



# Optimization

---

Hamidreza Baradaran Kashani  
Neural Networks Course (Fall 2022)





# Objectives

---

- Getting familiar with **Loss functions**
- Investigating **optimum points** and **conditions for optimality**
- Understanding the behavior of **quadratic functions** such as **MSE**
- Studying the most important iterative optimization algorithms like **Gradient descent**





# List of Contents



---

1. Performance Learning
2. Taylor Series
3. Gradient and Hessian
4. Directional Derivatives
5. Minimum Points
6. Conditions for Optimality
7. Quadratic Functions
8. Mean Square Error
9. Iterative Optimization
10. Gradient Descent
11. Newton's Method
12. Conjugate Gradient





# Performance Learning

- ❑ **Performance learning** is a neural network **training technique** in which the network parameters (weights and biases) are adjusted to optimize the **performance** of the network.
- ❑ **Loss function** (also called *Performance index*) is a quantitative measure of network performance.
- ❑ Loss Function   $\Rightarrow$  network performance 
- ❑ We want to search in the **performance surface** (parameter space) and reduce the performance index in order to find the **minimum point** (if exists).

# Taylor Series



- Assume we represent the performance index by  $F(x)$  where  $x$  is a scalar parameter we are adjusting and  $F(x)$  is an **analytic** function (all of its derivatives exist)
- **Taylor series expansion** about point  $x^*$  :

$$\begin{aligned} F(x) = & F(x^*) + \frac{d}{dx}F(x)\Big|_{x=x^*}(x-x^*) \\ & + \frac{1}{2}\frac{d^2}{dx^2}F(x)\Big|_{x=x^*}(x-x^*)^2 + \dots \\ & + \frac{1}{n!}\frac{d^n}{dx^n}F(x)\Big|_{x=x^*}(x-x^*)^n + \dots \end{aligned}$$



# Taylor Series



□ We will use the Taylor series expansion to **approximate** the loss function, by limiting the expansion to a finite number of terms.

□ **Example:**  $F(x) = \cos(x)$  Taylor series expansion about point  $x^* = 0$  is:

$$\begin{aligned} F(x) &= \cos(x) \\ &= \cos(0) - \sin(0)(x - 0) - \frac{1}{2}\cos(0)(x - 0)^2 + \frac{1}{6}\sin(0)(x - 0)^3 + \dots \\ &= 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 + \dots \end{aligned}$$

0<sup>th</sup>-order approximation of  $F(x)$  :  $F(x) \approx F_0(x) = 1$

2<sup>nd</sup>-order approximation of  $F(x)$  :  $F(x) \approx F_2(x) = 1 - \frac{1}{2}x^2$

4<sup>th</sup>-order approximation of  $F(x)$  :  $F(x) \approx F_4(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4$

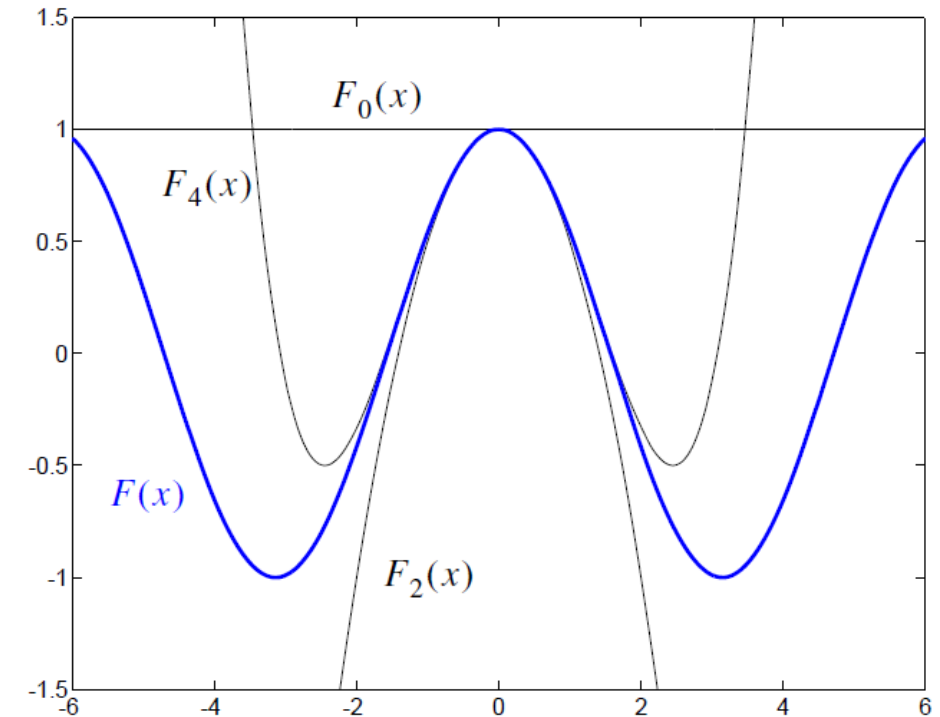


# Taylor Series



## Example (Cont.):

- If  $x$  is very close to  $x^* = 0$ , all three approximations are accurate.
- As  $x$  moves farther away from  $x^*$  only the higher-order approximations are accurate.
- The 2<sup>nd</sup>-order approximation is accurate over a wider range than the 0<sup>th</sup>-order approximation.
- The 4<sup>th</sup>-order approximation is accurate over a wider range than the 2<sup>nd</sup>-order approximation.



Cosine Function and Taylor Series Approximations



# Taylor Series

---



- ❑ Using Taylor series approximations of the loss function for:
  - ✓ Investigate the shape of the loss function in the neighborhood of possible optimum points.





# Taylor Series



□ **Vector Case:**  $F(\mathbf{x}) = F(x_1, x_2, \dots, x_n)$

$$\begin{aligned} F(\mathbf{x}) = & F(\mathbf{x}^*) + \frac{\partial}{\partial x_1} F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} (x_1 - x_1^*) + \frac{\partial}{\partial x_2} F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} (x_2 - x_2^*) \\ & + \dots + \frac{\partial}{\partial x_n} F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} (x_n - x_n^*) + \frac{1}{2} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} (x_1 - x_1^*)^2 \\ & + \frac{1}{2} \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} (x_1 - x_1^*) (x_2 - x_2^*) + \dots \end{aligned}$$

gradient

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \boxed{\nabla F(\mathbf{x})}^T \Big|_{\mathbf{x} = \mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*)$$

Hessian

$$+ \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \boxed{\nabla^2 F(\mathbf{x})} \Big|_{\mathbf{x} = \mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots$$



# Gradient and Hessian

□ The **gradient** of  $F(\mathbf{x})$ :  $\nabla F(\mathbf{x}) = \left[ \frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \frac{\partial}{\partial x_2} F(\mathbf{x}) \quad \dots \quad \frac{\partial}{\partial x_n} F(\mathbf{x}) \right]^T$

□ The **Hessian** of  $F(\mathbf{x})$ :  $\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_2 \partial x_n} F(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix}$





# Directional Derivatives

- The  $j^{\text{th}}$  element of the gradient  $(\frac{\partial F(\mathbf{x})}{\partial x_i})$  is the first derivative of the loss function  $F$  along the  $x_i$  axis.
- The  $j^{\text{th}}$  element of the diagonal of the Hessian matrix  $(\frac{\partial^2 F(\mathbf{x})}{\partial x_i^2})$  is the second derivative of the loss function  $F$  along the  $x_i$  axis.
- The **first** derivative of  $F$  along  $\mathbf{p}$  (an arbitrary vector):
$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$$
- The **second** derivative of  $F$  along  $\mathbf{p}$ :
$$\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x}) \mathbf{p}}{\|\mathbf{p}\|^2}$$





# Directional Derivatives

□ **Example:**  $F(\mathbf{x}) = x_1^2 + 2x_2^2$      $\mathbf{x}^* = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$      $\mathbf{p} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$

- Gradient at  $\mathbf{x}^*$ : 
$$\nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} \Big|_{\mathbf{x} = \mathbf{x}^*} = \begin{bmatrix} 2x_1 \\ 4x_2 \end{bmatrix} \Big|_{\mathbf{x} = \mathbf{x}^*} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

- Directional derivative:  
(along  $\mathbf{p}$ ) 
$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|} = \frac{\begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}}{\left\| \begin{bmatrix} 2 \\ -1 \end{bmatrix} \right\|} = \frac{\begin{bmatrix} 0 \end{bmatrix}}{\sqrt{5}} = 0$$



# Directional Derivatives

## □ Example (Cont.):

$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$$

- The first derivative of  $F$  along  $\mathbf{p}$  is the inner product between  $\mathbf{p}$  and the gradient normalized by the magnitude of  $\mathbf{p}$ .
- Any direction that is orthogonal to the gradient will have **zero slope**.
- The **maximum slope** will occur when the inner product of the direction vector and the gradient is a maximum (the direction vector is the same as the gradient).

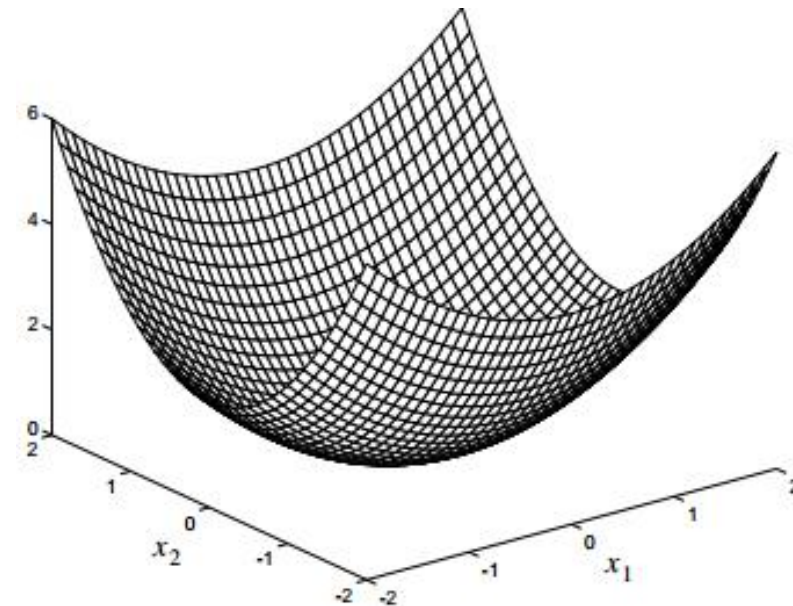
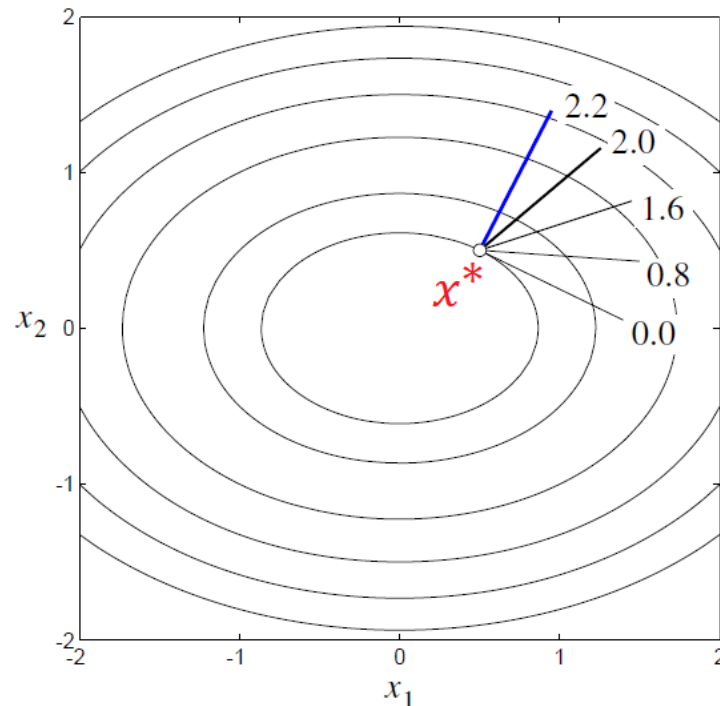




# Directional Derivatives

## □ Example (Cont.):

- The derivative in the direction of the gradient (orthogonal to the contour line) is **maximum**.
- The derivative in the direction orthogonal to the gradient (tangent to the contour line) is **zero**.





