An Example: Matrix Pseudoinverse

Solve $\quad 2x_1 + 3x_2 + x_3 = 4$

$$\underline{x}^* = \begin{bmatrix} 0.57 \\ 0.86 \\ 0.29 \end{bmatrix}$$

$L = X_1^2 + X_2^2 + X_3^2 + \lambda(2X_1 + 3X_2 + X_3 - 4)$

$\partial L/\partial X_1 = 2X_1 + 2\lambda = 0 \quad \Rightarrow \quad X_1 = -\lambda$

$\partial L/\partial X_2 = 2X_2 + 3\lambda = 0 \quad \Rightarrow \quad X_2 = -1.5\lambda$

$\partial L/\partial X_3 = 2X_3 + \lambda = 0 \quad \Rightarrow \quad X_3 = -0.5\lambda$

$\partial L/\partial \lambda = 2X_1 + 3X_2 + X_3 - 4 = 0$

$-2\lambda - 4.5\lambda - 0.5\lambda - 4 = 0$

$\lambda = -0.57$

$$\underline{x}^* = \begin{bmatrix} 0.57 \\ 0.86 \\ 0.29 \end{bmatrix}$$

$$2x_1 + 3x_2 + x_3 = 4$$

$$\begin{bmatrix} 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = 4$$

$$\underline{x} = A^T \left(A\, A^T\right)^{-1} \underline{b}$$

Ans.: $\underline{x}^* = \begin{bmatrix} 0.57 \\ 0.86 \\ 0.29 \end{bmatrix}$

Numerical Method: Unconstrained Problem

- Steepest Descent Method (Gradient based method)

- Cauchy (1847)

minimize $f\left(\underline{x}\right)$ $where\ \underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

- At any point on the objective function, the function decreases at the maximum rate along its negative gradient

- Move along this direction iteratively in small steps

Steepest Descent Derivation :

- $F = f(x_1, x_2, \ldots\ldots\ldots\ldots x_n)$

- $df = \left(\frac{\partial f}{\partial x_1}\right) dx_1 + \left(\frac{\partial f}{\partial x_2}\right) dx_2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + \left(\frac{\partial f}{\partial x_n}\right) dx_n \quad + \text{ h.o.t.}$

$$\simeq \begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \ldots\ldots & \frac{\partial f}{\partial x_n} \end{pmatrix} \begin{pmatrix} dx_1 \\ dx_2 \\ dx_n \end{pmatrix}$$

$$= \nabla f^T \, d\underline{x}$$

where:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \cdot \\ \cdot \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad \text{and} \quad d\underline{x} = \begin{pmatrix} dx_1 \\ \cdot \\ \cdot \\ dx_n \end{pmatrix}$$

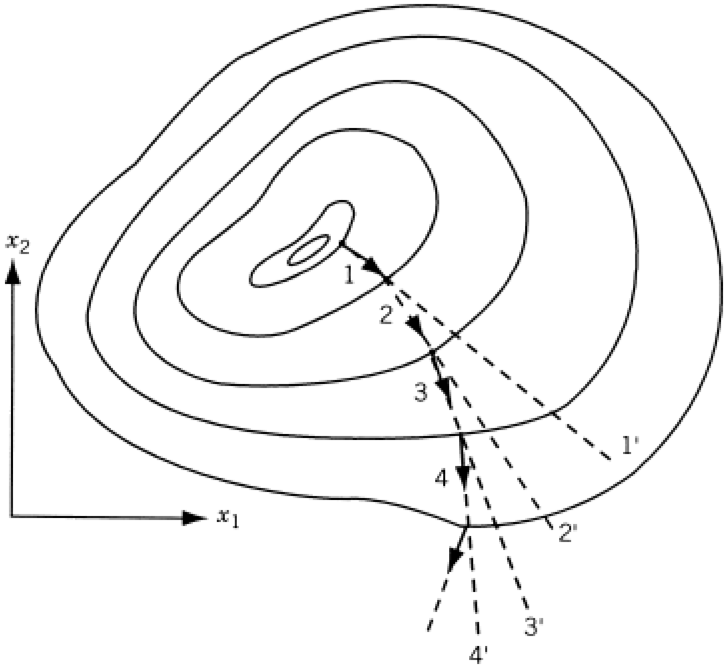
Let $d\underline{x} = \underline{u}\, ds$ where $\underline{u}$ is the unit vector along $d\underline{x}$.

