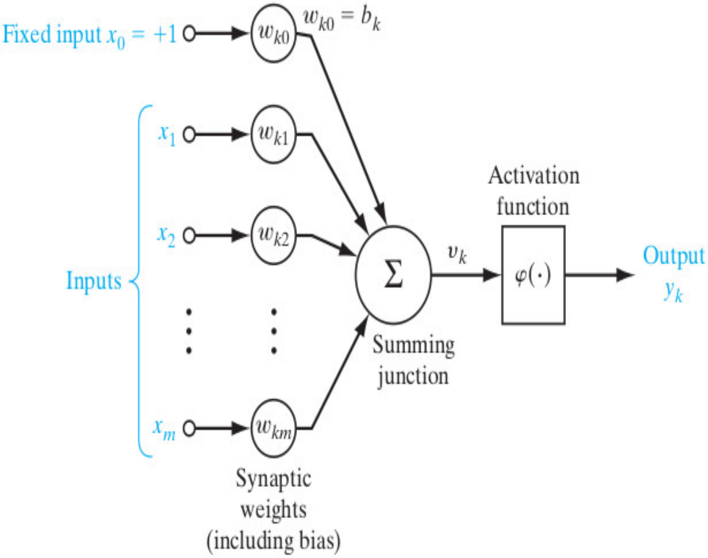
Deep Learning

Vijaya Saradhi

IIT Guwahati

Fri, 11th Sept 2020

Modified Neuron Model



Neural Networks as Directed Graphs

Directed Graphs

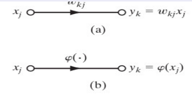
- Consists of links and nodes
- A node has associated signal x_j
- A directed link originates at node j and terminates at node k
- links are of two types
  - Synaptic links
  - Activation links

Neural Networks as Directed Graphs

Rules

Rule 1 A signal flows along a link only in one direction (arrow decides the flow)

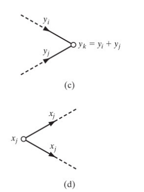
- Synaptic links: Node signal x_j is multiplied by weight w_{kj} to produce node signal y_k
- Activation links: This links behavior is governed by activation function $\phi(.)$



Neural Networks as Directed Graphs

Rules

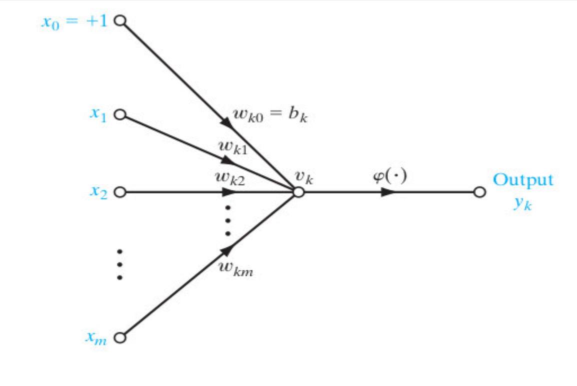
Rule 2 A node signal equal to the sum of all signals entering the node



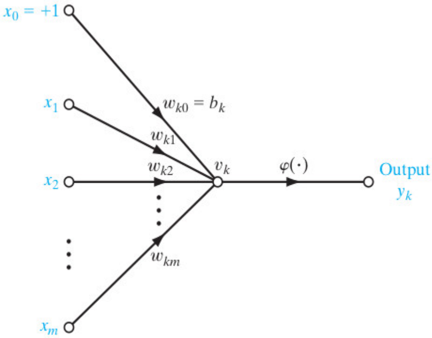
Rule 3 Signal at node is transmitted to each out going link with the same signal

Neuron Example as Directed Graphs

Neuron Model



Neuron Model - Directed Graph



- Rule 1 synaptic link: $x_0 \times w_{k0}$
- Rule 1 synaptic link: Second link: $x_1 \times w_{k1}$
- Rule 1 synaptic link: m^{th} link: $x_m \times w_{km}$
- Rule 2: Node v_k: $x_0 \times w_{k0} + x_1 \times w_{k1} + \cdots + x_m \times w_{km}$
- Rule 1: activation link between node v_k and y_k
- Rule 1: activation link:
$$y_k = \phi\left(\sum_{j=1}^{m} w_{kj} x_j\right)$$

Neural Network Architectures

Types

- Single-layer feedforward networks
- Multi-layer feedforward networks
- Recurrent networks

Single layer feedforward networks

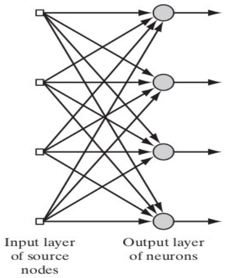


FIGURE 15 Feedforward network with a single layer of neurons.

