Regression Analysis using Swarm Intelligence

Submitted by
Arkanayan Shit
University Roll No.10500113001

A project report submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Science & Engineering

Department Computer Science & Engineering
Bankura Unnayani Institute of Engineering
Bankura - 722 146, West Bengal, India
January 2017

Dedicated to my parents

"We are what our thoughts have made us; so take care about what you think. Words are secondary. Thoughts live; they travel far."

- Swami Vivekananda

"Take up one idea. Make that one idea your life - think of it, dream of it, live on that idea. Let the brain, muscles, nerves, every part of your body, be full of that idea, and just leave every other idea alone. This is the way to success."

- Swami Vivekananda

"When an idea exclusively occupies the mind, it is transformed into an actual physical or mental state."

- Swami Vivekananda

"You have to dream before your dreams can come true."

-Dr. A. P. J. Abdul Kalam

"Dream is not that which you see while sleeping it is something that does not let you sleep."

-Dr. A. P. J. Abdul Kalam

Certificate of Approval

The forgoing project report is hereby approved as a creditable study of Technological subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite with degree for which it has been submitted. It is to be understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for the purpose for which it has been submitted.

Board of Examiners

-
-
-
-

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degrees or diplomas of the university or other institutes of higher learning, except which due acknowledgment has been made in this text.

....

Signature

Certificate

This is certified that the work contained in this report entitled, Regression Analysis using Swarm Intelligence, by Arkanayan Shit, has been carried out under the supervision of the undersign and this work has not been submitted elsewhere for any other degree.

(Signature of the Supervisor)
Mr. Tapas Si, Assistant Professor, Department of CSE
Bankura Unnayani Institute of Engineering
Bankura, West Bengal

(Signature of the Co-supervisor)
Mr. Asim Kumar Mahadani, Assistant Professor, Department of CSE
Bankura Unnayani Institute of Engineering
Bankura, West Bengal

(Signature of the HoD)
Mr. Aloke Roy, Associate Professor, Department of CSE
Bankura Unnayani Institute of Engineering
Bankura, West Bengal

Acknowledgments

I hereby wish to express my sincere gratitude and respect to Assistant Prof. Tapas Si, Dept. of CSE, Bankura Unnayani Institute Of Engineering, Bankura under whom I had proud privilege to work. His valuable guidance and encouragement have really led me to the path of completion of this project. Any amount of thanks would not be enough for the valuable guidance of my supervisor. I would also like to thank all the faculty member of CSE dept. for their devoted help. I also cordially thank all laboratory assistants for their cooperation. Finally, I would like to pen down my gratitude towards my family members for their continuous support and encouragement. It would have not been possible to complete my work without their support.

Abstract

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). A large part of Regression analysis deals with optimization problems. Optimization is finding the optimum (maximum or minimum) value of a co-efficients for which a function has minimum difference between predicted value and real value. Some optimization algorithms are Gradient Descent, Particle Swarm Optimization, Differential Evolution, Fireworks algorithm.

Keywords. Regression analysis, Gradient Descent, Particle Swarm Optimization, Differential Evolution, Fireworks algorithm

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. For example, relationship between rash driving and number of road accidents by a driver is best studied through regression.

Regression analysis is an important tool for modelling and analyzing data. Here, we fit a curve / line to the data points, in such a manner that the differences between the distances of data points from the curve or line is minimized. [1]

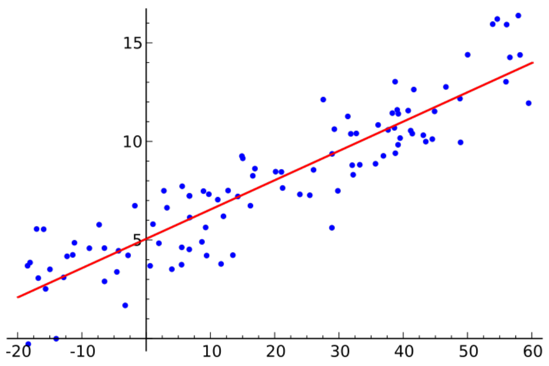

Figure 1.1: Regression Analysis [2]

There are quite a few algorithms that are used for regression analysis. There is one classical method named Gradient Descent. Also swarm intelligence is gaining popularity among researchers for regression analysis. Some of the algorithms that are based on swarm intelligence are Particle Swarm Optimization, Differential Evolution, Fireworks Algorithm.

Here I am doing comparative study of these algorithms with fireworks algorithm on six real world data sets.

