Optimization
Introduction
Definition
Classification
Derivative-Based Optimization
Steepest Descent

Optimization

BITSF312: Neural Networks and Fuzzy Logic

Ankush Jahagirdar

November 22, 2016

Topics to be Covered

- **Lecture 1:** Overview of Traditional Optimization Techniques
- **Lecture 2-3:** Genetic Algorithm
- **Lecture 4-5:** Differential Evolution
- **Lecture 6-7:** Particle Sworm Optimization
- **Lecture 8:** Neuro-Fuzzy/ Hybrid Techniques

14 Nov Extra Lecture
28 Nov Lecture Reserved ...
30 Nov Comprehensive Exam

Applications

- **Science**
- **Engineering**
- **Mathematics**
- **Economics**

Applications

- **Science**
  data fitting
- **Engineering**
- **Mathematics**
- **Economics**

Applications

- **Science**
  data fitting, solution of differential and integral equations
- **Engineering**
- **Mathematics**
- **Economics**

Applications

- **Science**
  data fitting, solution of differential and integral equations
- **Engineering**
- **Mathematics**
- **Economics**

Applications

- **Science**
  data fitting, solution of differential and integral equations
- **Engineering**
  design problems with constraints
- **Mathematics**
- **Economics**

Applications

Device sizing in electronic circuits

- variables: device widths and lengths
- constraints: manufacturing limits, timing requirements, maximum area
- objective: power consumption

Applications

Device sizing in electronic circuits

- variables: device widths and lengths
- constraints: manufacturing limits, timing requirements, maximum area
- objective: power consumption

Find out examples from your field!

Terminology

High and low points of a function — Terminology

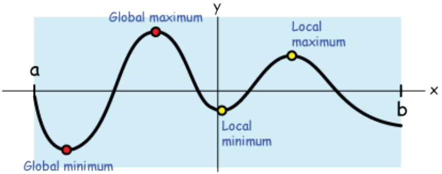
If c is a number in the domain [a, b] of function f, then f(c) is the

Global minimum value of f on [a, b] if $f(c) \leq f(x)$ for all x in [a, b],

Global maximum value of f on [a, b] if $f(c) \geq f(x)$ for all x in [a, b],

Local minimum value of f if $f(c) \leq f(x)$ when x is near c, and

Local maximum value of f if $f(c) \geq f(x)$ when x is near c.



Sometimes

Absolute is used instead of Global &/or

Relative is used instead of Local

Definition I

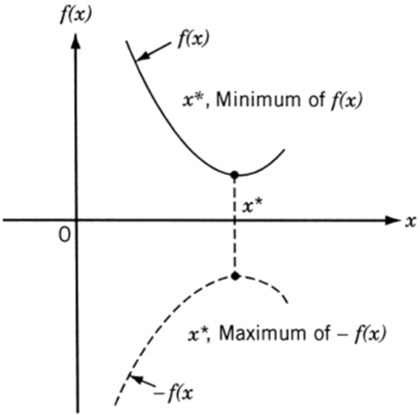
Definition

Optimization is defined as the method of minimizing or maximizing a function of several variables.

Note: Maximum of a function $f(x)$ is the negative of the minimum of $-f(x)$.

$$max\{f(x)\} = -min\{-f(x)\}$$

Definition II

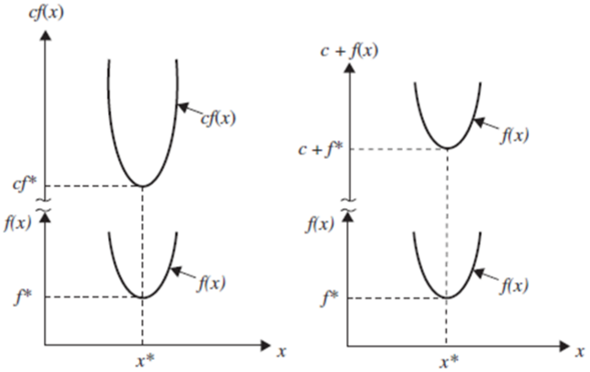


x^* corresponds to the minimum value of function $f(x)$, the same point also corresponds to the maximum value of the negative of the function,i.e. $f(x)$.