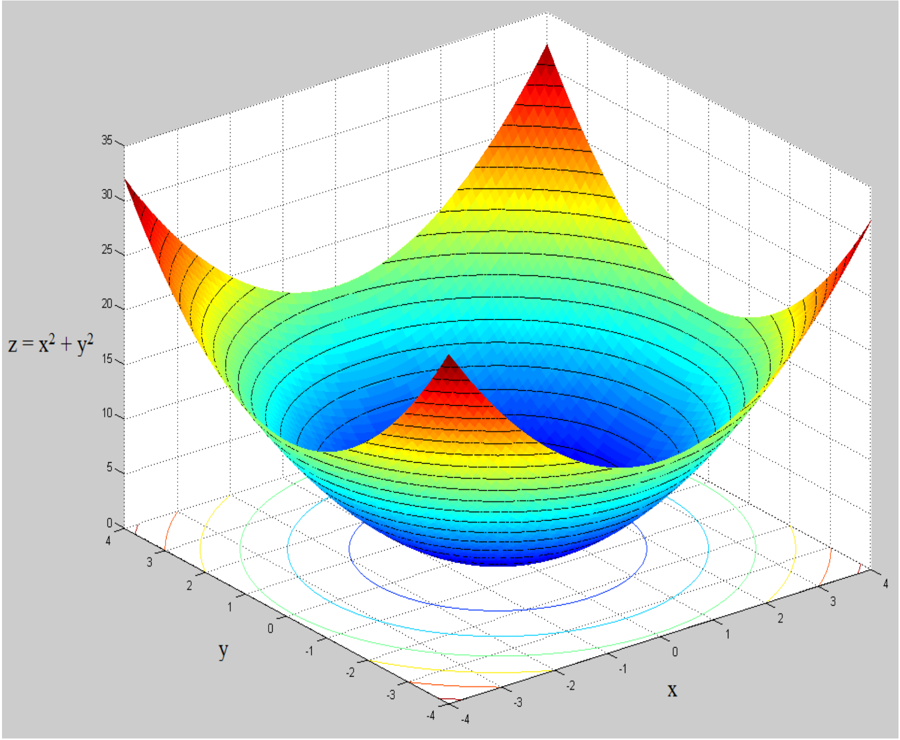
Therefore, $df = \nabla f^T \underline{u}\, ds$

$$\frac{df}{ds} = \nabla f^T \underline{u} = \nabla f \cdot \underline{u} = \| \nabla f \| \cos \Theta$$

- $\frac{df}{ds}$ is -ve maximum when $\Theta = 180°$

  i.e. angle between $d\underline{x}$ and ∇f is 180°.





Steepest ascent directions

- **Steps for Steepest Descent method:**

1. Select an initial arbitrary point $\underline{x}(1)$
2. For ith iteration, calculate the search direction as

$$\underline{S}(i) = -\nabla f\,(\underline{x}(i)).$$

3. Set the next search point as

$$\underline{x}(i+1) = \underline{x}(i) + \lambda_i\,\underline{S}(i) \qquad (\lambda_i > 0)$$

The step size λ_i can be optimized by solving $\frac{d}{d\lambda_i}\left(f\left(\underline{x}(i) + \lambda_i \underline{S}(i)\right)\right) = 0$

4. Test the new point for optimality using some stopping criterion.

If $\underline{x}(i)$ is optimum, then stop the process

Otherwise, iterate

- **Stopping Criteria**

i) change in the decision variable becomes small

$$\left|x_j(i+1) - x_j(i)\right| \leq \varepsilon \qquad for\ \ j = 1,2,\dots n$$

ii) normalized change in the objective function is small

$$\left|\frac{f(i+1) - f(i)}{f(i)}\right| \leq \varepsilon$$

iii) gradient vector (slope) becomes small

$$\left|\frac{\partial f}{\partial x_j}(i)\right| \leq \varepsilon \qquad for\ \ j = 1,2,\dots n$$

Numerical Example

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

Show one iteration assuming initial guess to be (2 , 2).