# Minimum Points

- ❑ **Strong Minimum:** The point  $\mathbf{x}^*$  is a strong minimum of  $F(\mathbf{x})$  if a scalar  $\delta > 0$  exists, such that  $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$  for all  $\Delta\mathbf{x}$  such that  $\delta > \|\Delta\mathbf{x}\| > 0$ .
- ❑ **Weak Minimum:** The point  $\mathbf{x}^*$  is a weak minimum of  $F(\mathbf{x})$  if it is **not** a strong minimum and a scalar  $\delta > 0$  exists, such that  $F(\mathbf{x}^*) \leq F(\mathbf{x}^* + \Delta\mathbf{x})$  for all  $\Delta\mathbf{x}$  such that  $\delta > \|\Delta\mathbf{x}\| > 0$ .
- ❑ **Local Minimum:** All strong minimum points are local minimum because the function  $F$  may be smaller at some points outside a neighborhood of them.
- ❑ **Global Minimum:** The point  $\mathbf{x}^*$  is a unique global minimum of  $F(\mathbf{x})$  if  $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$  for all  $\Delta\mathbf{x} \neq 0$ .



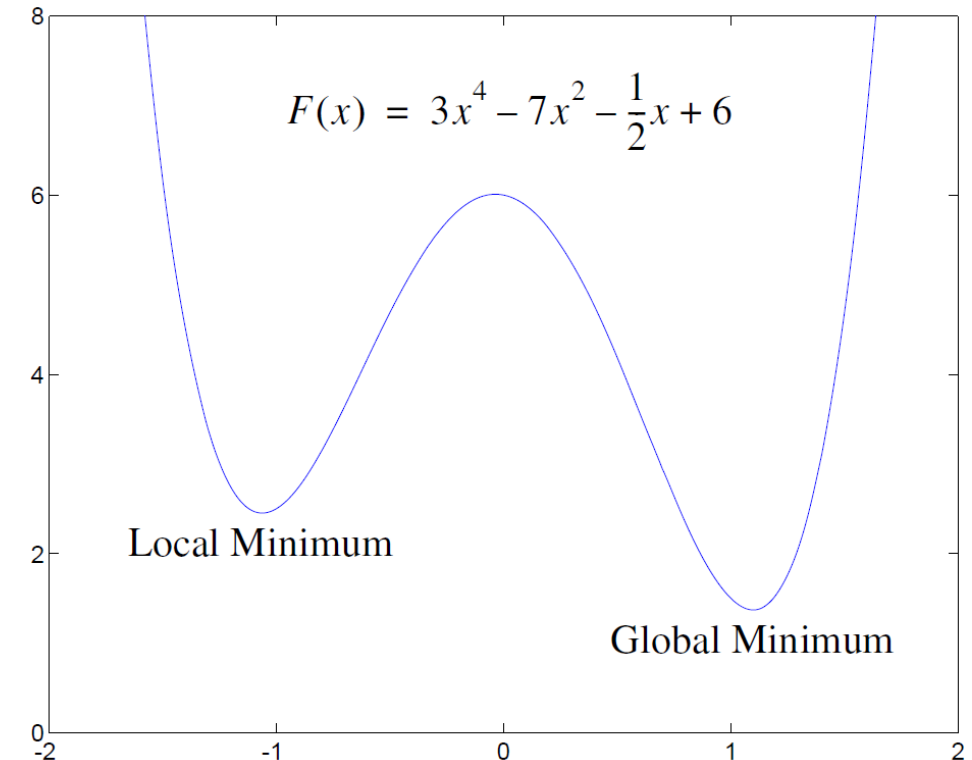


# Minimum Points

➤ **Example 1:** a scalar function

$$F(x) = 3x^4 - 7x^2 - \frac{1}{2}x + 6$$

- This function has two **strong minimum** at approximately **-1.1** and **1.1**
- The minimum at **-1.1** is a **local minimum**
- The minimum at **1.1** is **global minimum**
- This function has no **weak minimum**





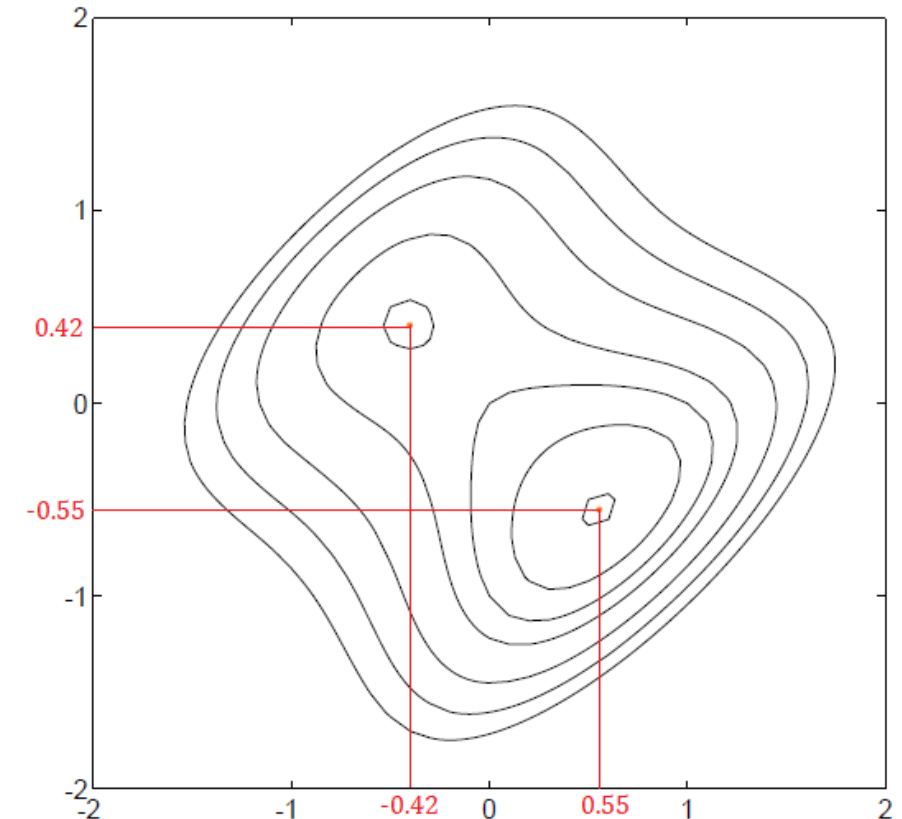
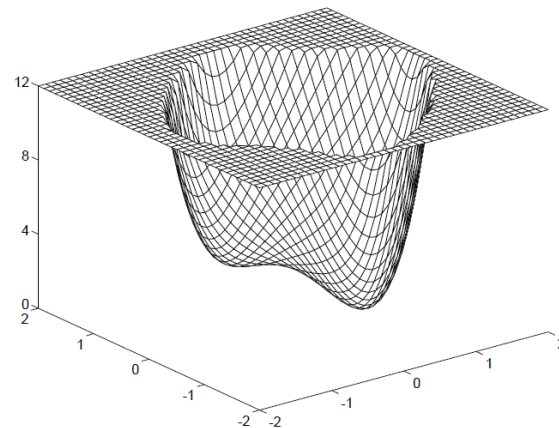


# Minimum Points

## ➤ Example 2:

$$F(x) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$

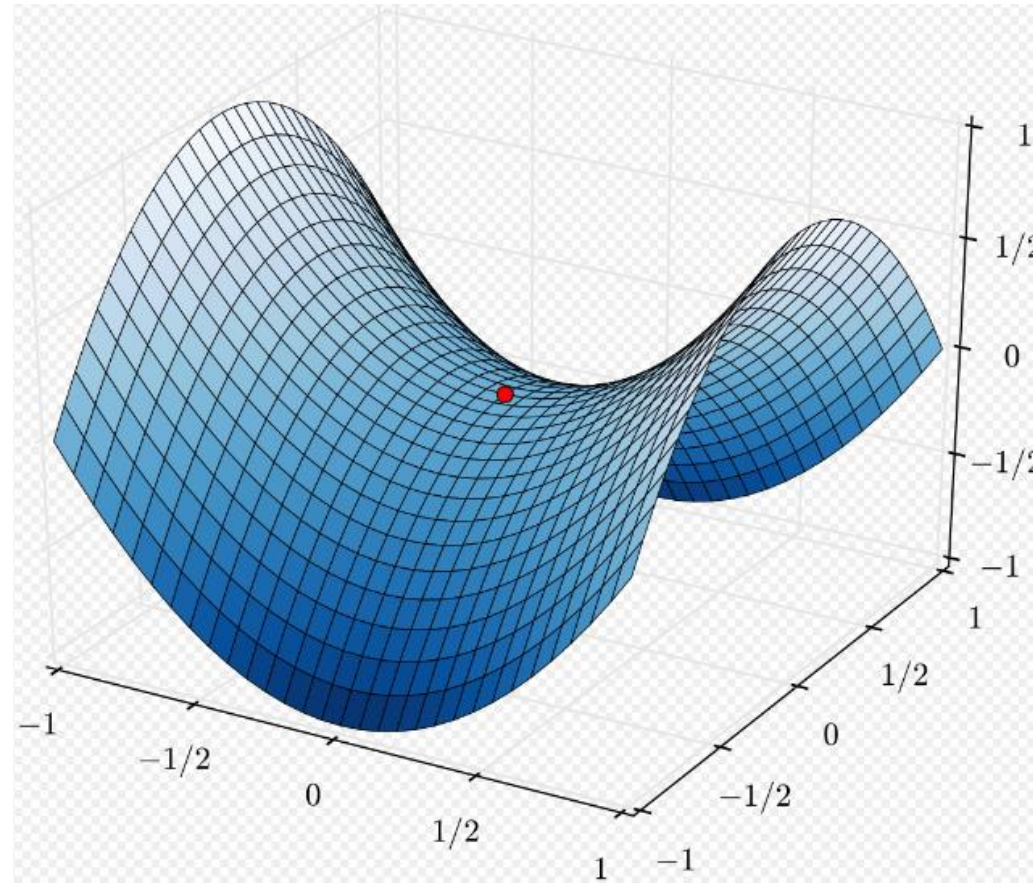
- $(-0.42, 0.42)$  is a strong **local** minimum point.
- $(0.55, -0.55)$  is the strong **global** minimum point.
- There is a **saddle point** at  $(-0.13, 0.13)$ , because it is a local maximum along the line  $x_1 = -x_2$  and a local minimum along the line  $x_1 = x_2$





