Far East Journal of Theoretical Statistics

Published Online: October 2013
Available online at http://pphmj.com/journals/fjts.htm
Volume 44, Number 2, 2013, Pages 93-120

SUGGESTED STATISTICAL MODEL TO DETERMINE THE IMPORTANT VARIABLES WHICH CAUSE THE MOST FREQUENT CRIMES IN EGYPTIAN SOCIETY

Moustafa Galal Moustafa, Medhat M. A. Abdelaal and Mahmoud Mohamed Eisa

Mathematics and Statistics Department
Faculty of Commerce
Ain Shams University
Egypt
e-mail: mostafa.galal.m@gmail.com
medhatal@hotmail.com
maheissa@gmail.com

Abstract

This paper aims to suggest a statistical model to estimate the number of crimes and to determine the most important variables and factors that have positive impact on increasing number of drug and theft crimes. These crimes are the most frequent in the Egyptian society. The study covers the period from 2000 to 2010 monthly data. The dependent variable is the number of offenses for which the judgment was issued for each crime. The independent variables were formed in five groups, each group contains several independent variables. These groups are (1) crime type, (2) offender career, (3) offender gender, (4) offender age and (5) offender educational level. Four statistical models were applied: multiple linear regression, multilayer perceptron neural

Received: May 30, 2013; Accepted: June 28, 2013
2010 Mathematics Subject Classification: 62Pxx.
Keywords and phrases: multilayer perceptron neural network, regression support vector machine, general regression neural network, factor analysis.

network, regression support vector machine and general regression neural network for each independent variables group via the dependent variable for each crime. Also, the factor analysis has been applied for each independent group, then these factors used as independent variables via the dependent variable using the four models. The results showed that the recommended model for the drug crime was RSVM using the factors and GRNN for theft crime using the factors.

1. Introduction

This paper aims to suggest a statistical model to estimate the number of crimes and to determine the most important variables and factors that have positive impact on increasing number of drug and theft crimes. The problem of the research is to identify the key variables and factors causing crime rates increase, where the results of this study can be used as an aid to the security bodies to prevent crime before it happens. The study data is the number of cases of drug and theft crimes because these two crimes are the most frequent and danger crimes in the Egyptian society.

The source of data is the monthly reports of public security in Arab Republic of Egypt, which issued by the Department of Public Security, Ministry of the Interior. These data are monthly time series of number of drug and theft crimes at the level of Arab Republic of Egypt from 2000 to 2010 for drug crimes, but for theft data crime the monthly time series data from 2001 to 2010.

Drug crime definition

Drug crime is the umbrella term used to describe different offenses involving controlled substances. Each state and the federal government have enacted laws against unlawful possession, use, distribution, or production of certain drugs. These include cocaine, heroin, marijuana and amphetamines. Such legislations aim to reduce illegal drug use, and reduce drug related crimes.

Theft crime definition

A criminal act in which property belonging to another is taken without

that person's consent. The term theft is sometimes used synonymously with larceny. Theft, however, is actually a broader term, encompassing many forms of deceitful taking of property, including swindling, embezzlement, and false pretenses. Some states categorize all these offenses under a single statutory crime of theft.

2. Study Techniques

Two types of techniques have been investigated, parametric analysis such as multiple linear regression (MLR) and non-parametric analysis such as multilayer perceptron neural network (MLP), regression support vector machine (RSVM) and generalized regression neural networks (GRNN).

2.1. Neural network

Artificial neural networks (ANN) are collections of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. The key element of the ANN paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements that are analogous to neurons and are tied together with weighted connections that are analogous to synapses.

When the network is executed (used), the input variable values are placed in the input units, and then the hidden and output layer units are progressively executed. Each of them calculates its activation value by taking the weighted sum of the outputs of the units in the preceding layer, and subtracting the threshold. The activation value is passed through the activation function to produce the output of the neuron. When the entire network has been executed, the outputs of the output layer act as the output of the entire network [7].

2.1.1. Multilayer perceptron

The multilayer perceptron (MLP) is one of the most widely implemented neural network topologies. Generally speaking, for multiple regression model, the MLP with two hidden layers is a universal pattern. When the

weights are properly normalized and the output classes are normalized, the MLP achieves the performance of the maximum a posteriori receiver, which is optimal from a regression model. In terms of mapping abilities, the MLP is believed to be capable of approximating arbitrary functions. This has been important in the study of nonlinear dynamics, and other function mapping problems. MLPs are normally trained with the BP algorithm [14]. In fact the renewed interest in ANN was in part triggered by the existence of BP. The BP rule propagates the errors through the network and allows adaptation of the hidden processing elements (PE).

Two important characteristics of the MLP are: its nonlinear PEs which have a nonlinearity that must be smooth (the logistic function and the hyperbolic tangent are the most widely used); and their massive interconnectivity, such as any element of a given layer feeds all the elements of the next layer [9].

