Generalized Regressive Neural Networks
Bruce Rosen

The object of a Generalized Regressive Neural Network (GRNN) is to estimate the output of a sample input \( x_* \) by creating a function (from existing input output data) by averaging the output values of all other output values in the training population. The weighting factors computed by the GRNN network are unequal, and are based on the distances between the sample input and each member of the training population.

Assume an input-output pattern is of the form \((x, y)\), and further assume we have \( p \) sample training patterns \((x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)\)

For convenience, as an example, let us use the following data of sample height and weight data:

<table>
<thead>
<tr>
<th>Person #</th>
<th>Height (in &quot; )</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6' 1&quot;</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>5’ 10”</td>
<td>190</td>
</tr>
<tr>
<td>3</td>
<td>3’ 0”</td>
<td>40</td>
</tr>
</tbody>
</table>

Let us assume we want to compute an estimate of the weight of a person * who is 6 feet tall, i.e. given \( x_* = 6' \), compute \( f(x_*) \) the output.

One way of estimating the output of input pattern \( x_* \) is to compute the weighted average of all the output patterns. A weighted average is a weighted sum of all the samples in our training population. For this to be an average, the weights (in our sum) must add up to 1.

For example, we could estimate how heavy person * is by computing the mean of all the samples of our population, weighing (and weighting) each output value equally i.e.

\[
f(x_*) = \frac{1}{3} \sum_{i=1}^{p} \frac{1}{P} y_i,
\]

\[
f(x_*) = \frac{1}{3} 200 + \frac{1}{3} 190 + \frac{1}{3} 40 = 143.
\]

Which is not very accurate.

Alternatively, we could estimate person *'s weight as an average made from an unequal weight of each member of our training population. For example we could weight our first 2 samples more heavily than the third, e.g.

\[
f(x_*) = \frac{2}{5} 200 + \frac{2}{5} 190 + \frac{1}{5} 40 = 164.
\]

Again, the weights sum to 1 (\( \frac{2}{5} + \frac{2}{5} + \frac{1}{5} = 1.0 \)), but our output (estimate) is not very good.
We could just use the weight of the person whose height is most similar to person *, and ignore all the other people (in which case, we assign their weight a value of 0).

\[
\begin{align*}
    f(x_*) &= 1 \cdot y_1 + 0 \cdot y_2 + 0 \cdot y_3, \\
    f(x_*) &= 1 \cdot 200 + 0 \cdot 190 + 0 \cdot 40 = 200.
\end{align*}
\]

which would give us a much better representation. This approach is called nearest neighbor estimation. However to find out which person in our sample population is most similar, we must create a distance function. The person who has the smallest distance in height to person * is the nearest neighbor.

The distance function we use is just the square of the distance between heights, i.e.

\[
d(x_*, x) = (x_* - x)^2
\]

therefore,

\[
\begin{align*}
    d(x_*, x_1) &= (6' - 6.08')^2 = 0.0064' \\
    d(x_*, x_2) &= (6' - 5.83')^2 = 0.0289' \\
    d(x_*, x_3) &= (6' - 3')^2 = 9.0'
\end{align*}
\]

Then clearly \( x_1 \) is the nearest neighbor, and we would use \( y_1 \) as the estimate for \( f(x_*) \).

Although this estimate is good, it ignores the height and weight information from persons 2 and 3.

What we really want to to do is provide an average which weighs each person’s weight contribution to our estimate by how similar each person is in height.

To do this, we create a weighting function that is inversely related to the distance between two samples. If we use the weighting function

\[
w(x_*, x) = e^{-d(x_*, x)}
\]

we have the nice property that the weight between two samples is 1 when the samples are identical, and decay asymptotically to 0 as the distance between the samples increases. In our example above, this would give weights of:

\[
\begin{align*}
    w(x_*, x_1) &= e^{-d(x_*, x_1)} = e^{-0.0064} = 0.9936 \\
    w(x_*, x_2) &= e^{-d(x_*, x_2)} = e^{-0.0289} = 0.9715 \\
    w(x_*, x_3) &= e^{-d(x_*, x_3)} = e^{-9.0} = 0.00012
\end{align*}
\]