# Minimum Points

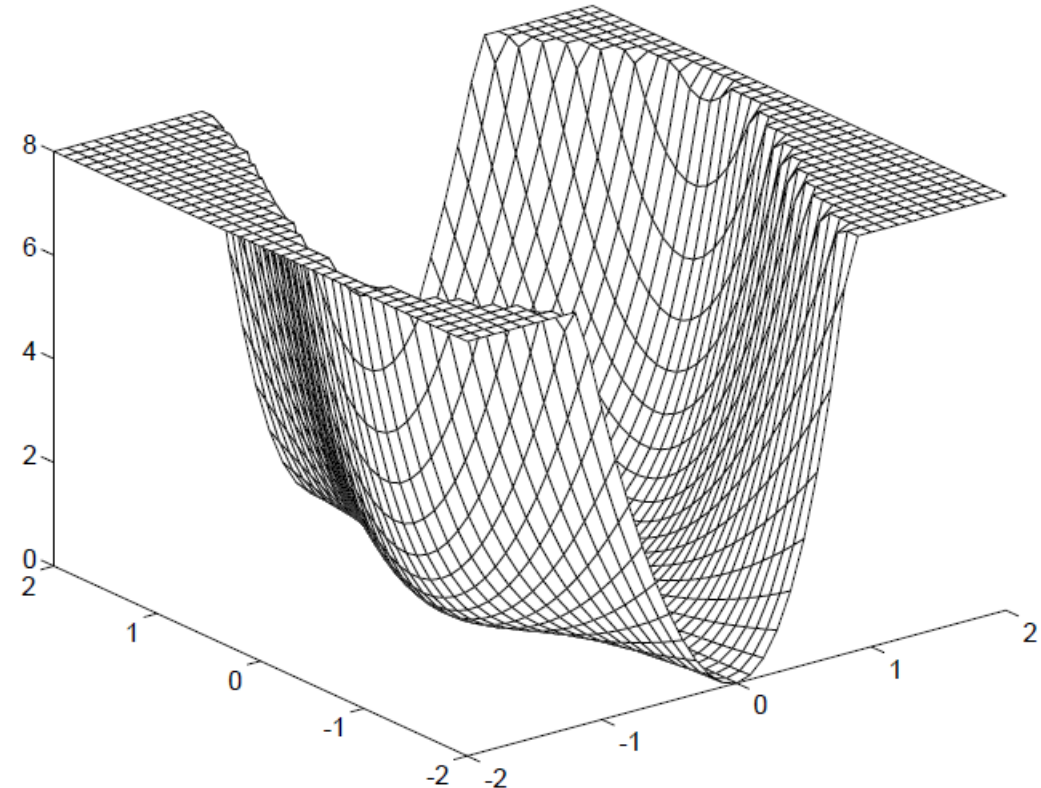
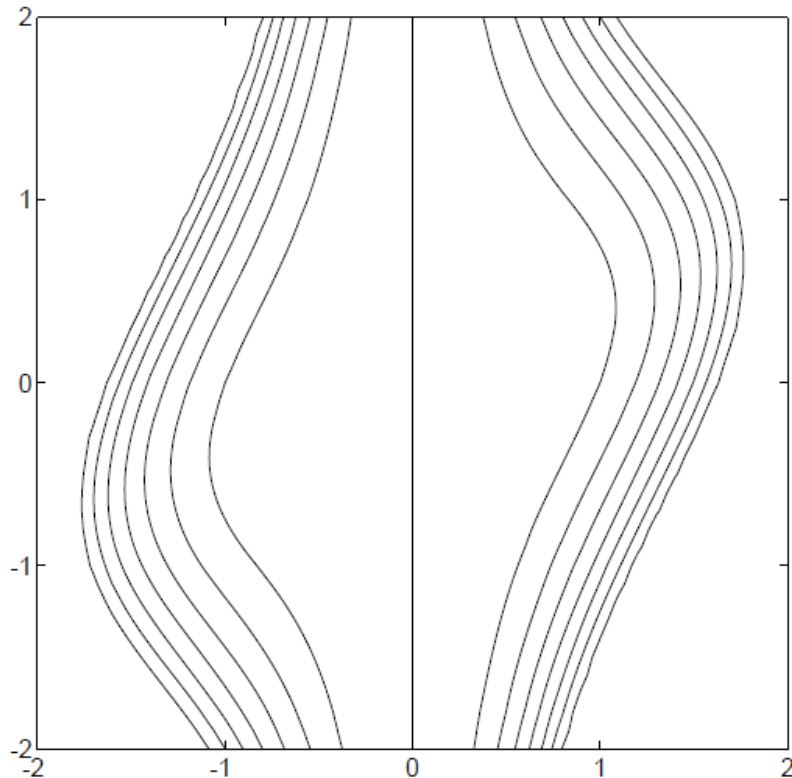
## ➤ Saddle Point:





# Minimum Points

- **Example 3:**  $F(x) = (x_1^2 - 1.5x_1x_2 + 2x_2^2)x_1^2$
- any point along the line  $x_1 = 0$  is a weak minimum





# Conditions for Optimality

❑ Which condition should be satisfied by an optimum (minimum) point?

❑ Taylor series expansion ( $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}^*$ ):

$$F(\mathbf{x}) = F(\mathbf{x}^* + \Delta \mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x} + \frac{1}{2} \Delta \mathbf{x}^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x} + \dots$$

❑ If  $\|\Delta \mathbf{x}\|$  is very small, then the higher order terms in this equation will be negligible. So, we have:

$$F(\mathbf{x}^* + \Delta \mathbf{x}) \cong F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x}$$





# Conditions for Optimality

- In order to have  $\mathbf{x}^*$  as a candidate minimum point, the second term should not be negative:

$$\nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x} \geq 0$$

- However, if this term is positive,

$$\nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x} > 0$$

- then this would imply that

$$F(\mathbf{x}^* - \Delta \mathbf{x}) \cong F(\mathbf{x}^*) - \nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x} < F(\mathbf{x}^*)$$

- But this is a contradiction, since  $\mathbf{x}^*$  should be a minimum point.







# Conditions for Optimality

- So, we should only have:

$$\nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}^*} \Delta \mathbf{x} = 0$$

- Since this must be true for any  $\Delta \mathbf{x}$ , we have

$$\nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}^*} = \mathbf{0}$$

- Therefore the gradient must be zero at a minimum point.
- This is a **first-order, necessary (but not sufficient) condition** for to be a **local minimum** point.
- Any point on which  $F$  has a **zero gradient** is called a **stationary point**.





# Conditions for Optimality

- Assume that we have a stationary point  $\mathbf{x}^*$ .
- Since the gradient of  $F(\mathbf{x})$  is zero at all stationary points, the Taylor series expansion will be

$$F(\mathbf{x}^* + \Delta\mathbf{x}) = F(\mathbf{x}^*) + \frac{1}{2}\Delta\mathbf{x}^T \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}^*} \Delta\mathbf{x} + \dots$$

- $\|\Delta\mathbf{x}\|$  is small and  $F(\mathbf{x})$  can be approximated by the first two terms in this equation. So, a strong minimum will exist at  $\mathbf{x}^*$  if

$$\Delta\mathbf{x}^T \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}^*} \Delta\mathbf{x} > 0$$

- It requires that the **Hessian matrix** be *positive definite*.





# Conditions for Optimality

- ❑ Matrix  $\mathbf{A}$  is **positive definite** if for any vector  $\mathbf{z} \neq 0$  we have:  $\mathbf{z}^T \mathbf{A} \mathbf{z} > 0$
- ❑ Matrix  $\mathbf{A}$  is **positive semidefinite** if for any vector  $\mathbf{z}$  we have:  $\mathbf{z}^T \mathbf{A} \mathbf{z} \geq 0$
- ❑ If all eigenvalues are **positive**, then the matrix is positive definite. If all eigenvalues are **nonnegative**, then the matrix is positive semidefinite.
- ❑ **Second-order necessary (but not sufficient) condition** for  $\mathbf{x}^*$  to be a **strong** minimum point is that the Hessian matrix be **Positive semidefinite**.
- ❑ **Second-order sufficient (but not necessary) condition** for  $\mathbf{x}^*$  to be a **strong** minimum point is that the Hessian matrix be **Positive definite**.







# Conditions for Optimality

➤ **Example:**  $F(\mathbf{x}) = x_1^4 + x_2^2$

- Finding stationary points:  $\nabla F(\mathbf{x}) = \begin{bmatrix} 4x_1^3 \\ 2x_2 \end{bmatrix} = \mathbf{0} \rightarrow$  The only stationary point is  $\mathbf{x}^* = \mathbf{0}$
- Testing the second-order condition:  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{0}} = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix} \bigg|_{\mathbf{x}=\mathbf{0}} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$
- The Hessian matrix is positive semidefinite, which is a necessary (but not sufficient) condition for  $\mathbf{x}^* = \mathbf{0}$  to be a strong minimum point.
- $\mathbf{x}^* = \mathbf{0}$  is a strong minimum point, but we cannot prove it from the conditions we have discussed.





# Conditions for Optimality

## □ Summary of the conditions:

- The **necessary** conditions for  $\mathbf{x}^*$  to be a minimum, strong or weak, of  $F(\mathbf{x})$  are:

$$\nabla F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}^*} = \mathbf{0} \text{ and } \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}^*} \text{ positive semidefinite.}$$

- The **sufficient** conditions for  $\mathbf{x}^*$  to be a strong minimum point of  $F(\mathbf{x})$  are:

$$\nabla F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}^*} = \mathbf{0} \text{ and } \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}^*} \text{ positive definite.}$$





# Quadratic Functions

- **Quadratic Functions** are important because:
  - they appear in many applications
  - **many functions can be approximated by quadratic functions in small neighborhoods, especially near local minimum points.**

- General form: 
$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$$

- The matrix **A** is symmetric. If it's not, it can be replaced by a symmetric matrix that produces the same  $F(\mathbf{x})$ .
- The gradient:  $\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$
- The Hessian:  $\nabla^2 F(\mathbf{x}) = \mathbf{A}$





# Quadratic Functions

---

- All higher derivatives of the quadratic function are **zero**.
- Therefore the first three terms of the Taylor series expansion give an **exact** representation of the function.
- We can say that **all analytic functions behave like quadratics over a small neighborhood** (i.e., when  $\|\Delta \mathbf{x}\|$  is small).





# Quadratic Functions

---

- We now want to investigate the general shape of the quadratic function.
- We can tell a lot about the shape by looking at the *eigenvalues* and *eigenvectors* of the Hessian matrix.
- The shape of this function can be seen more clearly if we perform a change of basis.
- We want to use the eigenvectors of the Hessian matrix  $\mathbf{A}$ , as the new basis vectors.
- Since  $\mathbf{A}$  is symmetric, its eigenvectors will be mutually orthogonal.





# Quadratic Functions

- Consider a quadratic function that has a stationary point at the **origin** ( $\mathbf{d} = \mathbf{0}$ ), and whose value there is **zero** ( $c = 0$ ). some characteristics of it will be:

$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$$

- 1) If the eigenvalues of the Hessian matrix are all **positive**, the function will have a **single strong minimum**.
- 2) If the eigenvalues are all **negative**, the function will have a **single strong maximum**.





# Quadratic Functions

---

- 3) If some eigenvalues are **positive** and other eigenvalues are **negative**, the function will have a **single saddle point**.
  
- 4) If the eigenvalues are all **nonnegative**, but some eigenvalues are **zero**, then the function will either have **a weak minimum** or will have **no stationary point**.
  
- 5) If the eigenvalues are all **nonpositive**, but some eigenvalues are **zero**, then the function will either have **a weak maximum** or will have **no stationary point**.





# Quadratic Functions

- We have assumed, for simplicity, that the stationary point of the quadratic function was at the origin, and that it had a zero value there ( $\mathbf{d} = \mathbf{0}$ ,  $c = 0$ ).
- If  $c$  is nonzero, the function is simply increased in magnitude by  $c$  at every point. The shape of the contours do not change.
- When  $\mathbf{d}$  is nonzero, and  $\mathbf{A}$  is invertible, the shape of the contours are not changed, but the stationary point of the function moves to  $\mathbf{x} = -\mathbf{A}^{-1}\mathbf{d}$ .
- If  $\mathbf{A}$  is not invertible and  $\mathbf{d}$  is nonzero then stationary points may not exist.





