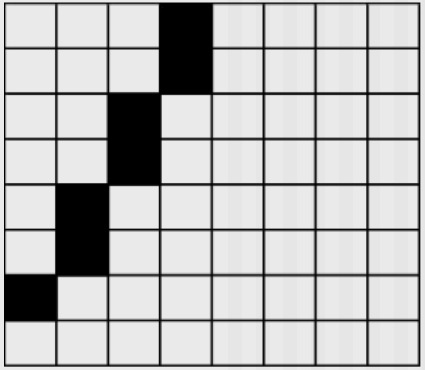
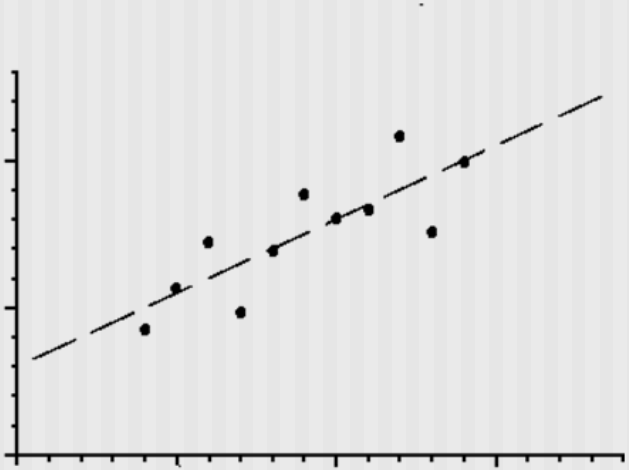
Lecture 3. Gradient Descent and Optimization

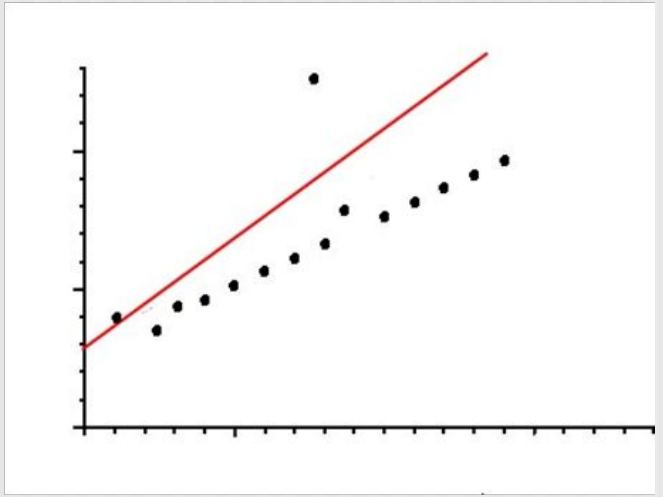
Suthep “Jogie” Madarasmi, Ph.D.

Linear Regression





min error



regression

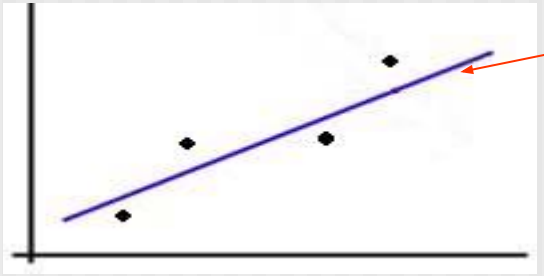
-sensitive to noise

unknown

$y_i = mx_i + b$

known

Gradient Descent Method



Best fit line given 4 points:

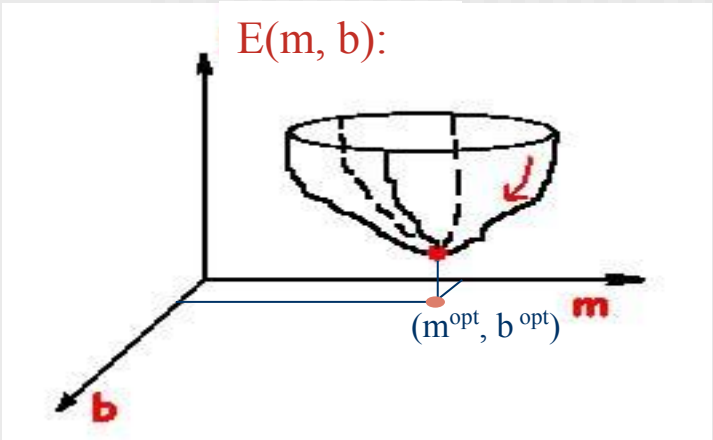
(x1, y1), (x2, y2), (x3, y3), (x4, y4).

Find best m, b for y = mx + b so that:

$$E = \sum_{i=1}^{4} (y_i - mx_i - b)^2 \quad \text{is minimum}$$

observed y

computed y



(m^{opt}, b^{opt}) is the value of (m, b) that minimizes E(m, b).

Gradient/Steepest Descent

E(m, b) Chain Rule:

$$\frac{dE}{dt} = \frac{\partial E}{\partial b}\frac{db}{dt} + \frac{\partial E}{\partial m}\frac{dm}{dt}$$

If we let: $\frac{dm}{dt} = -\frac{\partial E}{\partial m}$ and $\frac{db}{dt} = -\frac{\partial E}{\partial b}$

Using above update rule, the next E will be maximally smaller than current E:

$$E^{t+1} - E^{t} = \frac{dE}{dt} = -\left(\frac{\partial E}{\partial b}\right)^2 - \left(\frac{\partial E}{\partial m}\right)^2$$

Gradient Descent Update Rule

IF E(m, b):

$$m^{t+1} = m^{t} - \alpha \frac{\partial E}{\partial m} \qquad b^{t+1} = b^{t} - \alpha \frac{\partial E}{\partial b}$$

IF E(x, y, z):

$$x^{t+1} = x^{t} - \alpha \frac{\partial E}{\partial x} \qquad y^{t+1} = y^{t} - \alpha \frac{\partial E}{\partial y}$$

$$z^{t+1} = z^{t} - \alpha \frac{\partial E}{\partial z}$$

Gradient Descent Algorithm

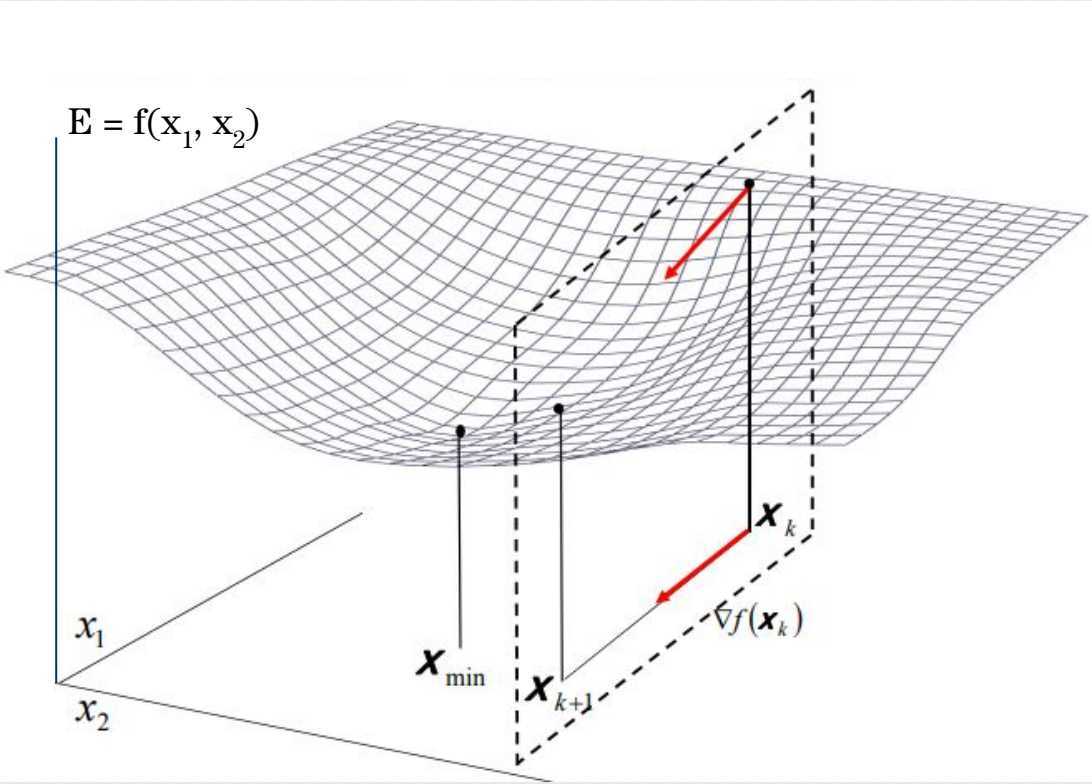
1. Start at random values for m_0, b_0
2. Update m, b by moving one α step in direction that will reduce E to the max (change in E or ΔE is max). This direction is either plus or minus some small value.

