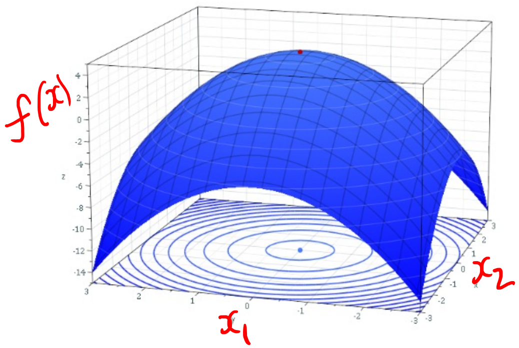
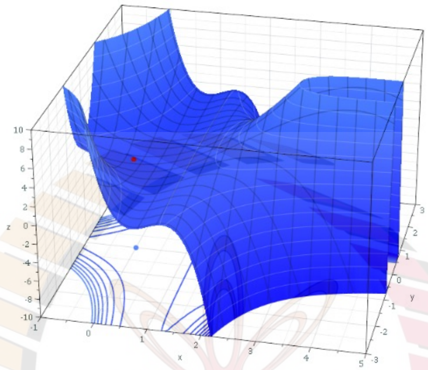
Machine Learning for Engineering and Science Applications

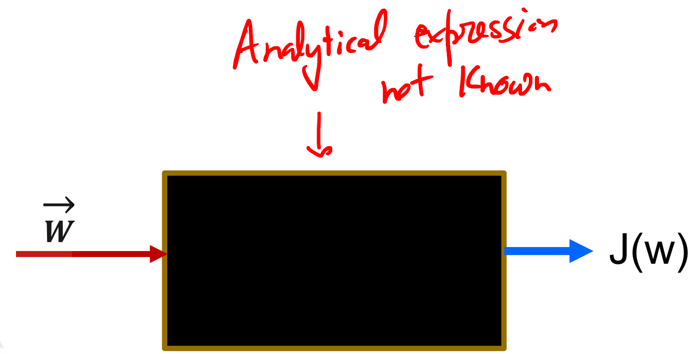
Introduction to Numerical Optimization

Gradient Descent-1

Need for Numerical Optimization







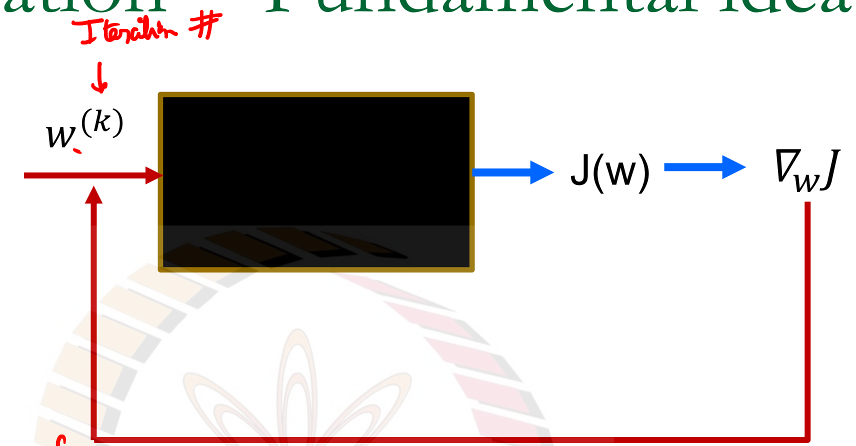
- Optimization we saw so far was analytical.

- This requires explicit expressions for the objective function in terms of the features (variables).
  - Example : $J(\mathbf{w}) = w_1^2 + w_2^2 + w_3^2 + 4$

- However, usually we only know the function as a “black” box.
  - In machine learning this “black box” is our Machine Learning Model (e.g. Neural network)

- So, we have to develop numerical (rather than analytical techniques)

Iterative optimization -- Fundamental idea



fn

- We want to drive $\nabla_w J$ to 0 but we do not have an analytical expression.

