvertible because its determinant equals 0. We therefore need to compute the dual variables using a small value of pre-whitening:

$$\boldsymbol{a} = \left(K + 10^{-6} I_N\right)^{-1} \boldsymbol{y} \approx \begin{bmatrix} -5 \cdot 10^5 \\ 1 \cdot 10^6 \\ -5 \cdot 10^5 \end{bmatrix}, \tag{52}$$

where $\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, and $\boldsymbol{y} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$. I have used the "approximately equal to" sign in equation 52 because the exact values of $\boldsymbol{a}$ must be computed to computer precision, and using the numbers shown above would result in a value of zero. Calculating the estimated y value at an x value of 2.5, using full precision, we get the same value that we saw in Figure 2:

$$\hat{y}(x) = a_1(1+2.5) + a_2(1+5) + a_3(1+7.5) = 2.25 . \tag{53}$$

Let us now move to the three-point quadratic solution. Now, $p = 2$, so we can write

$$K_{ij} = \begin{bmatrix} \left(1+x_1^2\right)^2 & \left(1+x_1x_2\right)^2 & \left(1+x_1x_3\right)^2 \\ \left(1+x_2x_1\right)^2 & \left(1+x_2^2\right)^2 & \left(1+x_2x_3\right)^2 \\ \left(1+x_3x_1\right)^2 & \left(1+x_3x_2\right)^2 & \left(1+x_3^2\right)^2 \end{bmatrix}, \tag{54}$$

which can be expanded to give

$$K_{ij} = \begin{bmatrix} 1+2x_1^2+x_1^4 & 1+2x_1x_2+x_1^2x_2^2 & 1+2x_1x_3+x_1^2x_3^2 \\ 1+2x_2x_1+x_2^2x_1^2 & 1+2x_2^2+x_2^4 & 1+2x_2x_3+x_2^2x_3^2 \\ 1+2x_3x_1+x_3^2x_1^2 & 1+2x_3x_2+x_3^2x_2^2 & 1+2x_3^2+x_3^4 \end{bmatrix} = \begin{bmatrix} 4 & 9 & 16 \\ 9 & 25 & 49 \\ 16 & 49 & 100 \end{bmatrix}, \tag{55}$$

which is invertible does not need pre-whitening to compute the dual variables:

$$\boldsymbol{a} = K^{-1}\boldsymbol{y} = \begin{bmatrix} -20.875 \\ 26.5 \\ -9.625 \end{bmatrix}. \tag{56}$$

Calculating the estimated y value at an x value of 2.5 we get the same value that we saw in Figure 3:

$$\hat{y}(x) = -20.875(1+2.5)^2 + 26.5(1+5)^2 - 9.625(1+7.5)^2 = 2.875 . \tag{57}$$

The general polynomial kernel is shown below, where p is the order of the polynomial:

$$K_{ij} = \begin{bmatrix} \left(1+x_1^2\right)^P & \left(1+x_1x_2\right)^P & \left(1+x_1x_3\right)^p \\ \left(1+x_1x_2\right)^P & \left(1+x_2^2\right)^P & \left(1+x_2x_3\right)^p \\ \left(1+x_1x_3\right)^p & \left(1+x_2x_3\right)^p & \left(1+x_2^2\right)^P \end{bmatrix} \tag{58}$$

This will allow us to compute polynomial fits of any order without having to build the initial matrices by adding columns.

Figure 9 shows the three regression fits for the three-point example, with solid lines for the kernel regression fits and dashed lines for the standard regression fits. The fits look identical for the linear and quadratic cases, but there is a slight deviation for the cubic fit due to the extra terms discussed earlier.

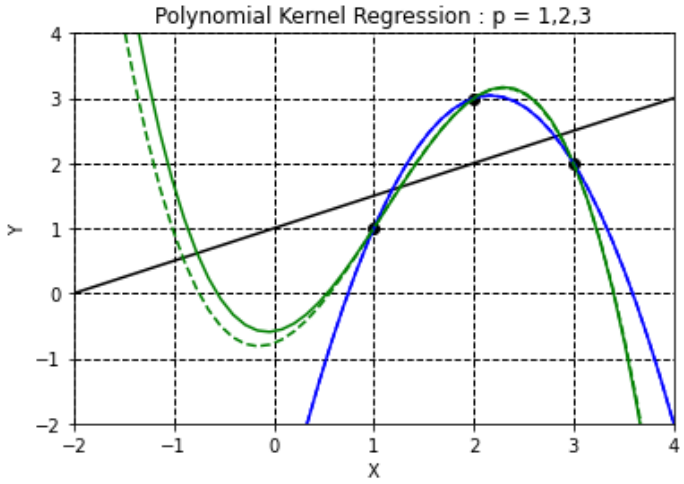


Figure 9: Polynomial kernel regression with the three-point example for p = 1,2, and 3.

Let's next apply kernel regression to the ten-point noisy sine wave example seen in Figure 10, which was created by generating a single period of a sine wave, corrupting it with Gaussian random noise and shifting it up by adding 1.0. Its x and y values are:

$$\boldsymbol{x}^T = [1,2,3,4,5,6,7,8,9,10], \text{ and}$$
$$\boldsymbol{y}^T = [1.07,1.22,1.93,1.77,1.24,1.04,0.39,-0.1,0.41,0.49]$$

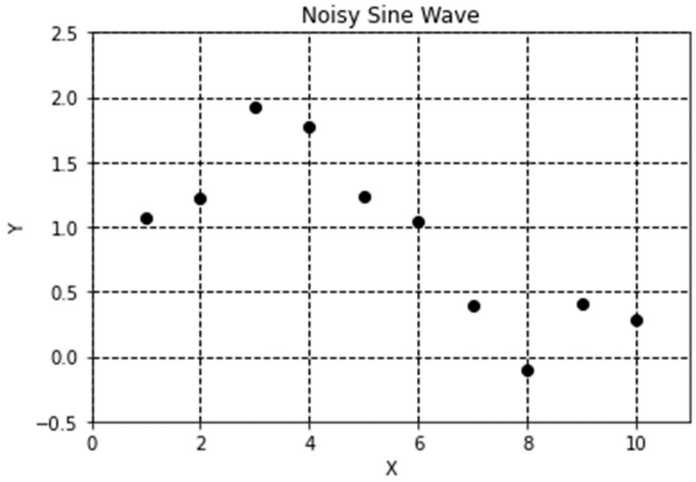


Figure 10: A noisy sine wave example with 10 points.