The MLP is trained with error correction learning, which means that the desired response for the system must be known. Error correction learning works in the following way: from the system response at *i*th PE at *n*th iteration, $Y_i(n)$ and the desired response $d_i(n)$ for a given input pattern an instantaneous error $e_i(n)$ can be expressed as follows:

$$e_i(n) = d_i(n) - Y_i(n). \tag{1}$$

Using the theory of gradient descent learning, each weight in the network can be adapted by correcting the present value of the weight with a term that is proportional to the present input and error at the weight, which can be expressed as follows:

$$w_{ij}(n+1) = w_{ij}(n) + \eta\delta_i(n)x_j(n). \tag{2}$$

The local error $\delta_i(n)$ can be directly computed from $e_i(n)$ at the output PE or can be computed as a weighted sum of errors at the internal PEs. The constant η is called the *step size*. This procedure is called the *BP algorithm*. BP computes the sensitivity of a cost functional with respect to each weight in the network, and updates each weight proportional to the sensitivity. The

beauty of the procedure is that it can be implemented with local information and requires just a few multiplications per weight, which is very efficient. Because this is a gradient descent procedure, it only uses the local information so can be caught in local minima. Moreover, the procedure is essentially noisy since we are using a poor estimate of the gradient, causing slow convergence.

Momentum learning is an improvement to the straight gradient descent in the sense that a memory term (the past increment to the weight) is used to speed up and stabilize convergence. In momentum learning the equation to update the weights becomes as follows:

$$w_{ij}(n+1) = w_{ij}(n) + \eta\delta_i(n)x_j(n) + \alpha(w_{ij}(n) - w_{ij}(n-1)), \qquad (3)$$

where α is the momentum factor. Normally α should be set between 0.1 and 0.9. Training can be implemented in two ways: either we present a pattern and adapt the weights (on-line training), or present all the patterns in the input file (an epoch), accumulate the weight updates, and then update the weights with the average weight update. This is called *batch learning*. They are theoretically equivalent, but the former sometimes has advantages in tough problems.

2.1.2. Back-propagation

To start BP, load an initial value for each weight (normally a small random value), and proceed until some stopping criterion is met. The three most common are: to cap the number of iterations, to threshold the output mean square error and checking the progress of learning is fundamental in any iterative training procedure. The learning curve illustrates how the mean square error changes with the training iteration. When the learning curve is flat, the step size should be increased to speed up learning. On the other hand, when the learning curve fluctuates up and down the step size should be decreased. In the extreme, the error can go little by little up, showing that learning is unstable. At this point the network should be reset. When the learning curve stabilizes after many iterations at an error level that is not acceptable, more hidden PEs or more hidden layers can be trained [14].

The MLP model using the BP algorithm is one of the well-known neural network; which consist of sets of nodes arranged in multiple layers with connections only between nodes in the adjacent layers by weights. The layer where the inputs information is presented is known as the input layer. The layer where the processed information is retrieved is called the output layer. All layers between the input and output layers are known hidden layers. For all nodes in the network, except the input layer nodes, the total input of each node is the sum of weighted outputs of the nodes in the previous layer. Each node is activated with the input to the node and the activation function of the node (equation (4)). The input and output of the *i*th node (except for the input layer) in a MLP mode, according to the BP algorithm computations can be expressed as follows:

$$\text{Input: } X_i = \sum W_{ij} O_j + B_i, \tag{4}$$

$$\text{Output: } O_i = F(X_i), \tag{5}$$

where

W_{ij} : the weight of the connection from node *i* to node *j*,

B_i : the numerical value called *bias*,

F : the activation function.

The sum in equation (4) is over all nodes *j* in the previous layer. The output function is a nonlinear function which allows a network to solve problems that a linear network cannot solve. In this study the sigmoid function given in equation (6) is used to determine the output state:

$$F(X_i) = \frac{1}{1 + \exp(-X_i)}. \tag{6}$$

BP learning algorithm is designed to reduce an error between the actual output and the desired output of the network in a gradient descent manner. The summed squared error (SSE) is defined as:

$$SSE = \frac{1}{2}\left[\sum_p \sum_i O_{pi} - t_{pi}\right]^2, \tag{7}$$

where p indexes the all training patterns and i indexes the output nodes of the network. O_{pi} and T_{pi} denote the actual output and the desired output of node, respectively when the input vector p is applied to the network [7]. A set of representative input and output patterns is selected to train the network. The connection weight W_{ij} is adjusted when each input pattern is presented. All the patterns are repeatedly presented to the network until the SSE function is minimized and the network learns the input patterns. The following iterative weight update rule can be used for the applications of the gradient:

$$\Delta W_{ij}(n+1) = \eta(\delta_i O_i + \alpha \Delta W_{ij}(n)), \tag{8}$$

where

Δ : the learning factor,

α : the momentum factor,

δ_i : the node error, for output node i is then given as

$$\delta_i = (t_i - O_i) O_i (1 - O_i). \tag{9}$$

The node error at an arbitrary hidden node is

$$\delta_i = O_i (1 - O_j) \sum_k \delta_k W_{ki}. \tag{10}$$

2.2. Regression support vector machine technique

One of the most useful applications of statistical analysis is the development of a model to explain the relationship between the variables. Many types of models have been developed, including regression support vector machines. Methods for analyzing and modeling data can be divided into two groups; “supervised learning” and “unsupervised learning”. Supervised learning requires input data that has both independent variables and a dependent variable (target) whose value is to be estimated. By various means, the process “learns” how to model predict the value of the target variable based on the predictor variables. Regression analysis is an example

of supervised learning. If the goal of an analysis is to predict the value of some variable, then supervised learning is recommended approach.