$f_1 = 2^2 + 2*2^2 + 2*2 = 16$

$$\nabla f_1 = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 \\ x_1 + 4x_2 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \end{pmatrix}$$

Therefore, $\underline{S}_1 = -\nabla f_1 = \begin{pmatrix} -6 \\ -10 \end{pmatrix}$

$\Rightarrow$ $\underline{x}_2 = \underline{x}_1 + \lambda \underline{S}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \lambda_1 \begin{pmatrix} -6 \\ -10 \end{pmatrix} = \begin{pmatrix} 2 - 6\lambda_1 \\ 2 - 10\lambda_1 \end{pmatrix}$

$f_2 = (2 - 6\lambda_1)^2 + 2*(2 - 10\lambda_1)^2 + (2 - 6\lambda_1)*(2 - 10\lambda_1)$

$df_2/d\lambda_1 = 2*(2 - 6\lambda_1)(-6) + 4*(2 - 10\lambda_1)(-10) + (2 - 6\lambda_1)(-10)$
$+ (-6)(2 - 10\lambda_1) = 0$

$\Rightarrow$ $-12*(2 - 6\lambda_1) - 40*(2 - 10\lambda_1) + (2 - 6\lambda_1)(-10)$
$+(-6)*(2 - 10\lambda_1) = 0$

$\Rightarrow$ $\lambda_1 = 0.23$

$$\underline{x}_2 = \begin{pmatrix} 2 - 6(0.23) \\ 2 - 10(0.23) \end{pmatrix} = \begin{pmatrix} 0.62 \\ -0.30 \end{pmatrix}$$

$$f_2 = (0.62)^2 + 2*(-0.3)^2 + (0.62)*(-0.3) = 0.38$$

An Example:

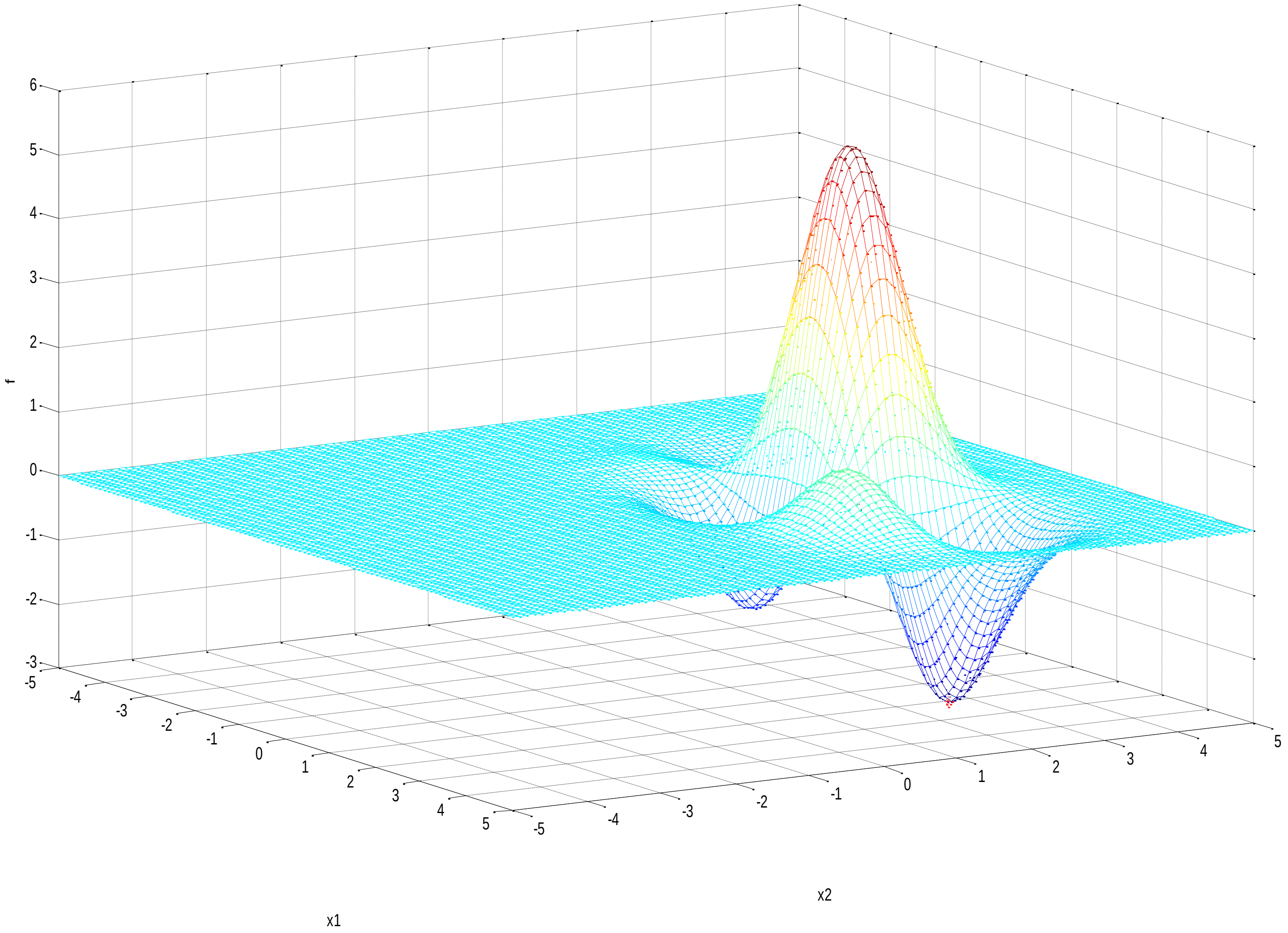
minimize $\quad f = -10\cos x_1 \cos x_2 \; e^{-\left\{\frac{(x_1-1)^2}{4} + (x_2-2)^2\right\}}$