Figure 11 shows the polynomial kernel regression fits for a noisy sine wave example, using p = 1, 2, and 3.

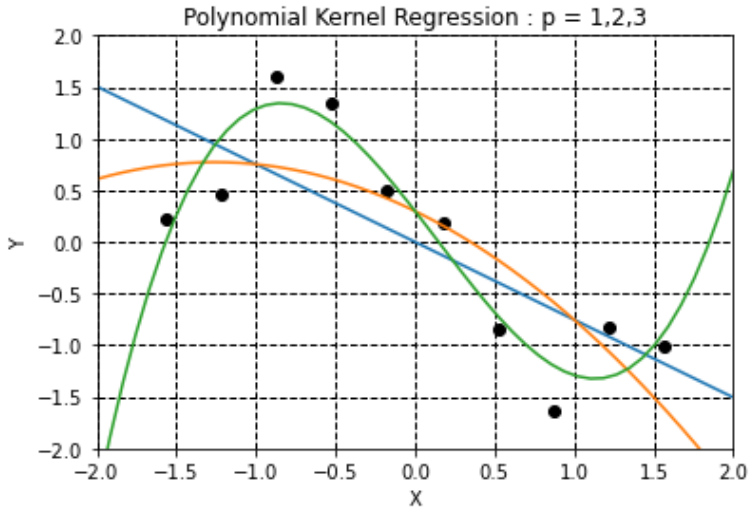


Figure 11: Polynomial kernel regression with the noisy sine example for p = 1,2, and 3.

Figure 12 shows the polynomial kernel regression fit for the noisy sine wave example using $p = 10$. Since $p = N$, the number of points, the fit is now perfect for each point. However, note the wild downward and upward swings at each end of the fit, which means that we have only done an exact fit within the range of points shown here. This is therefore an example of overfitting.

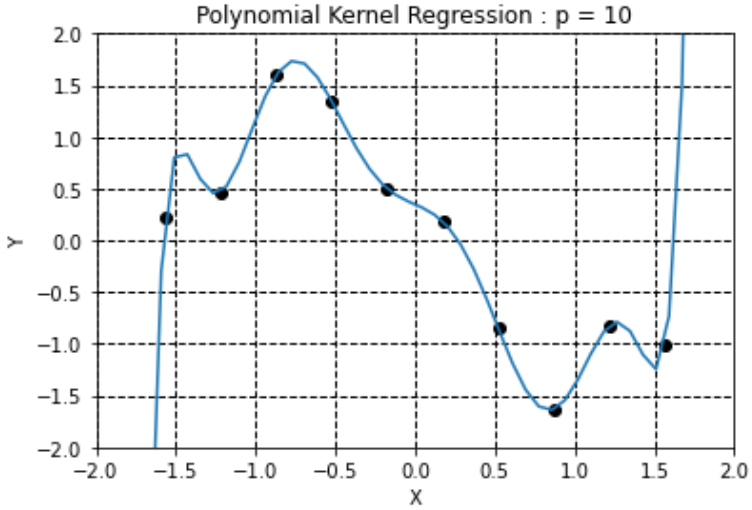


*Figure 12: Polynomial kernel regression with the noisy sine wave with p =*10.

Next, let's move to Gaussian kernel regression, which is also called the radial basis function neural network (RBFN), and can be written as:

$$K_{ij} = \exp\left[-\frac{(x_i - x_j)^2}{2\sigma^2}\right]. \tag{59}$$

For our three-point problem, this can be written out in full as

$$K = \begin{bmatrix} 1 & \exp\left[-\frac{(x_1-x_2)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_1-x_3)^2}{2\sigma^2}\right] \\ \exp\left[-\frac{(x_2-x_1)^2}{2\sigma^2}\right] & 1 & \exp\left[-\frac{(x_2-x_3)^2}{2\sigma^2}\right] \\ \exp\left[-\frac{(x_3-x_1)^2}{2\sigma^2}\right] & \exp\left[-\frac{(x_3-x_2)^2}{2\sigma^2}\right] & 1 \end{bmatrix}, \tag{60}$$

since $\exp\left[-\frac{(x_i-x_i)^2}{2\sigma^2}\right]=1$. The results of the Gaussian kernel inversion are applied to predict the unknown from the known points as follows:

$$\hat{y}(x) = a_1\phi_1 + a_2\phi_2 + \ldots + a_N\phi_N, \tag{61}$$

where $\phi_i = \exp\left[-\frac{(x_i-x\ )^2}{2\sigma^2}\right], i=1,\ldots,N,$ and $\boldsymbol{a} = \left(K+\lambda I_N\right)^{-1}\boldsymbol{y}$. For the three-point problem, this can be written as follows for each computed value:

$$\hat{y}(x) = a_1\exp\left[-\frac{(x_1-x)^2}{2\sigma^2}\right] + a_2\exp\left[-\frac{(x_2-x)^2}{2\sigma^2}\right] + a_3\exp\left[-\frac{(x_3-x)^2}{2\sigma^2}\right]. \tag{62}$$

The results of applying the Gaussian kernel to our three-point example is shown in Figure 13 for sigma values of 0.1 and 1.0. Notice that we get a perfect fit to the points and that the values never go below zero. A value of 1.0 gives a much smoother result than a value of 0.1.