1.1 Applications of Regression Analysis

Regression analysis estimates the relationship between two or more variables. Lets understand this with an easy example:

Let's say, you want to estimate growth in sales of a company based on current economic conditions. You have the recent company data which indicates that the growth in sales is around two and a half times the growth in the economy. Using this insight, we can predict future sales of the company based on current & past information.

There are multiple benefits of using regression analysis. They are as follows:

- It indicates the significant relationships between dependent variable and independent variable.
- It indicates the strength of impact of multiple independent variables on a dependent variable.

Predicting the Future

The most common use of regression in business is to predict events that have yet to occur. Demand analysis, for example, predicts how many units consumers will purchase. Many other key parameters other than demand are dependent variables in regression models, however. Predicting the number of shoppers who will pass in front of a particular billboard or the number of viewers who will watch the Super Bowl may help management assess what to pay for an advertisement. Insurance companies heavily rely on regression analysis to estimate how many policy holders will be involved in accidents or be victims of burglaries, for example. [3]

Optimization

Another key use of regression models is the optimization of business processes. A factory manager might, for example, build a model to understand the relationship between oven temperature and the shelf life of the cookies baked in those ovens. A company operating a call center may wish to know the relationship between wait times of callers and number of complaints. A fundamental driver of enhanced productivity in business and rapid economic advancement around the globe during the 20th century was the frequent use of statistical tools in manufacturing as well as service industries. Today, managers considers regression an indispensable tool. [3]

Chapter 2

Background Theory

Types of Regression Techniques

There are many different types of regression techniques. Some of those are discussed underneath.

2.1 Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line).

It is represented by an equation $Y = a + bX + e$, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable(s).

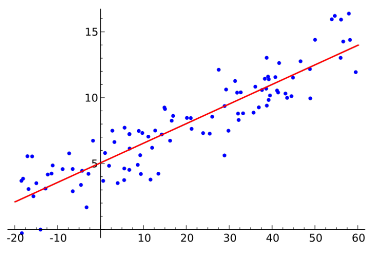

Figure 2.1: Linear Regression [2]

The difference between simple linear regression and multiple linear regression is that, multiple linear regression has (>1) independent variables, whereas simple linear regression has only 1 independent variable. [1]

2.2 Logistic Regression

Logistic regression is used to find the probability of event=Success and event=Failure. We should use logistic regression when the dependent variable is binary (0/ 1, True/ False, Yes/ No) in nature. Here the value of Y ranges from 0 to 1 and it can represented by following equation.

$$odds = \frac{p}{(1-p)} = \frac{probability\ of event\ occurrence}{probability\ of\ not\ event\ occurrence}$$

$$\ln odds = \ln \frac{p}{(1-p)}$$

$$logit(p) = \ln \frac{p}{(1-p)}$$

Above, p is the probability of presence of the characteristic of interest. A question that you should ask here is "why have we used log in the equation?".

Since we are working here with a binomial distribution (dependent variable), we need to choose a link function which is best suited for this distribution. And, it is *logit* function. In the equation above, the parameters are chosen to maximize the likelihood of observing the sample values rather than minimizing the sum of squared errors (like in ordinary regression). [1]

2.3 Polynomial Regression

A regression equation is a polynomial regression equation if the power of independent variable is more than 1. The equation below represents a polynomial equation:

$$y = a + bx^2$$

In this regression technique, the best fit line is not a straight line. It is rather a curve that fits into the data points. [1]

2.3.1 Important Points

- While there might be a temptation to fit a higher degree polynomial to get lower error, this can result in over-fitting. Always plot the relationships to see the fit and focus on making sure that the curve fits the nature of the problem. Here is an example of how plotting can help:

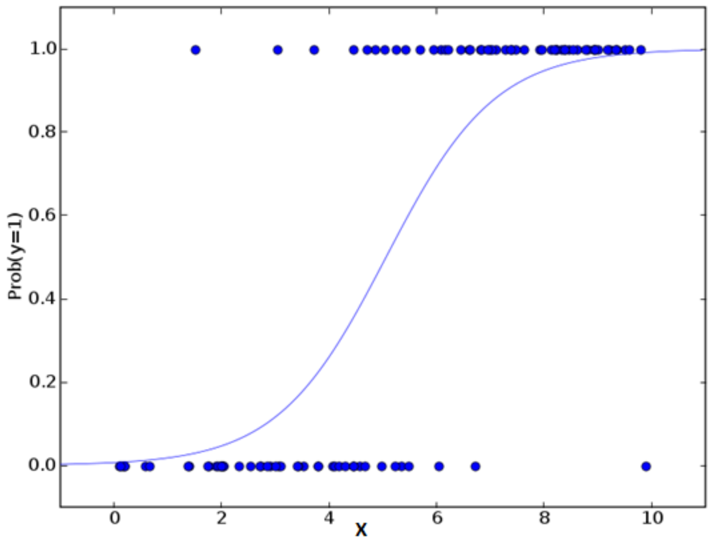

Figure 2.2: Logistic Regression [1]

