Non-Linear Regression

Data 624 Predictive Analytics

Niteen Kumar
Hovig Ohannessian
Gurpreet Singh
Peter Goodridge

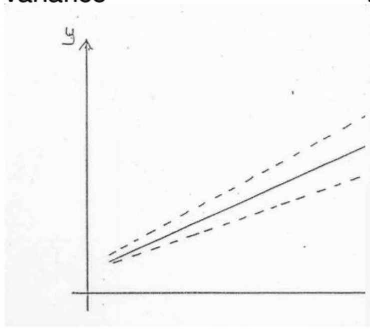
Regression

Regression is a statistical measurement used to determine the strength of relationship between one dependent variable and a series of other changing variables also known as independent variables. Linear and Multiple Regressions are commonly used types of Regression Analysis.

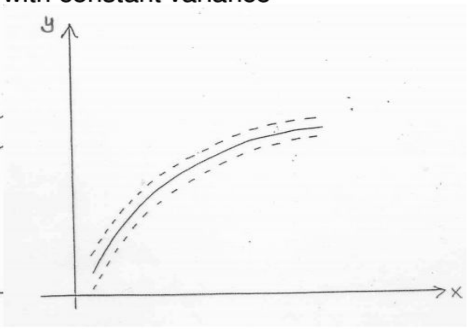
For advanced analysis, Non-Linear Regression Analysis is used. Both linear and nonlinear regression find the values of the parameters (slope and intercept for linear regression) that make the line or curve come as close as possible to the data. More precisely, the goal of both models is to minimize the sum of the squares of the vertical distances of the points from the line or curve.

Non Linear Regression

Untransformed: Linearity, but with increasing variance



After a **logarithmic transformation:** Non-linearity, but with constant variance



* **Nonlinear regression models are those that are not linear in the parameters**

- Transformation is necessary to obtain variance homogeneity but transformation destroys linearity
- Linearity does not fit, and the transformation seems to destroy other parts of the model assumptions (e.g. the assumption of variance homogeneity)
- Theoretical knowledge (e.g. from kinetics or physiology) indicates that the proper relation is intrinsically non-linear
- Interest is in functions of the parameters that do not enter linearly in the model (e.g. kinetic rate constants)

Comparison between Linear and Non-Linear Regression

Unlike the linear regression that estimates linear models, nonlinear regression can estimate nonlinear models with arbitrary relationships between independent and dependent variables.

Equation for linear regression: $\mathbf{y = ax + b + \varepsilon}$ *(a = slope, b=constant, ε = noise)*

Equation for nonlinear regression: $\mathbf{y = f(x,\beta) + \varepsilon}$, where:

- *x = a vector of p predictors*
- *β = a vector of k parameters*
- *f = a known regression function*
- *ε = an error term*

Transformation of Non-Linear to Linear Models

Some non linear regression problems can be moved to linear models by suitable transformations of the model formulations. Four common transformations to induce linearity are

1. Logarithmic transformation
2. Square root transformation
3. Inverse transformation
4. Square transformation.

Non-Linear Regression Models

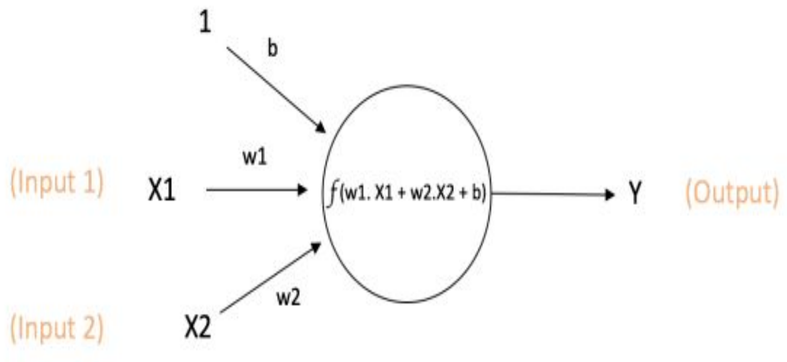
There are numerous regression models that are inherently nonlinear in nature, we will focus on following four models due to their popularity:

1. Neural Networks
2. Support Vector Machines (SVMs)
3. Multivariate Adaptive Regression Splines (MARS)
4. K-Nearest Neighbours (KNNs)

Neural Networks

Neural networks are powerful forecasting methods that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

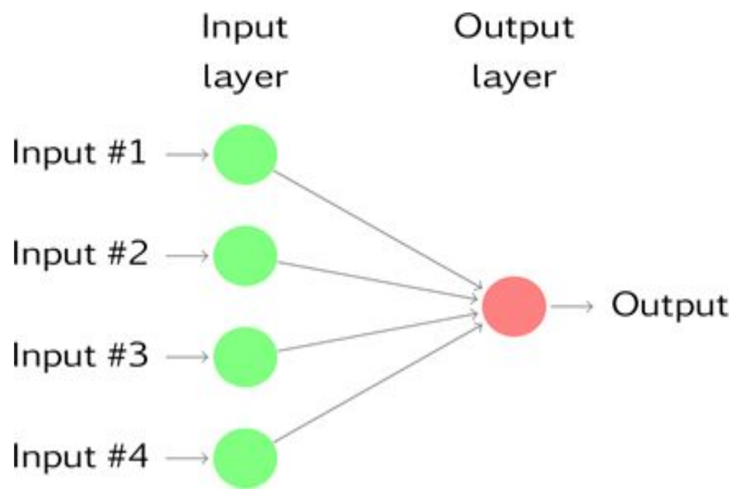
Neurons: Basic unit of neural network also known as node
Layers: A group of neurons. There is one input layer, one output layer, and one or more hidden layers.
Activation Function: Used to introduce non-linearity to model.
Learning Rate: The amount by which weights are changed and is used during backpropagation.
Gradient Descent: It is an optimization algorithm used for finding weights of ML algorithm
Backpropagation: Training method for feedforward neural networks used by neurons to adapt their weight to acquire new knowledge



Output of neuron $= Y = f(w1.X1 + w2.X2 + b)$

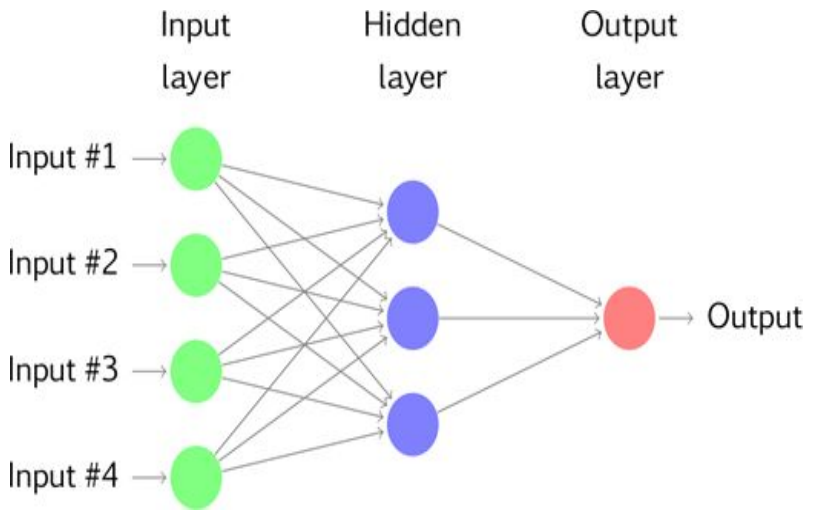
Neural Networks

The simplest neural network contains no hidden layer and are equivalent to linear regression. Four inputs in the figure are the predictors and coefficients attached to these predictors are called weights. The forecasts are obtained by linear combination of inputs.