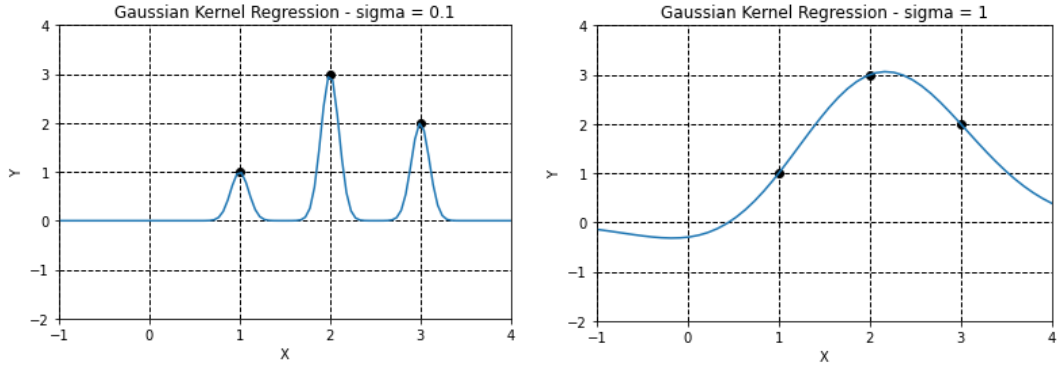


Figure 13: Gaussian kernel regression for the three-point example for $\sigma = 0.1$ *and* 1.0.

Next, let's move to the tanh kernel, which can be written as follows, (where I have set both *b* and *c* to 1 in the earlier formulation):

$$K_{ij} = \tanh\left[1 + x_i x_j\right]. \tag{63}$$

For our 3-point problem, this can be written out in full as:

$$K = \begin{bmatrix} \tanh\left[1+x_1^2\right] & \tanh\left[1+x_1x_2\right] & \tanh\left[1+x_1x_3\right] \\ \tanh\left[1+x_2x_1\right] & \tanh\left[1+x_2^2\right] & \tanh\left[1+x_2x_3\right] \\ \tanh\left[1+x_3x_1\right] & \tanh\left[1+x_3x_2\right] & \tanh\left[1+x_3^2\right] \end{bmatrix}. \tag{64}$$

Notice that like the polynomial kernel, and unlike the Gaussian kernel, the tanh kernel involves the products of the values, not their differences. The results of the tanh kernel inversion are applied to the unknown points as follows:

$$\hat{y}(x) = a_1\phi_1 + a_2\phi_2 + \ldots + a_N\phi_N, \tag{65}$$

where $\phi_i = \tanh\left[b + cxx_i\right]$, $i = 1,\ldots,N$, and $\boldsymbol{a} = \left(K + \lambda I_N\right)^{-1}\boldsymbol{y}$. For the three-point problem, this can be written as follows for each computed value, where we have set b to 0 and c to 0.25:

$$\hat{y}(x) = a_1 \tanh\left[0.25xx_1\right] + a_2 \tanh\left[0.25xx_2\right] + a_3 \tanh\left[0.25xx_3\right]. \tag{66}$$

The result of applying this function is shown in Figure 14. Notice that we get a perfect fit to the points but that, unlike the Gaussian fit, the solution swings towards negative numbers at the start and end of the plot.

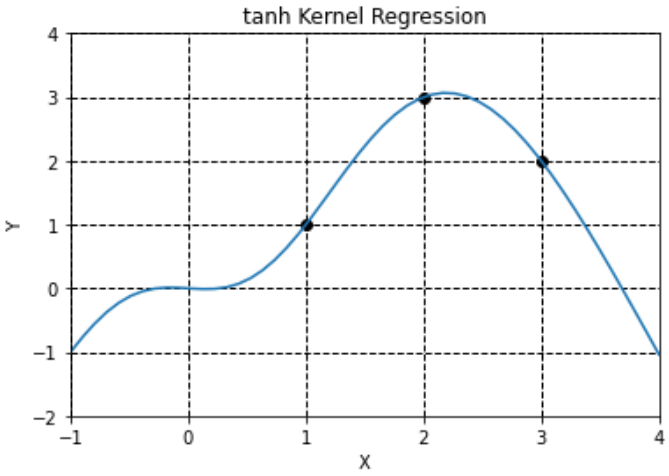


Figure 14: tanh kernel regression for the three-point example.

The result of applying Gaussian Kernel regression to our noisy sinusoidal example with $\sigma = 0.5$ is shown in Figure 15. Notice that we get a perfect fit to the points and that the fit before and after the observed points looks reasonable.

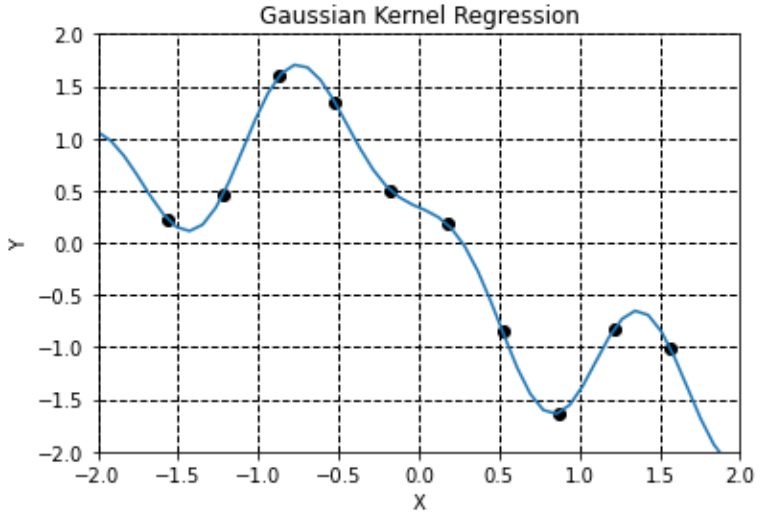


Figure 15: Gaussian kernel regression for the three-point example

Figure 16 shows the result of applying tanh kernel regression to our noisy sinusoidal example with $b = c = 1.0$. Notice that we get a perfect fit to the points and that the fit before and after the observed points looks reasonable.

