Linear Algebra

Shaurya Prakash
II Semester , Ph.D
Instructor: Dr. S.Karthikeyan

Department of Computer Science,
BHU , Varanasi

Outline

- Transpose
- Inverse
- Vectorization
- Eigen Values and Vectors
- Applications of Eigen Vectors

Transpose

Interchange row and columns.

Uses : To make matrix operation compatible.

$$\begin{pmatrix} 3 & 2 & 1 \\ 4 & 7 & 5 \\ 5 & 8 & 9 \end{pmatrix}^T = \begin{pmatrix} 3 & 4 & 5 \\ 2 & 7 & 8 \\ 1 & 5 & 9 \end{pmatrix}$$

Why do we go for Transposition?

$$\begin{pmatrix} 2 & 3 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \text{ X } \begin{pmatrix} 3 & 4 \\ 5 & 6 \\ 4 & 8 \end{pmatrix} = \text{Not Compatible..}$$

$$\begin{pmatrix} 2 & 3 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \text{ X } \begin{bmatrix} 3 & 4 & 5 \\ 4 & 6 & 8 \end{bmatrix} = \begin{pmatrix} 18 & 26 & 34 \\ 25 & 36 & 47 \\ 39 & 56 & 73 \end{pmatrix}$$

Transpose

$$\begin{pmatrix} 2 & 3 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \text{X} \begin{pmatrix} 3 & 4 \\ 5 & 6 \\ 4 & 8 \end{pmatrix}$$ = Not Compatible..

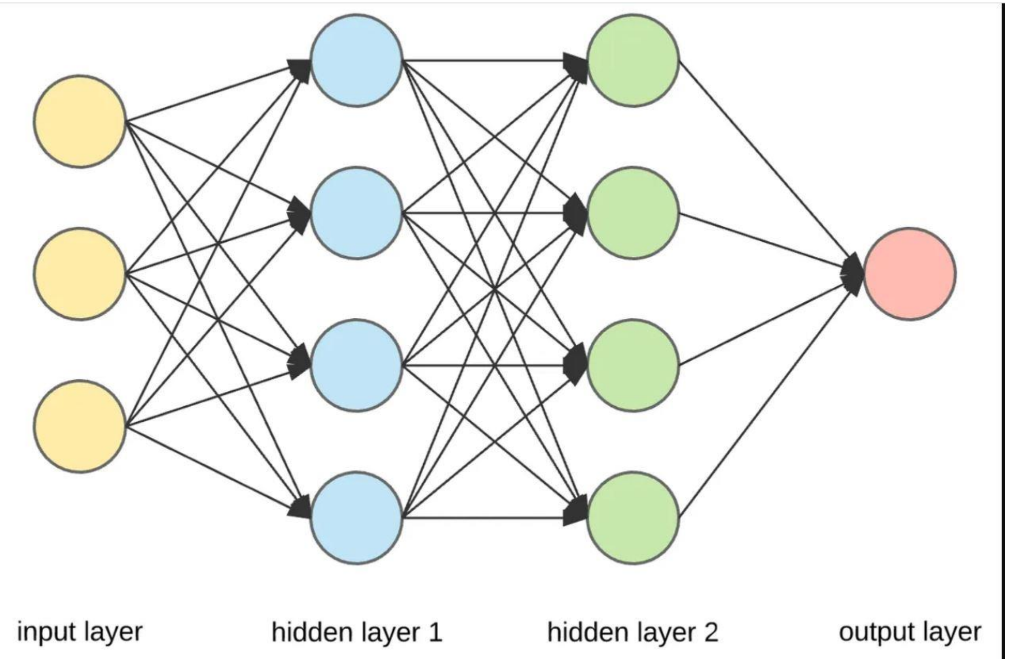
$$\begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 6 \end{bmatrix} \begin{pmatrix} 3 & 4 \\ 5 & 6 \\ 4 & 8 \end{pmatrix} = \begin{pmatrix} 43 & 66 \\ 55 & 84 \end{pmatrix}$$