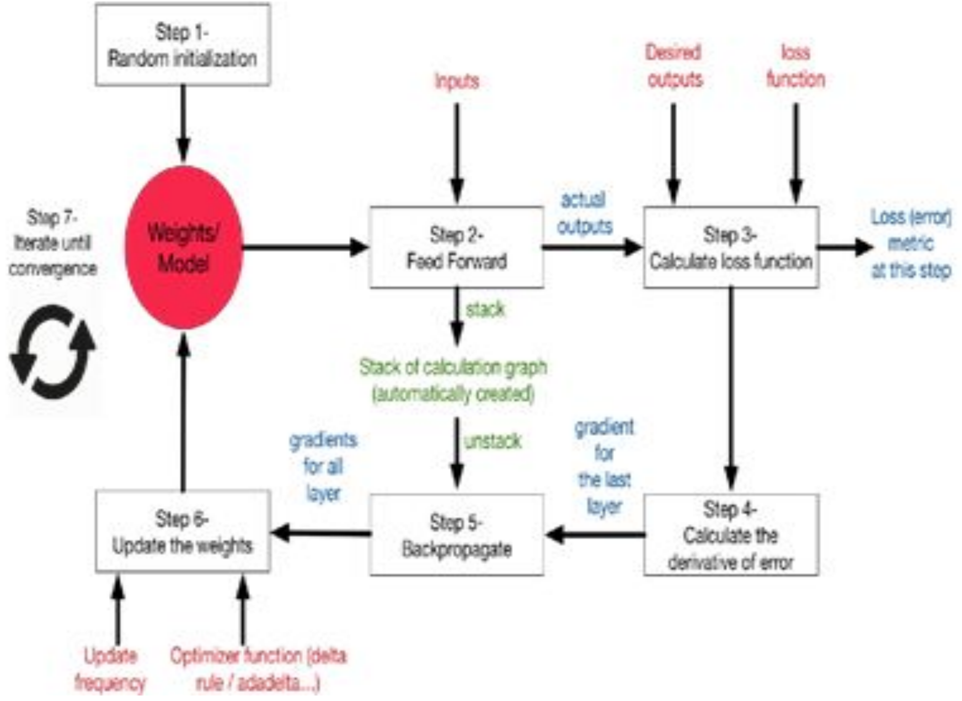
Neural Networks

Once we add an intermediate layer with hidden neurons, the neural network becomes non-linear. Each layer of nodes receives inputs from the previous layers. The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output.



Neural Networks

Mechanism : First step consists of random initialization of weights,we then pass the input into the model and output is generated straightforward through a process called forward propagation. At that point we have our actual and desired output and we compare the results and neural network learns from this step. Cost function is minimized to generate accurate results. Derivative of cost function assist the network in decreasing the error and updating the weights. Batches of data are passed iteratively through these steps to update the weights .


Step 1- Random initialization
Inputs
Desired outputs
loss function
Step 7- Iterate until convergence
Weights/ Model
Step 2- Feed Forward
actual outputs
Step 3- Calculate loss function
Loss (error) metric at this step
stack
Stack of calculation graph (automatically created)
unstack
gradients for all layer
gradient for the last layer
Step 6- Update the weights
Step 5- Backpropagate
Step 4- Calculate the derivative of error
Update frequency
Optimizer function (delta rule / adadelta...)

Support Vector Machines (SVMs)