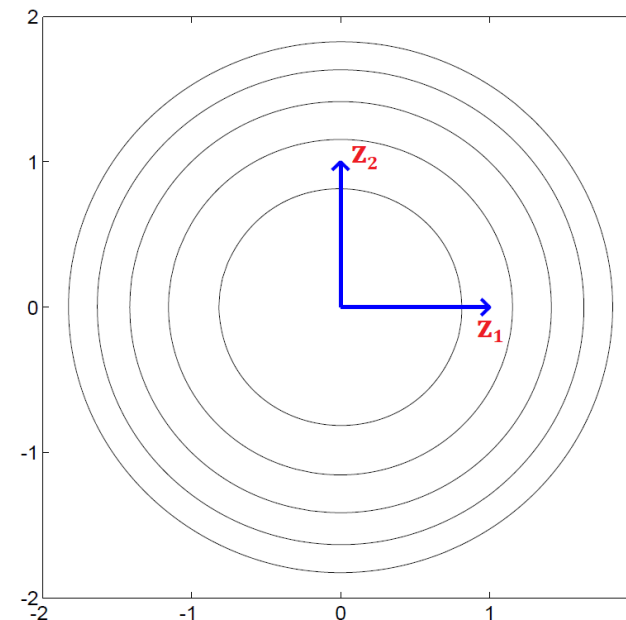
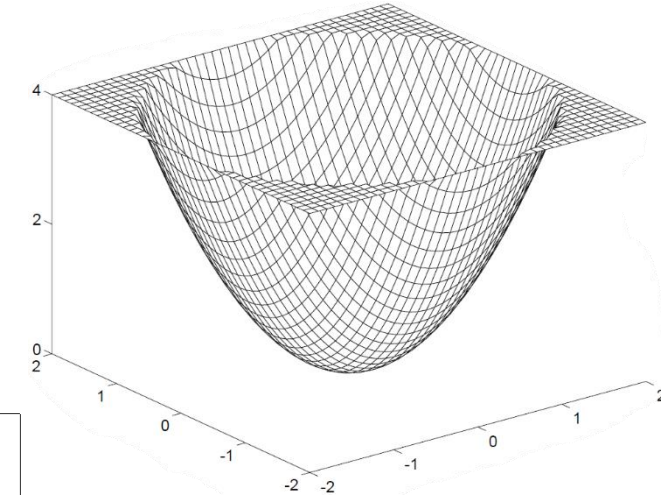


# Quadratic Functions

➤ **Example 1: Circular Hollow**  $F(\mathbf{x}) = x_1^2 + x_2^2 = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{x}$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \lambda_1 = 2, \mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \lambda_2 = 2, \mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Note that any two independent vectors could be the eigenvectors in this case.
- Since all the eigenvalues are equal, the curvature should be the same in all directions, and therefore the function should have circular contours.



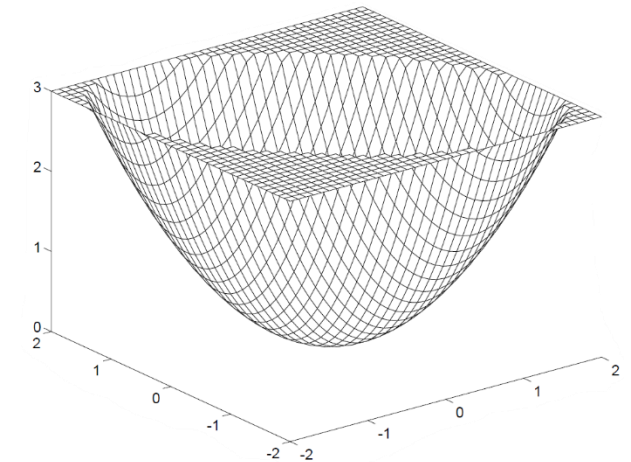
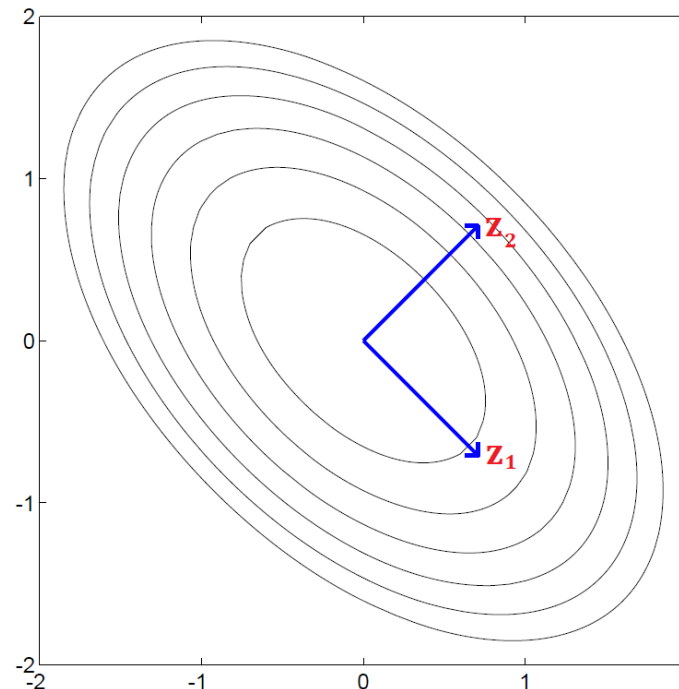


# Quadratic Functions

➤ **Example 2: Elliptical Hollow**  $F(\mathbf{x}) = x_1^2 + x_1x_2 + x_2^2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \lambda_1 = 1, \mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \lambda_2 = 3, \mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- In this case the **maximum curvature is in the direction of  $\mathbf{z}_2$**  so we should cross contour lines more quickly in that direction.



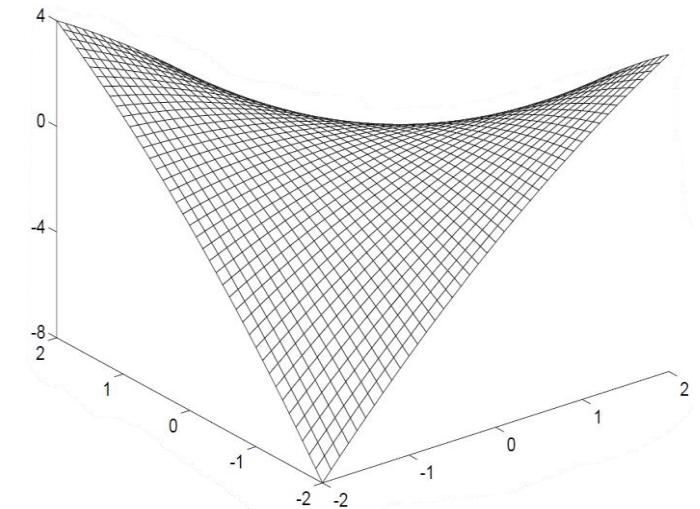
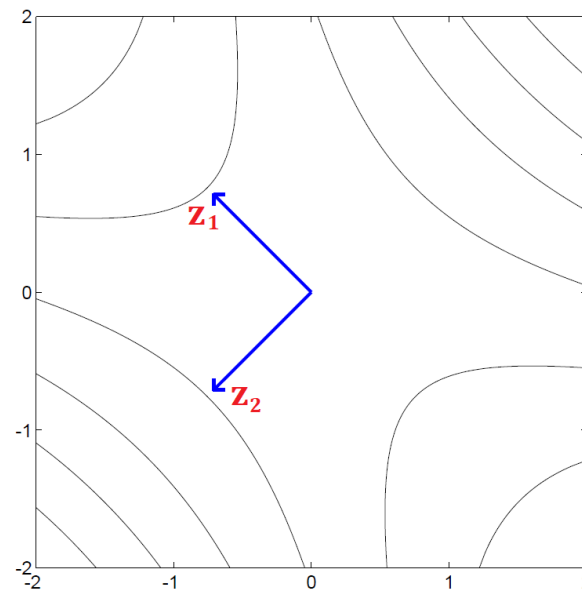


# Quadratic Functions

➤ **Example 3: Elongated Saddle**  $F(\mathbf{x}) = -\frac{1}{4}x_1^2 - \frac{3}{2}x_1x_2 - \frac{1}{4}x_2^2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} -0.5 & -1.5 \\ -1.5 & -0.5 \end{bmatrix} \mathbf{x}$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} -0.5 & -1.5 \\ -1.5 & -0.5 \end{bmatrix}, \lambda_1 = 1, \mathbf{z}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \lambda_2 = -2, \mathbf{z}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

➤ The first eigenvalue is **positive**, so there is **positive** curvature in the direction of  $\mathbf{z}_1$ . The second eigenvalue is **negative**, so there is **negative** curvature in the direction of  $\mathbf{z}_2$ .





# Quadratic Functions

## ➤ Example 3 (Cont.): Elongated Saddle

- Since the magnitude of the second eigenvalue is greater than the magnitude of the first eigenvalue, we will cross contour lines faster in the direction of  $\mathbf{z}_2$ .
- The Hessian matrix is **indefinite**. The stationary point is a **saddle point**. It is a **minimum** of the function along the **first** eigenvector (positive eigenvalue), and a **maximum** of the function along the **second** eigenvector (negative eigenvalue).





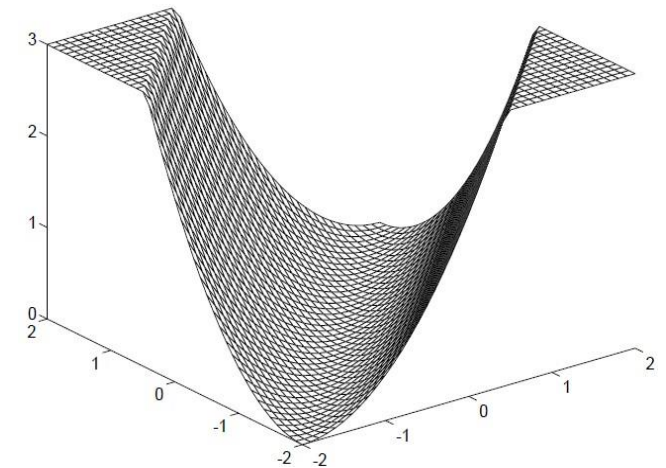
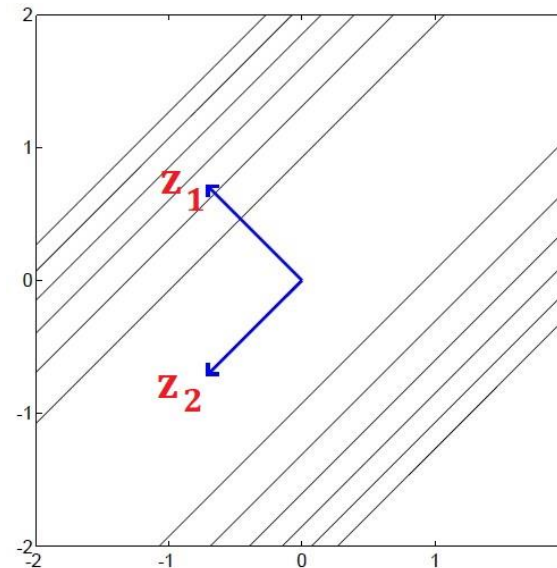
# Quadratic Functions

➤ **Example 4: Stationary Valley**  $F(\mathbf{x}) = \frac{1}{2}x_1^2 - x_1x_2 + \frac{1}{2}x_2^2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \mathbf{x}$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \lambda_1 = 2, \mathbf{z}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \lambda_2 = 0, \mathbf{z}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

- The second eigenvalue is zero, so it has zero curvature along  $\mathbf{z}_2$ .
- The Hessian matrix is positive semidefinite, and we have a weak minimum along the line:

$$x_1 = x_2$$





# Mean Square Error

❖ One of the most important loss functions in deep learning is **Mean Square Error (MSE)** which is usually used in **regression** tasks.