Example

```
In [1]:
import numpy as np
arr =np.array([[2,3,5],[3,4,6]])
brr =np.array([[3,4,5],[4,6,8]])

In [2]:
np.matmul(arr.T,brr)

Out[2]: array([[18, 26, 34],
               [25, 36, 47],
               [39, 56, 73]])

In [3]:
np.matmul(arr,brr.T)

Out[3]: array([[43, 66],
               [55, 84]])

In [ ]:
```

Uses of Transpose

Evaluation of weight matrix and aggregate value:



Using Transpose To Develop a Model for Neural Networks

$$A^{[1]} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad W^{[1]} = \begin{pmatrix} w_{11} & w_{21} & w_{31} & w_{41} \\ w_{12} & w_{22} & w_{32} & w_{42} \\ w_{13} & w_{23} & w_{33} & w_{43} \end{pmatrix} \quad b^{[1]} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

(3x1) (3x4)

Note : A and W cannot be multiplied directly.

$$z^{[L]} = W^{[L]T} * A^{[L-1]} + b^{[L]}$$

Using Transpose To Develop a Model for Neural Networks : Change **W to W^T**

$$A^{[1]}=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad W^{[1]T}=\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix} \quad b^{[1]}=\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$z^{[L]} = W^{[L]T} * A^{[L-1]} + b^{[L]}$$

Using Transpose To Develop a Model for Neural Networks : **Plug it into expression.**

$$W^{[1]T} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Vector(4x1) Vector(4x3) Vector(3x1) Vector(4x1)

$z^{[1]} = W^{[1]T} * A^{[0]} + b^{[1]}$ (Successfully computed !!)

Example

In [21]:

```
W = np.random.rand(3,4)
X = np.random.rand(3,1)
b=  np.random.rand(4,1)
c=  np.random.rand(3,1)

W,X,b
```

Out[21]:

```
(array([[0.899255  , 0.52149056, 0.75988558, 0.46614012],
        [0.24637199, 0.49076638, 0.83288382, 0.51380146],
        [0.38120651, 0.16910539, 0.93104312, 0.54162068]]),
 array([[0.60389517],
        [0.45503992],
        [0.60113909]]),
 array([[0.91962447],
        [0.39762239],
        [0.68937966],
        [0.69974358]]))
```

In [22]:

```
# Here W is 4X3 weight matrix.
#Step1 : take transpose of W.

Y = np.matmul(W.T,X)+b
Y
```

Out[22]:

```
array([[1.80394745],
       [1.03752217],
       [2.0869527 ],
       [1.54063289]])
```

Sanity Check (For each layer)

- Number of outputs = Number of neurons.
- 4 = 4
- Number of Inputs = Number of Columns in W^T .
- 3 = 3
- Number of outputs = Number of bias vector element.
- 4 = 4
- Number of rows in W^T = Number of neurons.
- 4 = 4

Inverse of Matrices

Invertible Matrices ?

*Simple **telltale signs** for invertible matrices !!*

- Matrix is full rank.
- The determinant of is not zero.
- The matrix has non-zero singular values.

Inverse of Matrices

- The inverse of a square matrix , sometimes called a reciprocal matrix, is a matrix such that

AA⁻¹ = I (Identity Matrix)

the matrix inverse is

$$A^{-1} = \frac{1}{|A|}\begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{12} \\ a_{33} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{23} & a_{21} \\ a_{33} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{11} \\ a_{23} & a_{21} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{11} \\ a_{32} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}.$$

Inverse of Matrices

- $(AB)^{-1} = B^{-1}A^{-1}$
- $AA^{-1} = I$ (Important)
- $A^T = A^{-1}$, $A^TA = I$ (Important :**symmetric matrices**)
- Inverse of Diagonal Matrices

$$D^k = \begin{bmatrix} d_1^{\,k} & 0 & \cdots & 0 \\ 0 & d_2^{\,k} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & d_n^{\,k} \end{bmatrix}$$

;

$$D^{-1} = \begin{bmatrix} 1/d_1 & 0 & \cdots & 0 \\ 0 & 1/d_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1/d_n \end{bmatrix}$$

What is a Matrix?

