# Overview of Optimization for ML

Sunita Sarawagi

# Motivation

- Many ML training algorithms can be posed as a continuous optimization problem

$$\min_{\{w_1,\dots,w_d\}} F(w_1, w_2, \dots, w_d) \iff \min_{w} F(\vec{w})$$

imize

$w_j \in \mathbb{R}$ $\quad\quad$ $w \in \mathbb{R}^d$

where the variables are the parameters of the model

F(w): loss over training data + regularizer

- Loss = continuous approximation of 0/1 error.
- Finding the minima of general functions could be intractably hard
  - Doable for certain types of functions → convex functions.

# Examples of functions

$d = 1$

$F(w_1) = 5w_1 + 7$ ; $F(w_1) = \log\left(1 + e^{-w_1}\right)$

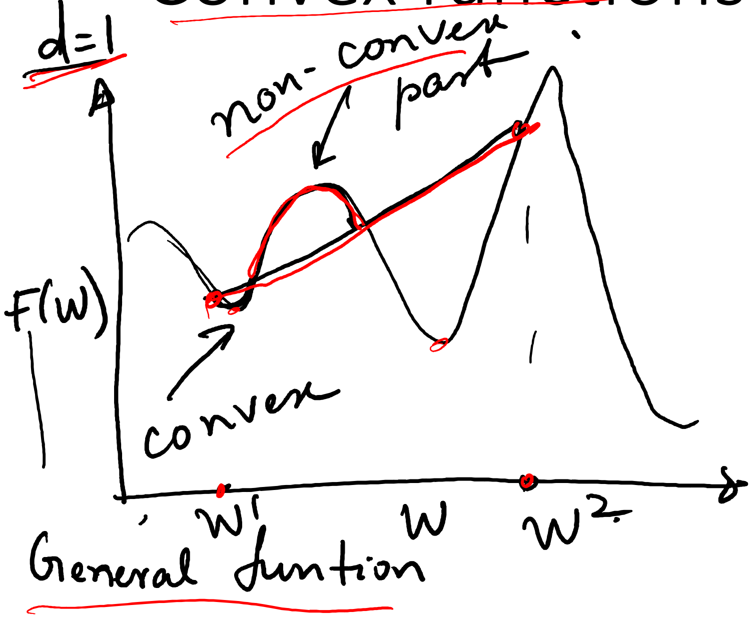
$$F(w_1) = \sum_{i=1}^{N} \left(y_i - w_1 x^i\right)^2$$