❖ If we put all parameters we are adjusting, including the bias, into one vector **x**, then in a **single neuron** we have:

$$\mathbf{x} = \begin{bmatrix} 1\mathbf{w} \\ b \end{bmatrix}$$

❖ Similarly, we include the bias input “1” as a component of the input vector:

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$







# Mean Square Error

❖ Now the network output can be written as:  $a = \mathbf{x}^T \mathbf{z}$

❖ The Mean Square Error is defined as:

$$F(\mathbf{x}) = E[e^2] = E[(t - a)^2] = E[(t - \mathbf{x}^T \mathbf{z})^2]$$

❖ The expectation is taken over all sets of **input/target** pairs. We can expand this as follows:

$$\begin{aligned} F(\mathbf{x}) &= E[t^2 - 2t\mathbf{x}^T \mathbf{z} + \mathbf{x}^T \mathbf{z} \mathbf{z}^T \mathbf{x}] \\ &= E[t^2] - 2\mathbf{x}^T E[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z} \mathbf{z}^T] \mathbf{x} \end{aligned}$$





# Mean Square Error

❖ This can be written in the following convenient form:

$$F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{R} \mathbf{x}$$

❖ Where:  $c = E[t^2]$ ,  $\mathbf{h} = E[t\mathbf{z}]$  and  $\mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$

- **h** : the cross-correlation between the input vector and its associated target.
- **R** : the input correlation matrix. The diagonal elements of this matrix are equal to the mean square values of the elements of the input vectors.







# Mean Square Error

- We can see the mean square error loss function is a **quadratic function** with  $\mathbf{d} = -2\mathbf{h}$  and  $\mathbf{A} = 2\mathbf{R}$ .

$$F(\mathbf{x}) = c + \mathbf{d}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} \quad F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{R} \mathbf{x}$$

- We previously saw that every property of a quadratic function depend on the hessian matrix. Now we can analyze the properties of MSE loss function based on its correlation matrix  $\mathbf{R}$ :
  - ❖ It can be shown that all correlation matrices are either positive definite or positive semidefinite, which means that they can never have negative eigenvalues.





# Mean Square Error

- ❖ If the correlation matrix has only positive eigenvalues (**positive definite**), the MSE function will have **one unique global strong minimum**.
- ❖ If the correlation matrix has some zero eigenvalues (**positive semi-definite**), the MSE function will either have **a weak minimum** or **no minimum** depending on the vector  **$\mathbf{d} = -2\mathbf{h}$** .

□ The stationary point of the MSE function:

$$\mathbf{x}^* = \mathbf{R}^{-1} \mathbf{h}$$





# Iterative Optimization

- We want to develop algorithms to optimize a loss function  $F(\mathbf{x})$ . The word “optimize” means to find the value of  $\mathbf{x}$  that minimizes  $F(\mathbf{x})$ .
- All of the optimization algorithm we will discuss are **iterative**. We begin from some initial (usually random) guess  $\mathbf{x}_0$  and then update our guess according to an equation of the form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \quad \text{OR} \quad \Delta \mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k \mathbf{p}_k$$

- $\alpha_k$  (learning rate): a positive scalar that determines the length of the steps.
- $\mathbf{p}_k$  (search direction): a vector along which the optimizer moves.





# Iterative Optimization

---

- Three different optimization algorithms will be discussed in this chapter:
  - Gradient descent (steepest descent),
  - Newton's method,
  - Conjugate gradient
  - They are distinguished by the choice of the search direction  $\mathbf{p}_k$ .
  
- Gradient descent and conjugate gradient use only the first derivative, but Newton's method use the first and the second derivatives of the loss function.





# Gradient Descent

□ We would like to **decrease** the function at each iteration:  $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$

□ Let  $\mathbf{g}_k$  be the gradient at the old guess  $\mathbf{x}_k$ :  $\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x} = \mathbf{x}_k}$

□ So the Taylor series expansion of  $F(\mathbf{x})$  about the old guess  $\mathbf{x}_k$  is:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k$$

• For  $F(\mathbf{x}_{k+1})$  to be less than  $F(\mathbf{x}_k)$ :  $\mathbf{g}_k^T \Delta\mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$





# Gradient Descent

- Since  $\alpha_k$  is positive therefore:  $\mathbf{g}_k^T \mathbf{p}_k < 0$ , Any vector  $\mathbf{p}_k$  that satisfies this equation is called a **descent direction**.
- **Question?**
  - What is the direction of steepest descent? (In what direction will the function decrease most rapidly?)
- The term  $\mathbf{g}_k^T \mathbf{p}_k$  is the inner product between  $\mathbf{g}_k$  (the gradient) and  $\mathbf{p}_k$  (the direction vector), so it will **be most negative** when the direction vector is the negative of the gradient:

$$\mathbf{p}_k = -\mathbf{g}_k$$

This  $\mathbf{p}_k$  is called the **steepest descent direction**.





# Gradient Descent

---

- Gradient descent (Steepest descent) algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$$

- This algorithm moves in the parameter space in the direction of steepest descent with step  $\alpha_k$ .





# Gradient Descent

➤ **Example:** (Fixed learning rate)  $F(\mathbf{x}) = x_1^2 + 25x_2^2$

• Initial guess:  $\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$       The gradient:  $\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix}$

• The gradient at  $\mathbf{x}_0$ :  $\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix}$

• We set  $\alpha = 0.01$

1







# Gradient Descent

## ➤ Example (Cont.):

- The **first** iteration:  $\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.01 \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix}$
- The **second** iteration:  $\mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix} - 0.01 \begin{bmatrix} 0.98 \\ 12.5 \end{bmatrix} = \begin{bmatrix} 0.4802 \\ 0.125 \end{bmatrix}$
- The steepest descent trajectory, for small learning rate (like  $\alpha = 0.01$ ), follows a path that is always **orthogonal** to the contour lines. This is because the gradient is orthogonal to the contour lines.
- With  $\alpha = 0.035$  we obtain a trajectory that oscillates but converges.

2

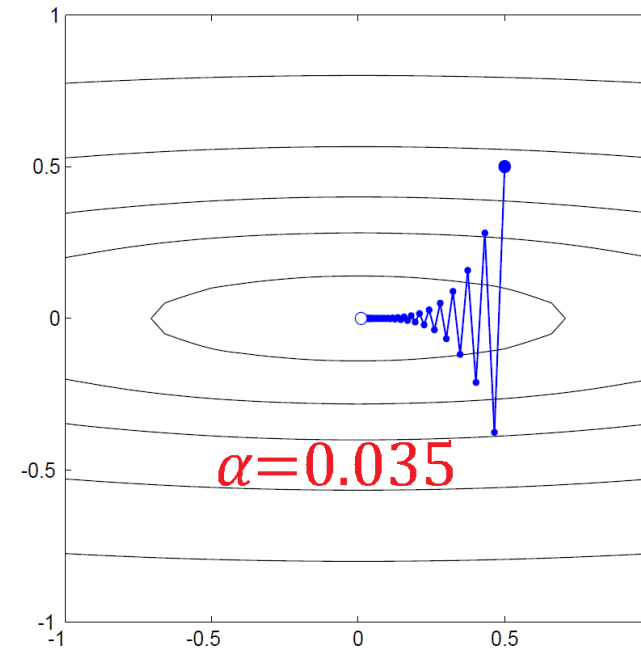
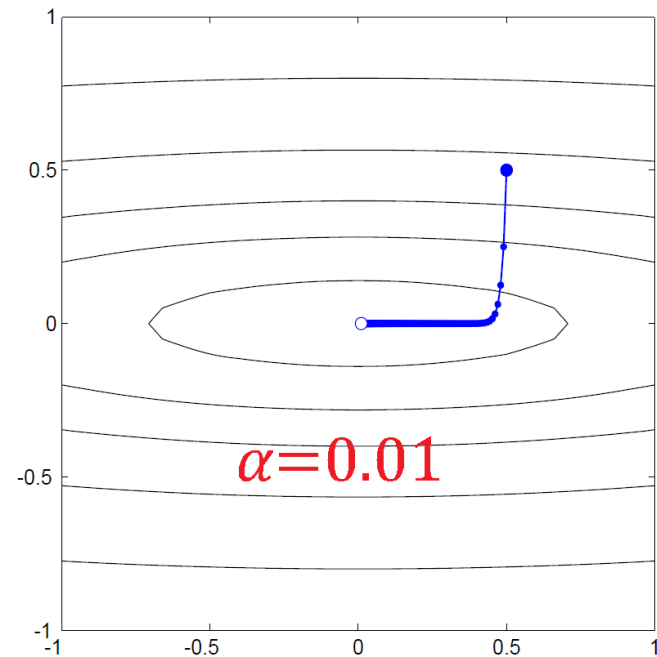




# Gradient Descent

- **Example (Cont.):** If we make the learning rate too large the algorithm will become **unstable** (the oscillations will increase instead of decaying)

3





# Gradient Descent

---

- We would like to **make the learning rate large**, since then we will be taking large steps and would expect to **converge faster**.
- However, if we make the learning rate **too large** the algorithm will become **unstable**.
- **Is there some way to predict the maximum allowable learning rate?**
- This is not possible for arbitrary functions, but for **quadratic functions** we can set an **upper limit**.





# Gradient Descent

## ❑ The choice of learning rate ( $\alpha_k$ )

- ❖ If the loss function is quadratic (for example MSE) then it can be shown this condition must hold:  $\alpha < \frac{2}{\lambda_{max}}$

$\lambda_{max}$ : the maximum eigenvalue of the Hessian matrix

- ❖ The maximum stable learning rate is inversely proportional to the maximum curvature of the quadratic function.
- ❖ If we assume that the quadratic function has a strong minimum point, then its eigenvalues must be positive numbers.