Definition III

Following operations on the objective function will not change the optimum solution x^*:

- Multiplication/division of $f(x)$ by a positive constant c
- Addition/subtraction of a positive constant c to/from $f(x)$



Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$

Optimization Problem

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$

which minimizes $f(\mathbf{X})$,

Optimization Problem

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$

which minimizes $f(\mathbf{X})$,

subject to the constraints

Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$

which minimizes $f(\mathbf{X})$,

subject to the constraints

$g_i(\mathbf{X}) \leq 0;\ i = 0, 1, 2, ..., m,$

Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$

which minimizes $f(\mathbf{X})$,

subject to the constraints

$g_i(\mathbf{X}) \leq 0;\ i = 0, 1, 2, ..., m,$

$l_j(\mathbf{X}) = 0;\ j = 0, 1, 2, ..., p.$

Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$ Design Vector

which minimizes $f(\mathbf{X})$,

subject to the constraints

$g_i(\mathbf{X}) \leq 0;\ i = 0, 1, 2, ..., m,$

$l_j(\mathbf{X}) = 0;\ j = 0, 1, 2, ..., p.$

Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$ Design Vector

which minimizes $f(\mathbf{X})$, Objective Function

subject to the constraints

$g_i(\mathbf{X}) \leq 0;\ i = 0, 1, 2, ..., m,$

$l_j(\mathbf{X}) = 0;\ j = 0, 1, 2, ..., p.$

Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$ Design Vector

which minimizes $f(\mathbf{X})$, Objective Function

subject to the constraints

$g_i(\mathbf{X}) \leq 0; \; i = 0, 1, 2, ..., m,$ Inequality Constraints

$l_j(\mathbf{X}) = 0; \; j = 0, 1, 2, ..., p.$

Optimization Problem

A constrained optimization problem can be stated as follows:

Find $\mathbf{X} = [x_1, x_2, ..., x_n]^T$ — Design Vector

which minimizes $f(\mathbf{X})$, — Objective Function

subject to the constraints

$g_i(\mathbf{X}) \leq 0;\ i = 0, 1, 2, ..., m,$ — Inequality Constraints

$l_j(\mathbf{X}) = 0;\ j = 0, 1, 2, ..., p.$ — Equality Constraints

Optimization Techniques

- Exact algorithms:
- Heuristics [1]:
- Metaheuristic:

[1] What are the differences between heuristics and metaheuristics? - ResearchGate.

Optimization Techniques

- Exact algorithms:
  - guaranteed optimal solution in a finite amount of time
- Heuristics [1]:
- Metaheuristic:

[1] What are the differences between heuristics and metaheuristics? - ResearchGate.

Optimization Techniques

- Exact algorithms:
  - guaranteed optimal solution in a finite amount of time
- Heuristics [1]:
  - do not have this guarantee; finds “good” solutions in a “reasonable” amount of time
- Metaheuristic:

[1] What are the differences between heuristics and metaheuristics? - ResearchGate.

Optimization Techniques

- Exact algorithms:
  - guaranteed optimal solution in a finite amount of time
- Heuristics [1]:
  - do not have this guarantee; finds “good” solutions in a “reasonable” amount of time
  - **specific** and **problem-dependent**
- Metaheuristic:

[1] What are the differences between heuristics and metaheuristics? - ResearchGate.

Optimization Techniques

- Exact algorithms:
  - guaranteed optimal solution in a finite amount of time
- Heuristics [1]:
  - do not have this guarantee; finds "good" solutions in a "reasonable" amount of time
  - **specific** and **problem-dependent**
- Metaheuristic:
  - high-level **problem-independent** algorithmic frame-work
  - provides a set of guidelines or strategies to develop heuristic optimization algorithms

[1] What are the differences between heuristics and metaheuristics? - ResearchGate.

Optimization Techniques

Traditional Methods

- Analytical methods (Direct search, Lagrangian mutlipliers, Calculus of variations, etc.)
- Mathematical programming (Dynamic, Geometric, Integer, Linear, Nonlinear programming techniques, etc.)
- Gradient methods (Methods of steepest descent/ascent)

Modern Methods/Metaheuristic Techniques

- Evolutionary Algorithms (Genetic Algorithm, Differential Evolution, etc.)
- Swarm Intelligence (Particle Swarm Optimization, Ant Colony Optimization, etc.)
- Probabilistic Optimization techniques (Simulated Annealing, etc.)