- Input layer
- Output layer
- Each node is a neuron model
- The arrow emerging out of single node is the output of the neuron model (y_k)

Single layer feedforward networks

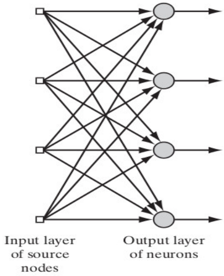
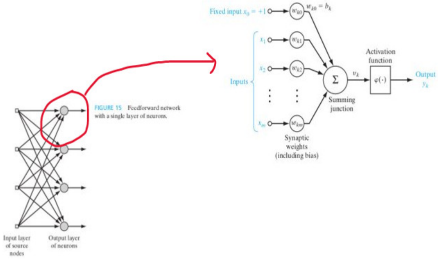


FIGURE 15 Feedforward network with a single layer of neurons.



Single layer feedforward networks

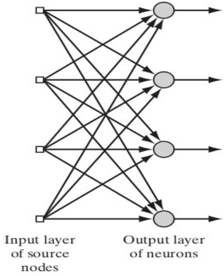


FIGURE 15 Feedforward network with a single layer of neurons.

- Let the inputs be: $x_1, x_2, \cdots, x_m$
- Let the weights on the first neuron be: $w_{11}, w_{12}, w_{13}, \cdots, w_{1m}$
- Let the weights on the second neuron be: $w_{21}, w_{22}, w_{23}, \cdots, w_{2m}$
- Output of the first neuron will be: $y_1 = \phi\left(\sum_{j=0}^{m} w_{1j} x_j\right)$
- Output of the second neuron will be: $y_2 = \phi\left(\sum_{j=0}^{m} w_{2j} x_j\right)$

Single layer feedforward networks

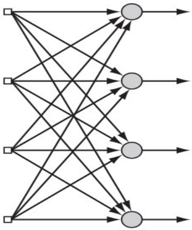


FIGURE 15 Feedforward network with a single layer of neurons.

- Network is feed forward as the inputs and weigths are passing along the direction of the arrows of the network in one direction
- One example of the environment is presented to this network
- Known quantities:
  - One input example (one spam email and its assocaited features) that is $x_{i1}, x_{i2}, \cdots, x_{im}$
  - Input examples class label: d_i

Single layer feedforward networks

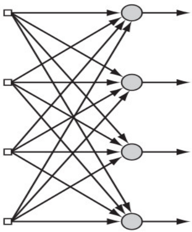
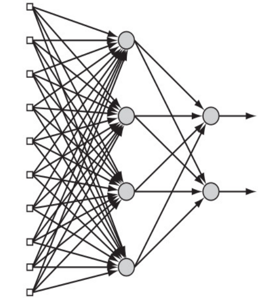


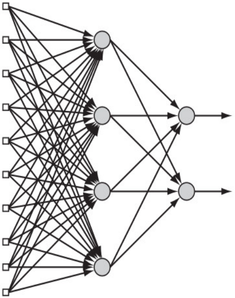
FIGURE 15 Feedforward network with a single layer of neurons.

- What is to be learned?
  - Weights for first neuron:
    $w_{11}, w_{12}, w_{13}, \cdots, w_{1m}$
  - Weights for second neuron:
    $w_{21}, w_{22}, w_{23}, \cdots, w_{2m}$
  - Weights for the last neuron:
    $w_{l1}, w_{l2}, w_{l3}, \cdots, w_{lm}$

Multi layer feedforward networks

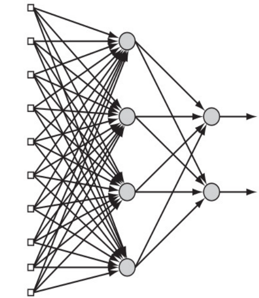


Multi layer feedforward networks



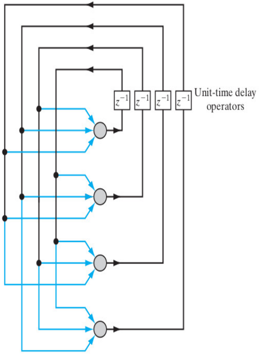
- Input layer, number of hidden layers and output layer
- Architecture is referred as: $m - h_1 - h_2 - q$
- m input features; h_1 hidden units in the first layer
- h_2 hidden units in the second layers and q-output nodes
- First layers is the input layer; last layer is the output layer