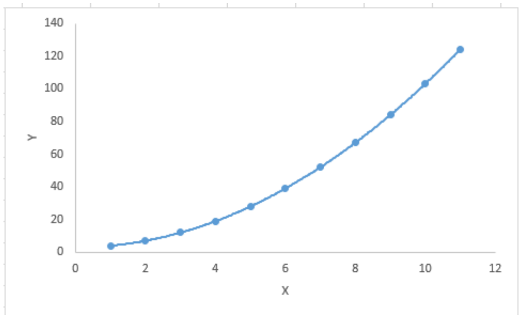

Figure 2.3: Polynomial Regression [1]

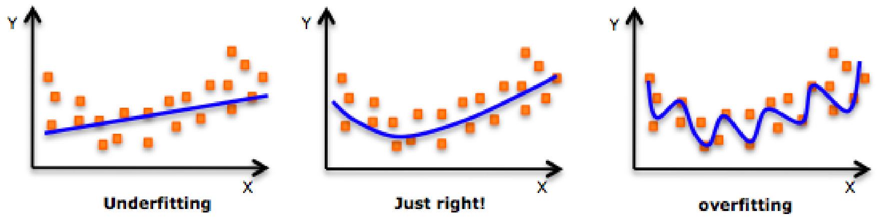

Figure 2.4: Underfitting & Overfitting [1]

- Especially look out for curve towards the ends and see whether those shapes and trends make sense. Higher polynomials can end up producing wierd results on extrapolation. [1]

Chapter 3

Materials & Methods

There are several techniques for regression analysis. Here we are using a few algorithms to analysis. Let us discuss some of those.

3.1 Gradient Descent

Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as **gradient ascent**.

Intuition for Gradient Descent

Think of a large bowl like what you would eat cereal out of or store fruit in. This bowl is a plot of the cost function (f).

Figure 3.1: Large Bowl [5]

A random position on the surface of the bowl is the cost of the current values of the coefficients (cost).

The bottom of the bowl is the cost of the best set of coefficients, the minimum of the function. The goal is to continue to try different values for the coefficients, evaluate their cost and select new coefficients that have a slightly better (lower) cost.

Repeating this process enough times will lead to the bottom of the bowl and you will know the values of the coefficients that result in the minimum cost. [4]

Method

Gradient descent is based on the observation that if the multi-variable function $F(\mathbf{x})$ is defined and differentiable in a neighbourhood of a point $\mathbf{a}$, then $F(\mathbf{x})$ decreases fastest if one goes from $\mathbf{a}$ in the direction of the negative gradient of F at $\mathbf{a}$,$-\nabla F(\mathbf{a})$. It follows that, if

$$\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a}) \mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$$

for γ small enough, then $F(\mathbf{a}) \geq F(\mathbf{b})$. In other words, the term $\gamma \nabla F(\mathbf{a})$ is subtracted from $\mathbf{a}$ because we want to move against the gradient, namely down toward the minimum. With this observation in mind, one starts with a guess $\mathbf{x}_0$ for a local minimum of F, and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ such that

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

We have

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$

so hopefully the sequence $(\mathbf{x}_n)$ converges to the desired local minimum. Note that the value of the step size γ is allowed to change at every iteration. With certain assumptions on the function F (for example, F convex and ∇F Lipschitz) and particular choices of γ (e.g., chosen either via a line search that satisfies the Wolfe conditions or the Barzilai-Borwein method shown as following),

$$\gamma_n = \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})^T [\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})]}{||\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})||^2}$$

convergence to a local minimum can be guaranteed. When the function F is convex, all local minima are also global minima, so in this case gradient descent can converge to the global solution.

This process is illustrated in the adjacent picture. Here F is assumed to be defined on the plane, and that its graph has a bowl shape. The blue curves are the contour lines, that is, the regions on which the value of F is constant. A red arrow originating at a point shows the direction of the negative gradient at that point. Note that the (negative) gradient at a point is orthogonal to the contour line going through that point. We see that gradient descent leads us to the bottom of the bowl, that is, to the point where the value of the function F is minimal. [6]

3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods.