- It is a transformation.
- Inverse will exist if a non singular matrix A has a unique matrix A^{-1} such that $AA^{-1} = I$.
- Relation between $A, A^{-1} \sim x, f(x)$.
- This concept is exploited in *change of basis transformation.*

Inverse of Matrices : Uses

Change of Basis :

- vector X has basis $u_1, u_2 \ldots u_n$.
- vector Y has basis $v_1, v_2 \ldots v_m$.
- Transformation A such that, $X = \sum x_i u_i$, $Y = \sum y_j v_j$

$$AX=Y$$

- vector X has basis $t_1, t_2 \ldots t_n$.
- vector Y has basis $w_1, w_2 \ldots w_m$.
- Transformation A' such that, $X = \sum x'_i t_i$, $Y = \sum y'_j w_j$

$$A'X' = Y'$$

Inverse of Matrices : Uses

- Change the each basis component to target component
- $t_i = \sum t_{ji} v_i$, $w_i = \sum w_{ji} u_j$
- Basis for $B_t = [t_1, t_2 \ldots\ldots\ldots]$, $B_w = [w_1, w_2 \ldots]$

$$A'X' = Y'$$

$$AB_t X' = B_w Y'$$

$$\boxed{B^{-1}_w AB_t = A'}$$

Example

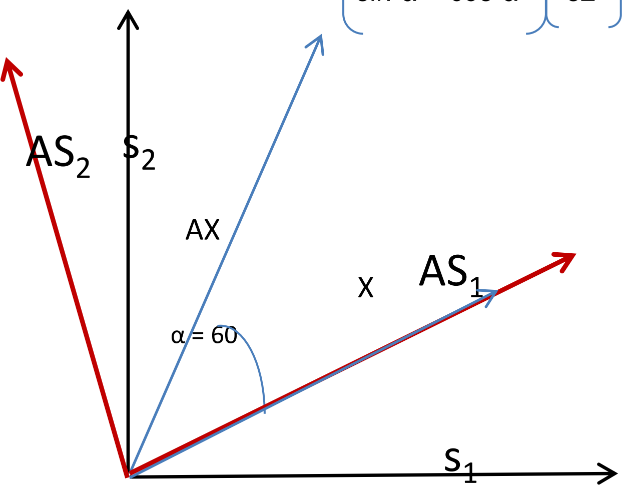
A vector is rotated through α degrees. Find transformation matrix. (NND Matrix representation page 6.6)

- X be a vector , AX be the transformed vector.

- If we rotate s_2 α degrees anticlockwise , transformation of s2.
- If we rotate s_1 α degrees anticlockwise , transformation of s1.

$AS_1 = s_1\cos(\alpha) + s_2\sin(\alpha)$ $AS_2 = s_1\cos(90+\alpha) + s_1\sin(90+\alpha)$