$$-5.0 \le x_1 \le 5.0\,; \quad -5.0 \le x_2 \le 5.0$$

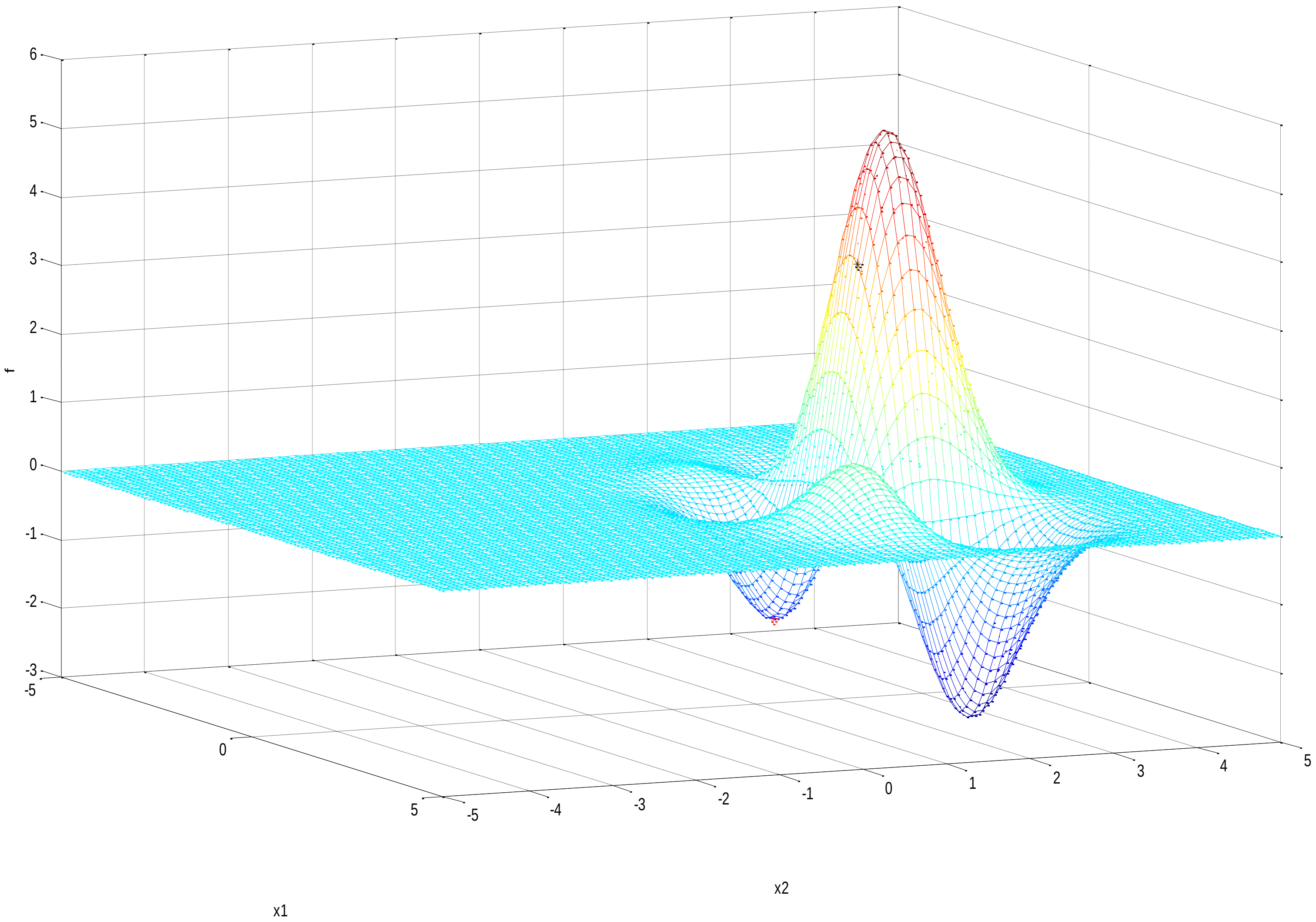
$$\underline{x}(0) = \begin{bmatrix} 2 \\ 1.5 \end{bmatrix}$$