However, \( w(x_*, x_1) + w(x_*, x_2) + w(x_*, x_3) > 1 \). Our weights do not sum to one. To maintain the correct proportions and ensure the weights sum to one, we must divide each weight by the sum of all weights, i.e.

\[
\begin{align*}
    w_{\text{new}}(x_*, x_1) &= \frac{w(x_*, x_1)}{w(x_*, x_1) + w(x_*, x_2) + w(x_*, x_3)}, \\
    w_{\text{new}}(x_*, x_2) &= \frac{w(x_*, x_2)}{w(x_*, x_1) + w(x_*, x_2) + w(x_*, x_3)}, \\
    w_{\text{new}}(x_*, x_3) &= \frac{w(x_*, x_3)}{w(x_*, x_1) + w(x_*, x_2) + w(x_*, x_3)}.
\end{align*}
\]
and in general

\[
\begin{align*}
    w_{\text{new}}(x_*, x_i) &= \frac{w(x_*, x_i)}{\sum_{j=1}^P w(x_*, x_j)}, \\
    w_{\text{new}}(x_*, x_i) &= \frac{e^{-\frac{(x_* - x_i)^2}{\sigma}}}{\sum_{j=1}^P e^{-\frac{(x_* - x_j)^2}{\sigma}}}.
\end{align*}
\]

Thus, our new weights become

\[
\begin{align*}
    w_{\text{new}}(x_*, x_1) &= \frac{0.936}{0.936 + 0.9715 + 0.00012} = \frac{0.936}{1.9083} = 0.5055, \\
    w_{\text{new}}(x_*, x_2) &= \frac{0.9064}{0.9064 + 0.9715 + 0.00012} = \frac{0.9064}{1.878} = 0.4943, \\
    w_{\text{new}}(x_*, x_3) &= \frac{0.9715}{0.936 + 0.9715 + 0.00012} = \frac{0.9715}{1.9083} = 0.00061,
\end{align*}
\]

and our new estimate becomes

\[
f(x_*) = \sum_{i=1}^P \frac{y_i e^{-\frac{(x_* - x_i)^2}{\sigma}}}{\sum_{j=1}^P e^{-\frac{(x_* - x_j)^2}{\sigma}}}.
\]

and our new estimate for \(f(x_*)\) is

\[
f(x_*) = 0.5055 \times 200 + 0.4943 \times 190 + 0.00061 \times 40 = 195.0194
\]

Of course, the above analysis is when \(x\) is a scalar. When \(x\) is a vector of \(N\) components \((\vec{x})\), the distance function is

\[
d(\vec{x}_*, \vec{x}) = \sum_{k=1}^N (\vec{x}_{*, k} - \vec{x}_k)^2.
\]

The network error can be determined by leave one out cross validation. The network performance can be improved by adjusting the widths of the Gaussian like functions. In this case, the weighting function is

\[
w(\vec{x}_*, \vec{x}) = e^{-d(\vec{x}_*, \vec{x})/\sigma}
\]

where the global variable \(\sigma\) is optimized by cross validation. Thus our new equation is

\[
f(x_*) = \sum_{i=1}^P \frac{y_i e^{-\frac{(x_* - x_i)^2}{\sigma}}}{\sum_{j=1}^P e^{-\frac{(x_* - x_j)^2}{\sigma}}} = \sum_{i=1}^P \frac{\sum_{k=1}^N (x_* - x_{i, k})^2 / \sigma}{\sum_{j=1}^P e^{-\sum_{k=1}^N (x_* - x_{j, k})^2 / \sigma}}.
\]

We can take this analysis one step further by employing multiple \(\sigma\) values, one for each component of the vector \(\vec{x}\) as

\[
f(x_*) = \sum_{i=1}^P \frac{\sum_{k=1}^N (x_* - x_{i, k})^2 / \sigma_k}{\sum_{j=1}^P e^{-\sum_{k=1}^N (x_* - x_{j, k})^2 / \sigma_k}},
\]

and optimizing each \(\sigma_k\) using leave one out cross validation.