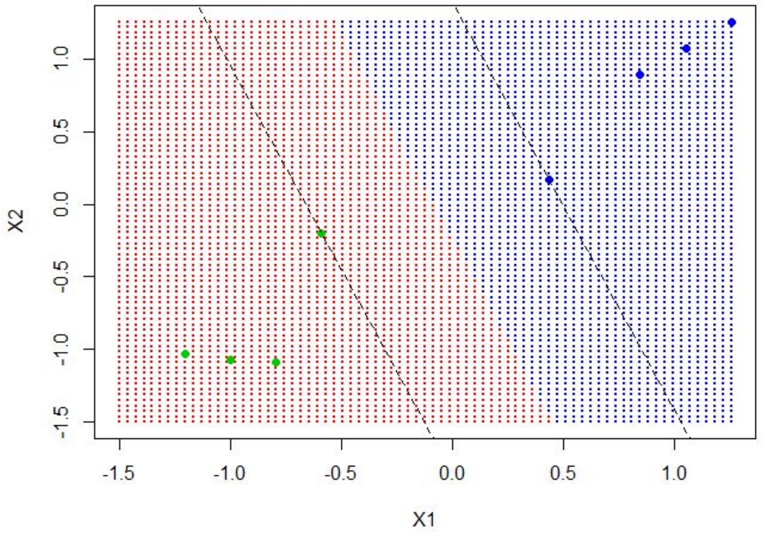
- SVMs originated as linear classifiers but can be extended to be nonlinear regressors
- The linear SVM has the familiar paramaization:

$$y = X\beta + \beta_0$$

- Y can be used to represent either a class or a continuous variable
- Computation of parameters and extensibility is a bit different from OLS

Linearly separable example

- No points in the margins
- Choice of separation made to maximize the size of the margins
- Only the two points on the margins contribute directly to the formulation of the boundary
- **Points on the dotted lines are the support vectors**



Computation

- Prediction only depends on the points with positive alpha (support vectors)

$$L = \frac{1}{2}||\beta||^2 + \sum \alpha_i [y_i(\beta \cdot x_i + b_0) - 1]$$

$$\beta = \sum \alpha_i y_i x_i$$

$$y = \sum \alpha_i y_i \langle x, x_i \rangle + \beta_0$$

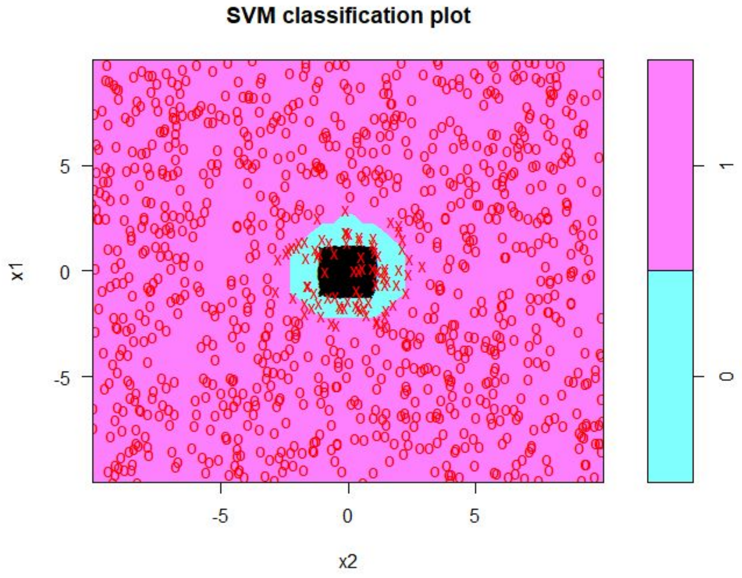
Kernel Motivation

Q: How do you find a line that separates points in the unit square from points outside it in 2D?

A: Map each point to 3D!

$$\Re^2 \rightarrow \Re^3$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix}$$



“Kernel Trick”

Q: Imagine h(x) is a much more complex transformation than last slide. What if we already have 5 dimensions and need to go to 1000?