Formally, let $f : \mathbb{R}^n \leftarrow \mathbb{R}$ be the cost function which must be minimized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given candidate solution. The gradient of f is not known. The goal is to find a solution a for which $f(a) \leq f(b)$ for all b in the search-space, which would mean a is the global minimum. Maximization can be performed by considering the function $h = -f$ instead. Let S be the number of particles in the swarm, each having a position $x_i \in \mathbb{R}^n$ in the search-space and a velocity $v_i \in \mathbb{R}^n$. Let pi be the best known position of particle i and let g be the best known position of the entire swarm. A basic PSO algorithm is then:

Algorithm 1 Basic PSO algorithm

```
1: for each particle i = 1, ..., S do
2:     Initialize the particle's position with a random vector: $x_i \sim U(b_{lo}, b_{up})$
3:     Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
4:     if $f(p_i) < f(g)$ then
5:         update the swarm's best known position: $g \leftarrow p_i$
6:     end if
7:     Initialize the particle's velocity: $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$
8:     while a termination criterion is not met do
9:         for each particle $i = 1, ..., S$ do
10:            for each dimension $d = 1, ..., n$ do
11:                Pick random numbers: $r_p, r_g \sim U(0, 1)$
12:                Update particle's velocity: $v_{i,d} \leftarrow \omega\, v_{i,d} + \phi_p r_p (p_{i,d} - x_{i,d}) + \phi_g r_g (g_d - x_i, d)$
13:                Update the particle's position: $x_i \leftarrow x_i + v_i$
14:                if $f(x_i) < f(p_i)$ then
15:                    Update the particle's best known position: $p_i x_i$
16:                    if $f(p_i) < f(g)$ then
17:                        Update the swarm's best known position: $g \leftarrow p_i$
18:                    end if
19:                end if
20:            end for
21:        end for
22:    end while
23: end for
```

The values b_{lo} and b_{up} are respectively the lower and upper boundaries of the search-space. The termination criterion can be number of iterations performed, or a solution with adequate objective function value is found. The parameters ω, ϕp, and ϕg are selected by the practitioner and control the behaviour and efficacy of the PSO method. [7]

3.3 Differential Evolution

n evolutionary computation, differential evolution (DE) is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as DE do not guarantee an optimal solution is ever found.

DE is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means DE does not require for the optimization problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. DE can therefore also be used on optimization problems that are not even continuous, are noisy, change over time, etc.

DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not needed.

DE is originally due to Storn and Price. Books have been published on theoretical and practical aspects of using DE in parallel computing, multiobjective optimization, constrained optimization, and the books also contain surveys of application areas. [9]

Basics

DE is a very simple population based, stochastic function minimizer which is very powerful at the same time. DE managed to finish 3rd at the First International Contest on Evolutionary Computation (1s tICEO) which was held in Nagoya, may 1996. DE turned out to be the best genetic type of algorithm for solving the real-valued test function suite of the 1st ICEO (the first two places were given to non-GA type algorithms which are not universally applicable but solved the test-problems faster than DE). The crucial idea behind DE is a scheme for generating trial parameter vectors.