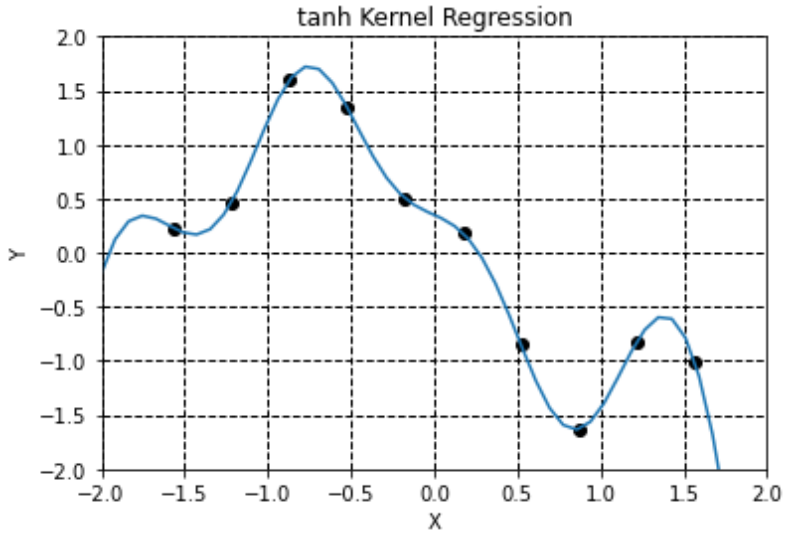


Figure 16: tanh Kernel regression for the noisy sine wave example.

THE GENERALIZED REGRESSION NEURAL NETWORK

Next, I will discuss a technique called the Generalized Regression Neural Network (GRNN), which is like the kernel methods we have been discussing except than no matrix inversion is involved. Recall that Gaussian kernel regression was written:

$$\hat{y}(x) = a_1\phi_1 + a_2\phi_2 + \ldots + a_N\phi_N(x_i)\,, \tag{67}$$

where $\phi_i = \exp\left[-\dfrac{(x_i - x)^2}{2\sigma^2}\right], i = 1,\ldots,N,$ and $\boldsymbol{a} = \left(K + \lambda I_N\right)^{-1}\boldsymbol{y}$.

The GRNN is written

$$\hat{y}(x) = \frac{y_1\phi_1 + y_2\phi_2 + \ldots + y_N\phi_N(x_i)}{\sum_{i=1}^{N}\phi_i}, \tag{68}$$

where $\phi_i = \exp\left[-\frac{(x_i - x)^2}{2\sigma^2}\right]$.

In other words, the GRNN is a weighted sum of the input points where the weights themselves are normalized Gaussian kernels. This can also be written as:

$$\hat{y}(x) = y_1k_1 + y_2k_2 + \ldots + y_Nk_N(x_i), \tag{69}$$

where $k_i = \frac{\exp\left[-\frac{(x_i - x)^2}{2\sigma^2}\right]}{\sum_{i=1}^{N}\phi_i}$ are the kernel functions. In matrix/vector format we have

$$\hat{y}(x) = \boldsymbol{y}^T K\mathbf{1}, \tag{70}$$

where $\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, K = \begin{bmatrix} k_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_N \end{bmatrix}$, and $\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$. The three-point problem is written:

$$\hat{y}(x) = \frac{\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}}{\sum_{i=1}^{N}\phi_i} \begin{bmatrix} \phi_1 & 0 & 0 \\ 0 & \phi_2 & 0 \\ 0 & 0 & \phi_3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{71}$$

where $\phi_i = \exp\left[-\frac{(x_i - x)^2}{2\sigma^2}\right]$. The results in the following sum:

$$\hat{y}(x) = y_1 \frac{\exp\left[-\frac{(x_1 - x)^2}{2\sigma^2}\right]}{\sum_{i=1}^{3}\phi_i} + y_2 \frac{\exp\left[-\frac{(x_2 - x)^2}{2\sigma^2}\right]}{\sum_{i=1}^{3}\phi_i} + y_3 \frac{\exp\left[-\frac{(x_3 - x)^2}{2\sigma^2}\right]}{\sum_{i=1}^{3}\phi_i}. \tag{72}$$

Figure 17 shows the result of applying the GRNN method for the three-point example using $\sigma = 0.25$. Note that the method extrapolates both the first and last values at the start and end of the plot.

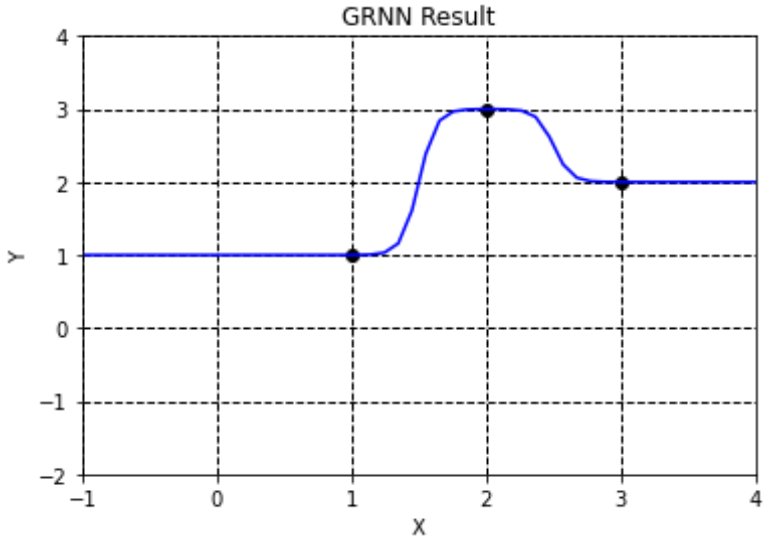


Figure 17: The GRNN method for the three-point example using $\sigma = 0.25$.