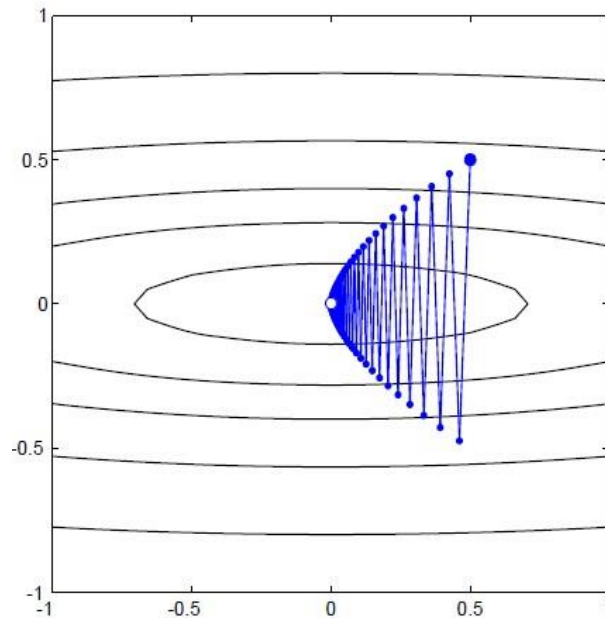


# Gradient Descent

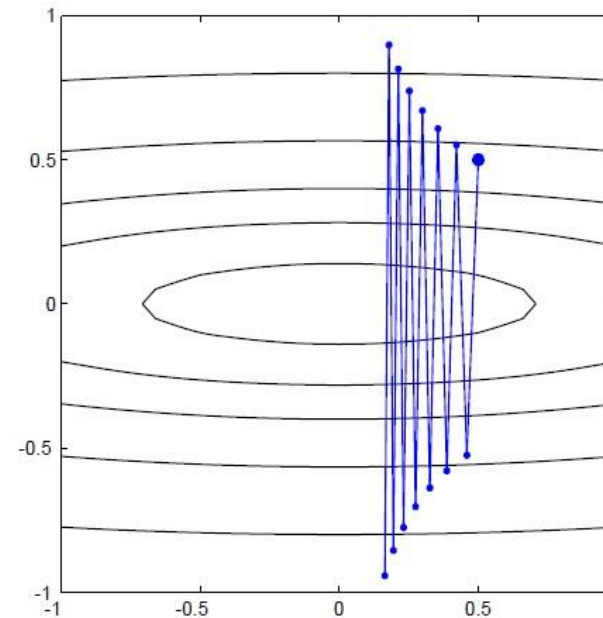
❖ Example:  $\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}$   $\left\{ (\lambda_1 = 2), \left( \mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \right\}, \left\{ (\lambda_2 = 50), \left( \mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \right\}$

- The maximum allowable learning rate:  $\alpha < \frac{2}{\lambda_{max}} = \frac{2}{50} = 0.04$

$\alpha = 0.039$   
stable



$\alpha = 0.041$   
unstable





# Gradient Descent

## □ The choice of learning rate ( $\alpha_k$ ): Minimizing along a line

- ✦ Another approach for selecting the learning rate is to minimize the loss function with respect to  $\alpha_k$  at each iteration. For quadratic functions we have:

$$\frac{d}{d\alpha_k} F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k + \alpha_k \mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k$$

- If we set this derivative equal to zero, then:

$$\alpha_k = - \frac{\nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k} = - \frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$$

- $\mathbf{A}_k$  is the Hessian matrix evaluated at the old guess  $\mathbf{x}_k$ :  $\mathbf{A}_k \equiv \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k}$

See page 9-8





# Newton's Method

□ The derivation of the steepest descent algorithm was based on the **first-order** Taylor series expansion

□ while the **Newton's method** is based on the **second-order** Taylor series:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta \mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta \mathbf{x}_k + \frac{1}{2} \Delta \mathbf{x}_k^T \mathbf{A}_k \Delta \mathbf{x}_k$$

□ Newton's method is trying to approximate the function  $F(\mathbf{x})$  as **quadratic** and then locate the stationary point of this quadratic approximation. Therefore we take the gradient of this quadratic function with respect to  $\Delta \mathbf{x}_k$  and set it equal to zero:

$$\mathbf{g}_k + \mathbf{A}_k \Delta \mathbf{x}_k = \mathbf{0} \quad \Rightarrow \quad \Delta \mathbf{x}_k = -\mathbf{A}_k^{-1} \mathbf{g}_k$$





# Newton's Method

---

□ The Newton's method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k$$

□ If the original function is **quadratic** (with a strong minimum) it will be always minimized in **one step** by using Newton's method.







# Newton's Method

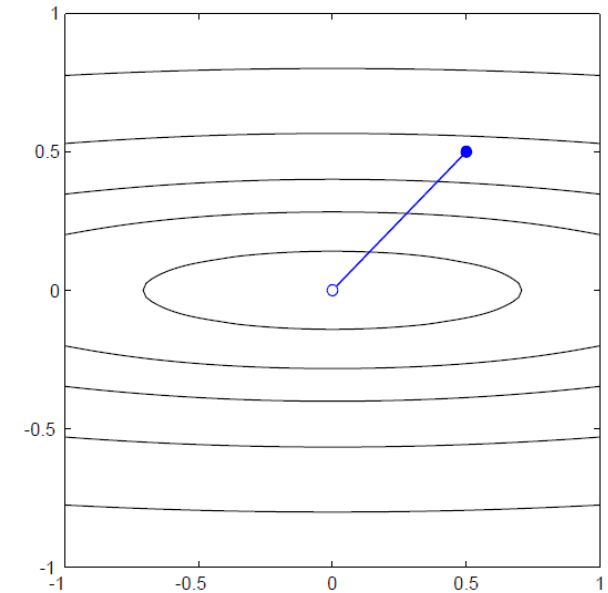
□ For example:  $F(\mathbf{x}) = x_1^2 + 25x_2^2$

- $\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix}, \nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}$

- Start from an **initial** guess:  $\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

- The **first** step of Newton's method:  
so it converges to the global minimum.

$$\mathbf{x}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$





# Newton's Method

- If the original function is **not quadratic**, then we cannot be sure that it will converge at all. This will depend on the **function** and the **initial guess**.

- For example:

$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$

- This function has 3 stationary points:

$$\mathbf{x}^1 = \begin{bmatrix} -0.41878 \\ 0.41878 \end{bmatrix}, \mathbf{x}^2 = \begin{bmatrix} -0.134797 \\ 0.134797 \end{bmatrix}, \mathbf{x}^3 = \begin{bmatrix} 0.55358 \\ -0.55358 \end{bmatrix}$$

- The first point is a **strong local minimum**, the second point is a **saddle point**, and the third point is a **strong global minimum**.
- The convergence of Newton's method for this problem depends on the **initial point**.

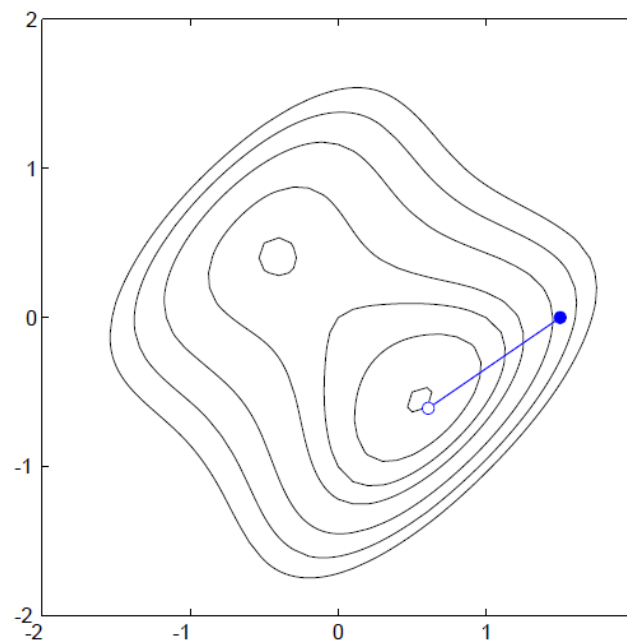




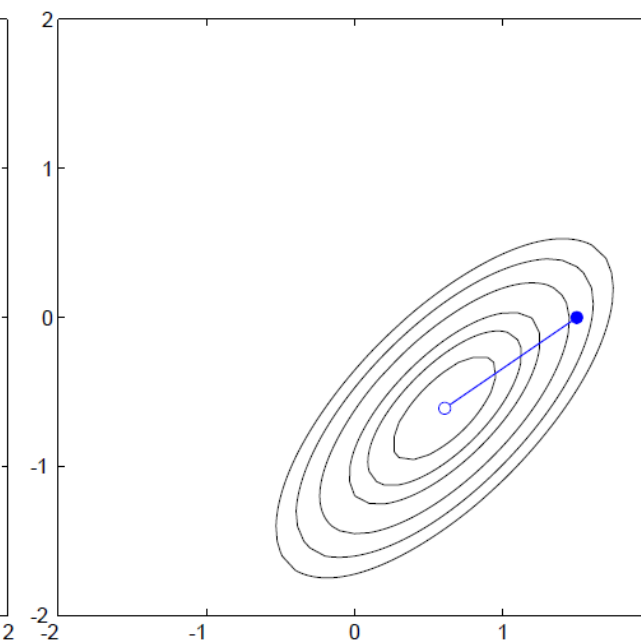
# Newton's Method

- Starting from the initial guess  $\mathbf{x}_0 = [1.5 \ 0]^T$ :

After several steps the algorithm converges to the **global** minimum (not in one step because it's **not** a quadratic function).



The contour plot of the original function



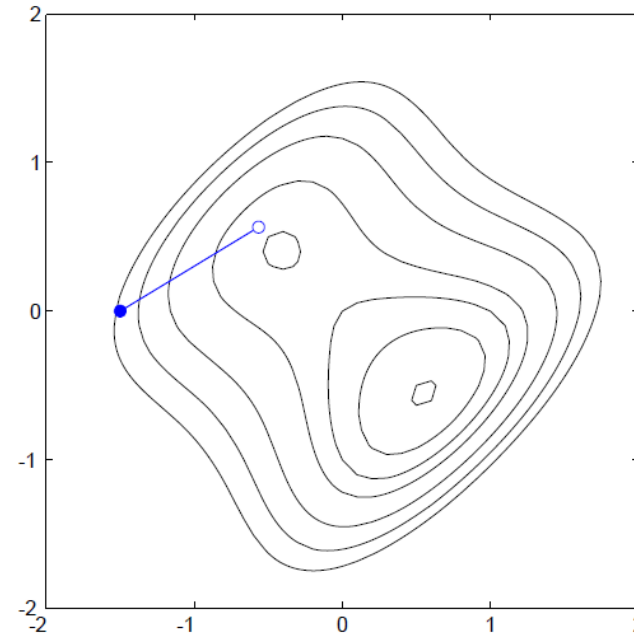
Quadratic approximation near the initial guess



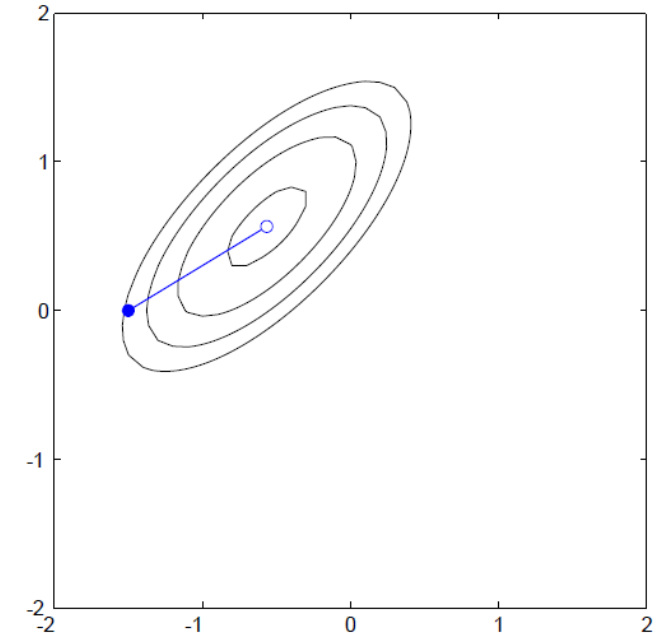


# Newton's Method

- Starting from the initial guess  $\mathbf{x}_0 = [-1.5 \ 0]^T$ :
  - After several steps the algorithm converges to the **local** minimum.
  - Newton's method (like Gradient descent) relies on the local features of the surface.
  - It cannot know the global character of the function and cannot distinguish between a local and a global minimum.