The flowchart of DE as follows,

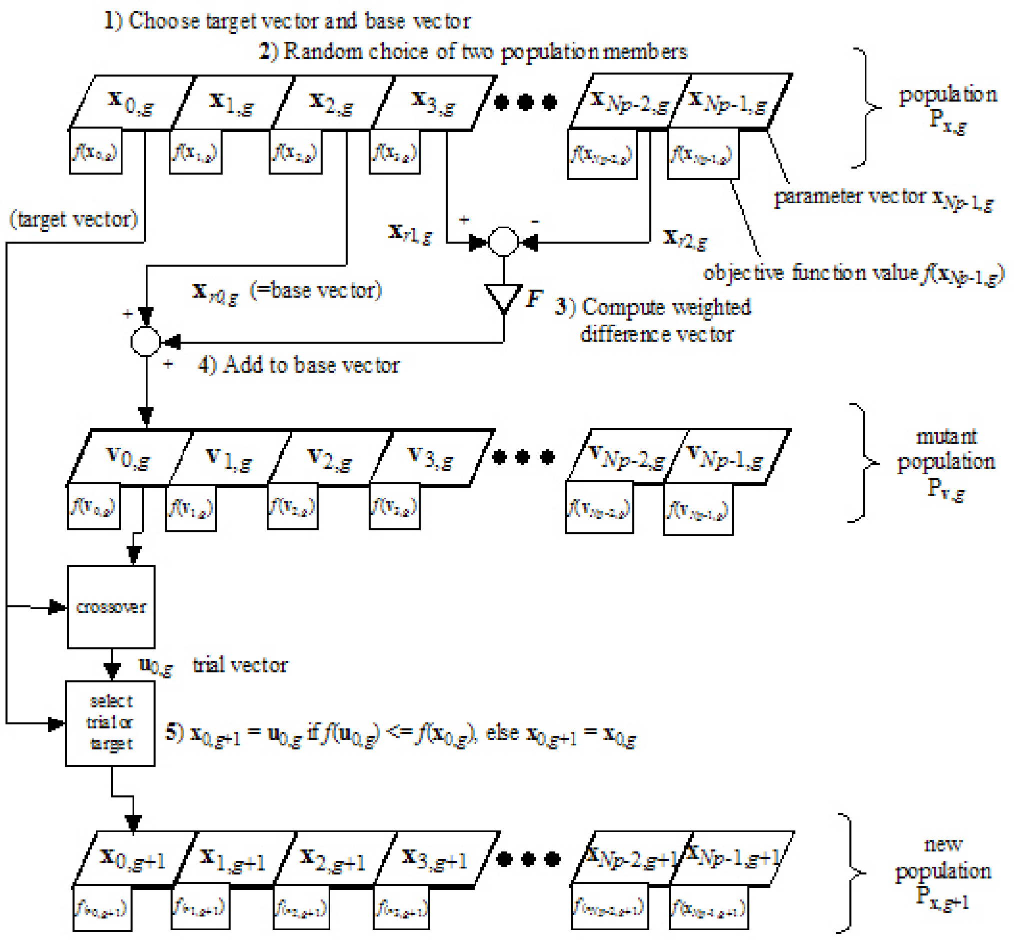
1) Choose target vector and base vector
2) Random choice of two population members
population $P_{x,g}$
parameter vector $x_{Np-1,g}$
objective function value $f(x_{Np-1,g})$
(target vector)
$x_{r1,g}$
$x_{r2,g}$
$x_{r0,g}$ (=base vector)
F
3) Compute weighted difference vector
4) Add to base vector
mutant population $P_{v,g}$
crossover
$u_{0,g}$ trial vector
select trial or target
5) $x_{0,g+1} = u_{0,g}$ if $f(u_{0,g}) <= f(x_{0,g})$, else $x_{0,g+1} = x_{0,g}$
new population $P_{x,g+1}$

Figure 3.2: Flowchart of Differential Evolution [8]

3.4 Fireworks Algorithm

Setting off fireworks is an important creative and joyful activity during Spring Festival in China. At this time, tens of thousands of fireworks explode in the night sky and show beautiful patterns of sparks. Usually, fireworks of different prices and specifications produce entirely different patterns. For example, fireworks of lower price produce less sparks with larger amplitude compared with higher price fireworks and vice versa. The way fireworks explode is similar to the way an individual searches the optimal solution in swarm intelligence algorithms. As a swarm intelligence algorithm, fireworks algorithm consists of four parts, i.e., the explosion operator, mutation operator, mapping rule and selection strategy. The effect of the explosion operator is to generate sparks around fireworks. The number and amplitude of the sparks are governed by the explosion operator. After that, some sparks are produced by mutation operator. The mutation operator utilizes Gaussian operator to produce sparks in Gaussian distribution. Under the effect of the two operators, if the produced spark is not in the feasible region, the mapping rule will map the new generated sparks into the feasible region. To select the sparks for next generation, the selection strategy is used. Fireworks algorithm runs iteratively until it reaches the termination conditions.

3.4.1 FA Framework

When a rework is set off, a shower of sparks will ll the local space around therework. In our opinion, the explosion process of a rework can be viewed asa search in the local space around a specic point where the rework is set othrough the sparks generated in the explosion. When we are asked to nd a pointxjsatisfying $f(x_j) = y$, we can continually set o reworks in potential spaceuntil one spark targets or is fairly near the point xj. Mimicking the process ofsetting o reworks, a rough framework of the FA is depicted in 3.3.

In the FA, for each generation of explosion, we first select n locations, where n fireworks are set off. Then after explosion, the locations of sparks are obtained and evaluated. When the optimal location is found, the algorithm stops. Otherwise, n other locations are selected from the current sparks and fireworks for the next generation of explosion.
From 3.3, it can be seen that the success of the FA lies in a good design of the explosion process and a proper method for selecting locations, which are respectively elaborated in the further subsections.