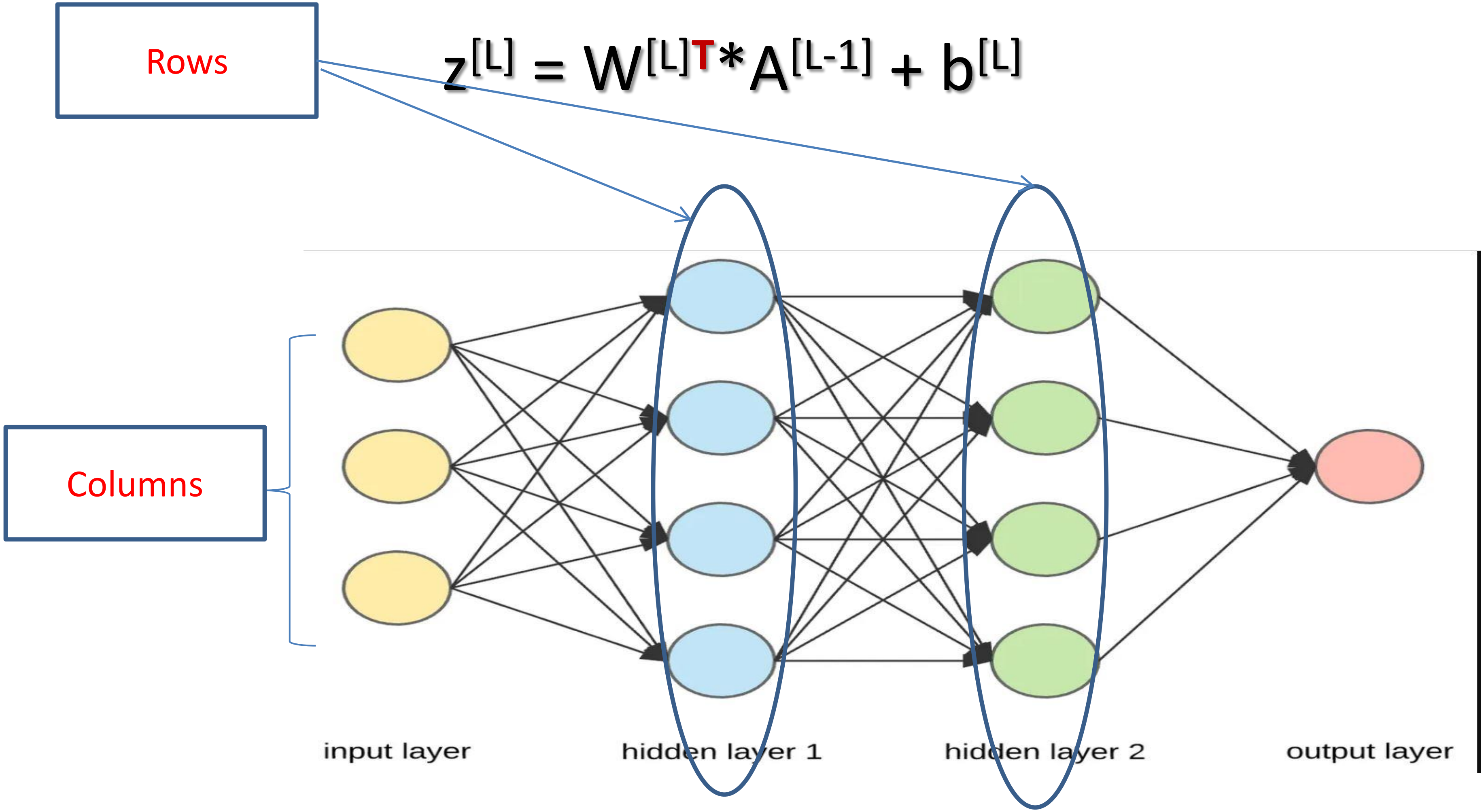
Combining $AS = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} s1 \\ s2 \end{pmatrix}$



Here Matrix is rotation transformation.

Vectorization : Revisit the network.