A: Instead of explicitly mapping each x vector to 1000 dimensional space, we can compute the dot product right in the comfort of 5 dimensional space.

$$y = \sum \alpha_i \, y_i \langle h(x), h(x_i) \rangle + \beta_0$$

$$y = \sum \alpha_i \, y_i K(x, x_i) + \beta_0$$

$$Polynomial: \; K(x, x_i) = (1 + \langle x, x_i \rangle)^d$$

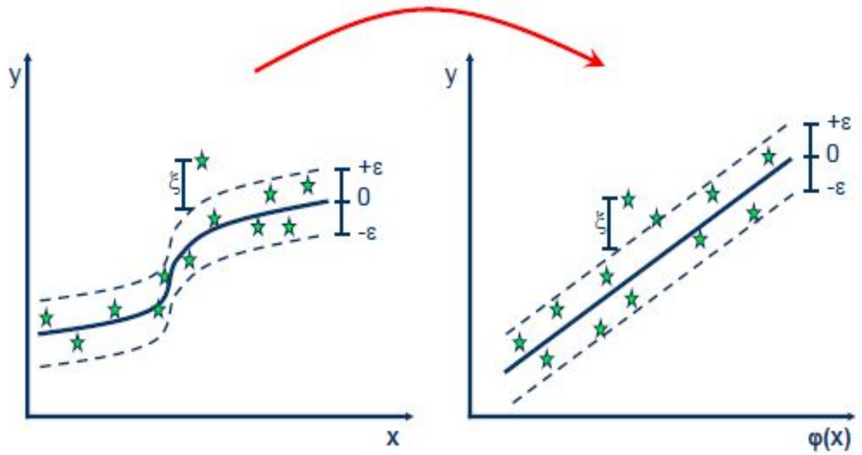
$$Radial \; Basis: \; K(x, x_i) = e^{-\lambda \|x - x_i\|^2}$$

$$Sigmoid: \; K(x, x_i) = tanh(k_1 \langle x, x_i \rangle + k_2)$$

SVM Regression

Minimize: $\sum V(y_i - f(x_i)) + \frac{1}{2} \|\beta\|^2$

- The function V reduces errors within a set range to zero (error band)
- Prediction function depends only on the values of y and observations (support vectors) outside the error band
- Kernel used in the same manner as classifier



Multivariate Adaptive Regression Splines

- In 1991, MARS was introduced by Jerome Friedman
- MARS produces continuous models for discrete data that can have multiple partitions and multilinear terms
- MARS provides a great stepping stone into nonlinear modeling and tends to be fairly intuitive due to being closely related to multiple regression techniques. They are also easy to train and tune
- The open-source version of MARS is called EARTH
- The MARS model for a dependent (outcome) variable *y*, and *M* terms , can be summarized in the following equation: $f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m \lambda_m (X)$
- A quick demo - http://rpubs.com/hovig613/484265

MARS - Advantages vs. Disadvantages

Advantages:

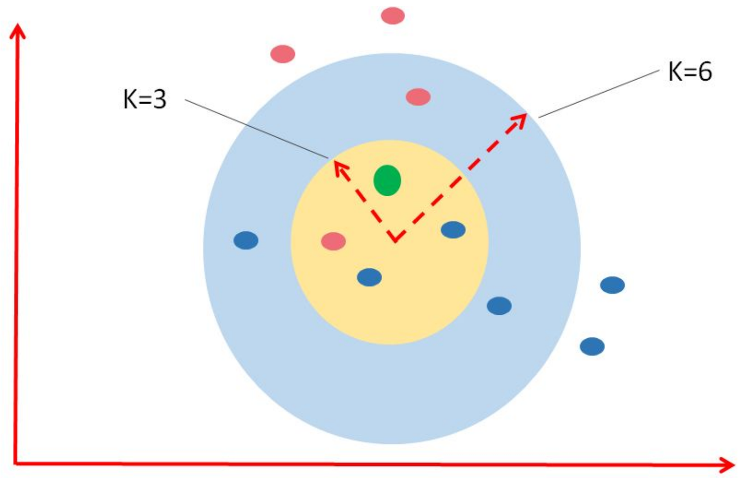
- Accurate if the local linear relationships are correct.
- Quick computation.
- Can work well even with large and small data sets.
- Provides automated feature selection.
- The non-linear relationship between the features and response are fairly intuitive.
- Can be used for both regression and classification problems.
- Does not require feature standardization.