Figure 18 shows the result of applying the GRNN method to the noisy sine wave example using $\sigma = 0.25$. Again, notice that the method extrapolates both the first and last values at the start and end of the plot.

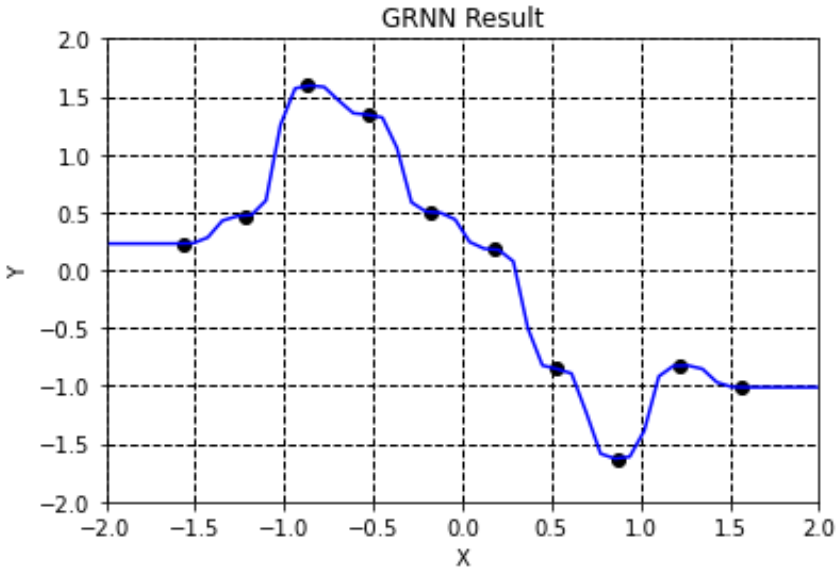


Figure 18: GRNN regression for the noisy sine wave example.

Finally, Figure 19 shows the comparison of all our results for the noisy sine wave problem. Again, they all fit the observed points perfectly. Also, the biggest difference is in how the interpolated values are computed as we move away from the first and last observed points. The worst extrapolation is using the polynomial fit, which shows wild swings. The best fit is given by the GRNN, which extrapolates the first and last points exactly.

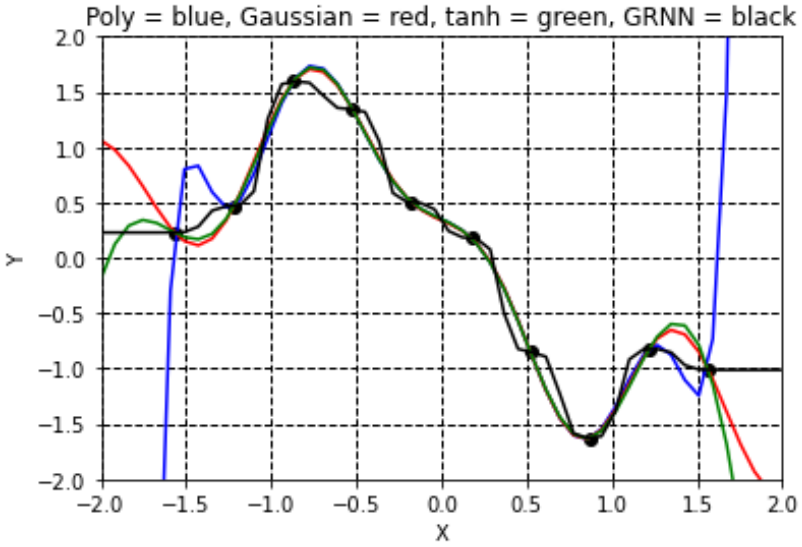


Figure 19: A comparison of all the regression fits for the noisy sine wave example.

REAL DATA EXAMPLE

Finally, I will apply several of the techniques we have discussed thrughout this paper to the prediction of reservoir parameters from seismic attributes (Hampson et al., 2001). I will use a channel sand example from the Western Canadian Sedimentary Basin, in which we predict pseudo-density logs. This involves finding a linear or nonlinear relationship between a set of M seismic attributes and some reservoir parameter of interest. The simplest such relationship is the linearly weighted sum of attributes A_i given by

$$P = w_0 + w_1 A_1 + w_2 A_2 + \cdots + w_M A_M \,. \tag{73}$$

The training samples consist of well log curves that intersect the seismic volumes, so the primal least-squares solution for N samples is:

$$\boldsymbol{w} = \left[A^T A + \lambda I \right]^{-1} A^T P \,, \tag{74}$$

where λ = a pre-whitening factor, I is the M+1 x M+1 identity matrix, $\boldsymbol{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_M \end{bmatrix}$,

$$A = \begin{bmatrix} 1 & A_{11} & \cdots & A_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & A_{N1} & \cdots & A_{NM} \end{bmatrix} \text{and } P = \begin{bmatrix} P_1 \\ \vdots \\ P_N \end{bmatrix}.$$

We modify the MLFN network shown in Figure 2 to solve the reservoir prediction by having the input consist of M attributes and the predicted output be the reservoir parameter. Any number of neurons can be used in the hidden layer (a rule of thumb is to use 2/3 the number of input attributes). The weights are initially set to random values and then undated using backpropagation.