General Startegy

- Start with some initial point X_0.
- Select a direction d_k.
- Move in that direction iteratively to explore the minimum.
- $X_{k+1} = X_k + \eta_k d_k$
  $\eta_k -$ step size/learning rate
- Termination Criteria (Wrong)
  $f(X_{k+1}) = f(X_k + \eta_k d_k) < f(X_k)$

Algorithms differ in selection of **Search Directions** and **Selection of Step Size**.

Gradient of a Function

$$g(X) = \nabla f(X) = [\frac{\partial f(X)}{\partial x_1}, \frac{\partial f(X)}{\partial x_2}, ..., \frac{\partial f(X)}{\partial x_n}]^T$$

Important Propoerties:

- If we move **along the gradient** direction from any point in n-dimensional space, the function value **increases at the fastest rate**. Hence the gradient direction i.e. $d = \nabla f(X)$ is called the direction of steepest ascent.
- If we move **against the gradient** direction from any point in n-dimensional space, the function value **decreases at the fastest rate**. Hence the opposite of gradient direction i.e. $d = -\nabla f(X)$ is called the direction of steepest descent.
- The maximum rate of change of $f(X)$ at any point x^* is the magnitude of the gradient vector i.e. $\sqrt{g^T g}$

Method of Steepest Descent

Steepest descent direction $-g$ is used as d_k.

Method of Steepest Descent

Steepest descent direction $-g$ is used as d_k.

Let's solve an example to understand!

Method of Steepest Descent

Steepest descent direction $-g$ is used as d_k.

Let's solve an example to understand!

Use the steepest descent method to search for the minimum for $f(x_1, x_2) = 25x_1^2 + x_2^2$ starting at point $X_0 = [1, 3]^T$ with step size of $\eta = 0.5$

Method of Steepest Descent

Steepest descent direction $-g$ is used as d_k.

Let's solve an example to understand!

Use the steepest descent method to search for the minimum for $f(x_1, x_2) = 25x_1^2 + x_2^2$ starting at point $X_0 = [1, 3]^T$ with step size of $\eta = 0.5$

$f(X_0) = 34$

$f(X_1) = 14.984$

$f(X_2) = 7.997$

$f(X_3) = 5.5169$

$f(X_4) =$ **4.6394**

$f(X_5) =$ **4.7537**

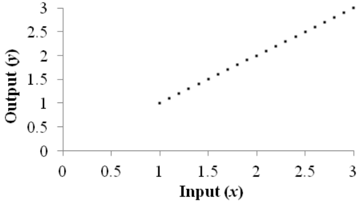
Method of Steepest Descent

Remember:

If step size is not given, you have to *find out the optimum value* for the step size at each iteration.

Linear Regression Example

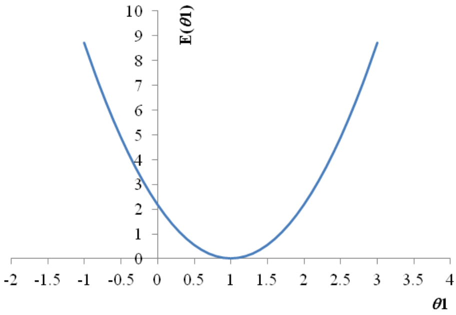
- Goal of linear regression is to fit a line to a set of points
- Consider total $m = 21$ input-output data samples shown in following graph:



- Suppose approximated line is given by: $h_\theta(x) = \theta_1 x$
- We have to calculate Mean Squared Error (MSE) '$E(\theta_1)$' for various values of 'θ_1' using following relation:

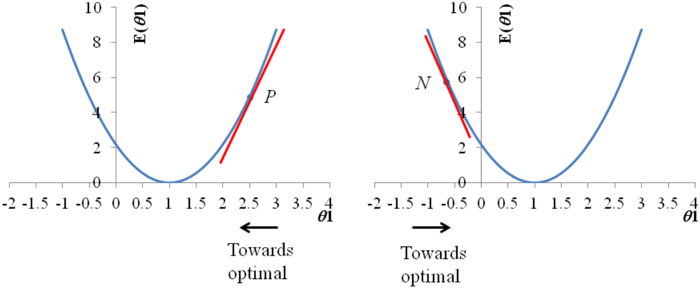
$$E(\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Linear Regression Example



- It can be seen that $\theta_1 = 1$ is the optimal value which gives least MSE.
- Next we will apply Gradient Descent algorithm with constant step size ($\alpha = 0.2$) to find the same value. Assuming initial value of $\theta_1 = 3$.