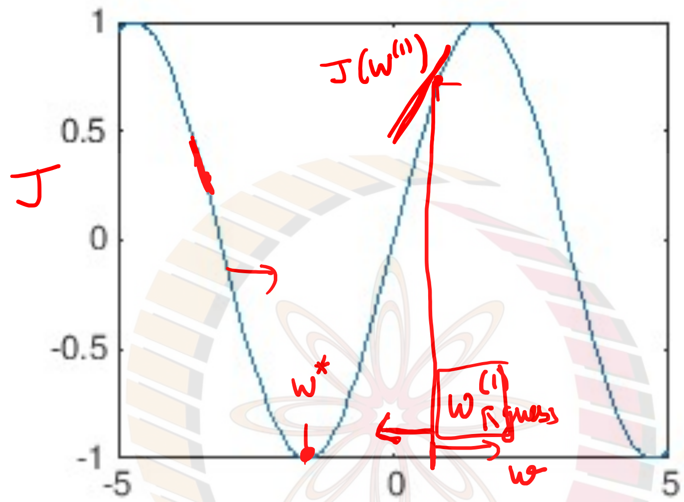
variable

Iterative Process

- Guess for w
- Run through the black box and find value of J(w)
  - This value may be obtained through a program instead of an expression
- Find $\nabla_w J$
  - We will discuss methods for determining $\nabla_w J$ numerically in later videos
- If $\nabla_w J = 0$, we stop, else we need to take a new guess
  - More precisely, improve our guess
- A very common method for improving guess is called Gradient Descent

Gradient Descent (Scalar case)

scalar. $J(w)$



$\frac{dJ}{dw} \neq 0$

$\frac{dJ}{dw} > 0$

$w = w - \alpha \left(\frac{dJ}{dw}\right)$

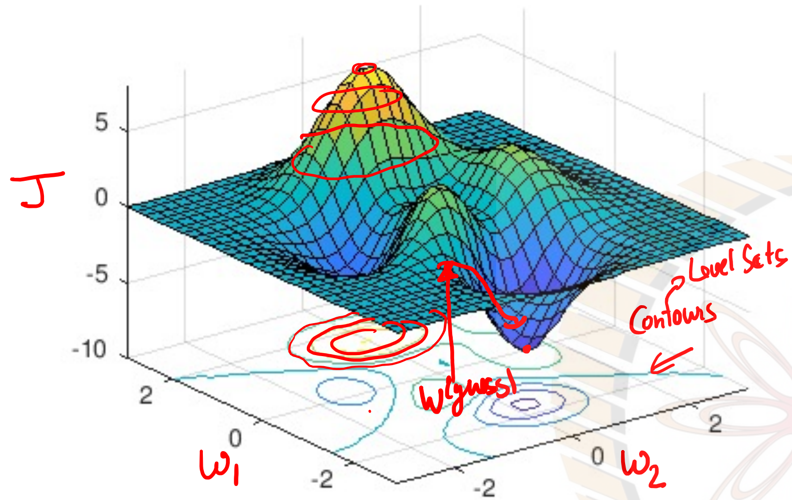
- Our task is to improve our guess for w such that we move from a region of higher gradient to a region of lower gradient

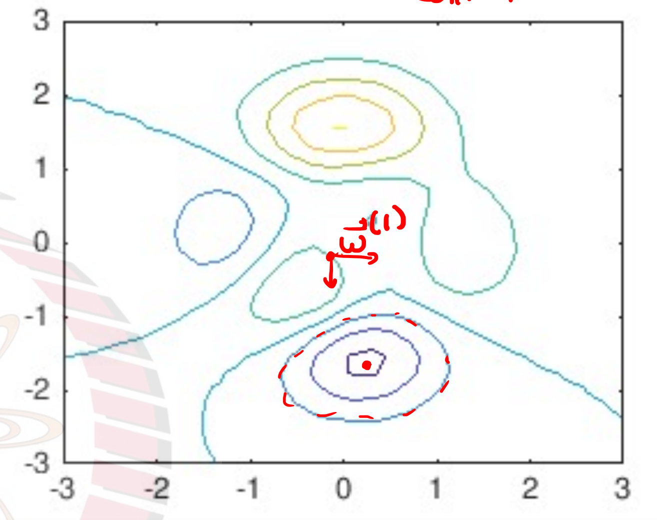
- For scalar (i.e. one component) w, this is easy