Multi layer feedforward networks



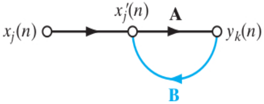
- Computation at the first node of the output layer:
- $y_{21} = \phi\left(\sum_{j=0}^{4} y_{1j} w_{2j}\right)$
- Output depends on the chosen activation function
- Input to the output layers is the 1st hidden layer
- Let its outputs are denoted as $y_{11}, y_{12}, y_{13}, y_{14}$
- The inputs in the 1st hidden layer are multiplied with the weights on the synaptic links going out of the first hidden

Recurrent networks



- Recurrent with no hidden layer
- Contains at least one feedback loop
- First neuron output is fed to rest of the three neurons
- Second neuron output is fed to rest of the other three neurons

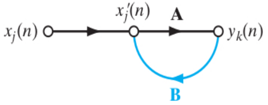
Feedback loop



$$y_k(n) = \mathbf{A}[x_j'(n)]$$

- Three Nodes are there $x_j(n), x_j'(n)$ and $y_k(n)$
- Two black colored directed links
- One blue colored directed link
- Node $x_j'(n)$ has two input links
  - One from node $x_j(n)$
  - One from node $y_k(n)$

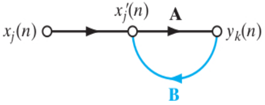
Feedback loop



$$y_k(n) = \mathbf{A}[x'_j(n)]$$
$$x'_j(n) = x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}$$

- Three Nodes are there $x_j(n), x'_j(n)$ and $y_k(n)$
- Two black colored directed links
- One blue colored directed link
- Node $x'_j(n)$ has two input links
  - One from node $x_j(n)$
  - One from node $y_k(n)$

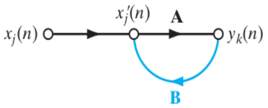
Feedback loop



$$y_k(n) = \mathbf{A}[x_j'(n)]$$
$$x_j'(n) = x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}$$
$$y_k(n) = \mathbf{A}[x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}\ ]$$

- Three Nodes are there $x_j(n), x_j'(n)$ and $y_k(n)$
- Two black colored directed links
- One blue colored directed link
- Node $x_j'(n)$ has two input links
    - One from node $x_j(n)$
    - One from node $y_k(n)$

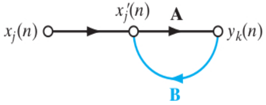
Feedback loop



$$
\begin{aligned}
y_k(n) &= \mathbf{A}[x'_j(n)] \\
x'_j(n) &= x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput} \\
y_k(n) &= \mathbf{A}[x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}\ ] \\
&= \mathbf{A}[x_j(n)] + \mathbf{A}\ \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}
\end{aligned}
$$

- Three Nodes are there $x_j(n), x'_j(n)$ and $y_k(n)$
- Two black colored directed links
- One blue colored directed link
- Node $x'_j(n)$ has two input links
  - One from node $x_j(n)$
  - One from node $y_k(n)$

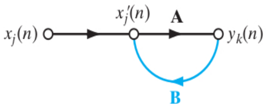
Feedback loop



- Three Nodes are there $x_j(n), x_j'(n)$ and $y_k(n)$
- Two black colored directed links
- One blue colored directed link
- Node $x_j'(n)$ has two input links
  - One from node $x_j(n)$
  - One from node $y_k(n)$

$$y_k(n) = \mathbf{A}[x_j'(n)]$$
$$x_j'(n) = x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}$$
$$y_k(n) = \mathbf{A}[x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}]$$
$$= \mathbf{A}[x_j(n)] + \mathbf{A}\underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}$$
$$= \mathbf{A}[x_j(n)] + \mathbf{AB}[y_k(n)]$$