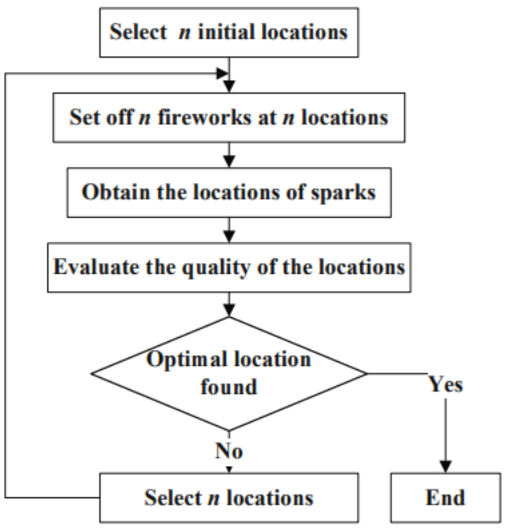
Select *n* initial locations
Set off *n* fireworks at *n* locations
Obtain the locations of sparks
Evaluate the quality of the locations
Optimal location found
Yes
No
Select *n* locations
End

Figure 3.3: Framework of fireworks algorithm [10]

3.4.2 Design of Fireworks Explosion

Through observing fireworks display, we have found two specific behavior of fireworks explosion. When fireworks are well manufactured, numerous sparks are generated, and the sparks centralize the explosion center. In this case, we enjoy the spectacular display of the fireworks. However, for a bad firework explosion, quite few sparks are generated, and the sparks scatter in the space.

The two manners are depicted in 3.4. From the standpoint of a search algorithm, a good firework denotes that the firework locates in a promising area which may be close to the optimal location. Thus, it is proper to utilize more sparks to search the local area around the firework. In the contrast, a bad firework means the optimal location may be far from where the firework locates. Then, the search radius should be larger. In the FA, more sparks are generated and the explosion amplitude is smaller for a good firework, compared to a bad one.

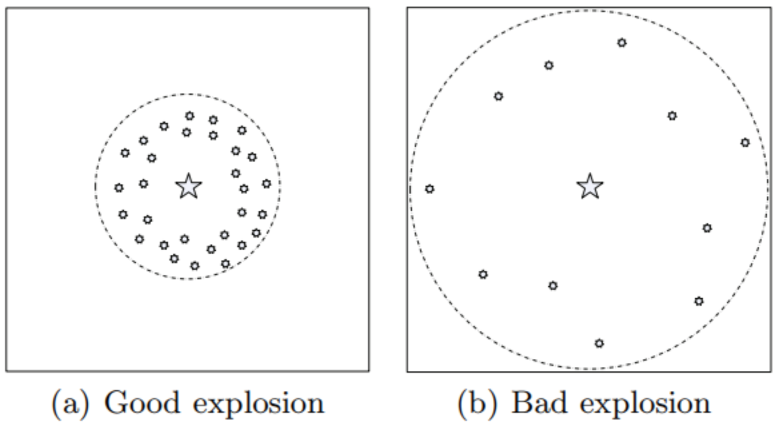

Figure 3.4: Two types of fireworks explosion [10]

Number of Sparks. Suppose the FA is designed for the general optimization problem:

$$Minimize f(x) \in \mathbb{R}, x_{min} \leq x \leq x_{max}, \tag{3.1}$$

where $x = x_1, x_2, ..., x_d$ denotes a location in the potential space, $f(x)$ is an objective function, and x_{min} and x_{max} denote the bounds of the potential space. Then the number of sparks generated by each firework x_i is defined as follows.

$$s_i = m \cdot \frac{y_{max} - f(x_i) + \epsilon}{\sum_{n}^{i=1}(y_{max} - f(x_i)) + \epsilon} \tag{3.2}$$

where m is a parameter controlling the total number of sparks generated by the n fireworks, $y_{max} = max(f(x_i))(i = 1, 2, ..., n)$ is the maximum (worst) value of the objective function among the n fireworks, and ϵ, which denotes the smallest constant in the computer, is utilized to avoid zero-division-error. To avoid overwhelming effects of splendid fireworks, bounds are defined for s_i, which is shown in 3.3.

$$\hat{s_i} = \begin{cases} \text{round(a $\cdot$m)} & if s_i < am \\ \text{round(b $\cdot$m)} & \text{if } s_i > bm, a < b < 1 \\ \text{round}(s_i) & \text{otherwise} \end{cases}, \tag{3.3}$$

where a and b are const parameters.