Linear Regression Example



- According to gradient descent algorithm θ_1 is updated as follows:

$$\theta_1^{(i+1)} = \theta_1^{(i)} - \alpha\left(\frac{dE}{d\theta_1}\right)^{(i)} \qquad where \qquad (\nabla E)^{(i)} = \left(\frac{dE}{d\theta_1}\right)^{(i)}$$

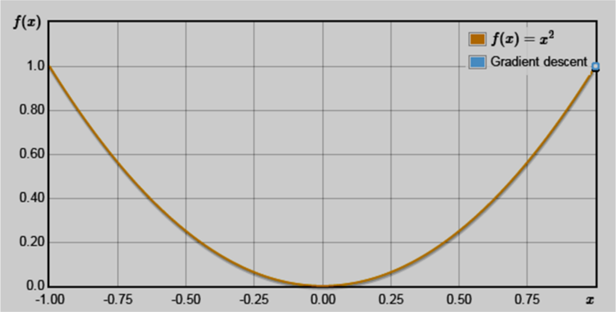
- Gradient $(\nabla E)^{(i)}$ at point 'P' is positive. Thus next value of θ_1 will decrease.
- Gradient $(\nabla E)^{(i)}$ at point 'N' is negative. Thus next value of θ_1 will increase.

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0$



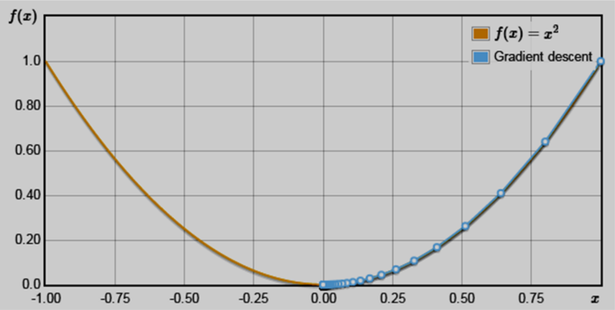
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.1$



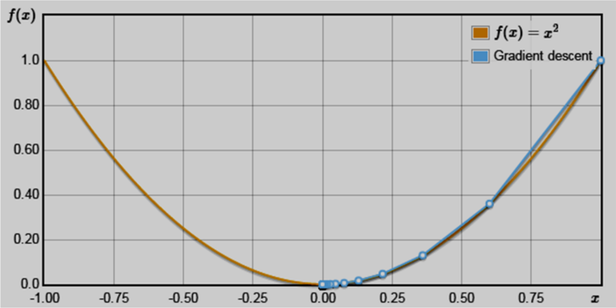
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.2$



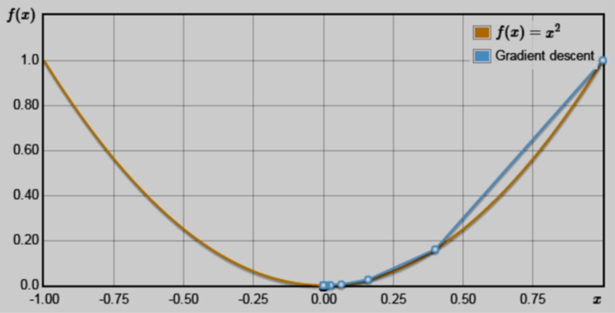
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.3$



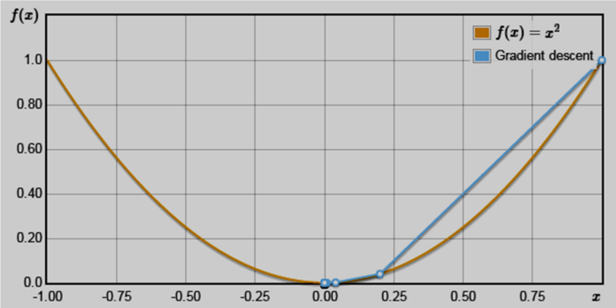
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.4$