$$m_{t+1} = m_t - \alpha \frac{\partial E}{\partial m}$$

$$b_{t+1} = b_t - \alpha \frac{\partial E}{\partial b}$$

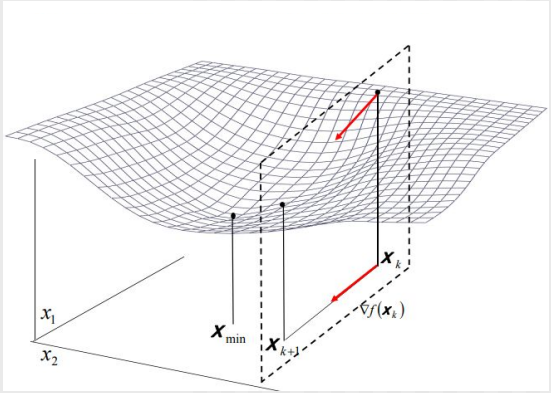
3. Repeat 2 until not much change in E

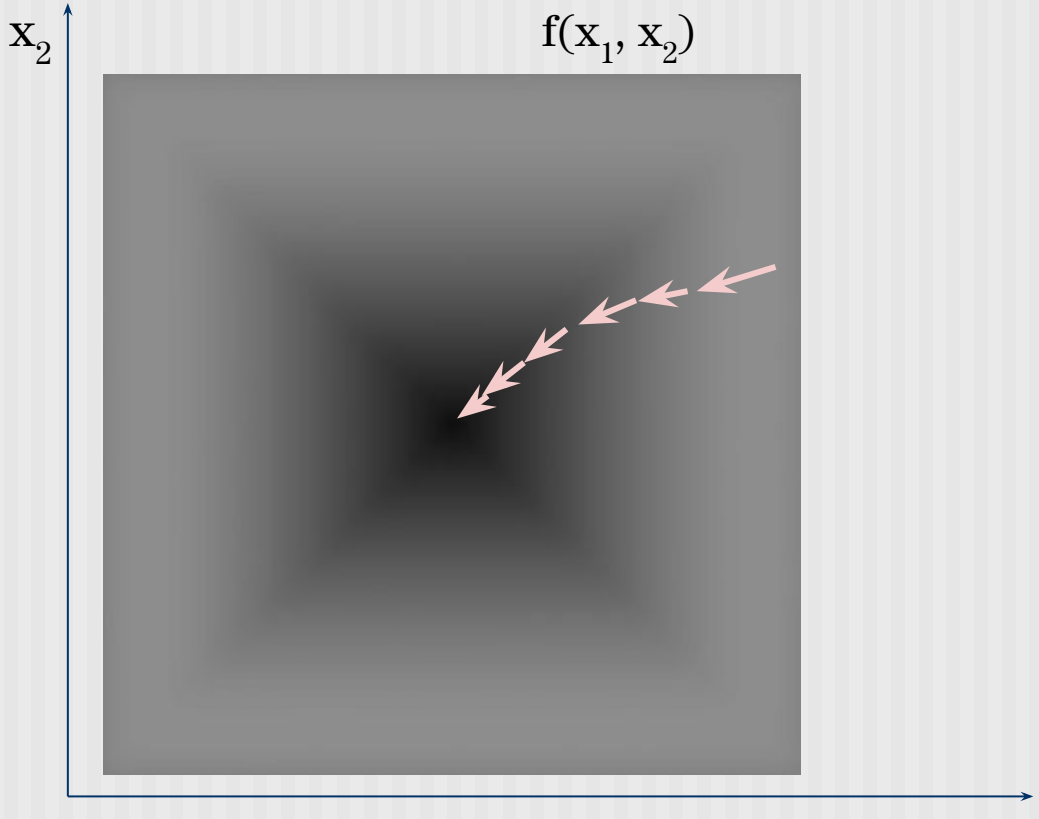
Gradient Descent Energy Terrain



$$\nabla f(x_1, x_2) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

Gradient Descent Energy Terrain





Gradient Descent Example

1. m^0 = random (-max, +max) or start at best guess.
   b^0 = random (-max, +max) or start at best guess.

2. Direction of Gradient

$$\frac{\partial E}{\partial m} = 2\sum (y_i - mx_i - b)(-x_i) \qquad \frac{\partial E}{\partial b} = 2\sum (y_i - mx_i - b)(-1)$$

$$\frac{dm}{dt} = -\frac{\partial E}{\partial m} \qquad m^{t+1} - m^t = 2\sum x_i (y_i - mx_i - b)$$

$$\frac{db}{dt} = -\frac{\partial E}{\partial b} \qquad b^{t+1} - b^t = 2\sum (y_i - mx_i - b)$$

$$m^{t+1} = m^t + \alpha \sum x_i (y_i - mx_i - b)$$

$$b^{t+1} = b^t + \alpha \sum (y_i - mx_i - b)$$

α is step size - must be small enough

3. Repeat 2 until E has no change, or m^t, b^t no change

Quiz 4

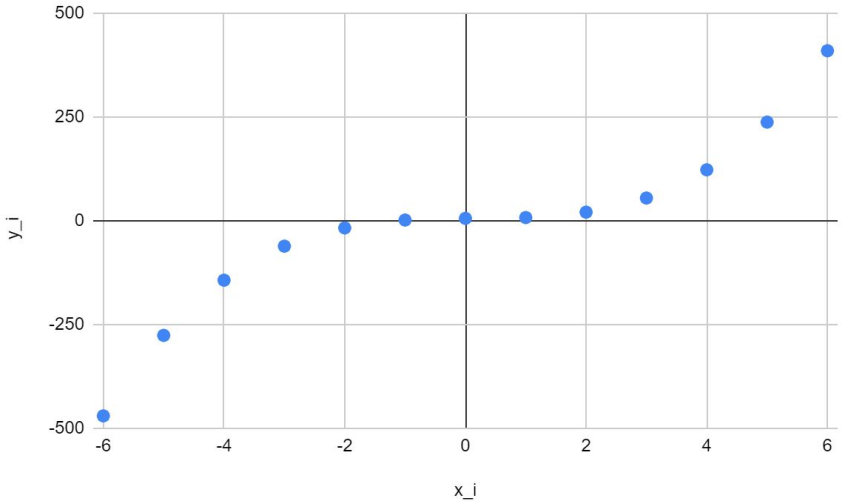
x_i	y_i	round(y_i)
-6	-469.8	-470
-5	-275.5	-276
-4	-143.2	-143
-3	-60.9	-61
-2	-16.6	-17
-1	1.7	2
0	6	6
1	8.3	8
2	20.6	21
3	54.9	55
4	123.2	123
5	237.5	238
6	409.8	410

1. The following points (x_i, y_i) are discrete samples from a function $f(x) = ax^3 + bx^2 + cx + d$.

   a. *5 points.* 0.5 hrs. Show the update rule equation used to find the current a, b, c, and d after each iteration. Make sure you show the mathematics on how this is derived.

   b. *15 points.* 2 hrs. Write a program to find the best fit a, b, c, and d using gradient descent. You must write the gradient descent loop yourself and not use any gradient descent libraries. Attach the source code as well. *Hint*: You should get a, b, c, and d close to 2, -1, 1.3, 6, respectively.

x_i	y_i
-6	-470
-5	-276
-4	-143
-3	-61
-2	-17
-1	2
0	6
1	8
2	21
3	55
4	123
5	238
6	410

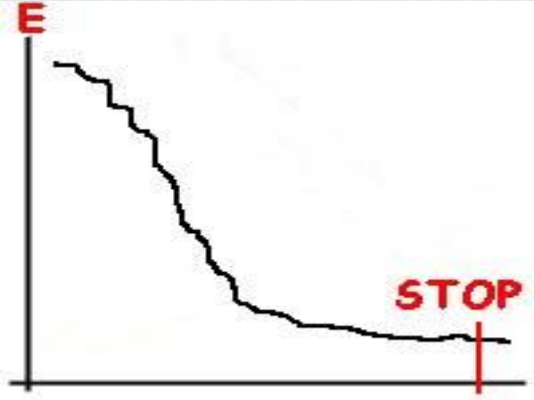


Gradient Descent

E (a, b, c) - function to be minimized

$$\frac{da}{dt} = -\frac{\partial E}{\partial a}; \quad \frac{db}{dt} = -\frac{\partial E}{\partial b}; \quad \frac{dc}{dt} = -\frac{\partial E}{\partial c}$$

$$a^{t+1} = a^{t} - \alpha \frac{\partial E}{\partial a} \qquad b'^{+1} = b' - \alpha \frac{\partial E}{\partial b} \qquad c^{t+1} = c^{t} - \alpha \frac{\partial E}{\partial c}$$



Step size α has to be small

$\alpha = 0.01$ might not work.
$\alpha = 0.0001$ might work.

Cannot find derivative by analytical method? Use numerical method.

How to numerically estimate the gradient for function $E(x_1, \ldots, x_n)$?

$$\delta = -\nabla E(x_1, x_2, \ldots, x_n) = -\left(\frac{\partial E}{\partial x_1}, \frac{\partial E}{\partial x_2}, \ldots, \frac{\partial E}{\partial x_n}\right)$$

$$\frac{\partial f(x_1,x_2,\ldots,x_n)}{\partial x_1} = \frac{f(x_1+h,x_2,\ldots,x_n)-f(x_1,x_2,\ldots,x_n)}{h}$$

$$\frac{\partial f(x_1,x_2,\ldots,x_n)}{\partial x_2} = \frac{f(x_1,x_2+h,\ldots,x_n)-f(x_1,x_2,\ldots,x_n)}{h}$$

In many calculators, h = 0.001, for double precision but it may still cause errors due to precision. A preferred method is to let:

$$h = x_i\sqrt{\varepsilon}$$

where ε is the machine number closest to 0, often about $2.2 \cdot 10^{-16}$.

Numerical method to finding partials example

$$\Re^2 \xrightarrow{f} \Re$$

$$f(x_1, x_2) = 3x_1^2 + 4x_1x_2 - x_2 - 12.$$

Numerically, find $\dfrac{\partial f(x_1, x_2)}{\partial x_1}$ at $(2, 5)$.

We know by derivatives that :

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 6x_1 + 4x_2; \quad \frac{\partial f(2,5)}{\partial x_1} = 6*2 + 4*5 = 32.$$

Let's see if we can get the same answer numerically.

Compare to numerical method

$$f(x_1,x_2)=3x_1^{\,2}+4x_1x_2-x_2-12.$$

$$\frac{\partial f(x_1,x_2)}{\partial x_1}=\frac{f(x_1+h,x_2)-f(x_1,x_2)}{h}$$

We use $h=x_1\sqrt{\varepsilon}=2\sqrt{2*10^{-16}}=2.83*10^{-8}$

$$\frac{\partial f(x_1,x_2)}{\partial x_1}\approx\frac{f(2+2.83*10^{-8},5)-f(2,5)}{2.83*10^{-8}}$$

$$=\frac{35.00000090560-35.00}{2.83*10^{-8}}$$

$$=32.00000014\quad(32.0003\text{ if use h}=0.001)$$

Wow! No derivative formulas needed in the numerical method.

Quiz 4 Problem using Numerical Method

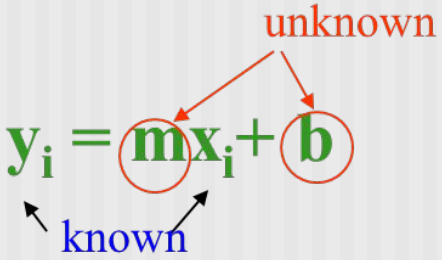
2. *10 points.* 1 hrs. Redo Problem 1b, but use the numerical method to calculate all your partial derivatives, where $h = x_i\sqrt{\epsilon}$.