$$\underline{x}^* = \begin{bmatrix} 2.49 \\ 2.43 \end{bmatrix} \qquad f^* = -2.8733$$

$$\lambda = 0.05 \qquad i = 15$$



f
6
5
4
3
2
1
0
-1
-2
-3
x1
-5
-4
-3
-2
-1
0
1
2
3
4
5
x2
-5
-4
-3
-2
-1
0
1
2
3
4
5



$$\underline{x}(0) = \begin{bmatrix} 0.5 \\ 2.0 \end{bmatrix} ; \quad \underline{x}^* = \begin{bmatrix} 0.34 \\ 1.08 \end{bmatrix} ; \quad f^* = -1.7093; \quad i = 19$$

- **Limitations**

  i) cannot be used for discontinuous objective functions

  ii) local minima

  iii) dependent on initial guess

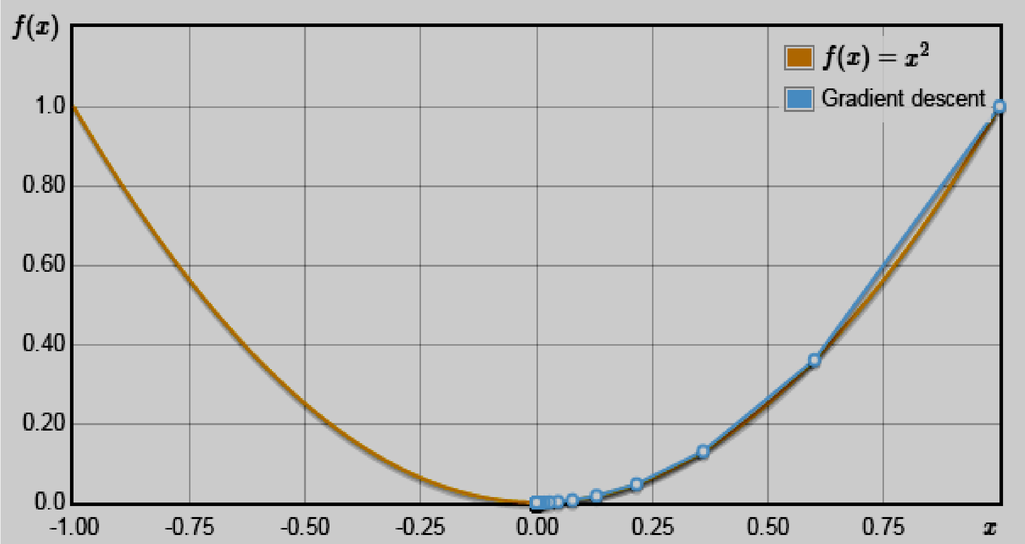
  iv) slow terminal convergence

- **Effect of Step Size**

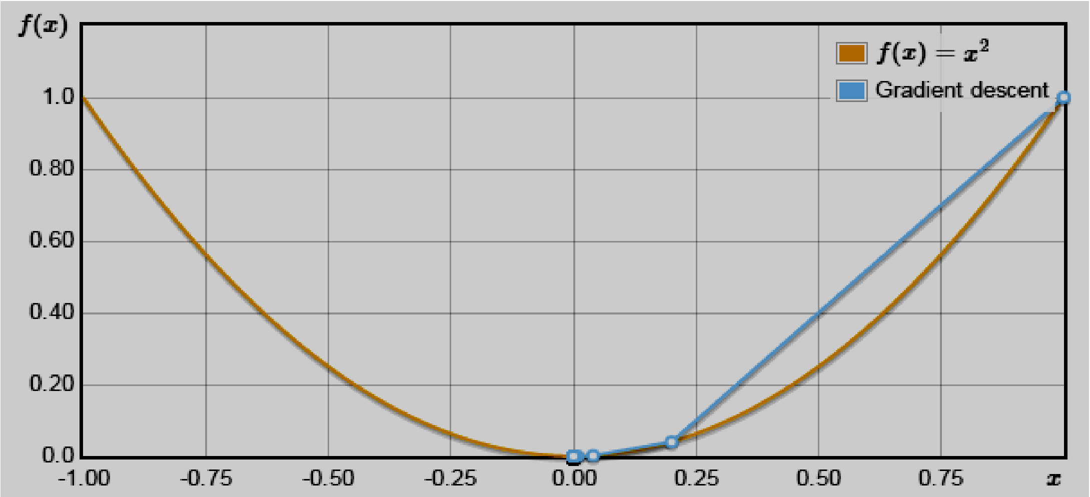
Objective function: $f(x) = x^2$

Initial point: $x = 1$

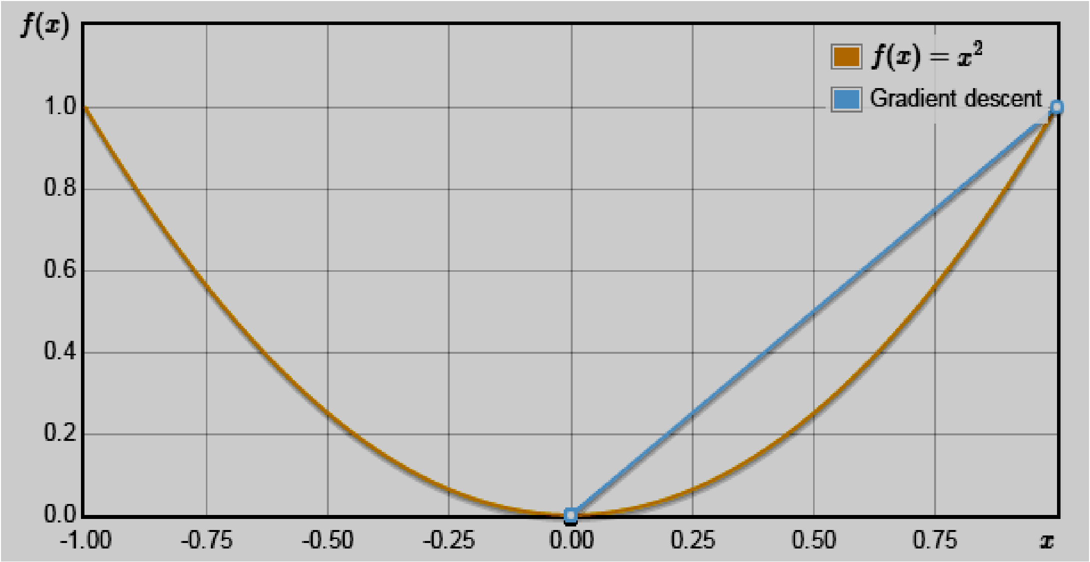
Variable step size ($\lambda = 0.2, 0.4, 0.5, 0.8, 1$)