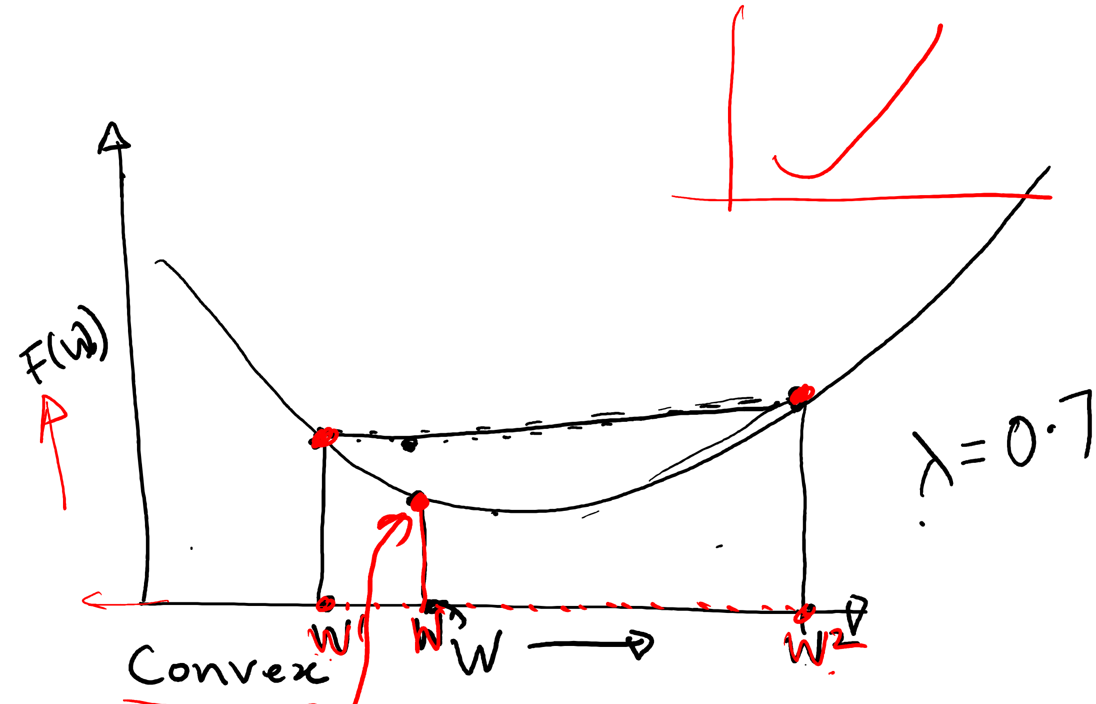
$(x^i, y_i) \equiv$ known constant

$d = 2$

$F(w_1, w_2) = w_1^2 + 2w_2 + w_1 w_2 + w_2^3 + \log(1 + w_1)$

# Convex functions

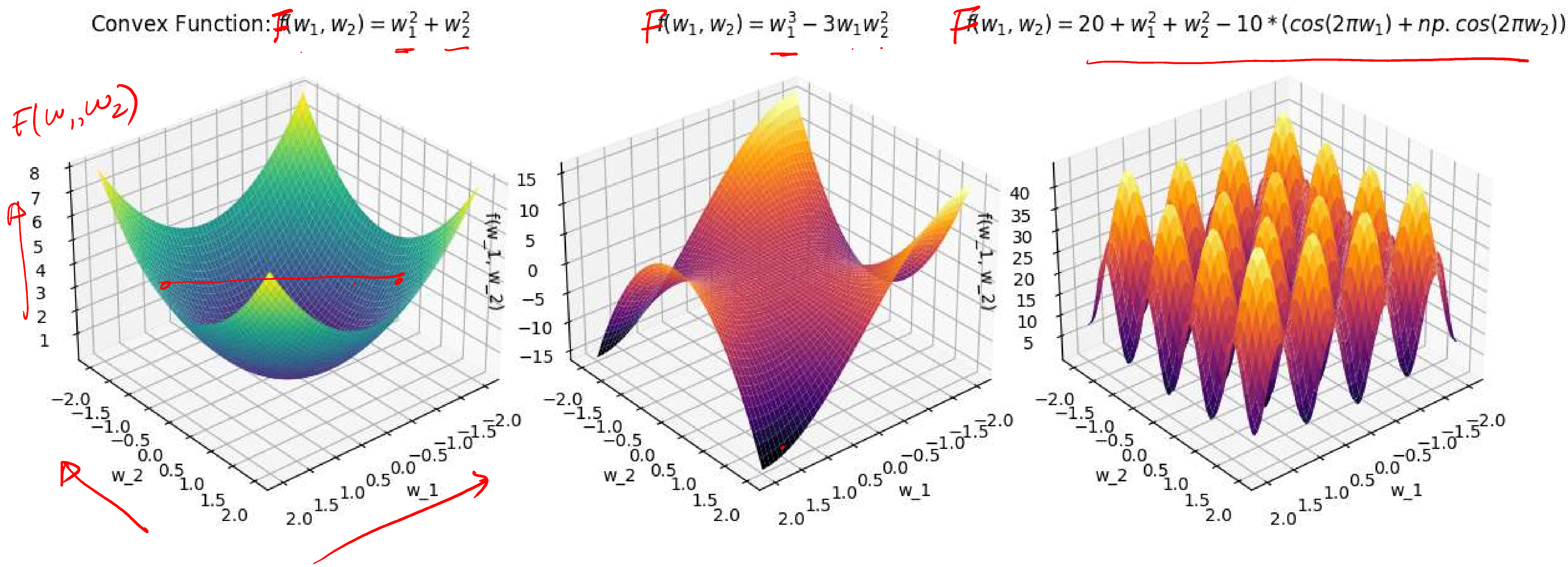



A function F(w) is convex in w if and only if (iff) for any $w^1, w^2 \in R^d, \lambda \in [0,1]$, $F(\underbrace{\lambda w^1 + (1-\lambda)w^2}_{\text{convex combination}}) \leq \lambda F(w^1) + (1-\lambda)F(w^2)$,

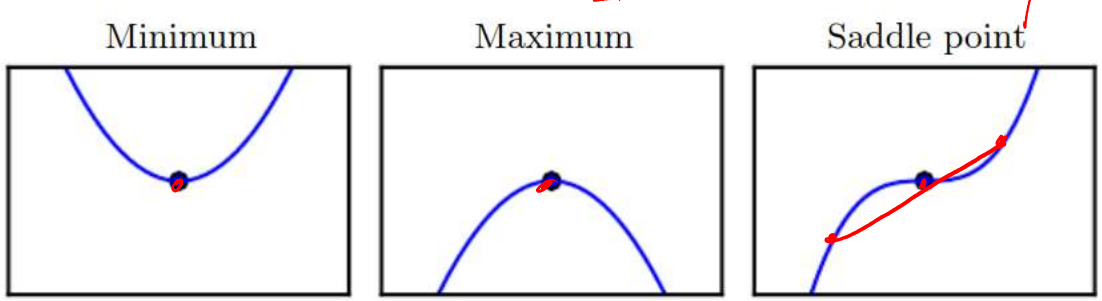
Concave functions = negative of convex functions.