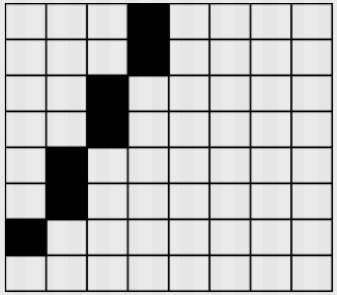
$$\frac{\partial}{\partial x_i} f(x_1, \ldots, x_i, \ldots, x_n) = \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_i - h, \ldots, x_n)}{2h}$$

$$E = \sum_{i=1}^{4} (y_i - mx_i - b)^2$$

$$m^{t+1} = m^t - \alpha \frac{\partial E}{\partial m} \qquad b^{t+1} = b^t - \alpha \frac{\partial E}{\partial b}$$

Pseudo-Inverse Linear Regression





- **To find equation of line- use all data & best fit**

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{10} \end{bmatrix}$$

$$Ax = B$$

$$A^T A\, x = A^T B$$

$$x = [A^T A]^{-1} A^T B$$

$$\begin{bmatrix} m \\ b \end{bmatrix} = \left(\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix}^T \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{10} \end{bmatrix}$$

$$y_i = mx_i + b$$

Pseudo Inverse Line Fitting

$$Ax = B$$

$$A_{n*2}x_{2*1} = B_{n*1}$$

$$[A^T_{2*n} \cdot A_{n*2}]\, x_{2*1} = A^T_{2*n} B_{n*1}$$

$$X = \underbrace{[A^TA]^{-1}A^T}_{A^{\#1}}B$$ (A min error solution regression)

$A^{\#1}$ - Pseudo Inverse

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{10} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_{10} \end{bmatrix} \Rightarrow \text{Pseudo Inverse} \Rightarrow \begin{bmatrix} m \\ b \end{bmatrix} = \left[\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{10} & 1 \end{bmatrix}^T \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{10} & 1 \end{bmatrix} \right]^{-1} \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{10} & 1 \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_{10} \end{bmatrix}$$

Quiz 4. Use Pseudo-Inverse.

3. *10 points.* 1 hrs. Solve Problem 1b using Pseudo-Inverse Linear Regression to find (a, b, c, d). You can use numpy or other tools to invert matrices.

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{10} \end{bmatrix}$$

$$\begin{bmatrix} m \\ b \end{bmatrix} = \left[\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix}^T \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix} \right]^{-1} \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_{10} & 1 \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{10} \end{bmatrix}$$

Pseudo Inverse = Min Least Squares Error

$$\vec{y} = A\vec{x} \qquad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\begin{aligned} \min_{x_1, x_2, x_3} E(x_1, x_2, x_3) &= \sum_{i=1}^{4} (y_i - f(x_1, x_2, x_3))^2 \\ &= ||\vec{y} - A\vec{x}||^2 \\ &= (\vec{y} - A\vec{x})^T (\vec{y} - A\vec{x}) \end{aligned}$$

$$\begin{aligned} E(x_1, x_2, x_3) &= (\vec{y} - A\vec{x})^T (\vec{y} - A\vec{x}) \\ &= \left(\vec{y}^T - \vec{x}^T A^T\right)(\vec{y} - A\vec{x}) \\ &= \vec{y}^T y - \vec{y}^T A\vec{x} - \vec{x}^T A^T \vec{y} + \vec{x}^T A^T A\vec{x} \\ &= \vec{y}^T y - \vec{y}^T A\vec{x} - \vec{y}^T A\vec{x} + \vec{x}^T A^T A\vec{x} \\ &= \vec{y}^T y - 2\vec{y}^T A\vec{x} + \vec{x}^T A^T A\vec{x} \end{aligned}$$

$$(\vec{y} - A\vec{x})^T (\vec{y} - A\vec{x}) = \begin{bmatrix} v_1 & v_1 & v_3 & v_4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

$$\begin{aligned} \frac{\partial E(x_1, x_2, x_3)}{\partial \vec{x}} &= \frac{\partial}{\partial \vec{x}} \left[\vec{y}^T y - 2\vec{y}^T A\vec{x} + \vec{x}^T A^T A\vec{x}\right] \\ &= -2\vec{y}^T A + 2\vec{x}^T A^T A \end{aligned}$$

$$\frac{\partial E(x_1, x_2, x_3)}{\partial \vec{x}} = 0$$

$$\begin{aligned} -2\vec{y}^T A + 2\vec{x}^T A^T A &= 0 \\ 2\vec{x}^T A^T A &= 2\vec{y}^T A \ ; \text{transpose both sides} \\ A^T A\,\vec{x} &= A^T \vec{y} \\ \vec{x} &= \left(A^T A\right)^{-1} A^T \vec{y} \end{aligned}$$

Prove 2 equations are indeed true

1. $$E(x_1, x_2, x_3) = \vec{y}^T y - \vec{y}^T A\vec{x} - \vec{x}^T A^T \vec{y} + \vec{x}^T A^T A\vec{x}$$
$$= \vec{y}^T y - \vec{y}^T A\vec{x} - \vec{y}^T A\vec{x} + \vec{x}^T A^T A\vec{x}$$
$$\vec{y}^T A\vec{x} = \vec{x}^T A^T \vec{y}\,;\ \text{Is this true?}$$

2. $$\frac{\partial E(x_1, x_2, x_3)}{\partial \vec{x}} = \frac{\partial}{\partial \vec{x}}\left[\vec{y}^T y - 2\vec{y}^T A\vec{x} + \vec{x}^T A^T A\vec{x}\right]$$

$$= -2\vec{y}^T A + 2\vec{x}^T A^T A$$

$$\frac{\partial}{\partial \vec{x}}\left[\vec{x}^T A^T A\vec{x}\right] = 2\vec{x}^T A^T A\,;\ \text{Is this true?}$$

$\vec{y}^T A\vec{x} = \vec{x}^T A^T \vec{y}$; Is this true?

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$\begin{bmatrix} \sum_{i=1}^{4} a_{i1}y_i & \sum_{i=1}^{4} a_{i2}y_i & \sum_{i=1}^{4} a_{i3}y_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{3} a_{1j}x_j & \sum_{j=1}^{3} a_{2j}x_j & \sum_{j=1}^{3} a_{3j}x_j & \sum_{j=1}^{3} a_{4j}x_j \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$\sum_{i=1}^{4}\sum_{j=1}^{3} a_{ij}y_i x_j = \sum_{i=1}^{4}\sum_{j=1}^{3} a_{ij}y_i x_j$$

$$\frac{\partial}{\partial \vec{x}}\left[\vec{x}^T A^T A \vec{x}\right] = 2\vec{x}^T A^T A \text{ ; Is this true?}$$

$$\vec{x}^T A^T A \vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^{3} a_{1j}x_j & \sum_{j=1}^{3} a_{2j}x_j & \sum_{j=1}^{3} a_{3j}x_j & \sum_{j=1}^{3} a_{4j}x_j \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{3} a_{1j}x_j \\ \sum_{j=1}^{3} a_{2j}x_j \\ \sum_{j=1}^{3} a_{3j}x_j \\ \sum_{j=1}^{3} a_{4j}x_j \end{bmatrix}$$

$$= \sum_{i=1}^{4} \left(\sum_{j=1}^{3} a_{ij}x_j \right)^2$$

$\frac{\partial}{\partial \vec{x}}\left[\vec{x}^T A^T A \vec{x}\right] = 2\vec{x}^T A^T A$; Is this true?

$$\frac{\partial}{\partial \vec{x}}\left[\vec{x}^T A^T A \vec{x}\right] = \frac{\partial}{\partial \vec{x}}\left[\sum_{i=1}^{4}\left(\sum_{j=1}^{3} a_{ij}x_j\right)^2\right]$$

$$= \sum_{i=1}^{4}\left[2 \cdot \left(\sum_{j=1}^{3} a_{ij}x_j\right) \cdot \frac{\partial}{\partial \vec{x}}\left[\sum_{j=1}^{3} a_{ij}x_j\right]\right]$$

$$= 2\left[\sum_{i=1}^{4} a_{i1}\sum_{j=1}^{3} a_{ij}x_j \quad \sum_{i=1}^{4} a_{i2}\sum_{j=1}^{3} a_{ij}x_j \quad \sum_{i=1}^{4} a_{i3}\sum_{j=1}^{3} a_{ij}x_j\right]$$

$\frac{\partial}{\partial \vec{x}}\left[\vec{x}^T A^T A \vec{x}\right] = 2\vec{x}^T A^T A$; Is this true?

$$\frac{\partial}{\partial \vec{x}}\left[\vec{x}^T A^T A \vec{x}\right] = 2\left[\sum_{i=1}^{4} a_{i1} \sum_{j=1}^{3} a_{ij}x_j \quad \sum_{i=1}^{4} a_{i2} \sum_{j=1}^{3} a_{ij}x_j \quad \sum_{i=1}^{4} a_{i3} \sum_{j=1}^{3} a_{ij}x_j\right]$$

$$\vec{x}^T A^T A = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

$$= \left[\sum_{j=1}^{3} a_{1j}x_j \quad \sum_{j=1}^{3} a_{2j}x_j \quad \sum_{j=1}^{3} a_{3j}x_j \quad \sum_{j=1}^{3} a_{4j}x_j\right] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

$$= \left[a_{11}\sum_{j=1}^{3} a_{1j}x_j + a_{21}\sum_{j=1}^{3} a_{2j}x_j \ldots \quad a_{12}\sum_{j=1}^{3} a_{1j}x_j + a_{22}\sum_{j=1}^{3} a_{2j}x_j \ldots \quad \ldots\right]$$

$$= \left[\sum_{i=1}^{4} a_{i1} \sum_{j=1}^{3} a_{ij}x_j \quad \sum_{i=1}^{4} a_{i2} \sum_{j=1}^{3} a_{ij}x_j \quad \sum_{i=1}^{4} a_{i3} \sum_{j=1}^{3} a_{ij}x_j\right]$$