Unsupervised learning does not identify a dependent variable (target), but rather treats all of the variables equally. In this case, the goal is not to predict the value of a variable but rather to look for patterns, groupings or other ways to characterize the data that may lead to understanding of the way the data relate. Cluster analysis, correlation, factor analysis (principle components analysis) and statistical measures are examples of unsupervised learning. Regression support vector machines (RSVM) is one of the best modeling methods.

In the manner of speaking of RSVM literature, a predictor variable is called an *attribute*, and a transformed attribute that is used to define the hyperplane is called a *feature*. The task of choosing the most suitable representation is known as feature selection. A set of features that describes one case (i.e., a row of predictor values) is called a *vector*. So the goal of RSVM modeling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyperplane are the support vectors.

To illustrate RSVM, let us assume that there is a linear relationship with N observations. The independent variable is x_i and the dependent variable is y_i given that $i = 1, 2, 3, ..., N$. The goal of RSVM is to produce a linear function which can make the best fit of the dependent variable y_i. The linear function can be expressed as follows:

$$y = f(x) = \langle b \bullet x \rangle + a, \tag{11}$$

where b and a are the regression parameters and $\langle b \bullet x \rangle$ is the dot product of b and x.

The dot product is the result of multiplying all the components of two vectors together and adding the results. The dot product of two vectors $b =$

$[b_1, b_2, ..., b_n]$ and $x = [x_1, x_2, ..., x_n]$ is defined as:

$$b \bullet x = \sum_{i=1}^{N} b_i x_i = b_1 x_1 + b_2 x_2 + \cdots + b_N x_N .$$

The optimum regression function can be found by minimizing the following function [4]:

$$M(b, S) = 0.5 \| b \|^2 + C \sum_{i=1}^{N} (S_i^- + S_i^+). \quad (12)$$

The constraints can be as follows:

$$y_i - \langle b, x_i \rangle - a \leq \varepsilon + S_i,$$

$$\langle b, x_i \rangle + a - y_i \leq \varepsilon + S_i^*,$$

$$S_i, S_i^* \geq 0,$$

where C is a constant greater than zero, and can be used to determine the trade off the smoothness of f and the amount up to which deviations larger than ε are accepted.

S_i^- and S_i^+ are two slack variables, which can be used to represent the upper and lower constraints of the regression function.

$\| b \|$ is norm of b. The norm is the square root of the inner product of the vector and itself, which can be as follows: $\| b \| = \sqrt{b_1^2 + b_2^2 + \cdots + b_N^2}$.

To find the optimum solution of M function, loss function must be determined. RSVM based on a loss function. Loss function is a function shows the maximum allowed deviation of the predicted values from the observed one. The most recommended loss functions are four. These four functions are: Huber, ε-insensitive, quadratic and Laplace. Figure 1 shows the difference between the four types of loss functions.

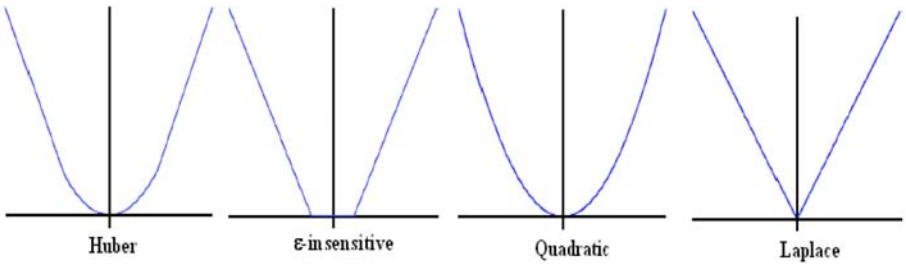

Figure 1. The four types of loss functions.

The first loss function is Huber loss function; it is a robust loss function that has optimal properties when the underlying distribution of the data is unknown. The second one is ε-insensitive loss function; it is an approximation to Huber's loss function but it can reduce sensitivity to the outliers. The third one is quadratic loss function; it corresponds to the predictable least squares error measure. The fourth one is Laplace loss function; it is less sensitive to outliers than the quadratic loss function.

In this paper, the ε-insensitive loss function was selected. For RSVM model which depends on the ε-insensitive loss function; the difference between the estimated values and the observed values of the dependent variable y_i can be calculated. If the difference is less than ε, the regression function is considered to be most popular and accurate [19]. The ε-insensitive loss function can be expressed as follows:

$$|S|_\varepsilon = \begin{cases} 0, & \text{if } |S| < \varepsilon, \\ |S| - \varepsilon, & \text{otherwise.} \end{cases} \tag{13}$$

Figure 2 shows that the difference between the RSVM and ordinary least square (OLS) estimation. In the left figure there are two bounds around the regression line and if the error between the prediction and the actual value is less than ε, the error is ignored geometrically, this can be thought of as fitting a tube of width 2ε to the data. The right figure shows traditional approach of OLS estimation which based on minimizing the squared error.

