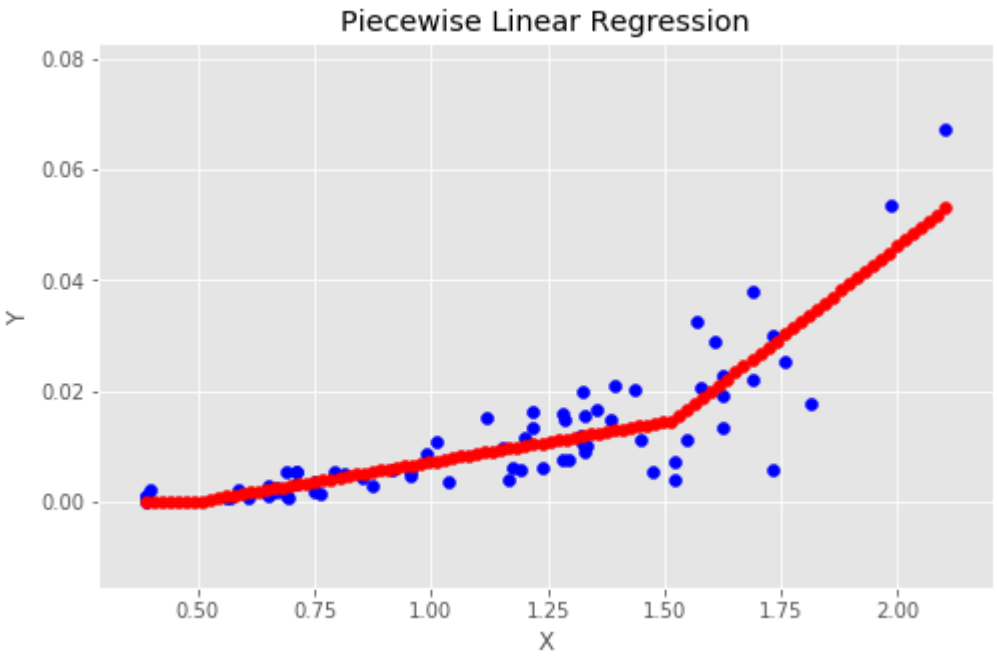
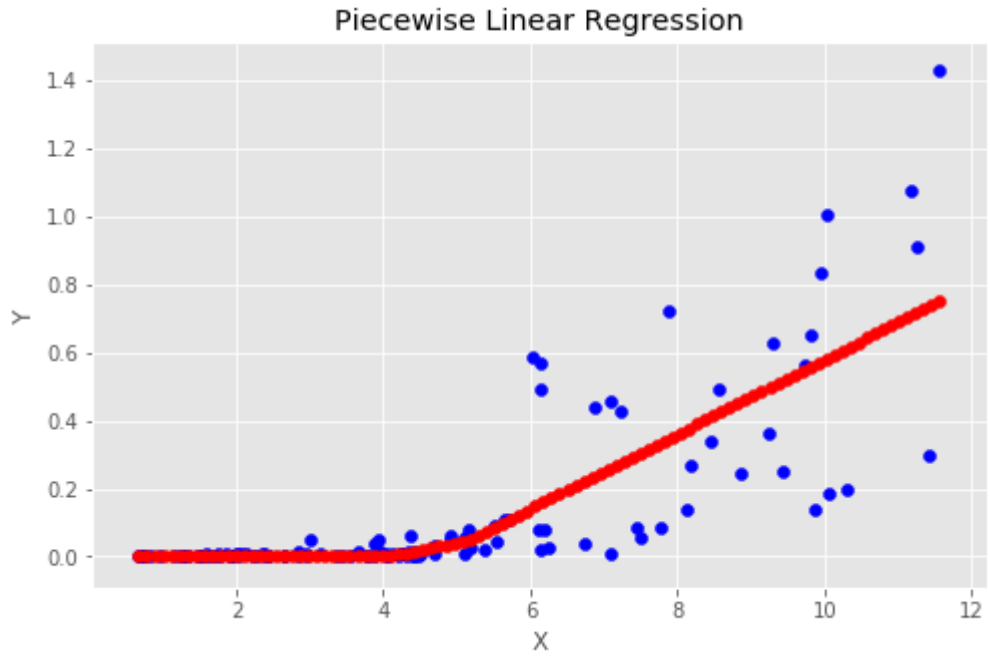
Piecewise Linear Regression using neural network





The math formula does not rendered with the default markdown. Read the pdf Readme instead.