Amplitude of Explosion. In contrast to the design of sparks number, the amplitude of a good firework explosion is smaller than that of a bad one. Amplitude of explosion for each firework is defined as follows.

$$A_i = \hat{A} \cdot \frac{f(x_i) - y_{min} + \epsilon}{\sum_{n}^{i=1}(f(x_i) - y_{min}) + \epsilon}, \tag{3.4}$$

where $\hat{A}$ denotes the maximum explosion amplitude,and $y_{min} = min(f(x_i))(i = 1, 2, ..., n)$ is the minimum (best) value of the objective function among the n fireworks.

Algorithm 2 Obtain the location of a spark

```
Initialize the location of the spark: $\tilde{x}_j = x_i$ ;
$z = round(d \cdot rand(0, 1))$;
Randomly select $z$ dimensions of $\tilde{x}_j$;
Calculate the displacement: $h = A_i \cdot rand(1, 1)$;
for each dimension $\tilde{x}_k^j \in$ { pre-selected $z$ dimensions of $\tilde{x}_j$ } do
    $\tilde{x}_k^j = \tilde{x}_k^j + h$;
    if $\tilde{x}_k^j < \tilde{x}_{min}^j$ or $\tilde{x}_k^j > \tilde{x}_k^{max}$ then
        map $\tilde{x}_k^j$ to potential space: $\tilde{x}_k^j = \tilde{x}_k^{min} + |\,\tilde{x}_k^j\,|\%(\tilde{x}_k^{max} - \tilde{x}_k^{min})$;
    end if
end for
```

Algorithm 3 Obtain the location of a specific spark

```
Initialize the location of the spark: $\tilde{x}_j = x_i$ ;
$z = round(d \cdot rand(0, 1))$;
Randomly select $z$ dimensions of $\tilde{x}_j$;
Calculate the coefficient of Gaussian explosion: $g$ = Gaussian(1, 1);
for each dimension $\tilde{x}_k^j \in$ { pre-selected $z$ dimensions of $\tilde{x}_j$ } do
    $\tilde{x}_k^j = \tilde{x}_k^j \cdot g$;
    if $\tilde{x}_k^j < \tilde{x}_{min}^j$ or $\tilde{x}_k^j > \tilde{x}_k^{max}$ then
        map $\tilde{x}_k^j$ to potential space: $\tilde{x}_k^j = \tilde{x}_k^{min} + |\,\tilde{x}_k^j\,|\%(\tilde{x}_k^{max} - \tilde{x}_k^{min})$;
    end if
end for
```

Algorithm 4 Framework of the FA

```
Randomly select $n$ locations for fireworks;
while stop criteria = false do
    Set off $n$ fireworks respectively at the $n$ locations:
    for each firework $x_i$ do
        Calculate the number of sparks that the firework yields: $\tilde{S}_i$, according to Eq.3.3
        Obtain locations of $\tilde{S}_i$ sparks of the firework $x_i$ using Algorithm 2
    end for
    for $k = 1 : \hat{m}$ do
        Randomly select a firework $x_j$ ;
        Generate a specific spark for the firework using Algorithm 3;
    end for
    Select the best location and keep it for next explosion generation;
    Randomly select n  1 locations from the two types of sparks and the current fireworks
according to the probability;
end while
```

Chapter 4

Results & Discussion

4.1 Data Set Information

In this this work, we have applied the proposed method on six well known real world data sets. The descriptions of the data sets are given underneath.

4.1.1 Concrete Slump Test

Data Set Characteristics:	Multivariate
Associated Tasks:	Regression
Attribute Characteristics:	Real
Number of Instances:	103
Number of Attributes:	10

4.1.2 Housing Data Set

Data Set Characteristics:	Multivariate
Associated Tasks:	Regression
Attribute Characteristics:	Categorical, Integer, Real
Number of Instances:	506
Number of Attributes:	14

4.1.3 Challenger USA Space Shuttle O-Ring Data Set

Data Set Characteristics:	Multivariate
Associated Tasks:	Regression
Attribute Characteristics:	Integer
Number of Instances:	23
Number of Attributes:	4

4.1.4 Concrete Compressive Strength Data Set

Data Set Characteristics:	Multivariate
Associated Tasks:	Regression
Attribute Characteristics:	Real
Number of Instances:	1030
Number of Attributes:	9

4.1.5 Red Wine Quality Data Set

Data Set Characteristics:	Multivariate
Associated Tasks:	Classification, Regression
Attribute Characteristics:	Real
Number of Instances:	1599
Number of Attributes:	12

4.1.6 White Wine Quality Data Set

Data Set Characteristics:	Multivariate
Associated Tasks:	Classification, Regression
Attribute Characteristics:	Real
Number of Instances:	4898
Number of Attributes:	12