The contour plot of the original function



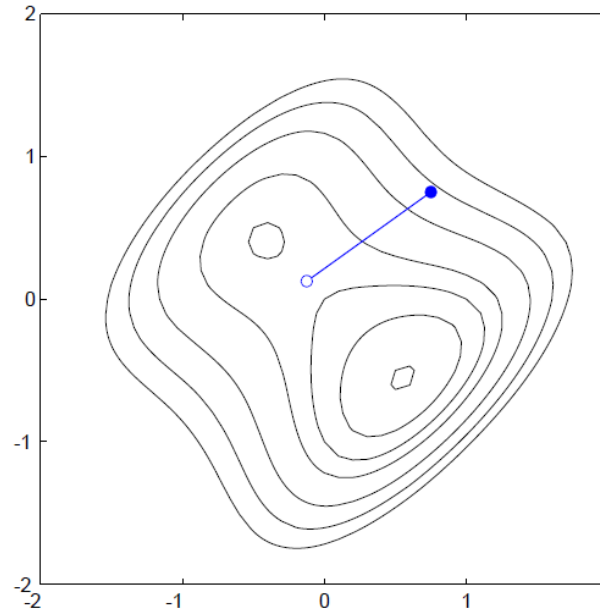
Quadratic approximation near the initial guess



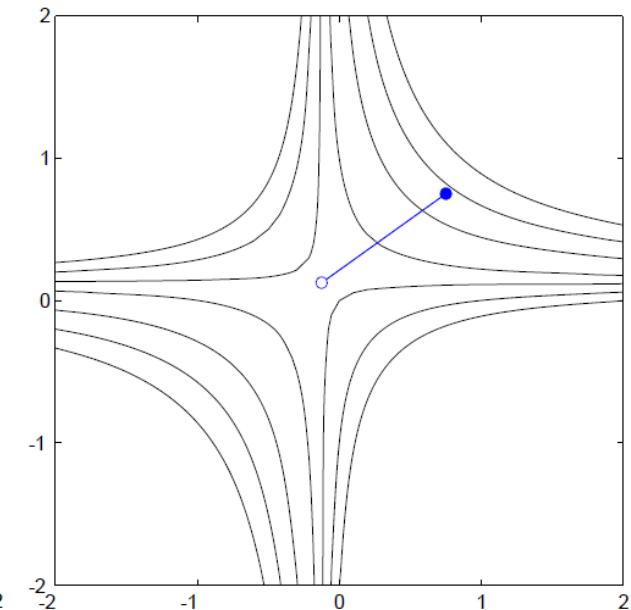


# Newton's Method

- Starting from the initial guess  $\mathbf{x}_0 = [0.75 \ 0.75]^T$ :
  - After several steps the algorithm converges to the **saddle point**.
  - This behavior (converging to a saddle point) is very unlikely with steepest descent.



The contour plot of the original function



Quadratic approximation near the initial guess

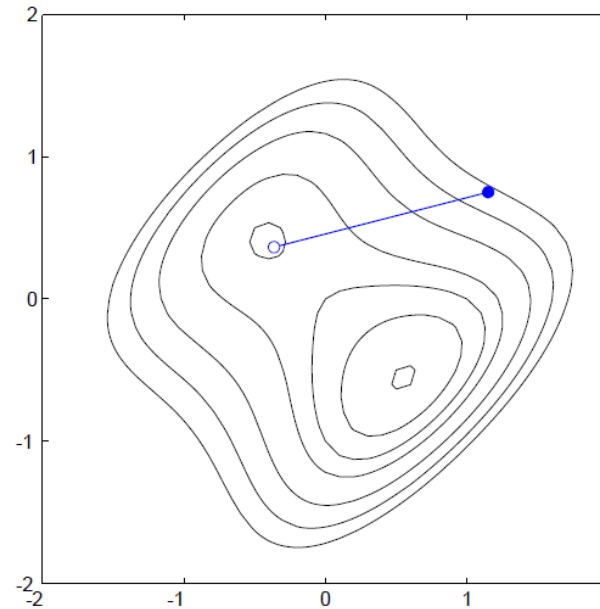




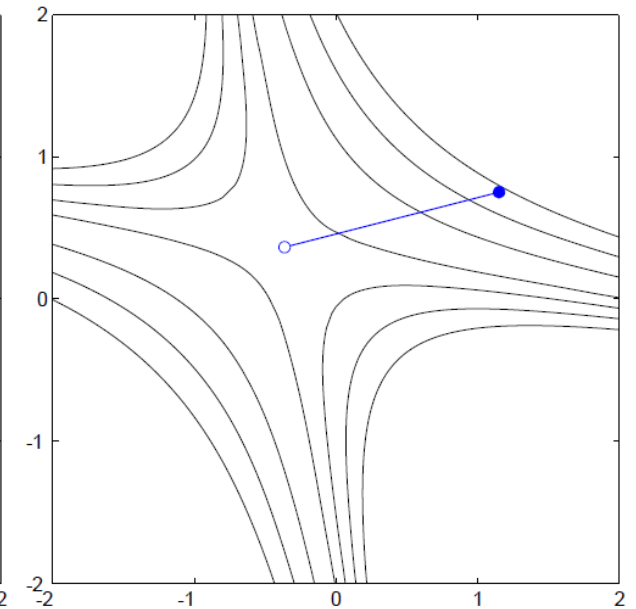
# Newton's Method

- Starting from the initial guess  $\mathbf{x}_0 = [1.15 \ 0.75]^T$ :

- Newton's method can produce *very unpredictable results*.
- For example in this case although the initial point is chosen farther away from the local minimum, after several steps the algorithm converges to the **local minimum** instead of saddle point.



The contour plot of the original function



Quadratic approximation near the initial guess





# Newton's Method

---

- Newton's method **converges quickly** in many because analytic functions can be accurately approximated by quadratic functions in a small neighborhood of a strong minimum.
- While Newton's method usually **produces faster convergence than steepest descent**, the **behavior of Newton's method can be quite complex**.
- In addition to the **problem of convergence to saddle points** (which is very unlikely with steepest descent), it is **possible for the algorithm to oscillate or diverge**.







# Newton's Method

---

- Gradient descent is guaranteed to converge, if the learning rate is not too large or if we perform a linear minimization at each stage.
- Unfortunately, Newton's method requires calculation and storage of the second derivatives, so it is computationally expensive. (Note that the gradient has  $n$  elements, while the Hessian has  $n^2$  elements.)





# Conjugate Gradient

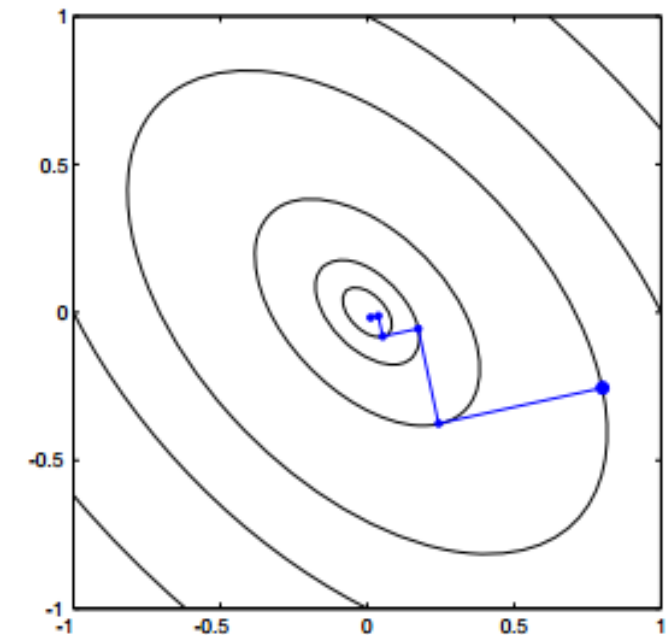
- ❑ **Newton's method** has a good property called **quadratic termination**, which means that it minimizes a quadratic function exactly in a finite number of iterations.
- ❖ Unfortunately, Newton's method requires calculation and storage of the **second derivatives**, so it is computationally expensive. (Note that the gradient has  $n$  elements, while the Hessian has  $n^2$  elements.)
- ❖ For neural networks that have thousands of weights or more, we would like to have methods that require only **first** derivatives (not **second** derivatives) but still have **quadratic termination**.





# Conjugate Gradient

- Recall the performance of the steepest descent algorithm, with linear searches at each iteration.
- The search directions at consecutive iterations were orthogonal.
- For quadratic functions with elliptical contours this produces a **zig-zag trajectory** of short steps. Quadratic search directions are not the best choice.
- Is there a set of search directions that will guarantee **quadratic termination**? One possibility is **conjugate directions**.



Steepest Descent with  
Minimization Along a Line





# Conjugate Gradient

- **Conjugate Vectors (Conjugacy Condition):** A set of vectors  $\{\mathbf{p}_k\}$  is mutually conjugate with respect to a positive definite Hessian matrix  $\mathbf{A}$  if and only if:

$$\mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = 0 \quad \text{for } k \neq j$$

- ❖ Conjugate directions will guarantee quadratic termination.
- ❖ There are an infinite number of mutually conjugate sets of vectors that span a given n-dimensional space.





# Conjugate Gradient

❖ One possible choice is the **eigenvectors of  $\mathbf{A}$** .

❖ To see that the eigenvectors are conjugate, replace  $\mathbf{p}_k$  with  $\mathbf{z}_k$

$$\mathbf{z}_k^T \mathbf{A} \mathbf{z}_j = \lambda_j \mathbf{z}_k^T \mathbf{z}_j = 0 \quad k \neq j$$

❖ where the last equality holds because the eigenvectors of a symmetric matrix are mutually orthogonal. Therefore the eigenvectors are both conjugate and orthogonal.

❖ We can minimize a quadratic function exactly **by searching along the eigenvectors of the Hessian matrix**, since they form the principal axes of the function contours.





# Conjugate Gradient

- ❖ But it's not practical because to find the eigenvectors we must first find the Hessian matrix.
- ❖ We want to find an algorithm that does **not** require the computation of **second derivatives**.
- ❖ It can be shown that:
  - if we make a sequence of exact linear searches along any set of conjugate directions  $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$  then the exact minimum of any quadratic function, with  $n$  parameters, will be reached in at most  $n$  searches.
- ❖ The question is: **“How can we construct these conjugate search directions without Hessian matrix?”**





# Conjugate Gradient

❖ For a quadratic function:  $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$

$$\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d},$$

$$\nabla^2 F(\mathbf{x}) = \mathbf{A}.$$

❖ We know

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}_k}$$

❖ the change in the gradient at iteration k+1 will be:

$$\Delta \mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k = (\mathbf{A} \mathbf{x}_{k+1} + \mathbf{d}) - (\mathbf{A} \mathbf{x}_k + \mathbf{d}) = \mathbf{A} \Delta \mathbf{x}_k$$







# Conjugate Gradient

❖ And previously we had:  $\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$

❖ We can now restate the **conjugacy conditions**:

$$\mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = 0 \quad k \neq j$$

$$\alpha_k \mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = \Delta \mathbf{x}_k^T \mathbf{A} \mathbf{p}_j = \Delta \mathbf{g}_k^T \mathbf{p}_j = 0 \quad k \neq j$$

❖ Note that we do **not** need to know the **Hessian matrix**.

❖ We have restated the **conjugacy conditions in terms of the changes in the gradient** at successive iterations of the algorithm.