Feedback loop

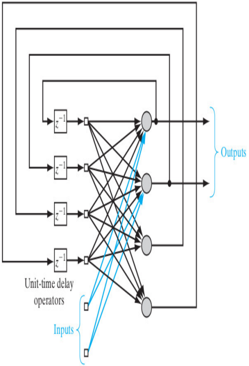


- Three Nodes are there $x_j(n), x_j'(n)$ and $y_k(n)$
- Two black colored directed links
- One blue colored directed link
- Node $x_j'(n)$ has two input links
  - One from node $x_j(n)$
  - One from node $y_k(n)$

$$
\begin{aligned}
y_k(n) &= \mathbf{A}[x_j'(n)] \\
x_j'(n) &= x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput} \\
y_k(n) &= \mathbf{A}[x_j(n) + \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput}\ ] \\
&= \mathbf{A}[x_j(n)] + \mathbf{A}\ \underbrace{\mathbf{B}[y_k(n)]}_{feedbackoutput} \\
&= \mathbf{A}[x_j(n)] + \mathbf{AB}[y_k(n)] \\
y_k(n) &= \frac{\mathbf{A}}{(1-\mathbf{AB})}[x_j(n)]
\end{aligned} \quad (1)
$$

Recurrent networks

with one hidden layer



Modern Architectures

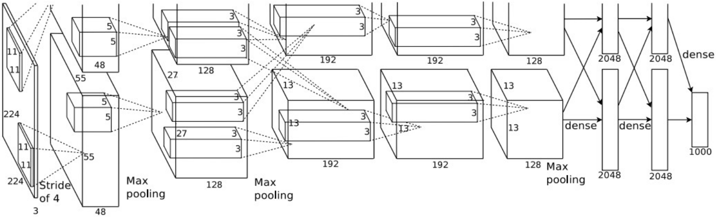


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Knowledge Representation

Definition

Stored information or models used by a person or a machine to interpret, predict and appropriately respond to the outside world.

Knowledge Representation

Discussion

Knowledge of the world consists of two kinds of information:

- Prior Information the known facts.
- Class related prior information example: 20% of emails belong to spam;
- Feature related prior information example 2: 90% of spam emails contain the word "Free Free Free"
- Incorporating such information is of

Knowledge Representation

Four main points

Rule 1 Similar inputs from similar classes should produce similar representations inside the network

Rule 2 Inputs to be categorized as separate classes should be given widely different representation in the network

Rule 3 Importance to specific features is given throguh involving large number of neurons

Rule 4 Prior information is achieved through design of neural network.

Introduction

- Obtain best result under given circumstance
- In engineering discipline the goal is to minimize the effort required or maximize the desired benefit
- These are expressed as function of certain decision variables
- Optimization can be defined as the process of finding conditions that gives maximum or minimum value of a function

Introduction

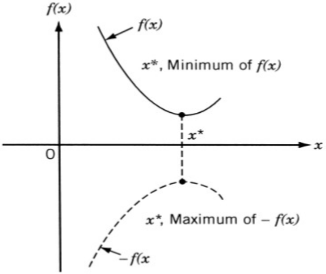
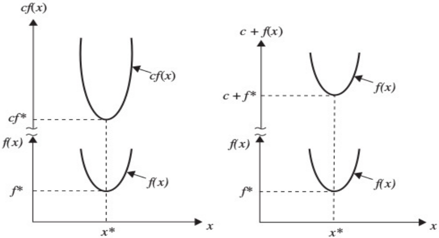


Figure 1.1 Minimum of $f(x)$ is same as maximum of $-f(x)$.



Statement Of Optimization Problem

- Optimization problem

$$
\begin{aligned}
\underset{\mathbf{x}}{\textit{minimize}}\ f \quad & f(\mathbf{x}) \\
\text{subject to} \quad & g_j(\mathbf{x}) \leq 0 \ \ \forall \ j = 1, 2, \cdots, m \\
& l_j(\mathbf{x}) = 0 \ \ \forall \ j = 1, 2, \cdots, p
\end{aligned}
$$

- $\mathbf{x}$: Design variables/ design vector
- $f(\mathbf{x})$: objective function
- $g_j(\mathbf{x})$ inequality constraints
- $l_j(\mathbf{x})$ equality constraints
- Constrained optimization problem