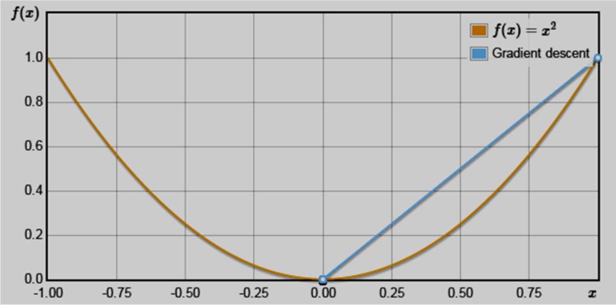
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.5$



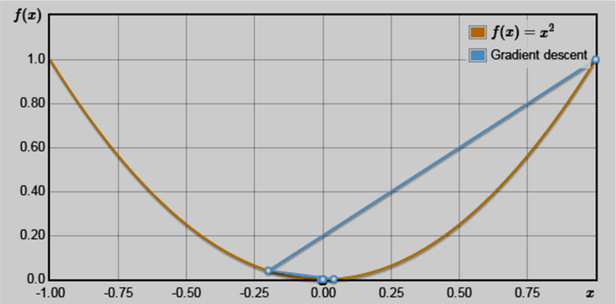
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.6$



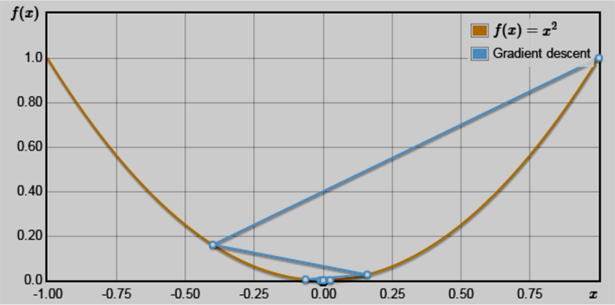
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.7$



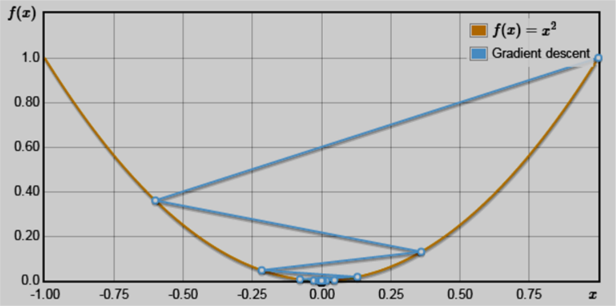
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.8$



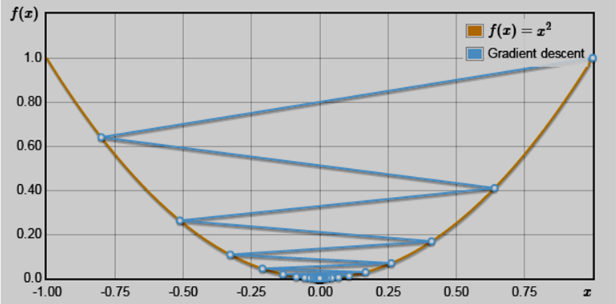
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 0.9$



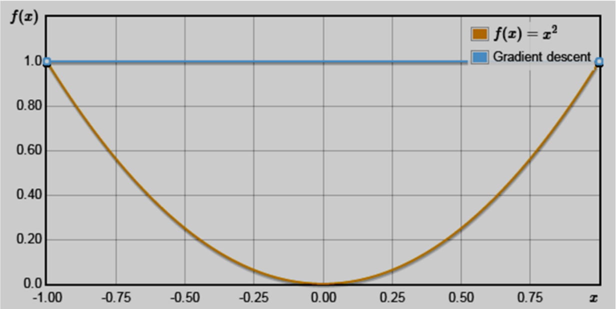
Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Effect of Step Size

Objective function: $f(x) = x^2$

Initial point: $x = 1$

Step size: $\alpha = 1$



Courtesy: http://www.onmyphd.com/?p=gradient.descent&ckattempt=1

Algorithm

Steepest Descent Method

- Use of the negative of the gradient vector as a direction for minimization was first made by Cauchy in 1847.

- Steps for Steepest Descent method:
  1. Select an initial arbitrary point $\mathbf{X}^{(1)}$. Set the iteration number $i := 1$.
  2. For i^{th} iteration, calculate direction of gradient descent of the function as follows: $\mathbf{S}^{(i)} = -\nabla f(\mathbf{X}^{(i)})$.
  3. Set the next search point as follows: $\mathbf{X}^{(i+1)} = \mathbf{X}^{(i)} + \alpha^{(i)} \mathbf{S}^{(i)}$ and determine the optimal step size $\alpha^{(i)}$ (>0) in the direction such that $\dfrac{df(\mathbf{X}^{(i+1)})}{d\alpha^{(i)}} = 0$.