Matrix Derivatives

Proposition 7 *Let the scalar* α *be defined by*

$$\alpha = \mathbf{y}^T \mathbf{A} \mathbf{x}$$

where $\mathbf{y}$ *is* $m \times 1$, $\mathbf{x}$ *is* $n \times 1$, $\mathbf{A}$ *is* $m \times n$, *and* $\mathbf{A}$ *is independent of* $\mathbf{x}$ *and* $\mathbf{y}$, *then*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{y}^T \mathbf{A}$$

and

$$\frac{\partial \alpha}{\partial \mathbf{y}} = \mathbf{x}^T \mathbf{A}^T$$

Matrix Derivatives

Proposition 8 *For the special case in which the scalar* α *is given by the quadratic form*

$$\alpha = \mathbf{x}^T \mathbf{A} \mathbf{x} \tag{43}$$

where $\mathbf{x}$ *is* $n \times 1$, $\mathbf{A}$ *is* $n \times n$, *and* $\mathbf{A}$ *does not depend on* $\mathbf{x}$, *then*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T \left(\mathbf{A} + \mathbf{A}^T \right) \tag{44}$$

Proof: *By definition*

$$\alpha = \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} x_i x_j \tag{45}$$

Differentiating with respect to the k*th element of* $\mathbf{x}$ *we have*

$$\frac{\partial \alpha}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{i=1}^{n} a_{ik} x_i \tag{46}$$

for all $k = 1, 2, \ldots, n$, *and consequently,*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T \mathbf{A}^T + \mathbf{x}^T \mathbf{A} = \mathbf{x}^T \left(\mathbf{A}^T + \mathbf{A} \right) \tag{47}$$

Matrix Derivatives

Proposition 9 *For the special case where* $\mathbf{A}$ *is a symmetric matrix and*

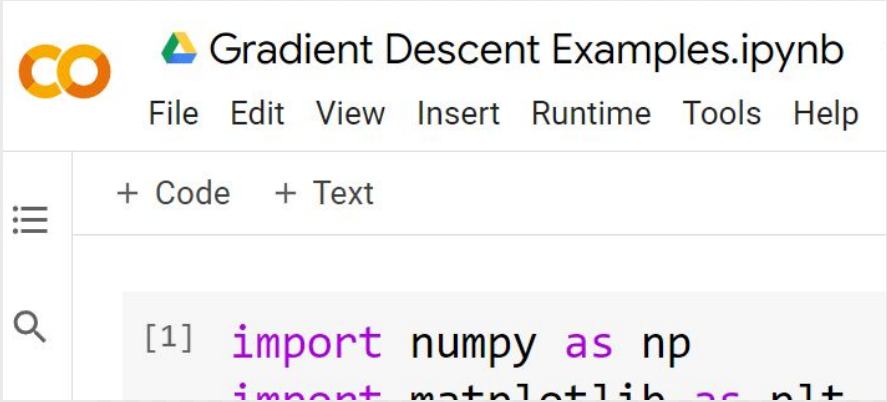
$$\alpha = \mathbf{x}^T \mathbf{A} \mathbf{x}$$

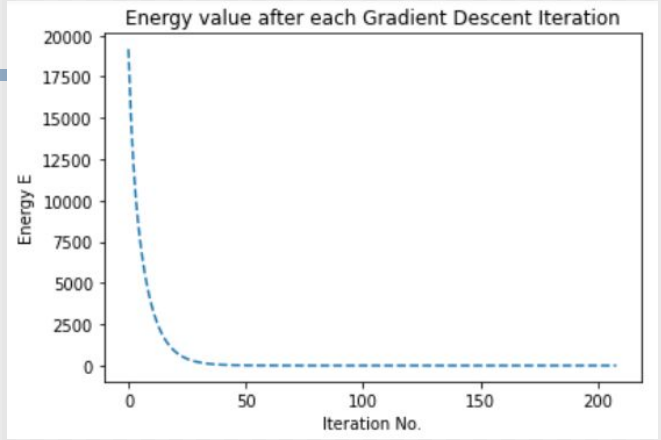
where $\mathbf{x}$ *is* $n \times 1$, $\mathbf{A}$ *is* $n \times n$, *and* $\mathbf{A}$ *does not depend on* $\mathbf{x}$, *then*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = 2\mathbf{x}^T \mathbf{A}$$

Proof: *This is an obvious application of Proposition 8.* $\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T \left(\mathbf{A} + \mathbf{A}^T \right)$

Run sample program

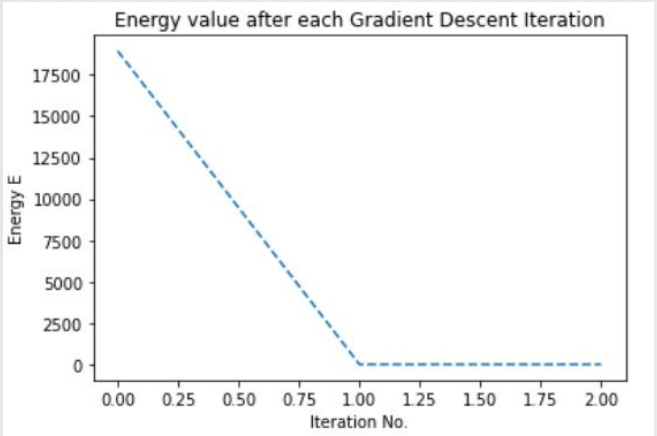




Example of find (m, b) for y = mx + b using 3 methods.

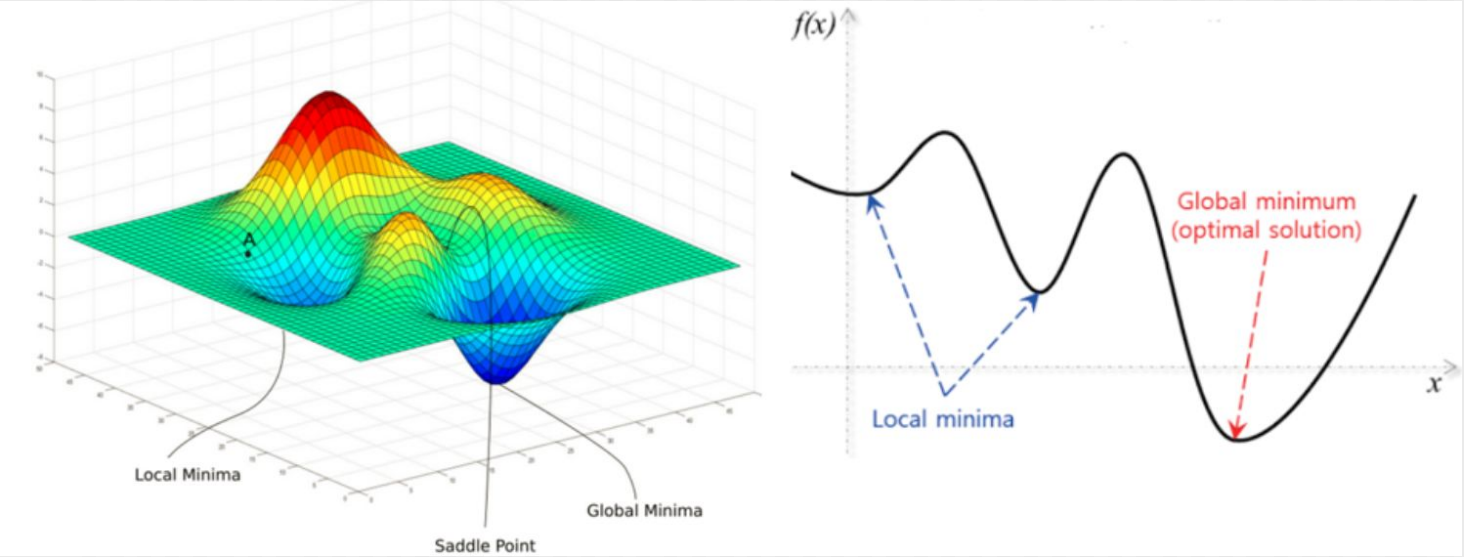
1. Simple Gradient Descent
2. Pseudo-Inverse
3. Gauss-Newton algorithm

- We also try derivatives using a numerical method.
- Scikit curve fitting libraries are also shown



Local Minimum Problem in Greedy Algorithms

- Gradient Descent is a greedy algorithm where in each step you take the best direction.
- Since the cost function is non-quadratic, it is likely to have many local minima.



From https://laptrinhx.com/neural-networks-part-3-understanding-back-propagation-learning-rates-3031386133/

Simulated Annealing. Stochastic/non-greedy.

Algorithm 2: Simulated Annealing Optimizer

$T \leftarrow T_{max}$
$\mathbf{x} \leftarrow$ generate the initial candidate solution
$E \leftarrow E(\mathbf{x})$ compute the energy of the initial solution
while $(T > T_{min})$ **and** $(E > E_{th})$ **do**
  $\mathbf{x}_{\text{new}} \leftarrow$ generate a new candidate solution
  $E_{\text{new}} \leftarrow$ compute the energy of the new candidate $\mathbf{x}_{new}$
  $\Delta E \leftarrow E_{new} - E$
  if `Accept` $(\Delta E, T)$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}_{new}$
    $E \leftarrow E_{new}$
  end
  $T \leftarrow \frac{T}{\alpha}$ cool the temperature
end
return x

From https://www.baeldung.com/cs/simulated-annealing

Simulated Annealing. Stochastic/non-greedy.