$$w^{new} = w^{old} - \alpha \left(\frac{dJ}{dw}\right)$$

Gradient Descent (vector case)

Contour $\Delta \vec{w}$





- For the vector case, we rely on a theorem that says

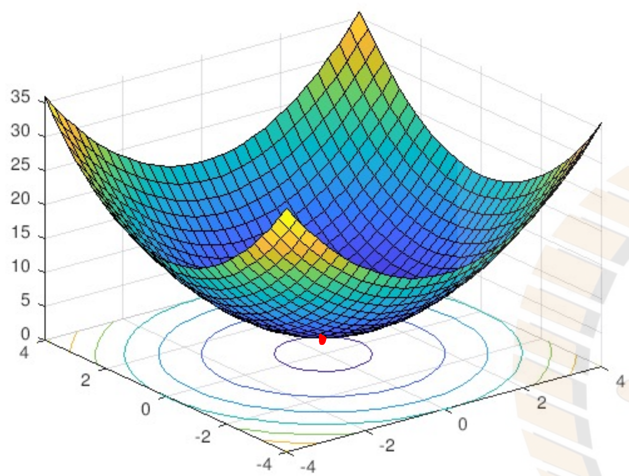
At any given point the gradient gives the direction of steepest descent

  - We will show a quick proof near the end of the video
- The general gradient descent algorithm is

$$\boxed{\vec{w}^{new} = \vec{w}^{old} - \alpha\, \overrightarrow{\nabla_w J}}$$

- α is called the learning rate is chosen by the user

Gradient Descent example

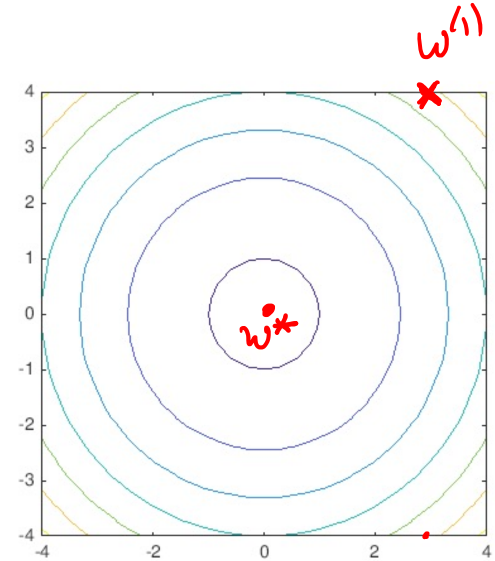


$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

$\vec{w}^{new} = \vec{w}^{old} - \alpha \, \vec{\nabla_w J}$

$w_1^{new} = w_1^{old} - \alpha \, (2w_1)$



Gradient Descent gives the iterative formula

$$w_1^{(k+1)} = w_1^{(k)} - \alpha \left(2w_1^{(k)}\right)$$
$$w_2^{k+1} = w_2^{k} - \alpha \left(2w_2^{k}\right)$$