Variations

- Design variables:
  - Single variable/Multivariable
  - Continuous values/integer values
- objective function
  - Linear
  - Non-linear
  - Convex
  - Single objective/multi objective
  - Unimodal/multimodal
- Constraints
  - No constraints
  - only $l_j(.)$ which are linear
  - both $g_j(.)$ and $l_j(.)$
  - Convex

Nature of objective functions

- When there are no constraints present the problem is an unconstrained optimization
- When there are constrains present the problem is known as constrained optimization
- Linear Optimization When $f(\mathbf{x})$ is linear and only linear constraints are present
- Non Linear Optimization when $f(\mathbf{x})$ is nonlinear
- Convex Optimization When $f(\mathbf{x})$ is convex and constraints are linear

Single variable optimization

Local optimal

f(x) has a minimum at $x = x^*$ if $f(x^*) \leq f(x^* + h)$ for all sufficiently small positive and negative values of h.
f(x) has a maximum at $x = x^*$ if $f(x^*) \geq f(x^* + h)$ for all sufficiently small positive and negative values of h.

Global optimal

$x = x^*$ found in the interval [a, b] such that x^* minimizes $f(x)$

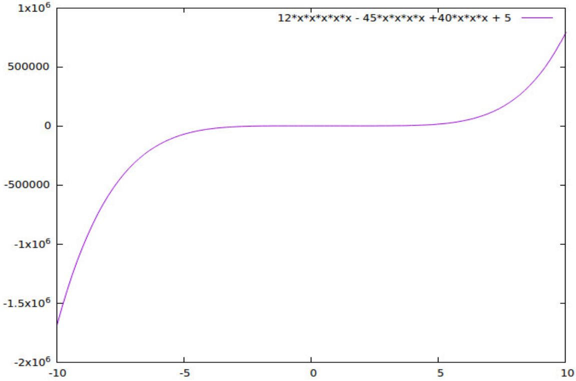
Single Variable

Necessary Condition

if $f(x)$ is defined in the interval [a, b] and has a local minimum at $x = x^*$; let the first order derivative of $f(x)$ exists at $x = x^*$ then

$$\frac{df(x)}{dx} = 0$$

Example



12*x*x*x*x*x - 45*x*x*x*x +40*x*x*x + 5
1x10^6
500000
0
-500000
-1x10^6
-1.5x10^6
-2x10^6
-10
-5
0
5
10

Example

$$f^{'}(x) = 60(x^4 - 3x^3 + 2x^2) = 60x^2(x-1)(x-2)$$

$f^{'}(x) = 0$ at x $=$ 0, 1 and 2.

Multi Variable

Necessary Condition

Let $\mathbf{x} = (x_1, x_2, \cdots, x_m)$

If f($\mathbf{x}$) has a maximum or minimium point at $\mathbf{x} = \mathbf{x}^*$. Assume partial derivatives of $f(\mathbf{x})$ exists at $\mathbf{x}^*$ then

$$\left.\frac{\partial f(\mathbf{x})}{\partial x_1}\right|_{x_1=x_1^*} = \left.\frac{\partial f(\mathbf{x})}{\partial x_2}\right|_{x_2=x_2^*} = \cdots = \left.\frac{\partial f(\mathbf{x})}{\partial x_m}\right|_{x_n=x_m^*} = 0$$

$$\left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} = \left.\begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_m} \end{bmatrix}\right|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}$$

Example

$$f(x_1, x_2) = x_1^3 + x_2^3 + 2x_1^2 + 4x_2^2 + 6$$

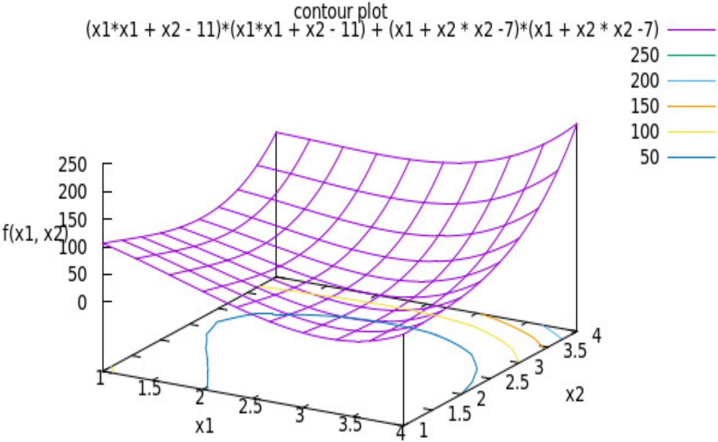
Necessary Condition

$$\frac{\partial f(x_1,x_2)}{\partial x_1} = 3x_1^2 + 4x_1 = x_1(3x_1 + 4) = 0$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} = 3x_2^2 + 8x_2 = x_2(3x_2 + 8) = 0$$