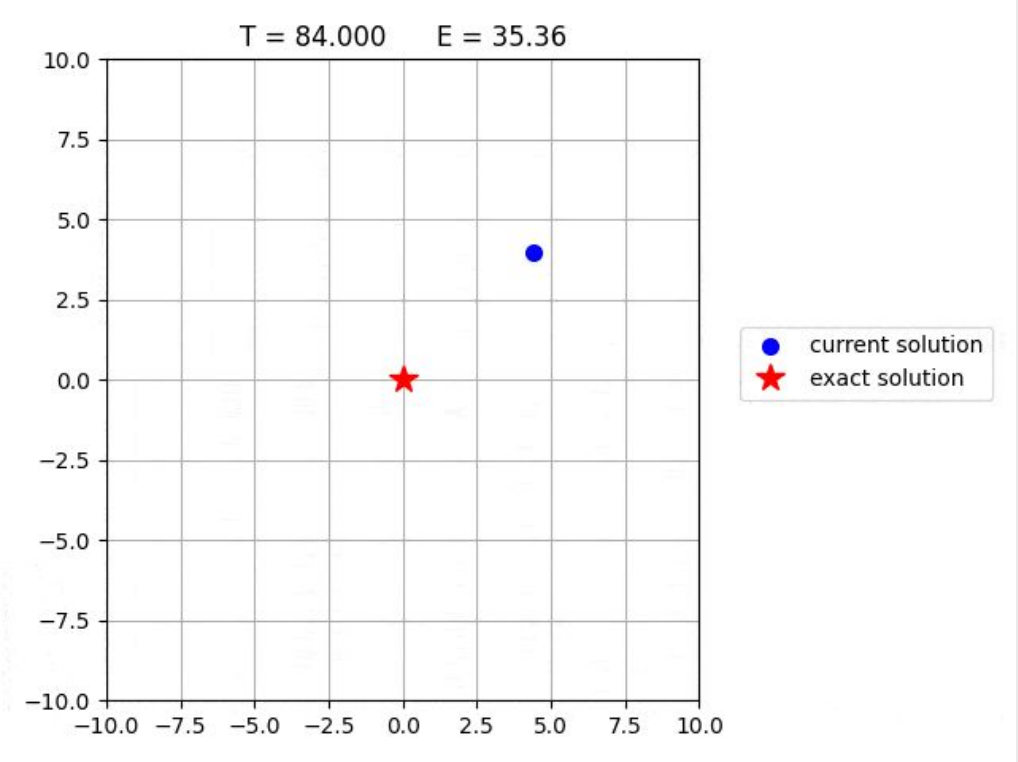
Algorithm 1: Acceptance Function

Data: T, ΔE - the temperature and the energy variation between the new candidate solution and the current one.
Result: Boolean value that indicates if the new solution is accepted or rejected.

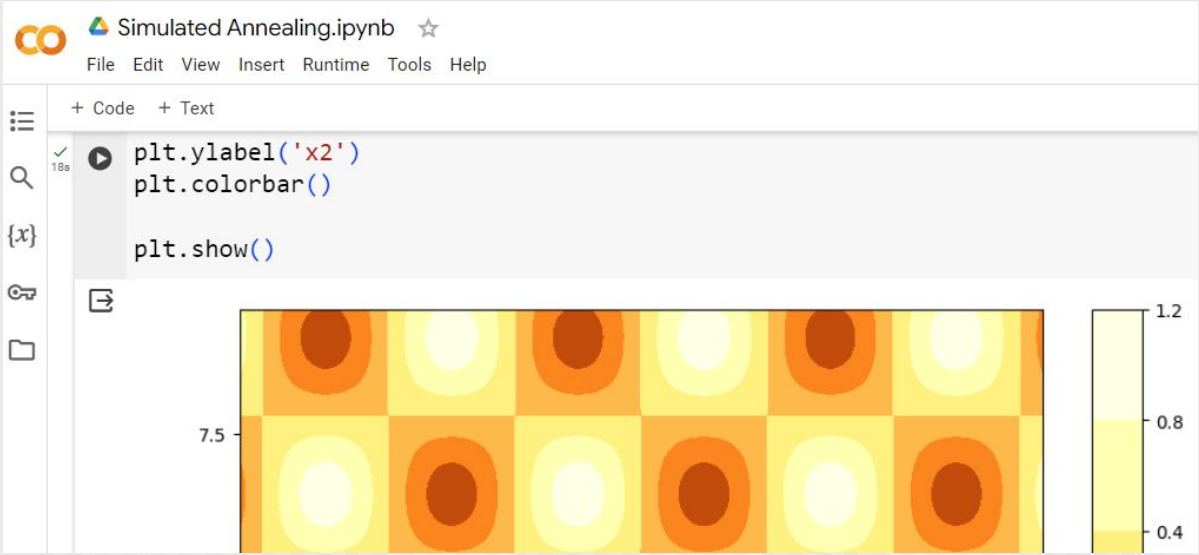
```
if (ΔE < 0) then
    return True;
else
    r ← generate a random value in the range [0, 1)
    if (r < exp(−ΔE/T)) then
        return True
    else
        return False
    end
end
```

From https://www.baeldung.com/cs/simulated-annealing

Example $f(x,y) = x^2 + y^2$. Search a grid of size 101 x 101 placed in the square area defined by (x,y) in [-10, 10] x [-10, 10] or step of 0.2. Cooling $\alpha = 0.84$ and the initial solution (x,y) = (4,4). At each step, a new solution is generated by randomly shifting the current solution by ± 0.2 in x and y direction.



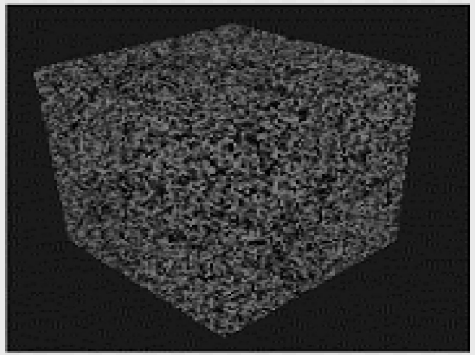
Simulated Annealing Example and Quiz Problem



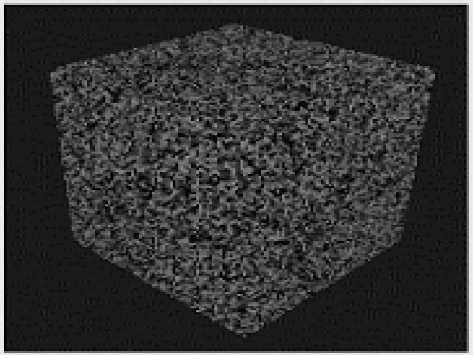
4. *10 points.* Change the function in "Simulated Annealing.ipynb" shown in class to the one in our lecture: $f(x,y) = x^2 + y^2$. Change the program to show more intermediate states (x1, x2, f) instead of only accepted states. Show the output movement image for (x1, x2) as a blue line.

Example Simulated Annealing in Stereo Vision

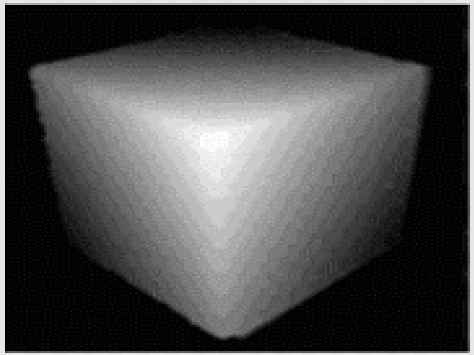
The use of 2 cameras (eyes) to get 2 images. We triangulate to compute depth:



Left Image



Right Image



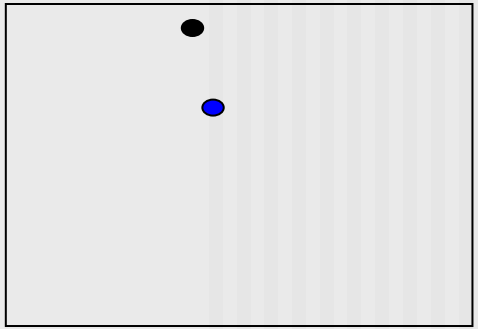
Disparity

The Stereo Pair

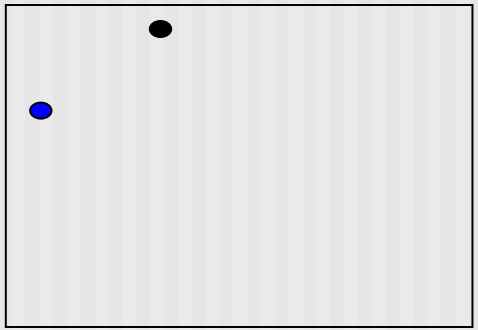
The Stereo Pair

Stereo Disparity

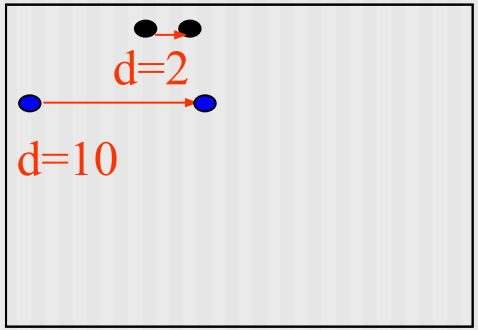
Disparity - different between projected point in left and right eye images.
Often represented as disparity with respect to right image.



$I_L(i, j)$



$I_R(i, j)$



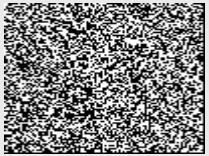
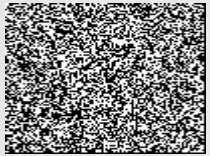
Overlay

Blue dot - nearer since higher disparity. $[I^R(i,j) - I^L(i,j+D(i,j))]^2$

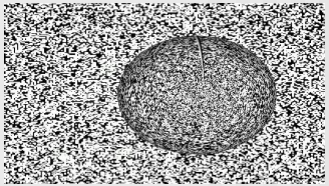
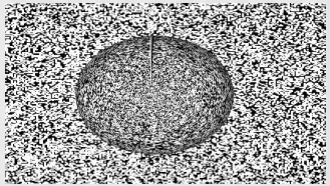
Stereo Test Images

Random-dot stereogram of floating square:

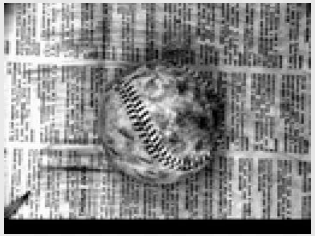
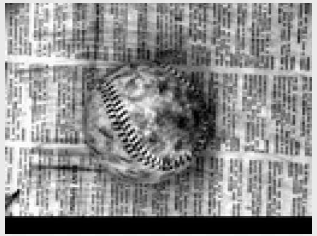

Random-dot stereogram of wedding cake:

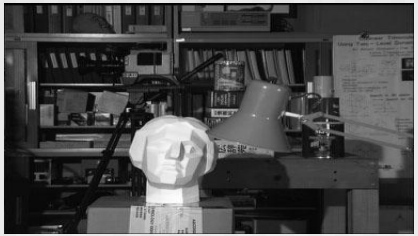
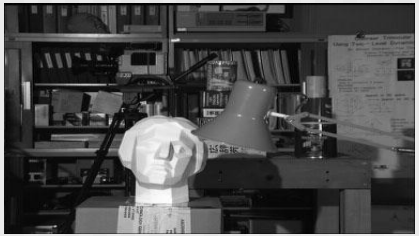
Random dot sphere:

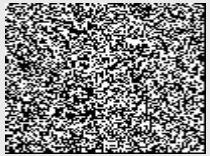



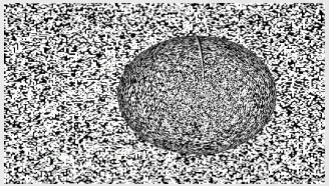
Baseball Test Image:

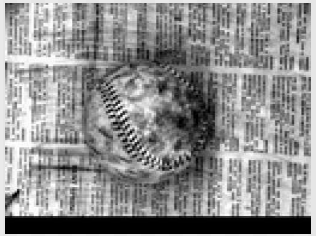



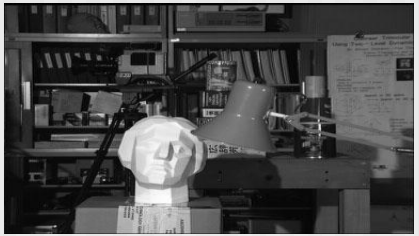
Tsukuba Test Image:

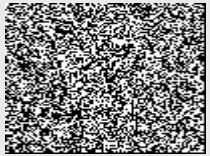



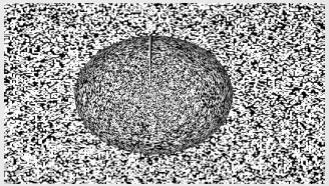


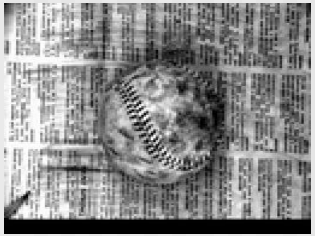


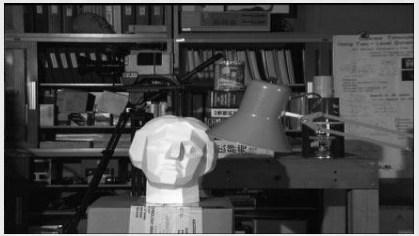




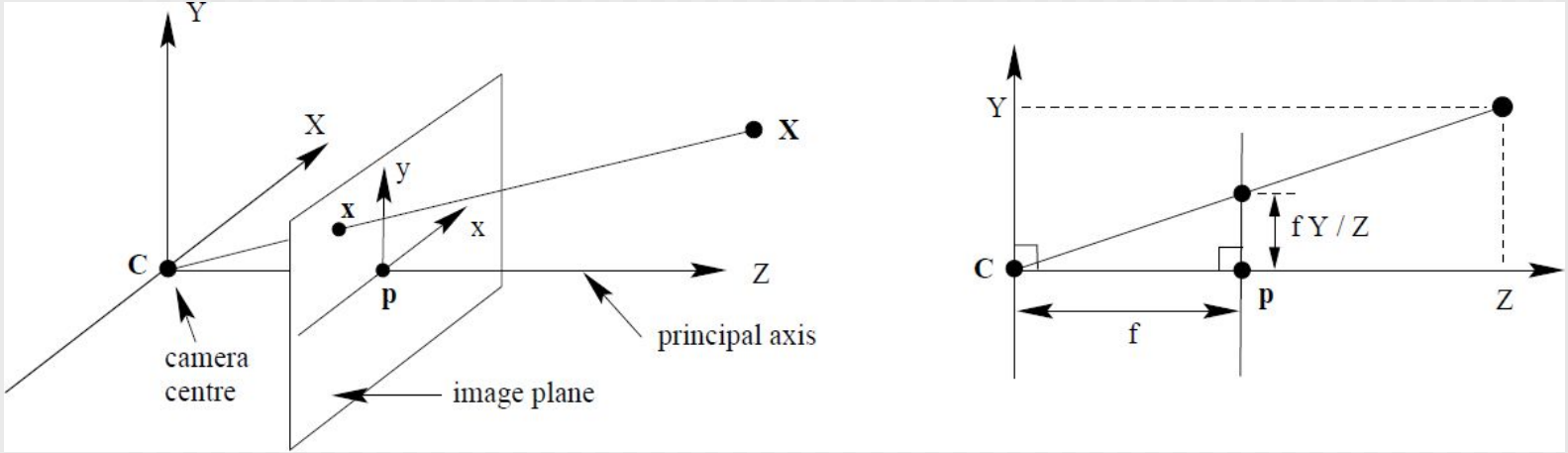




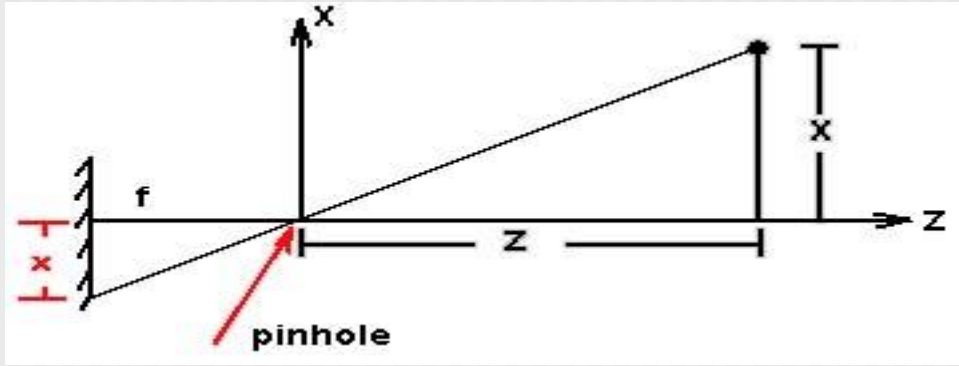




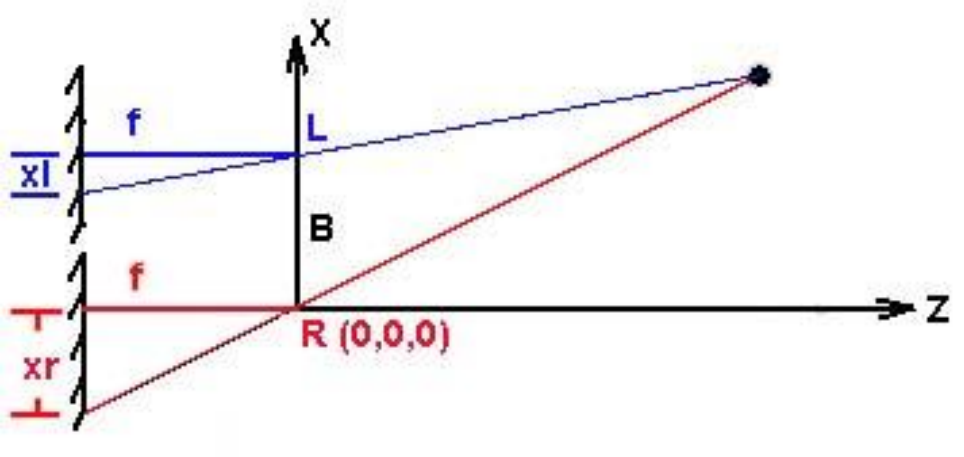
Pinhole Camera Projection Model



$$\frac{x}{f} = \frac{X}{Z} \quad \Rightarrow \quad x = \frac{Xf}{Z} \qquad y = \frac{Yf}{Z}$$



Stereo Vision: Disparity



$$x_r = \frac{Xf}{Z} \qquad x_l = \frac{(X-B)f}{Z}$$

B – Baseline Interocular Distance

Depth from Stereo

$$x_r = \frac{Xf}{Z} \qquad x_l = \frac{(X - B)f}{Z}$$

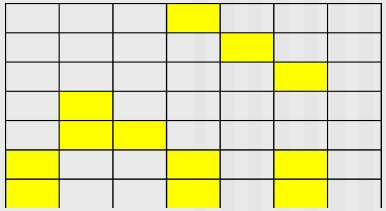
$$x_r - x_l = \delta = \frac{Xf}{Z} - \frac{Xf}{Z} + \frac{Bf}{Z} = \frac{Bf}{Z} = \frac{K}{Z}$$

$$\delta = \frac{K}{Z} \qquad \delta \propto \frac{1}{Z} \qquad Z = \frac{K}{\delta}$$

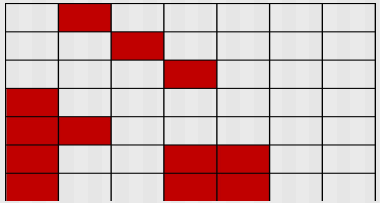
Stereo Disparity

Computing depth is easy once you have disparity

The CHALLENGE is in finding disparity

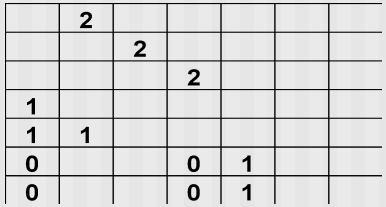


$I_L(i, j)$



$I_R(i, j)$

Overlay



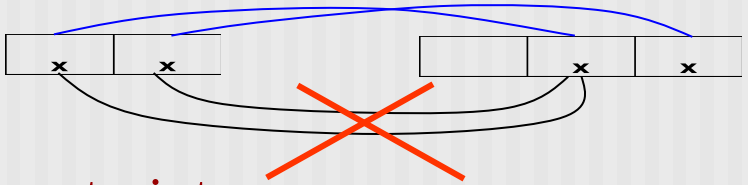
d(i, j)
disparity (δ)

From constraints to cost function

- Matching pixels should have similar color
- Nearby pixels are close in depth – smooth disparity.
- 2 points in one frame should not match the same point in another frame. Called 1:1 Constraint.