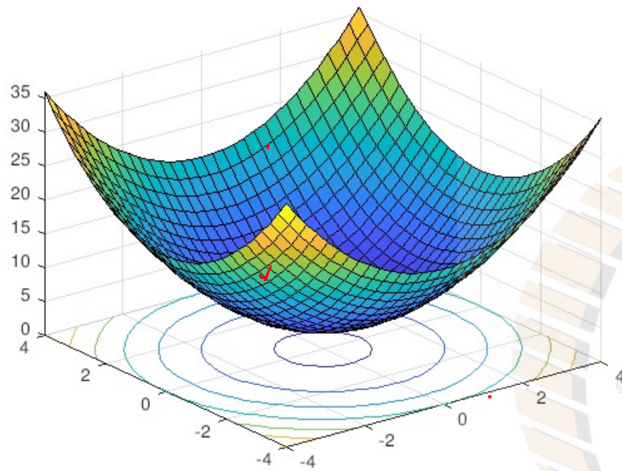
We know that the actual minimum is at $\mathbf{w} = [0 \;\; 0]^T$

Let us start with an initial guess of $\mathbf{w}^0 = [3 \;\; 4]^T$

Let us see different cases for various choices of α

$$\alpha = 2, 1\,, 0.1, 0.5$$

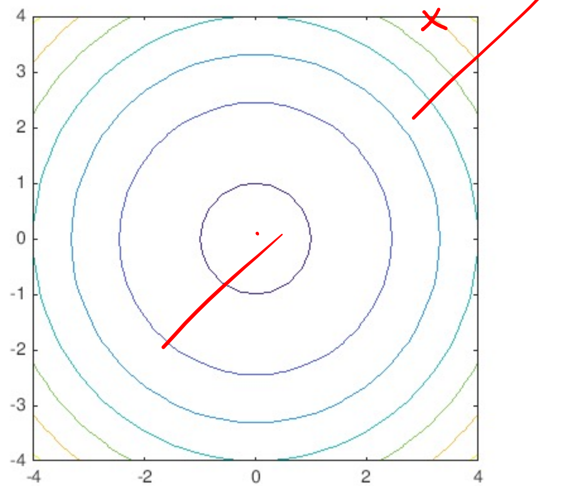
Gradient Descent example



$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

$$w_1^{k+1} = w_1^k - \alpha\left(2w_1^k\right)$$
$$w_2^{k+1} = w_2^k - \alpha\left(2w_2^k\right)$$

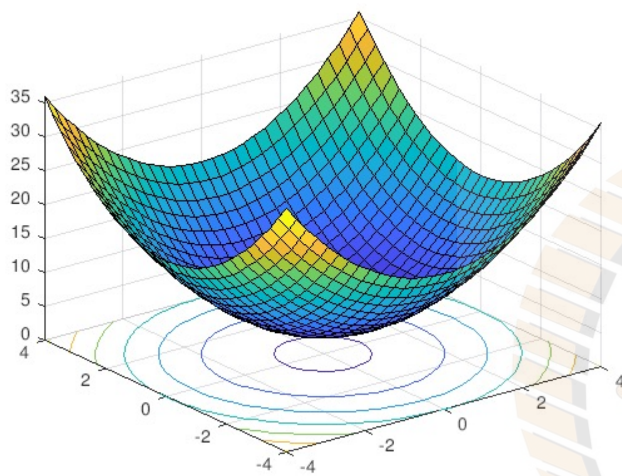


$\mathbf{w}^0 = [3 \quad 4]^T$ $\qquad$ $\alpha = 2$

Divergent

Iteration (k)	w^k	$\nabla_w J = 2[\mathrm{w}_1 \quad \mathrm{w}_2]$	J (cost)	$w^{k+1} = w^k - \alpha \nabla_\mathrm{w} \mathrm{J}$
0 Init guess	$[3 \quad 4]$	$[6 \quad 8]$	29	$[3 \quad 4] - 2 * [6 \quad 8] = [-9 \quad -12]$
1	$-[9 \quad 12]$	$-[18 \quad 24]$	229	$[27 \quad 36]$ Diverging
2	$[27 \quad 36]$	$[54 \quad 72]$	2029	$-[81 \quad 108]$