# Convex Vs Non-convex functions in 2-d

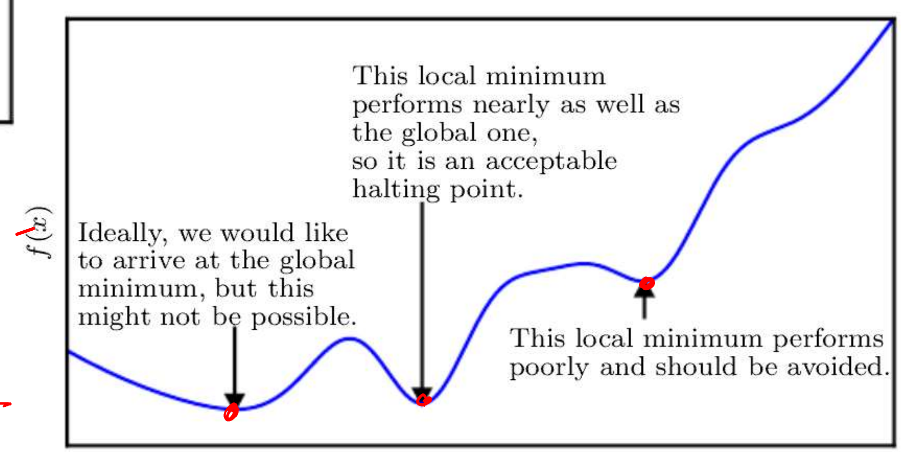


# Optimizing a function

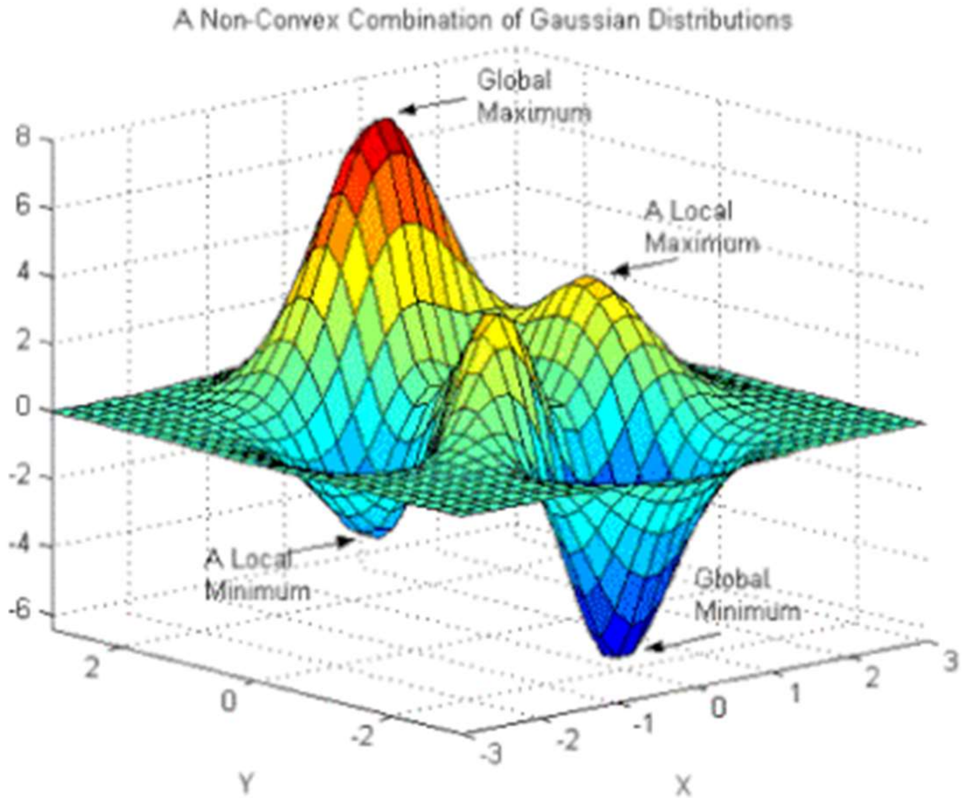
- $\min_{w} F(w),\ \ w \in R^d$
- A point $w^*$ is a **global minima** of F(w) if $F(w) \geq F(w^*)$
- A point $w^*$ is a **local minima** of F(w) if there exists an $\epsilon > 0$ $\ \ such\ that\ \ F(w) \geq F(w^*)\quad \forall |w - w^*| \leq \epsilon$