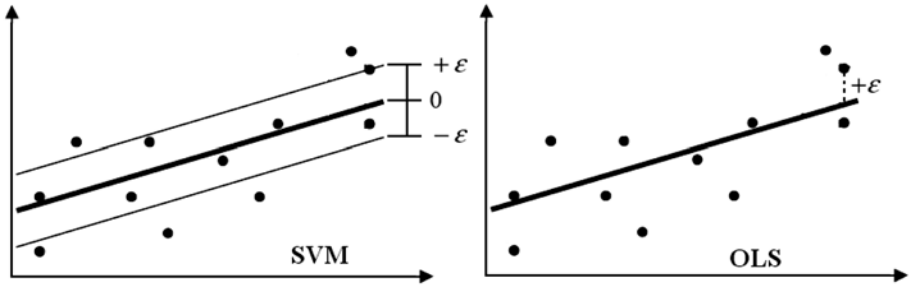

Figure 2. The loss function of RSVM and OLS.

Most modeling techniques are trying to find the best fit between the observed and predicted values, however, RSVM's ε-insensitive loss function focuses on optimizing a bound around the regression function, thus making it more strong against the outliers.

To find the solution of equation (12), Lagrange optimization must be used, the solution to this optimization problem can be expressed as follows:

Minimize the target function as follows:

$$\varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) + 0.5 \sum_{\substack{i=1 \\ j=1}}^{N} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i - \alpha_j) + \sum_{i=1}^{N} y_i (\alpha_i - \alpha_i^*). \tag{14}$$

The constraints can be as follows:

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0, \qquad 0 \le \alpha_i, \alpha_i^* \le C, \tag{15}$$

where α_i and α_i^* are the Lagrange multipliers.

RSVM models are built around a kernel function that transforms the input data into an *n*-dimensional space where a hyperplane can be constructed to partition the data. There are four kernel functions, linear, polynomial, radial basis function (RBF) and sigmoid (S-shaped). There is no way in advance to know which kernel function will be best for an application, but the RBF function has been found to do best job in the majority of cases.

The RBF kernel non-linearly maps samples into a higher dimensional space, so it can handle nonlinear relationships between target categories and predictor attributes; a linear basis function cannot do this. Furthermore, the linear kernel is a special case of the RBF. A sigmoid kernel behaves the same as a RBF kernel for certain parameters. The RBF function has fewer parameters to tune than a polynomial kernel, and the RBF kernel has less numerical difficulties.

This is the case of linear regression, but to illustrate the regression case of non-linear the data must be firstly linearized by mapping it into a higher dimensional space, called "*feature space*", by using kernel functions, that linear regression functions can be applied. The most recommended kernel function is the radial basis function (RBF) kernels [2]. The RBF kernel is defined as: $K(x,\ y) = ye^{-y(x-y)^2}$, where γ is a kernel parameter.

Insert kernel function in the previous model (14), this model can be adjusted as follows:

Minimize:

$$\varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) + 0.5 \sum_{\substack{i=1 \\ j=1}}^{N} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i \bullet x_j) + \sum_{i=1}^{N} y_i (\alpha_i - \alpha_i^*). \tag{16}$$

The same constraints can be used.

2.3. Generalized regression neural networks

The general regression neural network (GRNN) is one of the most popular neural networks. They have a parallel structure where the learning is one fold that is input to structure to output there is no iterative learning present such as in the case of MLP making them fast to some extents. Also, they perform well on noisy data then say Back-propagation Neural Networks (BPNN) if the available data is large enough. GRNN is also very unswerving and as the size of the dataset increases the error approaches towards zero.

The GRNN infrastructure consists of four layers input, hidden, summation and output layer.

- The input layer merely transports the data attributes to the next layer in a parallel archetype.
- The second layer consists of all the training samples.
- In the summation layer the summation units or neurons perform a dot product on the attributes of the weight vector of the second layer.
- Then in the output layer the respective local outputs are divided to get the predictions.

Input layer

There is one neuron in the input layer for each predictor variable. The input neurons (or processing before the input layer) standardize the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.

Hidden layer

This layer has one neuron for each case in the training data set. The neuron stores the values of the predictor variables for the case along with the target value. When presented with the *x* vector of input values from the input layer, a hidden neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the *RBF* kernel function using the sigma value(s). The resulting value is passed to the neurons in the pattern layer.

Pattern layer (summation layer)

There are only two neurons in the pattern layer. One neuron is the denominator summation unit and the other is the numerator summation unit. The denominator summation unit adds up the weight values coming from each of the hidden neurons. The numerator summation unit adds up the weight values multiplied by the actual target value for each hidden neuron.

Output layer

The decision layer divides the value accumulated in the numerator

summation unit by the value in the denominator summation unit and uses the result as the predicted target value.

Basically, there are two types of ANN's those that predict categorical output, and those that predict continuous output. The GRNN predicts continuous outputs. Two main functions are required of GRNN nodes. Those functions can be used to calculate the difference between all pairs of input pattern vectors, and to estimate the probability density function of the input variables. Calculation of the difference between input vectors is the simple Euclidean distance between data values in attribute space. Weighing the calculated distance of any point by the probability of other points occurring in that area yields a predicted output value. With those two tasks accomplished it is a rather straightforward process of predicting the output.