Stereo Matching Constraints

1. Data - Right Image color be same as left image
2. Smoothness - Disparity is smooth
3. 1 to 1 Matching



$$E = \sum_{ij} [(I^R(i,j) - I^L(i, j + D(i,j)))^2$$ Data Constraint

$$+ \lambda_1 (D(i+1, j) - D(i,j))^2$$ Smoothness Constraint

$$+ \lambda_1 (D(i, j+1) - D(i,j))^2$$ Smoothness Constraint

$$+ \begin{cases} \lambda_2 & \text{if } D(i,j+1) - D(i,j) == -1 \\ 0 & \text{otherwise} \end{cases}$$ 1 to 1 Matching Constraint

Gradient Descent Update Rule

Find best m,b with y = mx + b. The update rule for E(m, b):

$$m^{t+1} = m^{t} - \alpha \frac{\partial E}{\partial m} \qquad b^{t+1} = b^{t} - \alpha \frac{\partial E}{\partial b}$$

For computing disparity: $E\left(\vec{d}\right) = E(d_{ij}\,;\, i = 1 \ldots R,\, j = 1 \ldots C\,)$

The number of unknowns is R*C. For 100 x 100 pixel image this is 10,000 unknowns.

The Update Rule:

$$d_{ij}^{t+1} = d_{ij}^{t} - \alpha \frac{\partial E\left(\vec{d}\right)}{\partial d_{ij}}$$

$$\vec{d}^{\,t+1} = \vec{d}^{\,t} - \alpha \nabla E\left(\vec{d}\right)$$

Stereo Cost Function

$$E = V_D + \lambda_S V_S + \lambda_M V_M$$

$$V_D = \sum_x^C \sum_y^R \left(\frac{I^R(x,y) - I^L(x + d(x,y), y)}{255} \right)^2$$. The Data Constraint.

$$V_S = \sum_x^C \sum_y^R \frac{1}{2} \left(\frac{d(x,y) - d(x+1,y)}{D_{MAX}} \right)^2 + \frac{1}{2} \left(\frac{d(x,y) - d(x,y+1)}{D_{MAX}} \right)^2$$. The Smoothness Constraint.

$$V_M = \sum_x^C \sum_y^R \begin{cases} 1 & if\,(d(x,y) - d(x+1,y) == 1) \\ 0 & otherwise \end{cases}$$. The One-to-One Matching Constraint.

Note: You should use in the matching constraint a 3x3 neighboring area instead of just 1 pixel (it will help). So, in more detail, the data constraint should be:

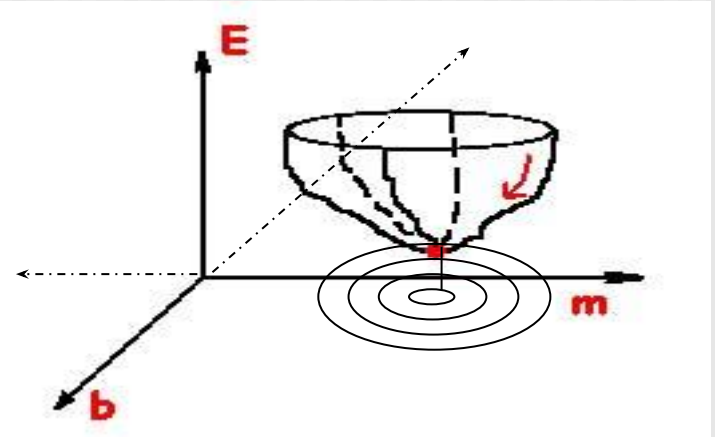
$$V_D = \sum_x^C \sum_y^R \frac{1}{9} \left(\sum_{i=-1}^1 \sum_{j=-1}^1 \left(\frac{I^R(x+j, y+i) - I^L(x + d(x,y) + j, y+i)}{255} \right)^2 \right)$$. The Data Constraint.

Gradient Descent Update Rule

Solving for 2 Unknowns (m, b):

$$E(m,b) = \sum (y_i - mx_i - b)^2$$

$$\left(\frac{dm}{dt}, \frac{db}{dt}\right) = -\alpha\left(\frac{\partial E}{\partial m}, \frac{\partial E}{\partial b}\right)$$



Solving for 10,000 unknowns d_{ij} :

$$E\left(\vec{d}\right) = E(d_{11}, d_{12}, d_{RC})$$

$$\left[\frac{dd_{11}}{dt} \quad \frac{dd_{12}}{dt} \quad \cdots \quad \frac{dd_{RC}}{dt}\right] = -\alpha\left[\frac{\partial E\left(\vec{d}\right)}{\partial d_{11}} \quad \frac{\partial E\left(\vec{d}\right)}{\partial d_{12}} \quad \cdots \quad \frac{\partial E\left(\vec{d}\right)}{\partial d_{RC}}\right]$$

$$E\left(\vec{d}\right) = E(d_{ij}\,;\, i = 1 \ldots R,\, j = 1 \ldots C\,)$$

$$d_{ij}^{t+1} = d_{ij}^{t} - \alpha \frac{\partial E\left(\vec{d}\right)}{\partial d_{ij}}$$

For E(**d**):

$$\vec{d}^{\,t+1} = \vec{d}^{\,t} - \alpha \nabla E\left(\vec{d}\right)$$

1. Computing the gradient ∇E(**d**) is difficult
2. The cost function is non-quadratic because non-linear data fitting model
3. Values of unknowns **d** are discrete integers

$$E = V_D + \lambda_S V_S + \lambda_M V_M$$

$$V_D = \sum_x^C \sum_y^R \left(\frac{I^R(x,y) - I^L(x + d(x,y), y)}{255} \right)^2$$. The Data Constraint.

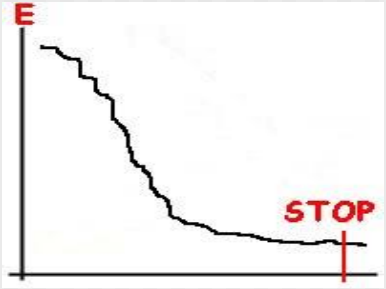
$$V_S = \sum_x^C \sum_y^R \frac{1}{2}\left(\frac{d(x,y) - d(x+1,y)}{D_{MAX}} \right)^2 + \frac{1}{2}\left(\frac{d(x,y) - d(x,y+1)}{D_{MAX}} \right)^2$$. The Smoothness Constraint.

$$V_M = \sum_x^C \sum_y^R \begin{cases} 1 & if\,(d(x,y) - d(x+1,y) == 1) \\ 0 & otherwise \end{cases}$$. The One-to-One Matching Constraint.

Discrete Gradient Descent Algorithm

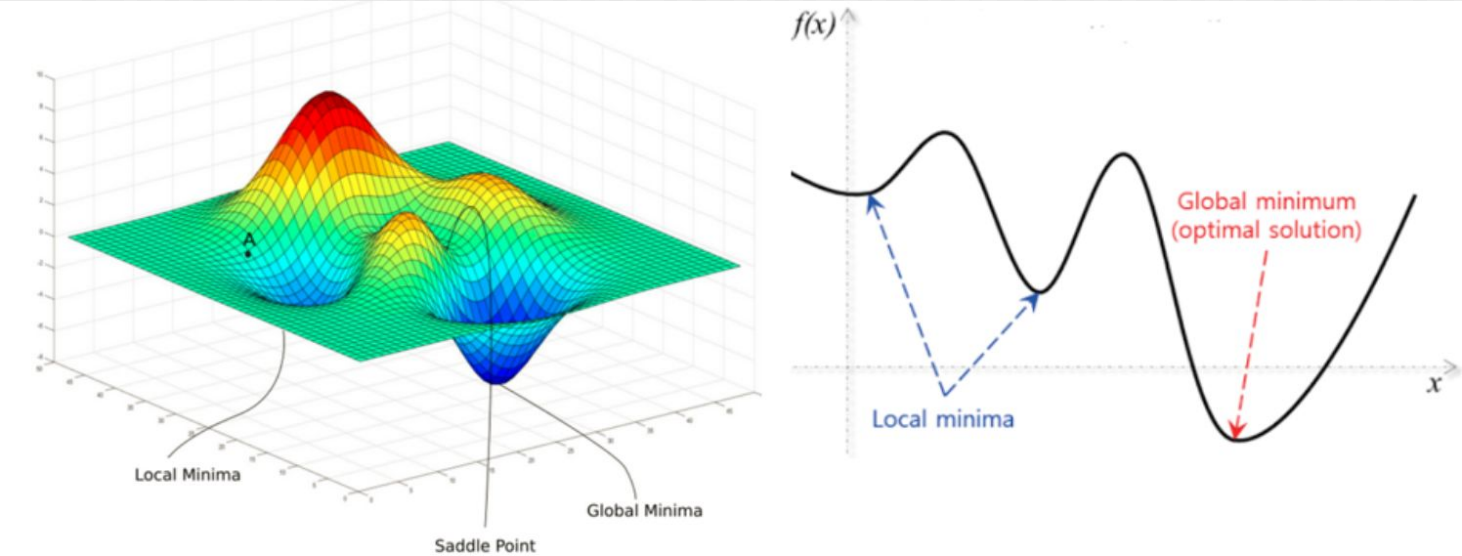
Looks at all possible values and chooses the value with the minimum energy.

1. t=1. $d^0(i,j)$ = Random 0..20 for disparity range.
2. For each pixel (i,j)
   For each discrete disparity s = 0..20
   when $d^t(i,j) = 0$; E_0 = ...
   when $d^t(i,j) = 1$; E_1 = ...
   ...
   when $d^t(i,j) = 20$; E_{20} = ...
   $d^{t+1}(i,j) = m$ where E_m is minimum of E_i; i = 0..20
3. t = t+1
4. Repeat steps 2-3 until E is stable.



Local Minimum Problem in Greedy Algorithms

- Gradient Descent is a greedy algorithm where in each step you take the best direction.
- Since the cost function is non-quadratic, it is likely to have many local minima.



From https://laptrinhx.com/neural-networks-part-3-understanding-back-propagation-learning-rates-3031386133/

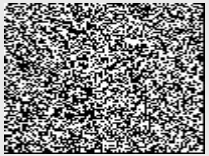
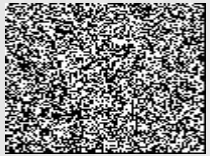
Stochastic Descent using Simulated Annealing

1. T = a high value. t=1. $d^0(i,j)$ = Random 0..20.
2. For each pixel (i,j)
   For each discrete disparity s = 0..20
   when $d^t(i,j) = 0$; $E_0 = \ldots$; $P_0 = e^{-E0/T}$
   when $d^t(i,j) = 1$; $E_1 = \ldots$; $P_1 = e^{-E1/T}$
   ...
   when $d^t(i,j) = 20$; $E_{20} = \ldots$; $P_{20} = e^{-E20/T}$
   $Z = \text{sum}(P_i)$ for i = 1..20.
   $P_i = P_i/Z$ for i = 1..20.
   $d^{t+1}(i,j)$ = Sampled state m from PDF P.
3. t = t+1. Reduct T by T = T * 0.99.
4. Repeat steps 2-3 until E is stable.