For the RBFN approach to reservoir prediction (Ronen et al.,1994), the equation to compute the weights is similar to equation 8 except that the Φ matrix is written

$$\boldsymbol{w} = \Phi^{-1} P\,, \tag{75}$$

where $\Phi = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \vdots & \ddots & \vdots \\ \phi_{N1} & \cdots & \phi_{NN} \end{bmatrix}$, $\phi_{ij} = \exp\left[-\dfrac{\left\| \boldsymbol{A}_i - \boldsymbol{A}_j \right\|^2}{2\sigma^2} \right]$, $\boldsymbol{A}_i^T = \left[A_{1i}, \ldots, A_{Ni} \right]$ are the column vectors from matrix A, and $\left\| \boldsymbol{A}_i - \boldsymbol{A}_j \right\|$ is the "distance in M-dimensional attribute space.

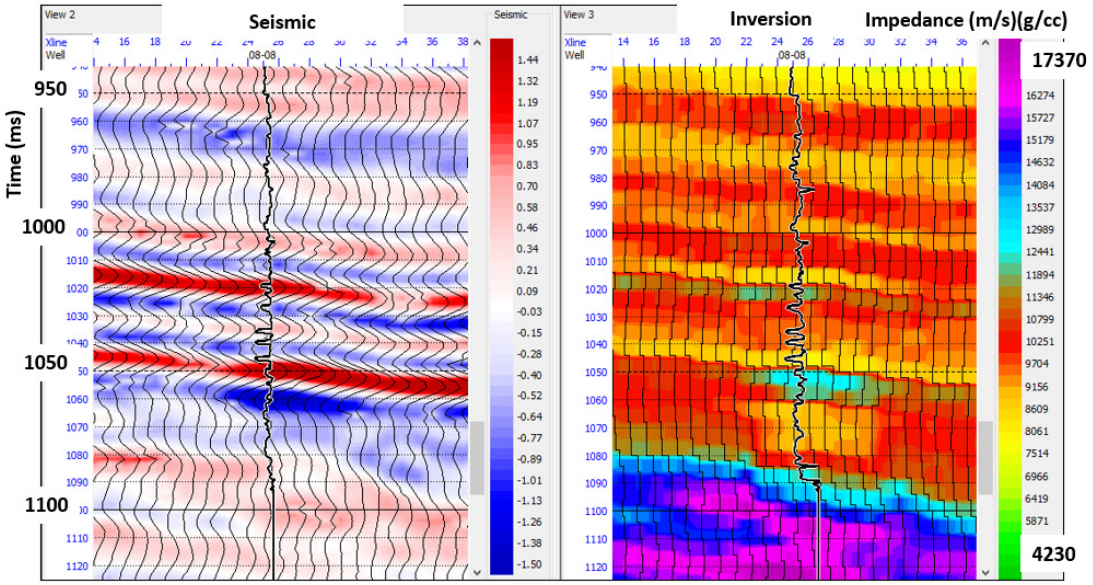
For each output sample, we apply the weights as follows:

$$P_0 = \sum_{i=1}^{N} w_i \exp\left[-\frac{\left\| \boldsymbol{A}_0 - \boldsymbol{A}_i \right\|^2}{2\sigma^2} \right], \tag{76}$$

In the GRNN (Hampson et al., 2001) approach, we compute the results as follows for the reservoir prediction problem:

$$P_0 = \frac{\sum_{i=1}^{N} P_i \exp\left[-\dfrac{\left\| \boldsymbol{A}_0 - \boldsymbol{A}_i \right\|^2}{2\sigma^2} \right]}{\sum_{i=1}^{N} \exp\left[-\dfrac{\left\| \boldsymbol{A}_0 - \boldsymbol{A}_i \right\|^2}{2\sigma^2} \right]}, \tag{77}$$

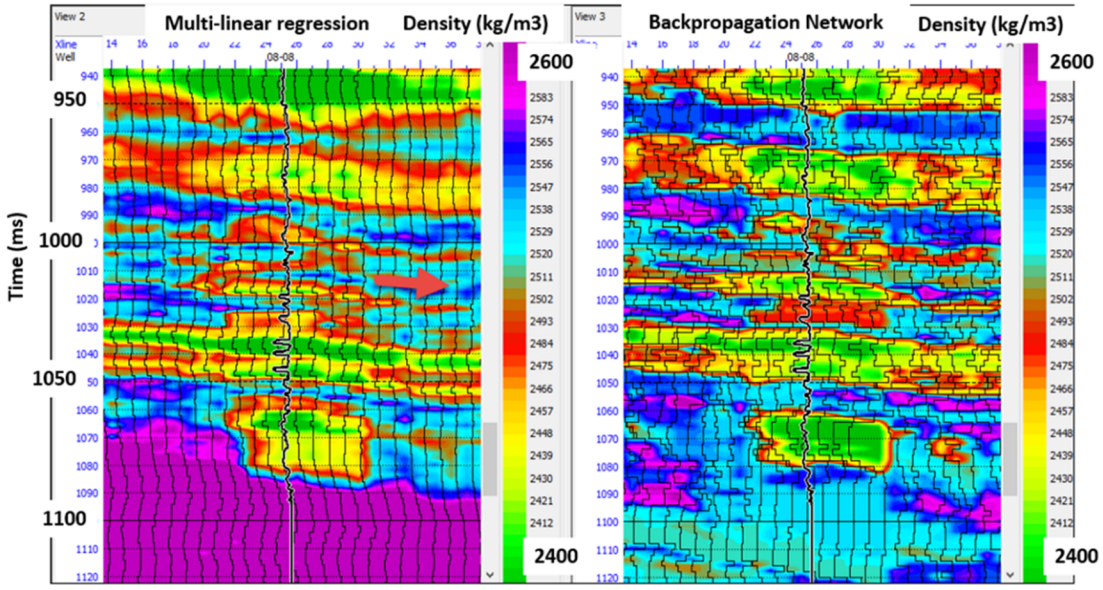
Figures 20 through 21 show the application of these four methods to a channel sand example from the Western Canadian Sedimentary Basin, in which we predict pseudo-density logs. Figure 20(a) shows a vertical cross-section from the input seismic volume, with the acoustic impedance log spliced in at its location. Figure 20(b) shows a vertical cross-section from the inverted acoustic impedance volume. In both figures, the channel sand is located below a time of 1070 msec.