Determining the joint probability density function $PDF = f\,XY(x,\ y)dy$ of the variables is the main difficulty in utilizing equation:

$$E_{Y|X}(X) = \frac{\int y * f_{XY}(x,\ y)dy}{\int f_{XY}(x,\ y)dy}. \tag{17}$$

The new x value can be expressed as follows:

$$D(x,\ x_i) = \sum_{j=1}^{p}\left(\frac{x_j - x_{ij}}{\delta_j}\right)^2, \tag{18}$$

where x is the input vector, x_i is the ith case vector, x_j is the jth data value in the input vector, x_{ij} is the jth data value in the ith case vector, and δ_j is the smoothing factor (Parzen's window) for the jth variable.

The predicted y is reasonable because it is similar to the y values which have x values similar to the new x value. Weights, or levels of similarity, are assigned to each point as determined by their distance from the new x, little weight is given to those points that have x values very different from that of the new x value because it is not very probable that the predicted y is affected by the more distant points. Calculating all the y values for the entire range of

x values would yield a curve. No priori assumption was made about the shape of that curve, nor was one made about the distribution of the data. We did, however, visualize the joint probability of *x* and *y* values.

With most Neural Networks, data are segregated into two classes: training data and validation data. The training data set are the data to which the weights or coefficients are initially applied. In regression, training data are equivalent to the data used for estimation. For neural nets to converge on a generalizable solution through iteration there must be a test set, against which prediction accuracy is compared. The test set may be randomly extracted from the larger sample or, in the case of time series, may consist of the most recent few events.

The leave-one-out (LOO) method of evaluating sigma values during the optimization process. This measures the error by building the model with all training rows except for one and then evaluating the error with the excluded row. This is repeated for all rows, and the error is averaged.

GRNN interpretation

Interpretation of GRNN analysis takes on three main forms - comparison of each variable's final smoothing factor (sigma), an overall model goodness of fit measure (*R*-squared), and visual analysis [21]. As stated earlier, the smoothing factor is a measure of each sample data point's sphere of influence. When sample data points for one variable have greater spheres of influence than sample data points for a second variable, the first variable is said to be more important in predicting an outcome than the second variable. Examination of the relative ranking of sigma weights reveals which input variables are most important in determining the output. Such an examination is analogous to comparing the standardized beta values in OLS regression.

When comparing GRNN results to OLS regression results, analyzing the visual relationship between predictions is also very useful. OLS regression assumes a linear relationship between data values - the GRNN does not. Likewise, other types of relationships can be modeled with regression techniques (i.e., exponential, logarithmic, polynomial), but the type of relationship must be assumed prior to running the regression. The GRNN, on

the other hand, fits the relationship between variables regardless of the form of their relationship because its predictions are a result of the joint probability density function between variables. No prior assumption about that relationship is made. Thus, plotting data points, regression lines, and GRNN lines makes possible a visual determination of which technique best fits the data. While in some cases the GRNN will not improve upon standard regression predictions, it nearly always will when the relationships are non-linear.

The final method of analyzing output from the GRNN is to calculate the percent of the variance in the output variable that is explained by the input variables (*R*-squared). *R*-squared in the GRNN is the same as in OLS regression analysis - it is the coefficient of multiple determination. In many cases *R*-squared is the final measure of which model, the GRNN or OLS regression, is the better predictor of the output. Also, multiple GRNN analyses may be run in which the same variables are used, but their scale of analysis is varied. In that situation the model whose *R*-squared value is highest represents the model with the most appropriate combination of scales.

GRNN limitations

Even though the GRNN is in many cases a better predictor than OLS regression, numerous statistics that OLS regression calculates to aid model interpretation are not present in the GRNN. OLS regression produces unstandardized beta values that represent a unit change in the output, given a unit change in the input. Since GRNN function approximation can be highly non-linear, calculation of such a statistics is unreasonable if not impossible. To a certain point, however, visual analysis of predicted values substitutes for the linear mathematical understanding given in OLS regression. OLS regression also provides significance levels for all of its statistics, another descriptor absent from GRNN analysis. Possible smoothing factors range from 0 to infinity, depending on variable importance and sample size. Without such significance levels, or confidence intervals, it is not possible to know how a small change in the smoothing factors would affect the predicted

value. A standardized method of assessing confidence intervals associated with each variable would certainly make GRNN interpretation more robust.

Generalized regression neural networks (GRNN) is a special case of radial basis networks (RBN). Compared with its competitor, e.g. standard feed forward neural network, GRNN has several advantages. First of all, the structure of a GRNN is relatively simple and static with 2 layers, namely pattern and summation layers. Once the input goes through each unit in the pattern layer, the relationship between the input and the response would be "memorized" and stored in the unit. As a result, # of units in the pattern layer is equal to # of observations in the training sample. In each pattern unit, a Gaussian PDF would be applied to the network input such that

$$Theta = EXP[-0.5(X-u)'(X-u)/(Sigma^2)], \tag{19}$$

where Theta is the output from pattern units, X is the input, u is training vector stored in the unit, and Sigma is a positive constant known as "spread" or "smooth parameter". Once Theta is computed, it is passed to the summation layer to calculate $Y/X = SUM(Y * Theta)/SUM(Theta)$, where Y/X is the prediction conditional on X and Y is the response in the training sample. In addition to the above, other benefits of GRNN are listed below [21]:

(1) The network is able to learning from the training data by "1-pass" training in a fraction of the time it takes to train standard feed forward networks.