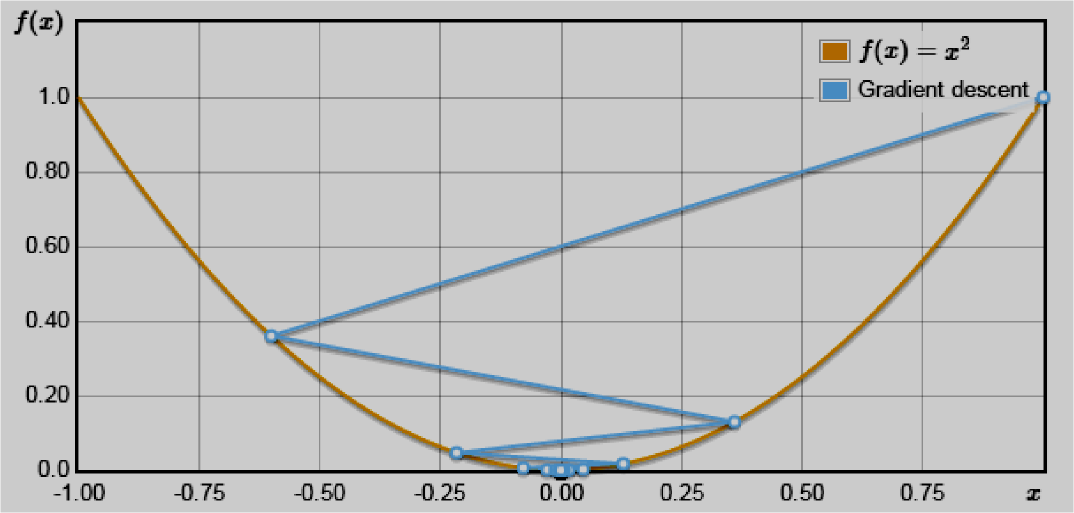
$\lambda = 0.2$



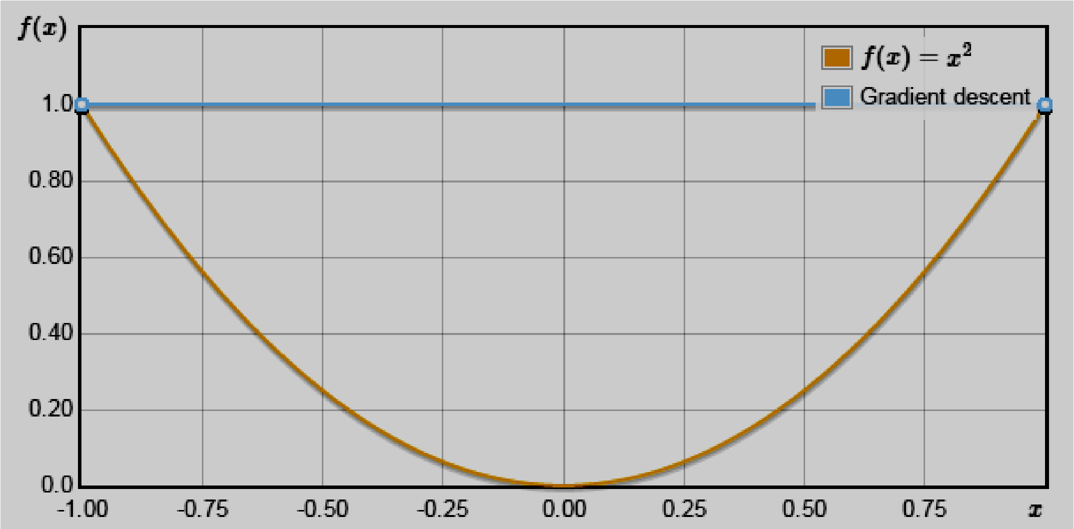
$\lambda = 0.4$



$\lambda = 0.5$
(optimal)



$\lambda = 0.8$



$\lambda = 1.0$

- Adaptive Step Size (gradual reduction)

- **Momentum Method**

  - ✓ $$\Delta\underline{x}(i) = \lambda\,\underline{S}(i) \quad \text{: without momentum}$$

    $$\Delta\underline{x}(i) = \lambda\,\underline{S}(i) + \alpha\,\Delta\underline{x}(i-1)\,; \qquad 0 < \alpha < 1$$

  - ✓ Faster convergence

  - ✓ If gradients have same signs in consecutive iterations then acceleration; else deceleration

Variants of Gradient Descent in the Context of ANN:

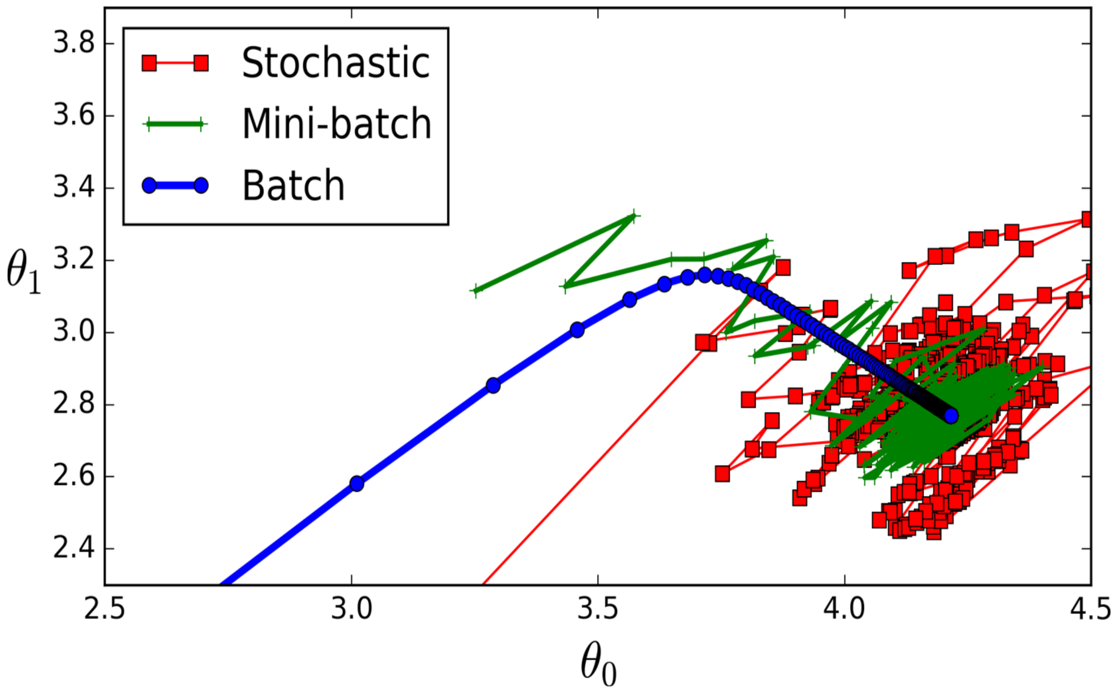
1) Batch GD/SGD/Mini-batch GD

Batch GD: - Weights are updated based on the average gradient of the entire training set
- Computationally expensive
- Smooth convergence to the minimum

SGD: - Weights are updated sequentially on presentation of each sample from the training set
- Less computation burden
- A zigzag path to the optima
- Less likely to get trapped in local minimum

Mini-batch GD: - Weights are updated based on the average gradient of a randomly chosen mini batch from the training set

- A compromise between Batch GD and SGD



2) Vanilla Momentum (VM) / Nesterov Accelerated Gradient (NAG)

VM:

$$g_t = \nabla_\theta J(\theta)$$

$$m_{t+1} = \alpha\, m_t - \eta\, g_t$$

$$\theta_{t+1} = \theta_t + m_{t+1}$$

i.e. $$\theta_{t+1} = \theta_t + \alpha\, m_t - \eta\, g_t$$

NAG:

$$g_t = \nabla_\theta J(\theta + \alpha\, m_t)$$

$$m_{t+1} = \alpha\, m_t - \eta\, g_t$$

$$\theta_{t+1} = \theta_t + m_{t+1}$$

i.e. $$\theta_{t+1} = \theta_t + \alpha\, m_t - \eta\, g_t$$

3) AdaGrad (Adaptive Gradient Descent)

- Running sum of squared gradients along each direction is stored
- Decrease learning rate along steep slope
- Increase learning rate along gentle slope

$$g_{t+1} = g_t + \{\nabla_\theta J(\theta)\}^2$$

$$\theta_{t+1} = \theta_t - \eta \frac{\nabla_\theta J(\theta)}{\sqrt{g_{t+1}} + 10^{-6}}$$

- Running sum of squared gradient may become large if training continues for a long time
- Premature convergence for complex error surfaces

4) <u>RMS Prop (Root Mean Square Propagation)</u>

- An improvement over AdaGrad
- Recent past values of squared gradients are emphasized
- Distant past values are discarded

$$g_{t+1} = \beta\, g_t + (1-\beta)\, \{\nabla_\theta\, J(\theta)\}^2$$

$$\theta_{t+1} = \theta_t - \eta\, \frac{\nabla_\theta\, J(\theta)}{\sqrt{g_{t+1}} + 10^{-6}}$$

- Usually, exponential forgetting factor β is 0.9

5) <u>Adam (Adaptive Moment Estimation)</u>

- Combines RMS Prop with momentum method

$$m_{t+1} = \beta_1 \, m_t \; + (1 - \beta_1) \, \nabla_\theta \, J(\theta)$$

$$g_{t+1} = \beta_2 \, g_t + (1 - \beta_2) \, \{\nabla_\theta \, J(\theta)\}^2$$

$$\widehat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}$$

$$\hat{g}_{t+1} = \frac{g_{t+1}}{1 - \beta_2^{t+2}}$$

$$\theta_{t+1} = \; \theta_t - \eta \; \frac{\widehat{m}_{t+1}}{\sqrt{\hat{g}_{t+1}} \; + 10^{-6}}$$

- Usually β_1=0.9 and β_2=0.999

6) Nadam

- Combines NAG (instead of vanilla momentum) with RMSProp