(a) (b)

Figure 20: Vertical cross-sections from (a) the seismic volume, and (b) the inverted acoustic impedance volume, where the channel sand is shown at a time of 1070 msec on both figures.

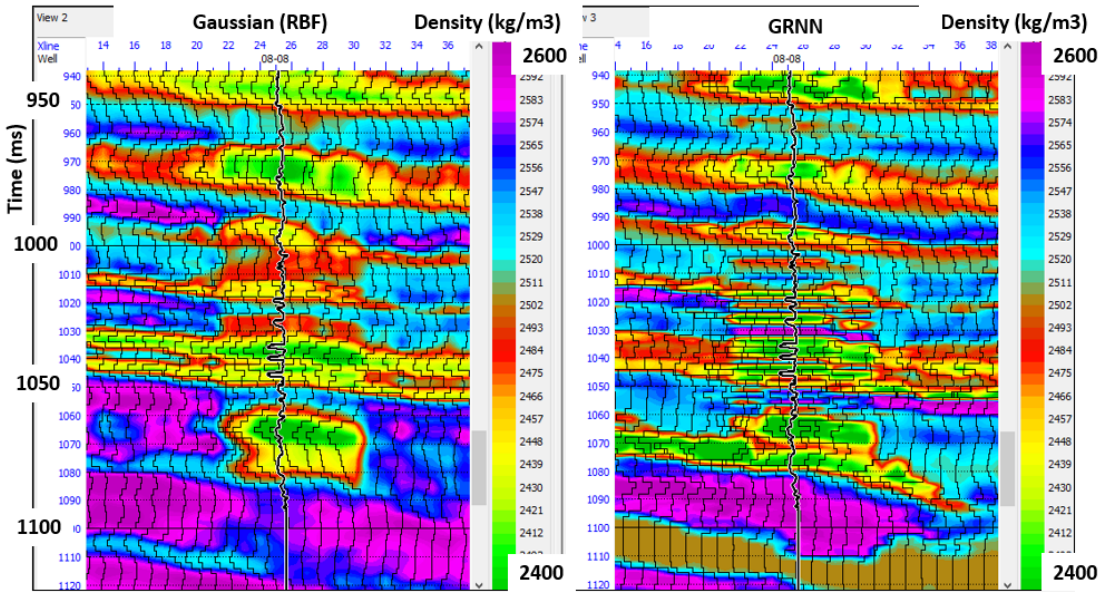
Figure 21(a) shows one line from the density volume predicted using multi-linear regression, with the density log spliced into the section. Figure 21(b) shows one line from the density volume predicted using the MLFN.



(a) (b)

Figure 21: Vertical cross-sections from (a) the multi-linear regression result, and (b) the backpropagation method, where the channel sand is shown at a time of 1070 msec on both figures.

Figure 22(a) shows one line from the density volume predicted using the RBFN, with the density log spliced into the center of the section. Finally, Figure 22(b)shows one line from the density volume predicted using the GRNN, again with the density log spliced into the center of the section. Note the extra resolution at the channel sand in the MLFN, RBFN and GRNN results over the inverted section



(b) (b)

Figure 22: Vertical cross-sections from (a) the RBFN and (b) the GRNN, where the channel sand is shown at a time of 1070 msec on both figures.

CONCLUSIONS

In this study I considered two different forms of the generalized inverse, the primal and dual, and showed how the dual form led to several very powerful kernel regression methods, that go well beyond the multi-linear regression techniques used in many applications. I illustrated these techniques using two simple datasets, one with only three points and the other with ten points, and then applied them to a seismic case study from the Blackfoot area in Alberta.

Regression methods can be broken into two categories: linear and nonlinear. There is only one linear approach, and we started by applying linear regression to the three-point problem, which did not fit the points exactly. There are a variety of nonlinear regression methods, and I stared with polynomial regression. Linear regression is the simplest polynomial regression approach and uses a first order polynomial. I also applied the quadratic and cubic polynomial regression techniques to the three-point problem and found that both gave a perfect fit to the three-point example, although the cubic method overfit the problem, since it had more weights that observed points. I then introduced the concept of the linear basis function model, in which three basis functions were considered: polynomial, Gaussian and hyperbolic tangent, or tanh. I then solved for the linear basis function weights using both the primal least-squares solution the dual least-squares solution. This lead directly to kernel regression, and we looked at three types of kernel regression techniques: polynomial kernel regression, Gaussian kernel regression, also called the radial basis function neural network, and the tanh kernel regression technique. The polynomial regression network performed very well on both the three-point problem and the ten-point noisy sine wave problem, where we were able to perform a fit to the data using a tenth order polynomial.

I also introduced the generalized regression neural network, or GRNN, and showed that although its application involves a Gaussian kernel it does not require the inversion of the kernel and instead applies the result “on-the-fly” to the data.