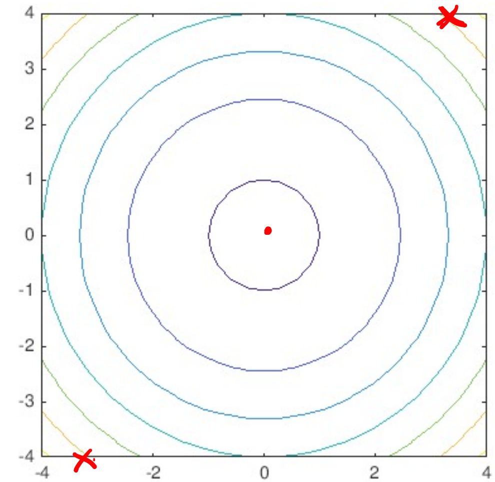
Gradient Descent example



$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

$$w_1^{k+1} = w_1^k - \alpha\left(2w_1^k\right)$$
$$w_2^{k+1} = w_2^k - \alpha\left(2w_2^k\right)$$

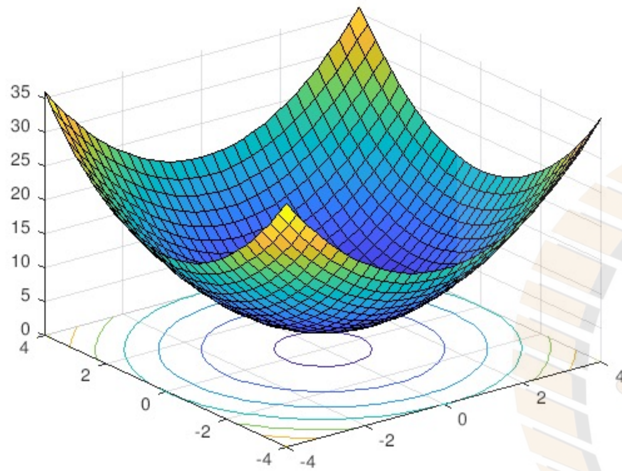


Oscillats

$\mathbf{w}^0 = [3 \quad 4]^T \qquad \alpha = 1$

Iteration (k)	w^k	$\nabla_w J = 2[w_1 \ \ w_2]$	J	$w^{k+1} = w^k - \alpha \nabla_w J$
0	$[3 \ \ 4]$	$[6 \ \ 8]$	29	$[3 \ \ 4] - 1 * [6 \ \ 8] = [-3 \ \ -4]$
1	$-[3 \ \ 4]$	$-[6 \ \ 8]$	29	$[3 \ \ 4]$
2	$[3 \ \ 4]$	$[6 \ \ 8]$	29	$[-3 \ \ -4]$