(2) The spread, Sigma, is the only free parameter in the network, which often can be identified by the V-fold or split-sample cross validation.

(3) Unlike standard feed forward networks, GRNN estimation is always able to converge to a global solution and will not be trapped by a local minimum.

RBF networks are very similar to GRNN networks. The main difference is that GRNN networks have one neuron for each point in the training file, whereas RBF networks have a variable number of neurons that is usually

much less than the number of training points. For problems with small to medium size training sets, GRNN networks are usually more accurate than RBF networks, but GRNN networks are impractical for large training sets.

GRNN networks have advantages and disadvantages compared to multilayer perceptron networks:

- It is usually much faster to train a GRNN network than a MLP network.
- GRNN networks often are more accurate than MLP networks.
- GRNN networks are relatively insensitive to outliers (wild points).
- GRNN networks are slower than MLP networks at classifying new cases.
- GRNN networks require more memory space to store the model.
- GRNN networks are very similar to RBF networks.

3. Data

The study variables of drugs and theft crimes were formed in five groups of independent variables. Each group contains several independent variables. These groups are (1) crime type, (2) offender career, (3) offender gender, (4), offender age and (5) offender educational level. The dependent variable is the number of offenses for which the judgment was issued for drug crimes (YD), and theft crimes (YT).

Independent variables

The crime type of drug is different than the crime type of theft, while the rest of the four independent groups are same for both drug and theft. The independent variables of the first group are listed in the following table:

Table 1. The independent variables of the crime type group

Symbol	Drug	Symbol	Theft
XD_{11}	Several types of drugs	XT_{11}	Steal homes
XD_{12}	Materials affecting the mental state	XT_{12}	Auto theft
XD_{13}	Plantations crimes	XT_{13}	Theft shops
XD_{14}	Cocaine crimes	XT_{14}	Robbery
XD_{15}	Heroin offenses	XT_{15}	Theft of public funds
XD_{16}	Opium crimes	XT_{16}	Pickpocket crimes
XD_{17}	Cannabis offense		

The other four groups have the same definition of each variable and different symbol as listed below in the following table:

Table 2. The independent variables groups and the variables of each group

Groups	Drug	Theft	Definition
Offender career	XD_{21}	XT_{21}	Different occupations
	XD_{22}	XT_{22}	Businessman
	XD_{23}	XT_{23}	Students
	XD_{24}	XT_{24}	Doctors and pharmacists
	XD_{25}	XT_{25}	Unskilled workers
	XD_{26}	XT_{26}	Skilled workers
	XD_{27}	XT_{27}	Unemployed
Offender gender	XD_{31}	XT_{31}	Female
	XD_{32}	XT_{32}	Male
Offender age	XD_{41}	XT_{41}	Without the date of birth
	XD_{42}	XT_{42}	Age greater than 30 years old
	XD_{43}	XT_{43}	Age between 30:25
	XD_{44}	XT_{44}	Age between 25:21
	XD_{45}	XT_{45}	Age between 21:18
	XD_{46}	XT_{46}	Age less than 18 years old
Offender's level of education	XD_{51}	XT_{51}	Higher than average education
	XD_{52}	XT_{52}	Average education or less
	XD_{53}	XT_{53}	Illiterates

4. Analysis and Results

Four techniques have been illustrated in the previous section, these techniques have been carried out for each crime, and for each group of independent variables via the dependent variable. Also the factor analysis was applied for each independent group separately to determine the most important variables for each factor and then the four techniques were applied for these factors. The following table shows the most important independent variables for each group of the drug crime and the coefficient of each variable.

Table 3. The factors of each group for drug crime

	Group 1		Group 2		Group 3	Group 4	Group 5
Drug	Fac1D	Fac2D	Fac3D	Fac4D	Fac5D	Fac6D	Fac7D
XD_{11}	0.766						
XD_{12}		0.995					
XD_{13}		0.651					
XD_{17}	0.839						
XD_{23}			0.829				
XD_{24}			0.611				
XD_{25}				0.891			
XD_{26}				0.896			
XD_{27}			0.886				
XD_{31}					0.935		
XD_{32}					0.935		
XD_{42}						0.730	
XD_{43}						0.952	
XD_{44}						0.954	
XD_{45}						0.727	
XD_{46}						0.691	
XD_{51}							0.041
XD_{52}							–0.216
XD_{53}							0.747

The following table shows the most important independent variables for each group of the theft crime and the coefficient of each variable:

Table 4. The factors of each group for drug crime

	Group 1	Group 2	Group 3	Group 4	Group 5
Theft	Fac1T	Fac2T	Fac3T	Fac4T	Fac5T
XT_{11}	0.719				
XT_{15}	0.763				
XT_{16}	0.835				
XT_{21}		0.774			
XT_{25}		0.710			
XT_{26}		0.947			
XT_{27}		0.608			
XT_{31}			0.450		
XT_{32}			0.450		
XT_{42}				0.699	
XT_{43}				0.714	
XT_{44}				0.715	
XT_{53}					1.003

The four models MLR, MLP, RSVM and GRNN were applied to the data of drug crime and theft crime. These techniques used the independent variables of each group separately via the dependent variable. Also these techniques used the factors of factor analysis via the dependent variable. The following figure illustrates the steps of the analysis.