# Conjugate Gradient

- The search directions ( $\mathbf{p}_j$ ) will be **conjugate** if they are **orthogonal** to the changes in the gradient ( $\Delta \mathbf{g}_k$ ).
- The first search direction,  $\mathbf{p}_0$ , is arbitrary. It is common to begin the search in the steepest descent direction:

$$\mathbf{p}_0 = -\mathbf{g}_0$$

- $\mathbf{p}_1$  can be any vector that is orthogonal to  $\Delta \mathbf{g}_0$





# Conjugate Gradient

- At each iteration we need to construct a vector  $\mathbf{p}_k$  that is orthogonal to  $\{\Delta\mathbf{g}_0, \Delta\mathbf{g}_1, \dots, \Delta\mathbf{g}_{k-1}\}$ . It is a procedure similar to **Gram-Schmidt orthogonalization** which we discussed in chapter Linear Algebra.
- It can be simplified to iterations of the form:  $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$
- The scalars  $\beta_k$  can be chosen by several different methods, which produce equivalent results for quadratic functions. The most common choices are:

$$\beta_k = \frac{\Delta\mathbf{g}_{k-1}^T \mathbf{g}_k}{\Delta\mathbf{g}_{k-1}^T \mathbf{p}_{k-1}} \text{ or } \beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \text{ or } \beta_k = \frac{\Delta\mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$





# Conjugate Gradient

## The Conjugate Gradient Algorithm:

- 1) Select the first search direction to be the negative of the gradient:  
 $\mathbf{p}_0 = -\mathbf{g}_0$
- 2) Take a step according to  $\Delta \mathbf{x}_k = \alpha_k \mathbf{p}_k$ , selecting the learning rate  $\alpha_k$  to minimize the function along the search direction.
- 3) Select the next search direction according to  $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$ , using one of  $\beta_k$  formulas.
- 4) If the algorithm has not converged, return to step 2.





# Conjugate Gradient

□ **Example:**  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$

- The Gradient:  $\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix}$

- Initial guess:  $\mathbf{x}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix}$

- The **first** search direction:  $\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})^T \big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}$

1





# Conjugate Gradient

## □ Example (Cont.):

- The learning rate: 
$$\alpha_0 = -\frac{\begin{bmatrix} 1.35 & 0.3 \end{bmatrix} \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}}{\begin{bmatrix} -1.35 & -0.3 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}} = 0.413$$

- The **first** step of conjugate gradient:

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + 0.413 \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix} = \begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix}$$

- The Gradient at  $\mathbf{x}_1$ : 
$$\mathbf{g}_1 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix} = \begin{bmatrix} 0.11 \\ -0.5 \end{bmatrix}$$





# Conjugate Gradient

3

## □ Example (Cont.):

- $\beta_1$  using second formula: 
$$\beta_1 = \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0} = \frac{\begin{bmatrix} 0.11 & -0.5 \end{bmatrix} \begin{bmatrix} 0.11 \\ -0.5 \end{bmatrix}}{\begin{bmatrix} 1.35 & 0.3 \end{bmatrix} \begin{bmatrix} 1.35 \\ 0.3 \end{bmatrix}} = \frac{0.2621}{1.9125} = 0.137$$

- The **second** search direction: 
$$\mathbf{p}_1 = -\mathbf{g}_1 + \beta_1 \mathbf{p}_0 = \begin{bmatrix} -0.11 \\ 0.5 \end{bmatrix} + 0.137 \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix} = \begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix}$$

- The learning rate: 
$$\alpha_1 = -\frac{\begin{bmatrix} 0.11 & -0.5 \end{bmatrix} \begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix}}{\begin{bmatrix} -0.295 & 0.459 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix}} = \frac{0.262}{0.325} = 0.807$$





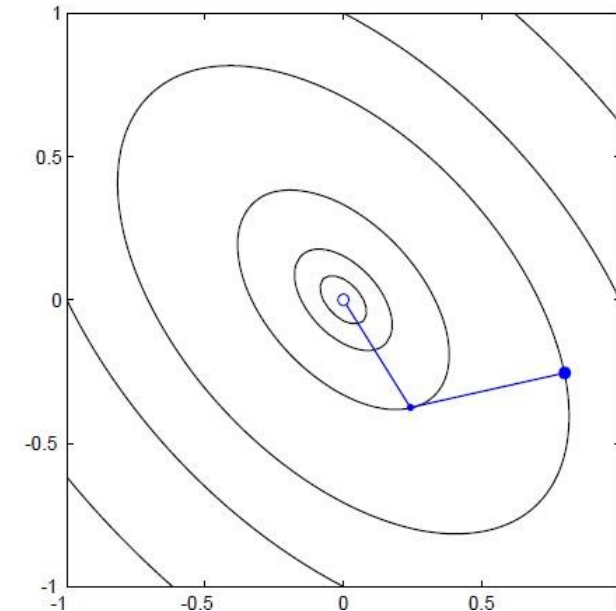


# Conjugate Gradient

□ **Example (Cont.):** The **second** step of conjugate gradient:

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 = \begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix} + 0.807 \begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- The algorithm converges exactly to the minimum in two iterations.
- Since this is a two-dimensional quadratic function (Point 3 Page 69)

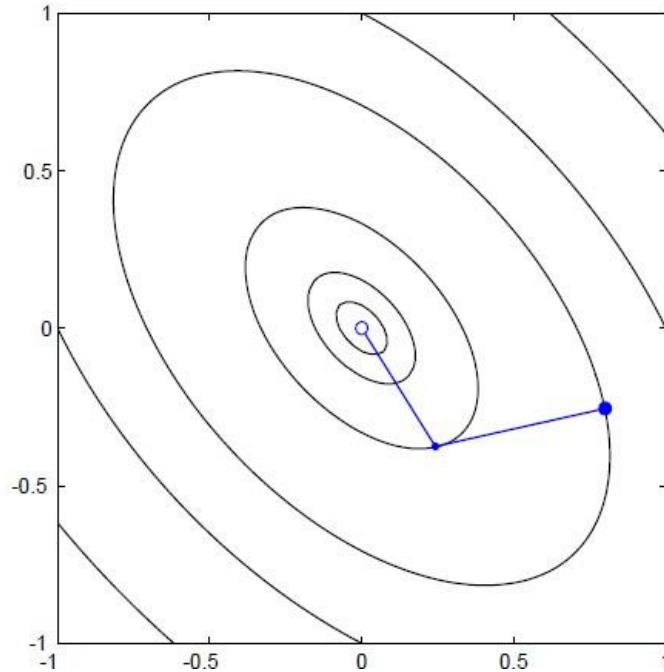


4

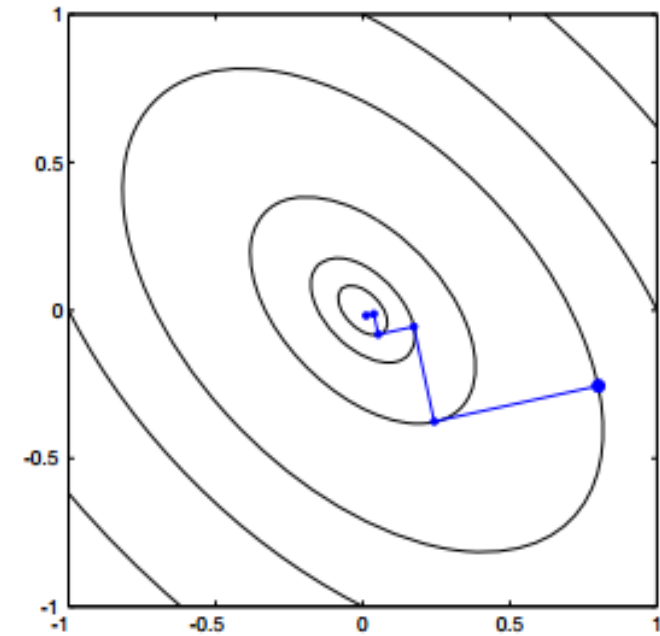




# Conjugate Gradient

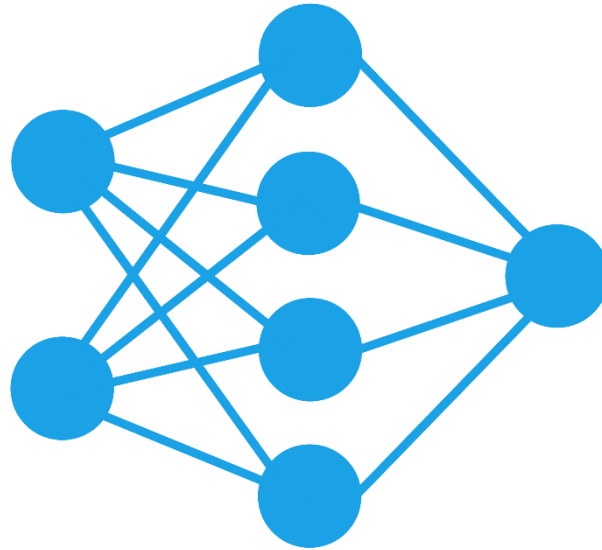


Conjugate Gradient  
Algorithm



Steepest Descent





# Thanks for your attention

---

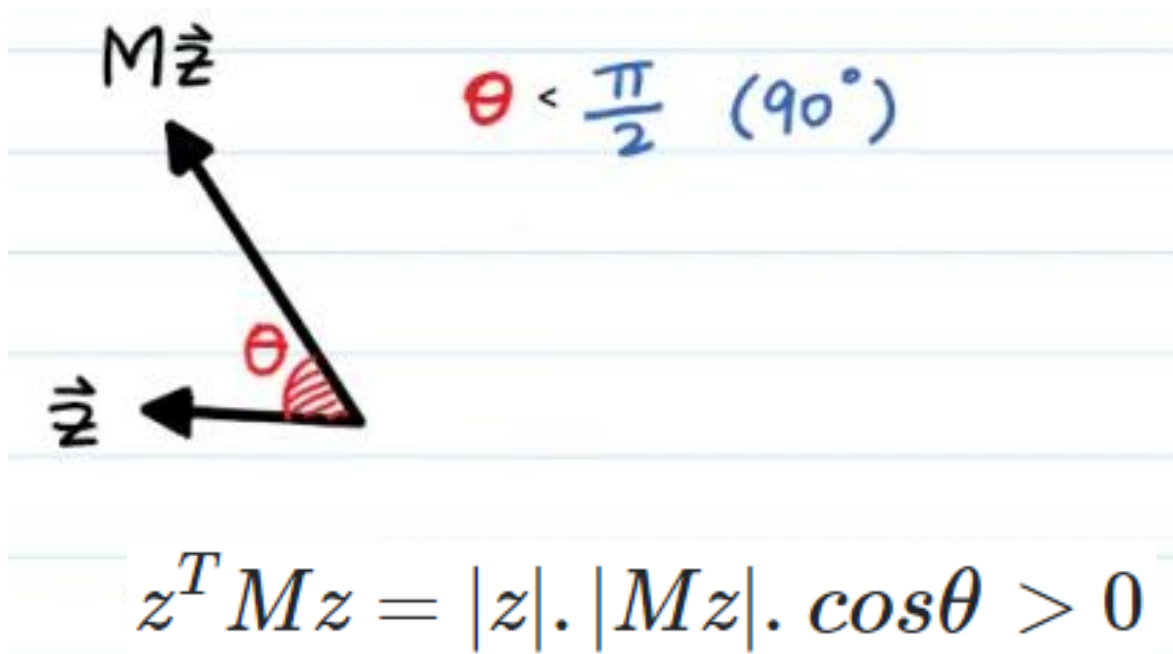
End of chapter 5

Hamidreza Baradaran Kashani



# Example: Positive definite

□ Matrix  $M$  is **positive definite** if for any vector  $z \neq 0$  we have:  $z^T M z > 0$





# Example: Positive definite

- Prove that the **identity matrix** is a positive definite matrix

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$z^T I z = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a^2 + b^2$$

- The right side of the above equation is always positive, so the identity matrix is positive definite.





# Exercise: Positive definite

□ Prove that  $M$  is a positive definite matrix.

$$M = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