Algorithm

4. Test the new point for optimality using following criterion:

   (a) When the gradient vector becomes zero i.e. $\nabla f(\mathbf{X}^{(i+1)}) = 0$.

   (b) When the change in function value in two consecutive iterations is small i.e. $|f(\mathbf{X}^{(i+1)}) - f(\mathbf{X}^{(i)})| \le \varepsilon_1$.

   (c) When the components of the gradient ∇f are small i.e. $|\partial f/\partial x_j| \le \varepsilon_2$ where $j = 1, 2, \ldots, n$.

   (d) When the change in the design vector in two consecutive iterations is small i.e. $|\mathbf{X}^{(i+1)} - \mathbf{X}^{(i)}| \le \varepsilon_3$.

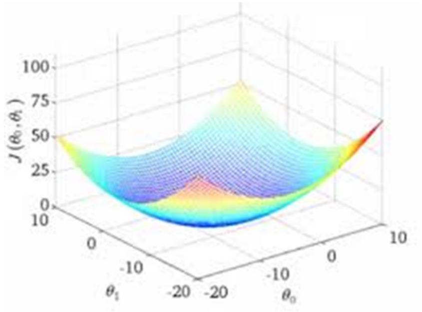
   If $\mathbf{X}^{(i+1)}$ is optimum, then stop the process. Otherwise, go to Step 5.

5. Set the new iteration number $i := i + 1$ and go to Step 2.

Note: Steps for Steepest Ascent are same as those, except Step 2, for Steepest Descent. Set direction gradient ascent as $\mathbf{S}^{(i)} = \nabla f(\mathbf{X}^{(i)})$.

Limitations

Steepest descent converges to single local minima as shown in Fig. Convergence to optimal global point does not depend on initial point but step size (or learning rate as in neural network learning algorithm).



Surface with single local minima

Limitations

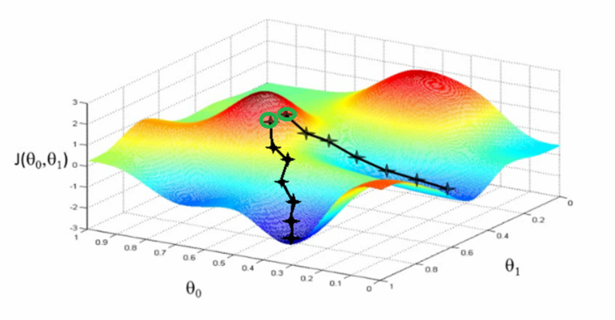
Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Steepest descent converges to one of local minima as shown in Fig. Convergence to local minima depends on initial point as well as step size (or learning rate as in neural network learning algorithm).



Surface with multiple local minima

Limitations of Traditional Methods

- Local in nature
- Requirement of Continuity and Derivative Existence
- Enumerative methods lack efficiency
- Effective for a particular class of problem

“Mans longing for perfection finds expression in the theory of optimization. It studies how to describe and attain what is Best, once one knows how to measure and alter what is Good or Bad.
Optimization theory encompasses the quantitative study of optima and methods for finding them.”

- Beightler, Philips and Wilde (1979)

How are Genetic Algorithms Different?

- GAs work with a **coding** of the parameter set, not the parameters themselves.
  - Parameters as strings.
- GAs search from a **population** of points, not a single point.
  - Population of strings.
- GAs use objective function **(payoff) information**, not derivatives or other auxiliary knowledge.
  - GAs are blind! They only require payoff values associated with individual strings.
  - Knowledge-directed GAs
- GAs use **probabilistic** transition rules, not deterministic rules.

Genetic Algorihm

- GA was introduced by Holland in 1975 and popularized by his student Goldberg.
- GAs does not suffer from the problem of getting stuck in local minima.
- Large amount of randomness does not allow stagnation.