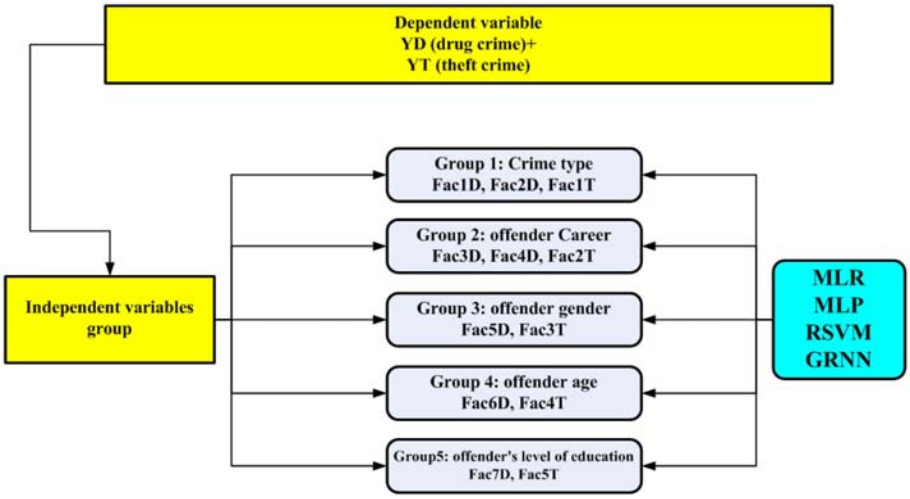

Figure 3. The sequences of the analysis.

Each group of independent variables has been used each technique, so for each group there were four models for each group (the total number of models were 20 models). In addition to the 20 models there are four models for the four techniques using the results of the factor analysis. The results of the analysis can be presented in four measurers' the variable importance, highest proportion of variance explained by model (R^2) for each model and for each group, the mean sum of squares (MSE) and mean absolute error (MAE).

Variable importance

There are three methods for computing the importance of predictor variables:

Use split information

Calculate the importance of each variable by adding up the improvement in classification gained by each split that used the predictor. Generally, this method produces good results, and it can be calculated quickly.

Type 1 margins

Firstly calculate the misclassification rate for the model using the actual data values for all predictors. Then for each predictor, it randomly permutes (rearranges) the values of the predictor and computes the misclassification rate for the model using the permuted values. The difference between the misclassification rate with the correctly ordered values and the misclassification rate for the permuted values is used as the measure of importance of the predictor. This method of calculating variable importance often is more accurate than calculating the importance from split information, but it takes much longer to compute because of the time required to permute the rows for each predictor.

Type 1 + 2 margins

Firstly calculate the importance using type 1 margins as described above. It then examines each data row and determines how many trees in the forest correctly voted for the row with the original data minus the number of trees that correctly voted for the row using the permuted data. The two measures of importance are then averaged. This is usually the most accurate measure of importance, but it is also the slowest to compute. In the case of a regression, this method is the same as the "Type 1 margins" method.

The third method is the most recommended one and it has been used in this research.

Proportion of variance explained by model

This is the proportion of the initial variance in the training data that is explained by the genetic expression programming (GEP) model. Where initial variance is the variance for the training data set using the mean value of the target variable as the predicted value for all rows:

$$\text{Var\%} = \frac{\text{Initial Variance} - \text{Variance}}{\text{Initial Variance}}.$$

Variance is computed as shown in the next section.

Mean squared error (MSE)

This is the mean value of the squared difference between the actual target value and the predicted target value which can be expressed as follows:

$$\text{Variance} = \sum_{i=1}^{n} (P_i - T_i)^2,$$

$$\text{Fitness} = \frac{1}{1 + \dfrac{\text{Variance}}{n}},$$

where P_i is the predicted value for row I and T_i is the actual target value; n is the number of rows in the training data set.

Mean absolute error

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. The equation is given in the library references. Expressed in words, the MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average.

4.1. The results of drug crime data

The following table shows the variable importance, highest proportion of variance explained by model (R^2) for each model and for each group, the mean sum of squares (MSE) and mean absolute error (MAE).

Table 5. The results of drug data using four techniques

Model	Variable importance	Var%	MSE	MAE	Group
RSVM	Fac1D, Fac2D	90.681	484.287	10.019	Factor analysis factors
MLR	XD_{17}	89.856	1039.276	17.512	Crime type
RSVM	XD_{27}, XD_{26}	89.367	24839.122	124.996	Career
RSVM	XD_{32}	82.454	24554.177	125.967	Gender
GRNN	XD_{41}	82.924	23026.663	118.212	Age
GRNN	XD_{52}, XD_{53}	83.190	22126.663	115.048	Level of education

The importance score for the most important predictor is scaled to a value of 100. Other predictors will have lower scores. The most important variables for each model and for each group of independent variables are listed in the above table. The independent variables with score less than 40% could be ignored in the analysis [22].