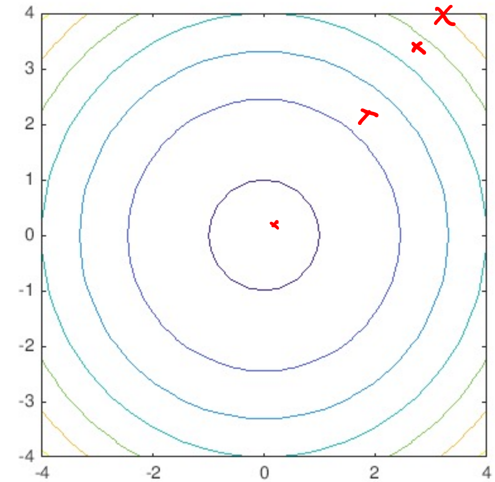
Gradient Descent example



$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

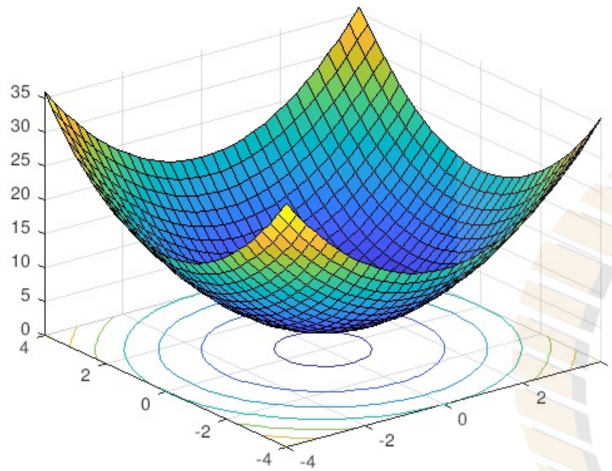
$$w_1^{k+1} = w_1^k - \alpha\left(2w_1^k\right)$$
$$w_2^{k+1} = w_2^k - \alpha\left(2w_2^k\right)$$



$\mathbf{w}^0 = [3 \quad 4]^T \qquad \alpha = 0.1$

Iteration (k)	w^k	$\nabla_w J = 2[\mathrm{w}_1 \;\; \mathrm{w}_2]$	J	$w^{k+1} = w^k - \alpha \nabla_\mathrm{w} \mathrm{J}$
0	[3 4]	[6 8]	29	[3 4] − 0.1 ∗ [6 8] = [2.4 3.2]
1	[2.4 3.2]	[4.8 6.4]	20	[1.92 2.56]
2	[1.92 2.56]	[3.84 5.12]	14.24	[1.536 2.048]
30	[0.0037 0.005]	…	4.0000	…

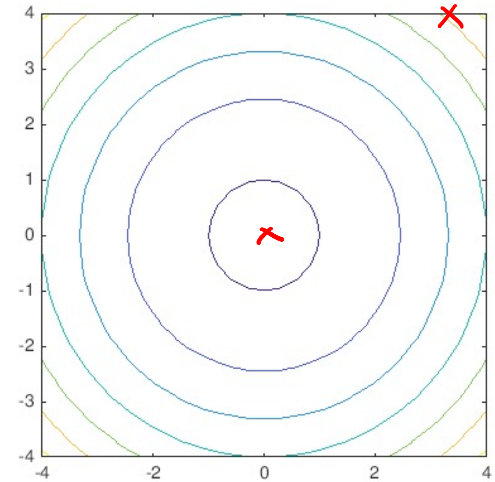
Gradient Descent example



$$J(w) = w_1^2 + w_2^2 + 4$$

$$\nabla_w J(w) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

$$w_1^{k+1} = w_1^k - \alpha\left(2w_1^k\right)$$
$$w_2^{k+1} = w_2^k - \alpha\left(2w_2^k\right)$$



$\mathbf{w}^0 = [3 \quad 4]^T$ $\alpha = 0.5$

Converges Rapidly

$W = W - \alpha \boxed{\nabla J}$

Iteration (k)	w^k	$\nabla_w J = 2[\mathrm{w}_1 \quad \mathrm{w}_2]$	J	$w^{k+1} = w^k - \alpha\nabla_\mathrm{w}\mathrm{J}$ $[0,0] = [0,0] - \alpha[0,0]$
0	$[3 \quad 4]$	$[6 \quad 8]$	29	$[3 \quad 4] - 0.5 * [6 \quad 8] = [0 \quad 0]$
1	$[0 \quad 0]$	$[0 \quad 0]$	4	$[0 \quad 0]$
2	$[0 \quad 0]$	$[0 \quad 0]$	4	$[0 \quad 0]$

Some lessons from the example

- It is possible for the gradient descent algorithm to
  - Diverge ($\alpha = 2$)
  - Oscillate without diverging or converging ($\alpha = 1$
  - Converge slowly ($\alpha = 0.1$)
  - Converge rapidly ($\alpha = 0.5$)

- All these behaviors can manifest for the sample example depending on the learning rate α

- The choice of α is part of algorithm design

- α is a *hyperparameter* – a parameter that must be set before learning begins

In the next video

Some details of the algorithm will be covered

- Proof of the steepest descent property
- Stopping criterion
- Calculating gradients when there is no analytical expression