Genetic Algorithm: Simple Example

Maximize: $f(x) = x^2$; $x\epsilon[0, 31]$

Encoding

Population

Reproduction

Crossover

Mutation

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Steps for Genetic Algorithm (GA)

- Consider a maximization problem:
  - Maximize $f(x)$
  - Initial population range is given as $x_i^{(L)} \leq x_i \leq x_i^{(U)}$ $where$ $i = 1,2,......,N$

- Important steps for GA realization
  - Variables Encoding
  - Fitness function Evaluation and Elite Selection
  - Application of GA operators: Reproduction, Crossover, Mutation
  - Check termination criteria

Example

Consider an objective function to be minimized in the interval $0 \leq x_1, x_2 \leq 6$.

$$f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$$

GA variables

- GA begins by defining a chromosome or an array of variable values to be optimized.

- If the chromosome has N variables (an N-dimensional optimization problem) given by $x_1, x_2, \ldots, x_N$, then the chromosome is written as an N element row vector.

  $chromosome = [x_1, x_2, \ldots, x_N]$

- A group of 'n' chromosomes form a population of 'n' members.

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Encoding of GA variables

- Coding is the method by which the variables x_i coded into string structures.
- Binary coding is generally used to translate the range of the function variables. This essentially means that a certain number of initial guesses are made within the range of the function variables and these are transformed into a binary format.
- Length of the binary coding is generally chosen with respect to required accuracy.

Genetic Algorithm

- For l - bit binary encoding, there are 2^l combinations or codes possible. Accuracy in variable values are given by

$$Accuracy = \frac{x_i^{(U)} - x_i^{(L)}}{2^l}$$

- Following linear mapping rule is used for the purpose of l - bit binary encoding:

$$x_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^l - 1}\left[\sum_{j=0}^{l-1}\left(S_j 2^j\right)\right]$$

where $S_j \in \{0, 1\}$

Example

Optimization
Introduction
Definition
Classification
Derivative-Based Optimization
Steepest Descent

Suppose the required accuracy is 0.006.

Example

Suppose the required accuracy is 0.006.

$$\frac{6-0}{2^l} = 0.006$$

$$2^l = 1000$$

$$l = 10 bits$$

Example

Suppose the required accuracy is 0.006.

$$\frac{6-0}{2^l} = 0.006$$

$$2^l = 1000$$

$$l = 10 bits$$

$$5.601 = 0 + \frac{6-0}{2^{10}-1}(\text{decoded value})$$

$$(\text{decoded value}) = 954.97$$

in binary $= 1110111011$

Evaluation of GA Fitness Function

- GA works on the principle of "survival of the fittest".

- The 'good points' or the points which yield larger values for the function have higher probability to continue in the next generation. The 'bad points' or the points which yield smaller values for the function have lower probability to continue in the next generation.

- GA maximizes a given function, so it is necessary to transform a minimization problem to a maximization problem. This transformation does not alter the location of the minimum value.

Genetic Algorithm

- Depending upon whether initial objective function $f(x)$ needs to be maximized or minimized, the fitness function $F(x)$ is defined as:
  - $F(x) = f(x)$ for maximization problem
  - $F(x) = 1/(1+f(x))$ for minimization problem

- The fitness function value for a particular coded string is known as the string's fitness and it is used to decide whether a particular string carries on to the next generation or not.

Genetic Algorithm

The genetic algorithm creates three types of children for the next generation:

- Elite children are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.
- Crossover children are created by combining the vectors of a pair of parents.
- Mutation children are created by introducing random changes, or mutations, to a single parent.

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Reproduction

- Reproduction operation is also known as Selection operation since this operation decides the strings from current population to be selected for Crossover and Mutation operations

- The end result of this operation is the formation of a 'mating pool' where strings are selected in a probabilistic manner using following rule:

  Probability of selection into mating pool $\propto$ *Fitness value of string*

Genetic Algorithm

- Probability of selection of the i^{th} string into the mating pool is

$$p_i = \frac{F_i}{\sum_{j=1}^{n} F_j}$$

where F_i is the fitness of the i^{th} string, n is the population size

- For minimization problem, we needed to maximize its probability of selection. That is the reason behind defining the fitness function as $F(x) = 1/(1+f(x))$.

Genetic Algorithm

- Roulette wheel based selection operation has following steps:
  - Using F_i, calculate p_i.
  - Calculate the cumulative probability P_i.
  - Generate n random numbers between 0 and 1.
  - Copy the string that represents the chosen random number in the cumulative probability range into the mating pool.