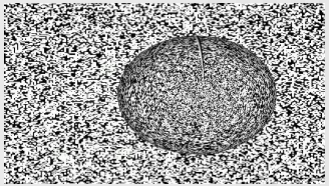
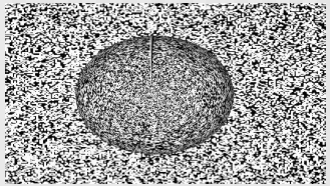
Stereo Test Images

Random-dot stereogram of floating square:

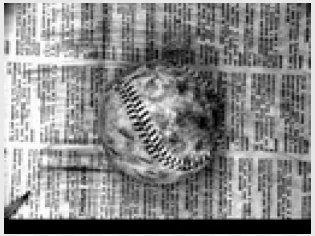
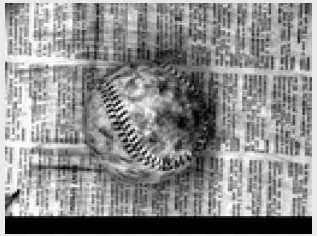

Random-dot stereogram of wedding cake:

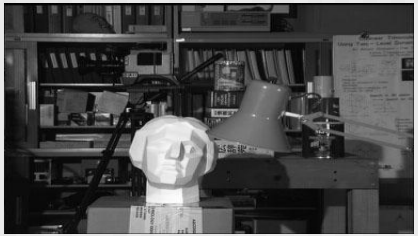
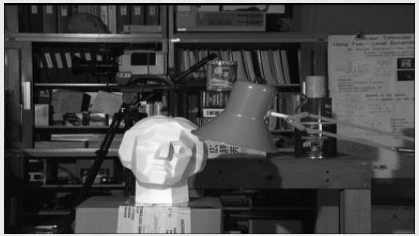
Random dot sphere:

Baseball Test Image:

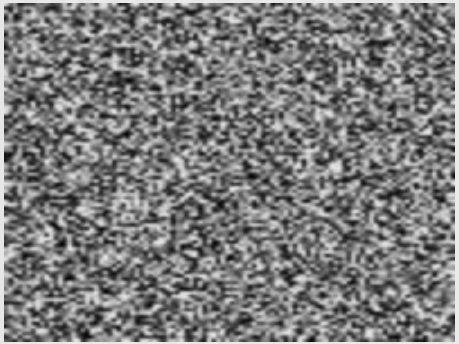



Tsukuba Test Image:

Stereo Result using Stochastic Descent

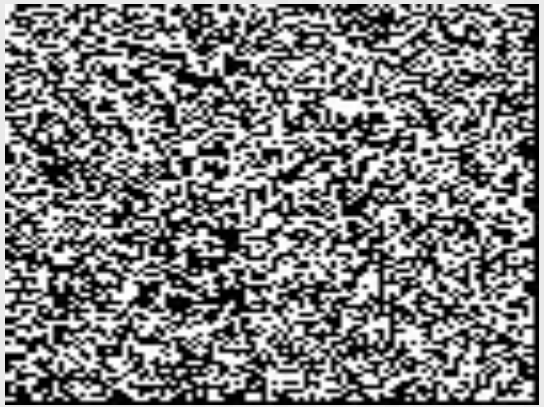
Random-dot stereogram of floating square:



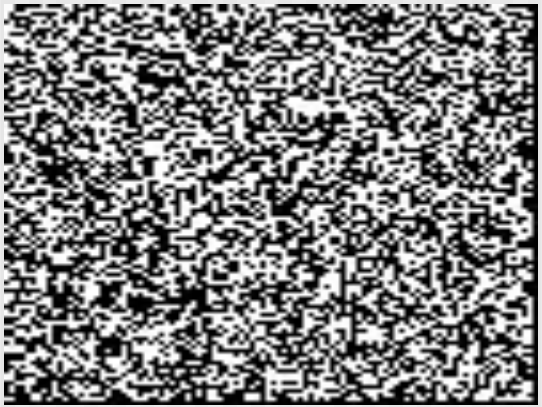
Stereo Result using Stochastic Descent

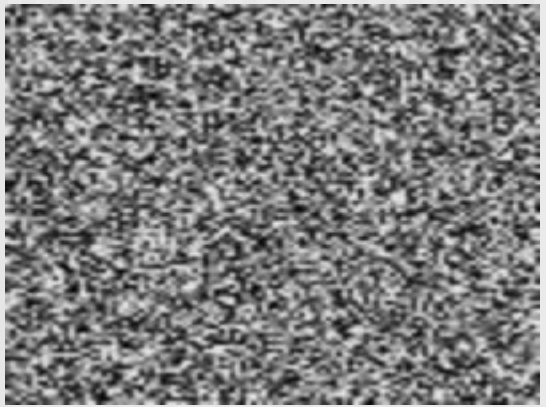
Wedding Cake Random Dot Stereogram:

Left Image



Right Image

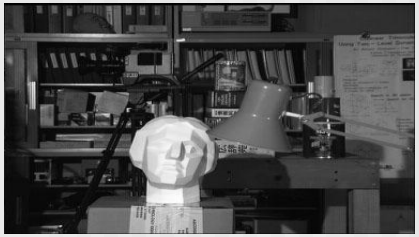




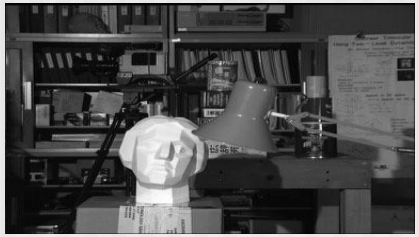
Stereo Result using Stochastic Descent

The Tsukuba benchmark test image:

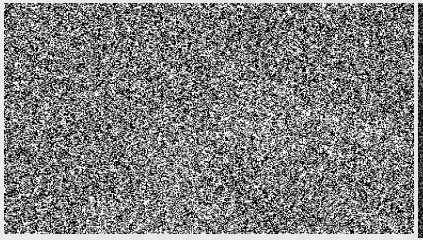
Left Image

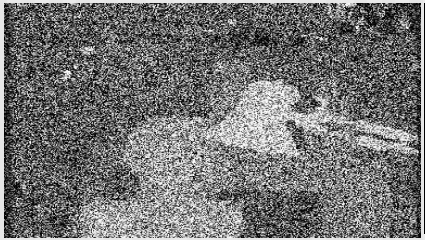


Right Image



Ground Truth Disparity





Iteration 0 4 30 99

How to get PDF P(i) from E(i)?

proportional

$$P_i \propto e^{-\frac{E_i}{T}}$$

$$P_i = \frac{e^{-E_i/T}}{Z}$$

E – Energy Cost

$e = 2.718..$

- Probability of each state is inversely proportional to state's E Cost
- Z: probability normalizing factor to make $\sum P_i = 1$.

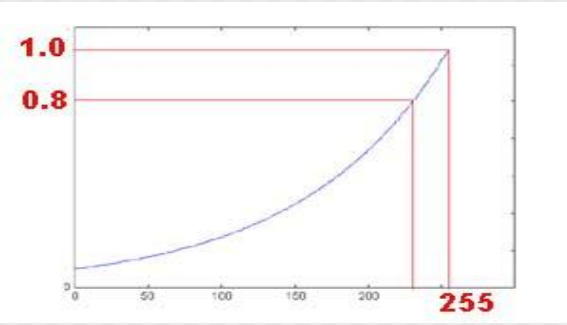
How to sample for a new state from PDF?

Sample for a new state X from PDF P(S = k). Set F(x,y) = X.

Non-Greedy: Select a state X ~~with minimum E~~ so that the chance of selecting a state is inversely proportional to the Cost E of that state.

Consider a 5 state example: S0..S4.

State	PDF P(S = i)	CDF P(S ≤ i)
S0	0.2	0.2; [0.0, 0.2]
S1	0.1	0.3; (0.3, 0.4]
S2	0.1	0.4; (0.3, 0.4]
S3	0.2	0.6; (0.4, 0.6]
S4	0.4	1.0; (0.6, 1.0]



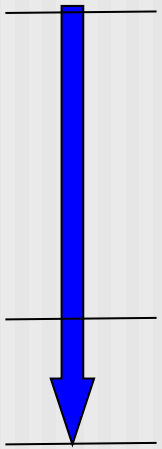
Sampling:

If Rand(0..1) = 0.43 then F(x,y) is 3.
If Rand (0..1) = 0.14 then F(x,y) is 0.
If Rand (0..1) = 0.61 then F(x,y) is 4.

Why reduce T?

$$P_i = \frac{e^{-E_i/T}}{Z}$$

T High - Random Walk
(all states equally likely)



100% - Too High, random states accepted.

25% - A good starting T

Approach Gradient Descent. Ending T.

T Low or nearly zero becomes
Gradient Descent (Greedy)

Why reduce T? T high vs. T low.

Analyze following 3 States 0, 1, and 2 at 3 temperature values:

	E(0) = 2 $P_0 = e^{-2/T}$	E(1) = 3 $P_1 = e^{-3/T}$	E(2) = 4 $P_2 = e^{-4/T}$
T = 100	P' = 0.98 P = 0.33	P' = 0.97 P = 0.33	P' = 0.96 P = 0.33
T = 10	P' = 0.82 P = 0.36	P' = 0.74 P = 0.33	P' = 0.67 P = 0.31
T = 0.1	P' = $2.06*10^{-9}$ P = 0.999	P' = $9.3*10^{-14}$ P = 0.00001	P' = $4.2*10^{-18}$ P = 0.000001

T controls the distribution of the probability distribution function. At High T the system is less biased by E. As T lowers it becomes more **greedy** and gives higher chance of selecting a lower E solution. At a very low T, the system becomes fully greedy, making the min energy state have a probability of 1; essentially, turning into gradient descent to make sure the algorithm finally converges.