I finished by applying all these methods to a seismic reservoir prediction problem, which involved predicting density from a set of multiple seismic attributes. In this case study, I used multi-linear regression, the RBFN and GRNN kernel-based algorithms, and also the multi-layer feed forward, or MLFN, method which is also called the backpropagation method. The theory of MLFN was not discussed in the study today but it is a well described method in the literature. All the methods gave similar results, but each had its own advantages and disadvantages. The most robust was multi-linear regression, but it underpredicted the density log. The other three methods gave better fits to the density log but were in danger of overfitteing and produced spurious artefacts.

APPENDIX

THE DUAL REGRESSION SOLUTION

Polynomial regression using the covariance, or inner product, matrix X^TX with ridge regression is called the primal solution and is written:

$$\boldsymbol{w} = \left(X^T X + \lambda I_{p+1} \right)^{-1} X^T \boldsymbol{y}, \tag{A1}$$

where X is an N row by p+1 column matrix, and the autocorrelation matrix $A = X^TX$ is a square p+1 x p+1 matrix. The term λ is a pre-whitening factor that is multiplied by the p+1 x p+1 identity matrix I_{p+1}. To derive the dual solution, we start by bringing the inverted term back to the left side:

$$\left(X^T X + \lambda I_{p+1} \right) \boldsymbol{w} = X^T \boldsymbol{y} . \tag{A2}$$

This can then be re-arranged as

$$X^T X \boldsymbol{w} + \lambda \boldsymbol{w} = X^T \boldsymbol{y} . \tag{A3}$$

Subtracting the first term on the left-hand side from both sides and multiplying by the inverse pre-whitening factor gives

$$\boldsymbol{w} = \lambda^{-1} \left(X^T \boldsymbol{y} - X^T X \boldsymbol{w} \right). \tag{A4}$$

Factoring out the transpose of X then gives

$$\boldsymbol{w} = X^T \lambda^{-1} \left(\boldsymbol{y} - X \boldsymbol{w} \right). \tag{A5}$$

Note that equation A5 can be rewritten as

$$\boldsymbol{w} = X^T \boldsymbol{a}, \tag{A5}$$

where $\boldsymbol{a} = \lambda^{-1} \left(\boldsymbol{y} - X \boldsymbol{w} \right)$ contains parameters which are called the dual variables. Notice that we can also multiply both sides of the expression for a by l, giving

$$\lambda \boldsymbol{a} = \boldsymbol{y} - X \boldsymbol{w} . \tag{A6}$$

Now comes a “trick”, where we re-substitute the $X^T\boldsymbol{a}$ form into $\boldsymbol{w}$ in equation A6:

$$\lambda \boldsymbol{a} = \boldsymbol{y} - X \boldsymbol{w} = \boldsymbol{y} - X X^T \boldsymbol{a} , \tag{A7}$$

where $\boldsymbol{a} = \lambda^{-1}\left(\boldsymbol{y} - X\boldsymbol{w}\right)$ contains parameters which are called the dual variables. We then can group the terms as

$$\left(XX^T + \lambda\right)\boldsymbol{a} = \boldsymbol{y}\,, \tag{A8}$$

We now re-substitute this into the definition of $\boldsymbol{w}$ to get:

$$\boldsymbol{a} = \left(XX^T + \lambda I_N\right)^{-1}\boldsymbol{y}\,. \tag{A9}$$

Finally, re-substitute into the weight equation to get the final form of the dual regression solution:

$$\boldsymbol{w} = X^T\boldsymbol{a} = X^T\left(XX^T + \lambda I_3\right)^{-1}\boldsymbol{y}\,. \tag{A10}$$

Recalling that the equation for primal regression is

$$\boldsymbol{w} = \left(X^TX + \lambda I_{p+1}\right)^{-1}X^T\boldsymbol{y}\,, \tag{A11}$$

we can combine equations A10 and A11 to show that

$$X^T\left(XX^T + \lambda I_N\right)^{-1} = \left(X^TX + \lambda I_{p+1}\right)^{-1}X^T\,. \tag{A12}$$

That is, the weights can be computed using either an inverse containing the N x N matrix XX^T or the p+1 x p+1 matrix X^TX.

REFERENCES

Bishop, C.M, 2006, Pattern Recognition and Machine Learning: Springer-Verlag.

Hampson, D., Schuelke, J.S., and Quirein, J.A., 2001, Use of multi-attribute transforms to predict log properties from seismic data: Geophysics, 66, 220-231.

Hastie, T., Tibshirani, R., and Friedman, J., 2009, The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd Edition: Springer Series in Statistics.

Murphy. K.P., 2012, Machine Learning: A Probabilistic Perspective: MIT Press, Cambridge, Mass.

Powell, M.J.D., 1987, Radial basis functions for multivariable interpolation: a review. In J.C. Mason and M.G. Cox (Eds.), Algorithms for Approximation, 143-167. Oxford: Clarendon Press.

Ronen, S., Schultz, P.S., Hattori, M., and Corbett, C., 1994, Seismic-guided estimation of log properties, Part 2: Using artificial neural networks for nonlinear attribute calibration: The Leading Edge, 13**,** Issue 6, 674-678.

Rummelhart, D.E., Hinton, G.E., and Williams, R.J., 1986, Learning representations of back-propagation errors: Nature, 323, 533-536.

Russell, B.H., 2019, Machine learning and geophysical inversion — A numerical study: The Leading Edge, 38**,** Issue 7, 512-519.

Schölkopf, B. and Smola, A., 2002, Learning with Kernels: MIT Press, Cambridge, Mass.

Shawe-Taylor, J. and Cristianini, N., 2004, Kernel Methods for Pattern Analysis: Cambridge University Press.

Specht, D.F., 1991, A general regression neural network: IEEE Transactions on Neural Networks, 2(6), 568-576.

ACKNOWLEDGEMENTS

I wish to thank my colleagues at the CREWES Project and GeoSoftware for their support and ideas, as well as the sponsors of the CREWES Project.