Disadvantages:

- Not accurate if the local linear relationships are incorrect.
- Typically not as accurate as more advanced non-linear algorithms (random forests, gradient boosting machines).
- The **earth** package does not incorporate more advanced spline features (i.e. Piecewise cubic models).
- Missing values must be pre-processed.

K-Nearest Neighbours (KNNs)

K-nearest neighbors, or K-NN, is one of the simplest machine learning algorithms used for both classification and regression problem types.



K-Nearest Neighbours (KNNs) - Approach

The basic KNN method depends on how the user defines distance between samples

- Euclidean distance (i.e., the straight-line distance between two samples) is the most commonly used metric

$$\left(\sum_{j=1}^{P} (x_{aj} - x_{bj})^2 \right)^{\frac{1}{2}}$$

Euclidean distance

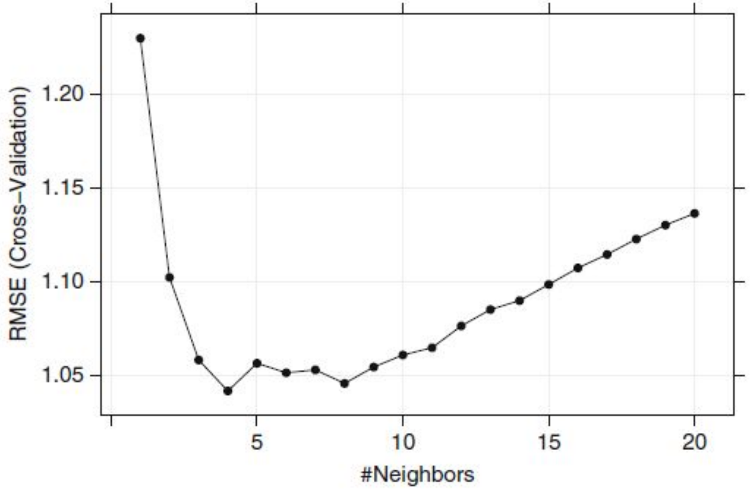
$$\left(\sum_{j=1}^{P} |x_{aj} - x_{bj}|^q \right)^{\frac{1}{q}} \quad \begin{cases} q=1\text{ , Manhattan distance} \\ q=2\text{ , Euclidean distance} \end{cases}$$

Minkowski distance

** **Xa** and **Xb** are two individual samples*

K-Nearest Neighbours (KNNs) - Model

K values and Model Overfitting and Underfitting



K small value - Overfit
K large value - Underfit
The optimal number of neighbors is 4

K-Nearest Neighbours (KNNs) - Risk and Mitigation

Issue/ Risk		Resolution/ Mitigation
Predictors with largest scale in dataset - Introduce bias	⇨	Preprocessing - All predictors be centered and scaled
Missing values - Unable to compute the distance	⇨	Preprocessing - Impute the missing data using a na¨ıve estimator such as the mean of the predictor

References

- Notes on the **earth** package by Stephen Milborrow: http://www.milbo.org/doc/earth-notes.pdf
- EarthNet Online Neural Network Glossary: https://envisat.esa.int/handbooks/meris/CNTR4-2-5.html
- Forecasting Principles and Practices : https://otexts.com/fpp2/nnetar.html
- Quick Intro to Neural Networks: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/
- Lagrange multipliers: https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/lagrange-multipliers-and-constrained-optimization/v/the-lagrangian?modal=1
- SVM Math: https://www.youtube.com/watch?v=_PwhiWxHK8o
- SVM Graph: https://www.saedsayad.com/support_vector_machine_reg.htm
- SVM Math: https://www.youtube.com/watch?v=mTyT-oHoivA&t=276s
- SVM Graph Code: https://www.datacamp.com/community/tutorials/support-vector-machines-r