$z^{[L]} = W^{[L]T} * A^{[L-1]} + b^{[L]}$



Vectorization

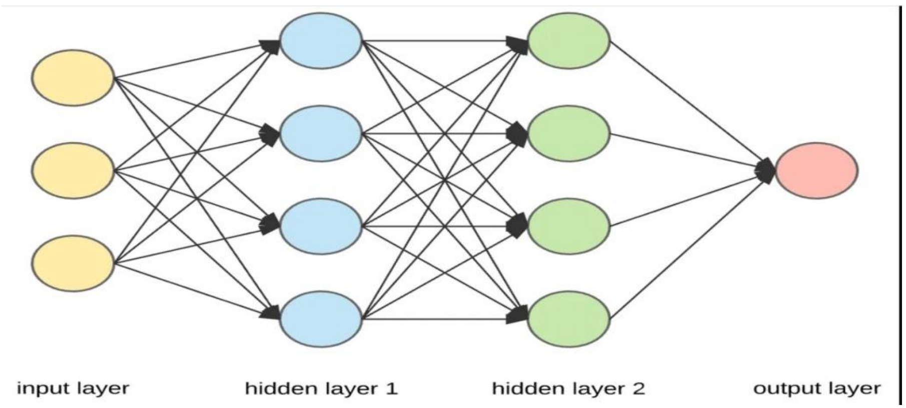
Input   : $X = (3x1)$

Layer 1 : $W^T = (4x3)$

Layer 2 : $W^T = (4x3)$

Layer 3 : $W^T = (1x4)$

Output : $Y = (1x1)$



Eigen Values and Vectors

$$AX = \lambda I$$

$$(AX - \lambda I) = 0$$

- We get values for λ , called eigen values and eigen vectors.
- Eigen is all about scaling without invariance i.e. eigen vectors are those vector whose dirction do not change after a transformation.

Eigen Values and Vectors

- Shock absorber should only change dimensions linearly .
- So direction of shock absorber should be along eigen vectors.

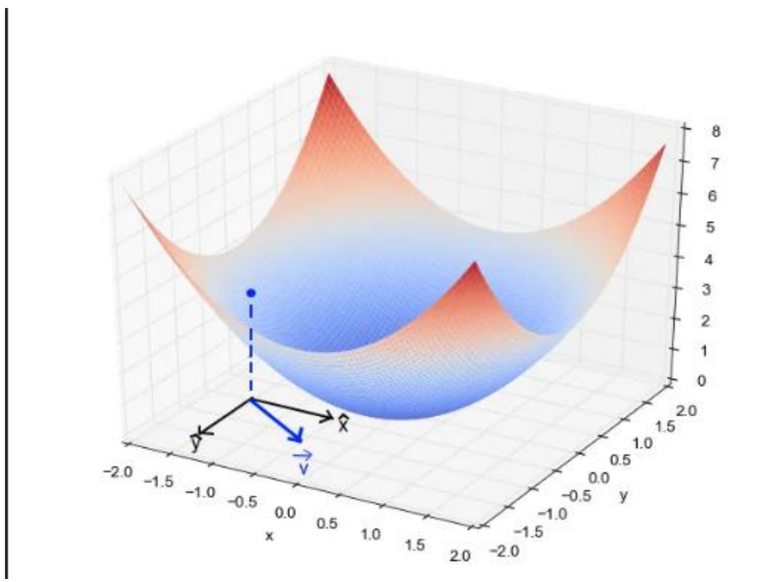
Eigen Values and Vectors :Uses

- Learning Rates :

$$\alpha = 2/\lambda_{max} \text{ (Steepest Descent)}$$

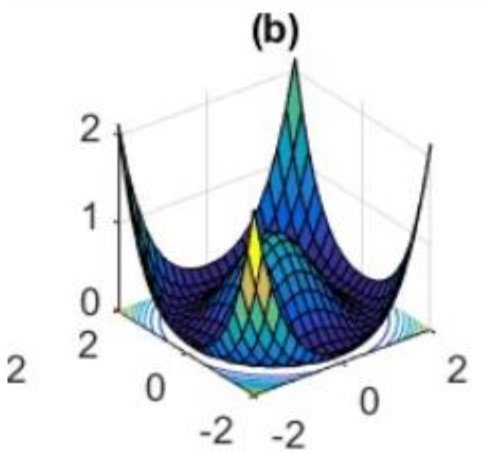
Eigen Values : Performance Surfaces

- λ_{max} is largest among all eigen values.
- λ_i > 0 , for each I . Positive Definite , Global Mininma , Strong Minima