The highest values of proportion of variance explained by model, the smallest MSE and MAE were corresponding to SVM model for the factors of drug crime.

4.2. The results of theft crime data

The following table shows the variable importance, highest proportion of variance explained by model (R^2) for each model and for each group, the mean sum of squares (MSE) and mean absolute error (MAE).

Table 6. The results of theft data using four techniques

Model	Variables	R-squared%	MSE	MAE	Group
GRNN	Fac1T, Fac4T	89.888	0.258	0.377	Factor analysis factors
GRNN	XT_{16}	89.420	1.339	0.874	Crime type
GRNN	XT_{41}	88.750	2.870	1.291	Age
GRNN	XT_{27}, XT_{26}	88.727	2.924	1.106	Career
GRNN	XT_{52}, XT_{53}	88.599	3.218	1.316	Level of education
GRNN	XT_{32}	88.389	3.701	1.398	Gender

The highest values of proportion of variance explained by model, the smallest MSE and MAE were corresponding to GRNN model for the factors of theft crime.

Conclusion and Future Work

For drug crime, it is recommended to use the factor analysis of the original data and then use RSVM model. The reasons of this recommendation are that the factors included most important independent variables for each group and then RSVM model can be used to determining the most important factors that affect the dependent variable.

For theft crime, it is recommended to use the factor analysis of the original data and then use GRNN model. The reasons of this recommendation are that the factors included most important independent variables for each group and then GRNN model can be used to determining the most important factors that affect the dependent variable.

These techniques can be applied for other types of crimes and can give a technical support for the security bodies to prevent the crimes before it happens.

References

[1] M. H. Ahmadian, Artificial neural networks - a new tool in technology, Tech Directions 54(9) (1995), 33-36.

[2] C. Chang and C. J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology, 2011, pp. 1-27.

[3] B. Cheng and D. M. Titterington, Neural networks: a review from a statistical perspective, Statist. Sci. 9 (1994), 2-54.

[4] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, New York, NY, 2000.

[5] V. Dhar and R. Stein, Intelligent Decision Support Methods: The Science of Knowledge Work, Prentice Hall Business Publishing, Upper Saddle River, New Jersey, 1996.

[6] D. A. Dickey and W. A. Fuller, Distribution of the estimators for autoregressive time series with a unit root, J. Amer. Statist. Assoc. 74 (1979), 427-431.

[7] S. E. Fahlmann, An empirical study of learning speed in back-propagation networks, CMU Technical Report, CMU-CS-June, 1988, pp. 88-162.

[8] G. D. Garson, A comparison of neural network and expert systems algorithms with common multivariate procedures for analysis of social science data, Social Science Computer Review 9(3) (1991), 398-434.

[9] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan Publishing, New York, 1994.

[10] Iffat A. Gheyas and Leslie S. Smith, A neural network approach to time series forecasting, Proceedings of the World Congress on Engineering, Vol. II, July 1-3, 2009, London, U.K.

[11] I. Kaastra and M. S. Boyd, Forecasting futures trading volume using neural networks, J. Futures Markets 15(8) (1995), 953-970.

[12] L. Leslie, M. Richman, H. Mansouri and C. Shafer, Application of support vector regression to scatterometer, School of Meteorology, The University of Oklahoma, JCSDA 5th Workshop on Satellite Data Assimilation, Marriott Inn and Conference Center, University of Maryland, 2007.

[13] Richard A. Johnson and Dean W. Wichern, Applied Multivariate Statistical Analysis, 5th ed., Prentice Hall, New Jersey, 2002.

[14] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation, Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, MA, 1986.

[15] S. N. Qasem and S. M. Shamsuddin, Generalization improvement of radial basis function network based on multi-objective particle swarm optimization, J. Artificial Intelligence 3 (2010), 1-16.

[16] S. E. Said and D. A. Dickey, Testing for unit roots in autoregressive-moving average models of unknown order, Biometrika 71 (1984), 599-607.

[17] Avraham Shtub and Ronen Versano, Estimating the cost of steel pipe bending, a comparison between neural networks and regression analysis, International J. Production Economics 62(3) (1999), 201-207.

[18] Robert H. Shumway and David S. Stoffer, Time Series Analysis and Its Applications, Soc. B 54, 2000, pp. 83-127. The Cleveland Clinic Foundation, Cleveland, Ohio.

[19] A. J. Smola, Regression estimation with support vector learning machines, Master's Thesis, Technique University at Munchen, 1996.

[20] A. J. Smola and A. Scholkopf, A tutorial on support vector regression, NeuroCOLT2 Technical Report NC2-TR-1998-030, 1998.

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

[22] D. R. Thomas, P. Zhu and Y. J. Decady, Point estimates and confidence intervals for variable importance in multiple linear regression, J. Educational and Behavioral Statistics 32(1) (2007), 61-91.

[23] R. Tibshirani, A comparison of some error estimates for neural network models, Neural Comput. 8 (1996), 152-163.