These equations satisfy at (0, 0), $(0, -\frac{8}{3})$, $(-\frac{4}{3}, 0)$ and $(-\frac{4}{3}, -\frac{8}{3})$

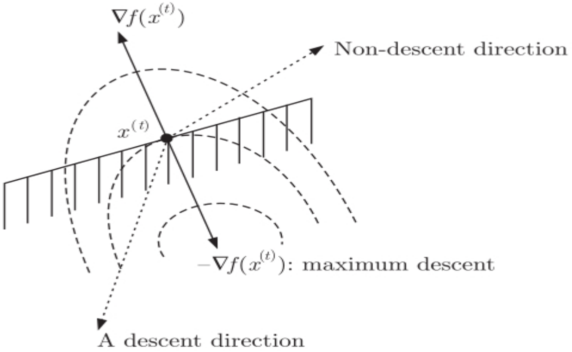
Contours


contour plot
(x1*x1 + x2 - 11)*(x1*x1 + x2 - 11) + (x1 + x2 * x2 -7)*(x1 + x2 * x2 -7)
250
200
150
100
50
f(x1, x2)
250
200
150
100
50
0
1
1.5
2
2.5
3
3.5
4
x1
1
1.5
2
2.5
3
3.5
4
x2

Descent Direction

Definition

A search direction $\mathbf{d}^t$ is a descent direction at point $\mathbf{x}^t$ if the condition $\bigtriangledown f(\mathbf{x}^t).\mathbf{d}^t \leq 0$ is satisfied



Descent Direction

Condition

$$
\begin{aligned}
f(\mathbf{x}^{(t+1)}) &< f(\mathbf{x}^t) \\
&< f(\mathbf{x}^t + \alpha \bigtriangledown f(\mathbf{x}^t).\mathbf{d}^t)
\end{aligned} \tag{2}
$$

Maximum Descent Direction

Condition

When $\mathbf{d}^t = -\bigtriangledown f(\mathbf{x}^t)$ maxium decrease in function value is obtained

Let $\mathbf{d}^t = (1,0)^T$ Example: $f(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$

Let $\mathbf{x}^t = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Let $\mathbf{d}^t = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$\bigtriangledown f\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} -46 \\ -38 \end{pmatrix}$

$(-46 - 38)\begin{pmatrix} 1 \\ 0 \end{pmatrix} = -46$

Maximum Descent Direction

Condition

When $\mathbf{d}^t = -\bigtriangledown f(\mathbf{x}^t)$ maxium decrease in function value is obtained

Example: $f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$

Let $\mathbf{x}^t = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

When $\mathbf{d}^t = -\bigtriangledown f(\mathbf{x}^t) = \begin{pmatrix} 46 \\ 38 \end{pmatrix}$

$\bigtriangledown f\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} -46 \\ -38 \end{pmatrix}$

$(-46 - 38)\begin{pmatrix} 46 \\ 38 \end{pmatrix} = -3560$

Gradient Descent

Algorithm

Step 1 Choose a maximum number of iterations M to be performed, an initial point $x^{(0)}$, two termination parameters ϵ_1, ϵ_2, and set $k = 0$.

Step 2 Calculate $\nabla f(x^{(k)})$, the first derivative at the point $x^{(k)}$.

Step 3 If $\|\nabla f(x^{(k)})\| \leq \epsilon_1$, **Terminate**;

Else if $k \geq M$; **Terminate**;

Else go to Step 4.

Step 4 Perform a unidirectional search to find $\alpha^{(k)}$ using ϵ_2 such that $f(x^{(k+1)}) = f(x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}))$ is minimum. One criterion for termination is when $|\nabla f(x^{(k+1)}) \cdot \nabla f(x^{(k)})| \leq \epsilon_2$.

Step 5 Is $\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k)}\|} \leq \epsilon_1$? If yes, **Terminate**;

Else set $k = k + 1$ and go to Step 2.