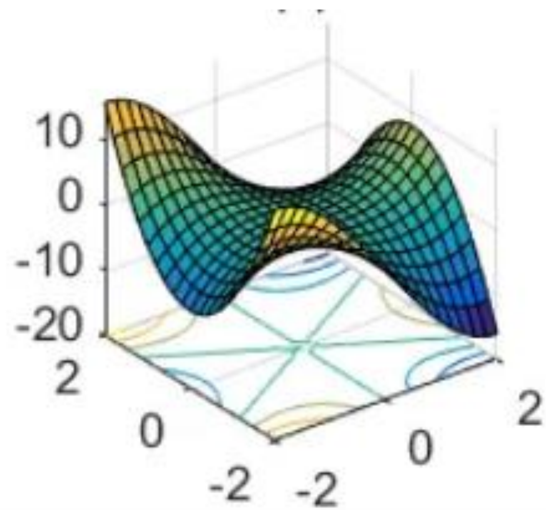
Eigen Values : Performance Surfaces

- λ_i >= 0 . Positive Semi Definite. Weak minima. No Global minima.



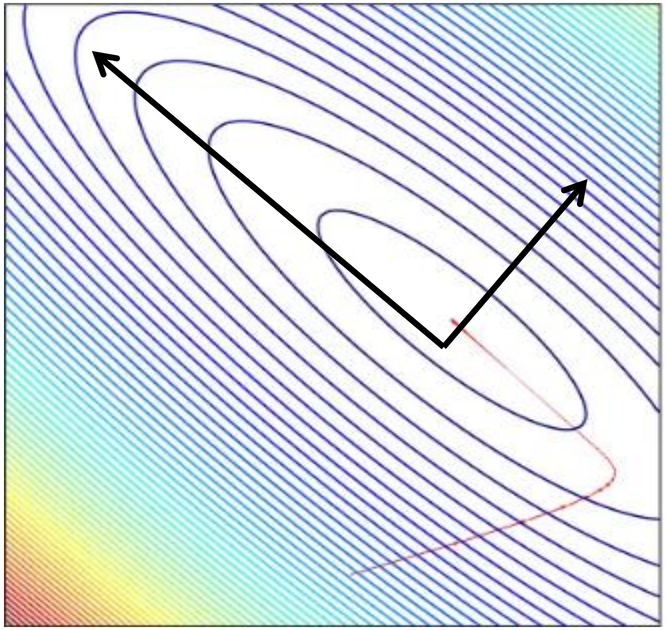
Eigen Values : Performance Surfaces

- λ_i are negative and positive , saddle point.



Eigen Values : Magnitude

- Magnitude of eigen value is proportional to the rate of contour crossing , irrespective of sign of eigen values.

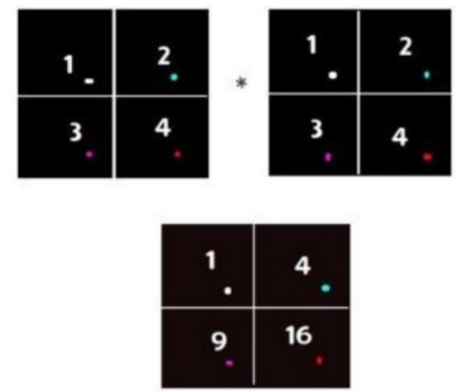


Eigen Values : Magnitude

- Magnitude of eigen value is proportional to the rate of contour crossing , irrespective of sign of eigen values.
- In case of Steepest descent , maximum gradient of a surface is along eigen vector, corresponding to largest eigen value.
- Steepest descent is special case of gradient descent .

Multiplication : Element wise and Dot

- Element wise



- Dot

```
Let the two 2D array are v1 and v2:-

v1=[[1, 2], [3, 4]]

v2=[[1, 2], [3, 4]]

Than numpy.dot(v1, v2)  gives output of :-

[[ 7 10]

 [15 22]]
```

References

[1] . Laurene Faussett,An *Introduction to Artificial Neural Network*. PearsonPublisher, 2018.

[2] .Jacek M. Zurada,*Introduction to Artificial Neural Network*. City:St.Paul Jaico Publisher, 2008, p.1-251.

Exploring More…

Thankyou