4.2 Parameter Settings

4.2.1 Gradient Descent

Learning Rate α:	0.6
Number of function evaluations:	2000

4.2.2 Particle Swarm Optimization

Number of particles:	10
Number of function evaluations:	2000
Inertia weight =	0.729
c1, c2 =	1.44945
x_{max} =	1
x_{min} =	0

4.2.3 Differential Evolution

Population N :	10
Number of function evaluations:	2000
Crossover Rate:	0.8
Scale Factor:	0.9

4.2.4 Fireworks

Initial Population (N) :	10
Number of function evaluations:	2000
a:	0.04
b:	0.8
$\hat{m}$:	10
m:	50
Maximum Amplitude ($\hat{A}$):	40

4.3 Results

Table 4.1: Training Result Table

Datasets	GD		PSO		DE		FWA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Slump Test	2.1158	0.1093	9.1280	0.4221	8.2466	2.7469	16.6108	0.6138
Housing Dataset	29.2291	1.0910	36.3147	2.1599	58.8854	2.7284	44.6538	2.7284
O-ring Dataset	0.0021	0.0114	23.1370	2.0607	0.0367	1.7544	57.1096	1.7544
Concrete Dataset	83.9624	1.1753	88.7813	0.9970	86.7842	5.0554	97.2449	5.0554
Red Wine Quality	128.1562	1.3891	139.6847	1.9411	141.9707	4.2652	142.2366	4.2652
White Wine Quality	380.6936	2.5659	440.3686	6.4462	409.2238	4.0194	406.1980	4.0194

Table 4.2: Testing Result Table

Datasets	GD		PSO		DE		FWA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Slump Test	0.3766	0.0843	1.9112	0.3724	1.8287	0.5692	3.9014	0.5667
Housing Dataset	7.1900	0.9507	8.7433	0.8530	12.1576	2.9148	10.5913	1.4469
O-ring Dataset	3.5628×10^{-5}	0.0002	2.3429	1.6763	0.0061	3.5629	12.5373	1.7544
Concrete Dataset	20.7782	1.1501	22.3451	1.6806	19.9450	6.2046	21.9134	1.5426
Red Wine Quality	31.3149	1.3765	34.4185	1.5232	33.1242	9.4062	32.5130	1.7021
White Wine Quality	95.4184	2.4815	108.4537	3.8595	95.2971	27.4521	93.3859	2.5384

Table 4.3: Run Time

Datasets	GD	PSO	DE	FWA
	Time	Time	Time	Time
Slump Test	3.40E-05	1.50E-05	0.018889	3.695438
Housing Dataset	1.50E-05	1.60E-05	0.075811	3.410051
O-ring Dataset	1.50E-05	2.40E-05	0.018523	3.524133
Concrete Compressive Strength Dataset	1.40E-05	3.60E-05	0.075699	3.110807
Red Wine Quality	1.40E-05	1.60E-05	0.065636	3.853955
White Wine Quality	1.40E-05	2.00E-05	0.129604	4.339977

4.3.1 Discussion

From the above Table 4.1 it can be seen that for almost all the data sets Gradient Descent Chapter.3.1 trumps all of the other algorithms. The error is the lowest of all the other algorithms. Also from the run time table 4.3 it is evident that the runtime complexity of Gradient Descent is several magnitude lower for the same 2000 function evaluations. Also the standard deviation is lowest among them. That signifies robustness.

The above is also evident for Testing Result Table 4.2.

Chapter 5

Conclusions

In the above pages, four different algorithms (Gradient Descent, Particle Swarm Optimization, Differential Evolution, Fireworks Algorithm) are compared against six data sets. It is found that despite having simplest implementation and technique, Gradient Descent performs best among the rest of the algorithms. It is also more efficient, more robust and have less complexity.

Bibliography

[1] https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression

[2] https://en.wikipedia.org/wiki/Linear_regression

[3] http://smallbusiness.chron.com/application-regression-analysis-business-77200.html

[4] http://machinelearningmastery.com/gradient-descent-for-machine-learning/

[5] William Warby

[6] https://en.wikipedia.org/wiki/Gradient_descent

[7] https://en.wikipedia.org/wiki/Particle_swarm_optimization

[8] http://www1.icsi.berkeley.edu/~storn/code.html

[9] https://en.wikipedia.org/wiki/Differential_evolution

[10] Y. Tan, and Y. Zhu, Firework Algorithm for Optimization, In: Y. Tan et al.(Eds): ICSI 2010, Part I, LNCS 6145, pp.355364, 2010, Springer- Verlag Berlin Heidelberg 2010.