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Crossover

- Crossover operation forms offspring chromosomes for next generation by using parent chromosomes from the mating pool of the current generation. Parent chromosomes chosen from mating pool are random since they resulted from probabilistic roulette wheel selection operation.

- For Single-point crossover operation, a random crossover site, say integer H, is chosen such that $0 \leq H \leq N \times l$, and all the bits to the right of the H^{th} position i.e. $(H+1)^{\text{th}}$ bit onwards in the two parent strings are swapped to get two children strings.

Genetic Algorithm

- Single-point crossover at site 11 is illustrated below for 16-bit chromosome:

Crossover site

Parent 1: **00110010111 000110** Offspring 1: **00110010111 011100**

Parent 2: **01110101000 011100** Offspring 2: **01110101000 000110**

- What happens if the crossover sites are $H = 0, N \times l$.

- Crossover operation introduces randomness into the current population to avoid getting trapped in local searches.

Genetic Algorithm

- Crossover operation may result in better or worse strings. If few worse offsprings are formed, they will not survive for long since upcoming generations reproduction will eliminate them.

- What if majority of new offsprings are worse? To avoid such situations, we do not select all strings from the mating pool of current population for reproduction.

- If crossover probability is p_c, then we use following number of crossover parents (CP) to generate equal numbers of crossover children (CC): $CC = round(p_c \times (n - EC))$

Genetic Algorithm

- Crossover operation can be summarized as follows:
  - Select crossover parents (CP) from mating pool resulted by selection operation to generate equal numbers of crossover children (CC): $CC = round(p_c \times (n - EC))$.
  - Select first two pairs of strings and generate a random integer number between 0 to n to decide single-crossover site.
  - Perform single-crossover operation by swapping all the bits to the right of crossover site

Mutation

- Mutation involves making changes in the population members directly, that is, by flipping randomly selected bits in certain strings.
- The aim of mutation is to change the population members by a small amount to promote local searches when the optimum is nearby.
- Mutation is performed by deciding a mutation probability p_m and selecting strings from mating pool of selection operation on which mutation is to be performed.

Genetic Algorithm

- Mutation operation can be summarized as follows:
  - Select mutation parent (MP) from mating pool resulted by selection operation to generate mutation child (MC): $MC = (n - EC - CC)$.
  - Generate random numbers to decide whether mutation is to be performed on a particular bit of the population member or not.
  - If the random number is greater than mutation probability, do no flip particular bit (no mutation for the bit). Otherwise, flip the particular bit (bit is mutated).

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Next Generation Population

- Following is order for next generation population of size = 20, elite count = 2, and crossover probability = 0.8.
  - Elite 1
  - Elite 2
  - Crossover child 1
  - Crossover child 2
  - ...
  - Crossover child 14
  - Mutation child 1
  - ...
  - Mutation child 4

Optimization

Ankush Jahagirdar

Optimization
Introduction
Definition
Classification

Derivative-Based Optimization
Steepest Descent

Termination criteria

- Following termination criteria can be used to Stop GA procedure.
  - Maximum number of Generations, say G = 100, is reached.
  - After certain number of generations, say G = 50, change in fitness function values for two consecutive generations is than threshold value, say

$\{(f_{G=52})-(f_{G=51}) < 10^{-6}$.

Numerical Example for GA

Use unconstrained GA to minimize the objective function:

$f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

Following additional information are provided:

(a) Boundary values of variables: $0 \leq x_1, x_2 \leq 6$.

(b) 10-bit binary encoding of variables (x_1, x_2) represent initial population, given in Table 1.

(c) Take elite count = 2, crossover probability = 0.8, mutation probability = 0.05.

(d) Random number in generation 1 for Roulette wheel selection operation, is given in Table 1.

(e) From the mating pool, first take required numbers of consecutive strings for crossover operations and then take required number of consecutive strings for mutation operations.

(f) Assume Single-point crossover sites as 11, 9, 1, 12, 10, 17, 5 for respective crossover operations.

(g) Assume following bits have probability less than 0.05: bit 9 for mutation parent 1; bit 15 for mutation parent 2; bit 2 for mutation parent 3: bit 5 for mutation parent 4.

Show calculations for first generation to obtain (i) linearly mapped initial population (ii) fitness function (iii) elite children (iv) mating pool (v) crossover children (vi) mutation children (vii) next generation population.