Motivation

Relationships that can be explained by linear regression are limited in practice. Polynomial or other complex machine learning models are hard to explain, and could behave extreme outside of the data range. Piecewise linear regression, with flexible number of segments and break points may work when regression is too simple but patches of regression could express the phases of the relationship.

Some examples of piecewise linear regression applications are linked below:

- A Tutorial on the Piecewise Regression Approach Applied to Bedload Transport Data
- Water-cement ration v.s. compressive strength
- Piecewise Linear Regression: A Statistical Method for the Analysis of the Relationship between Traffic Signal Parameters and Air Pollutant Emissions

Previous works

[1] A Tutorial on the Piecewise Regression Approach Applied to Bedload Transport Data

- Break point estimates need to be provided by user
- Use of SAM NLIN, Nonlinear least squares regression

[2] segmented: An R Package to Fit Regression Models with Broken-Line Relationships

- Break point estimates need to be provided by user
- Iterative linear regression

[3] A Learning Algorithm for Piecewise Linear Regression

- Clustering and regression. multi-variables. The line may be disconnected.
- separate gate for each hidden node.

Proposed method - Neural network application

Let's try to use simple neural network with 1 hidden layer with ReLU (Rectified linear unit) activation function. The benefit is that we can remove the manual input and let the data decide the number of segments and breakpoints, with a very simple feed forward network. In comparison with [3], it is quite similar idea, but this model is much simple. The gate is ReLU with no separate parameters, and there's no clustering etc. It's just summing up output of ReLU with no bias :

$$y = (1, \ldots, 1)(W^T x + c)^+$$

Here, y is a dependent variable. x is independent variables. It is a column vector with different variables in row direction. W contains slopes of different input variables in the row direction and the hidden nodes in the column direction. The result of $W^T x$ places hidden nodes in row direction. The bias c is a column vector with a bias for each hidden nodes in row direction. Let me provide more concrete example. The ith row of $W^T x + c$ is an input to a hidden node h_i, say z_i. The z_i for 2 variables input $x = [x_1, x_2]^T$ can be written as

$$z_i = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + c_i = w_1 * x_1 + w_2 * x_2 + c_i$$

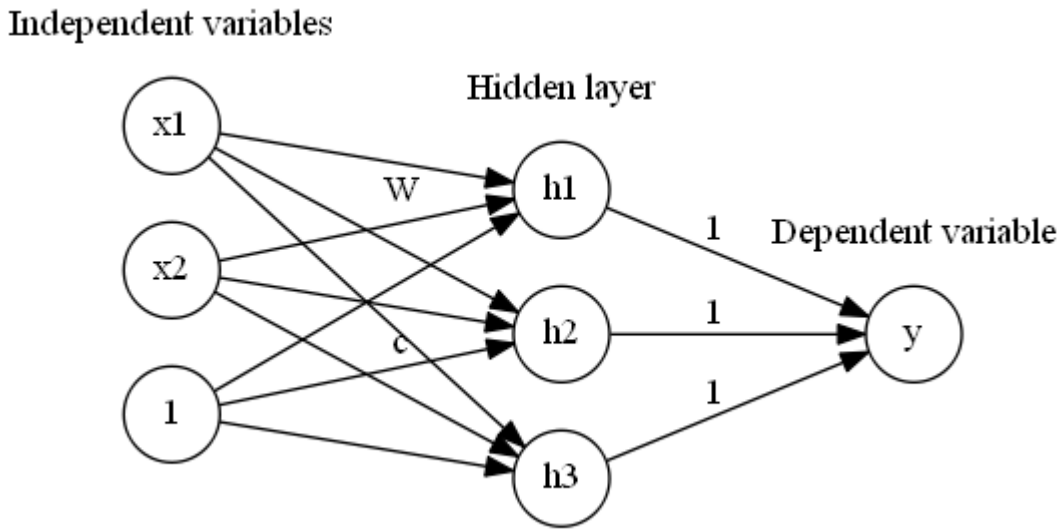
Here, w_1 and w_2 are slopes for x_1 and x_2 respectively. c_i is a bias.

The $(.)^+$ represent ReLU or $max\{0, .\}$. Finally, applying (1,...,1) just means adding up all the rows, in other words, the outputs of all the hidden nodes with no bias.

By adding 1 on the last row on x, and adding c^T on the last row on W, the 1st formula can be written as

$$y = (1, \ldots, 1)(W^T x)^+$$

Here's the graphical representation of the model. Note that the bias is expressed as node "1" and bias edges c.



We apply L1 regularization on W to regulate the number of segments.

Sample data

Sample data