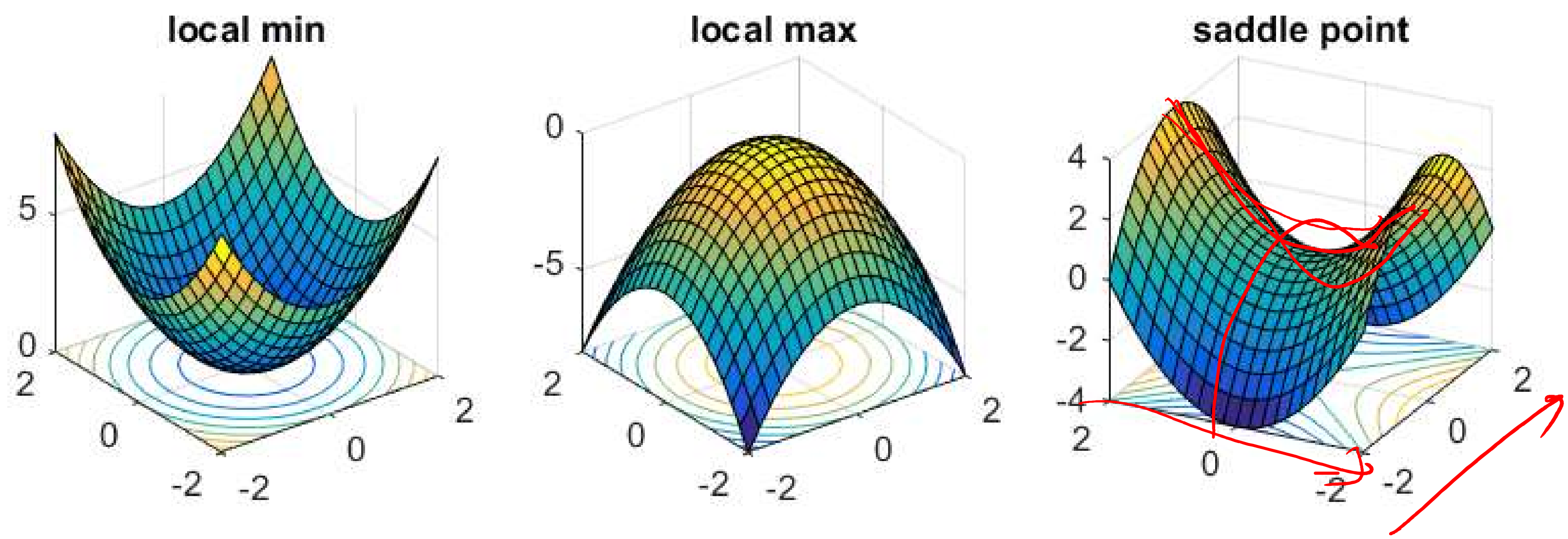



d = 1



# Local and global minima in 2-D





local min
local max
saddle point

# Gradient of a differentiable function

- Derivative of a function on a single variable measures the rate of change of the function.

$$F(w_1) = w_1^2 + 2w_1^3$$

$$\frac{\partial F(w_1)}{\partial w_1} = 2w_1 + 6w_1^2$$

$$F(w_1) = \log(1 + e^{-w_1})$$

$$\frac{\partial F(w_1)}{\partial w_1} = \frac{-e^{-w_1}}{1 + e^{-w_1}}$$

$$F(w_1) = \sum_{i=1}^{N} (y^i - w_1 x^i)^2$$

$$\frac{\partial F(w_1)}{\partial w_1} = \sum_{i=1}^{N} 2(y^i - w_1 x^i)(-x^i)$$

# Gradients of multi-variable functions

- For multivariable functions $F(w_1, \dots, w_d)$ we can define partial derivative of F w.r.t each of the variables.

$$\frac{\partial F}{\partial w_1}, \frac{\partial F}{\partial w_2}, \dots \frac{\partial F}{\partial w_d}$$

- Gradient of F(w) denoted as $\nabla F(w)$: vector of partial derivative of the function

$$\nabla_w F(\vec{w}) = \begin{bmatrix} \frac{\partial F}{\partial w_1} \\ \frac{\partial F}{\partial w_2} \\ \vdots \\ \frac{\partial F}{\partial w_d} \end{bmatrix}$$

# Example of gradient

$d=2$

$$F(w_1, w_2) = \underline{F(\mathbf{w})} = \frac{1}{2}\left(w_1^2 + 10\, w_2^2\right)$$

$$\nabla F(w) = \begin{bmatrix} \frac{\partial F}{\partial w_1} \\ \frac{\partial F}{\partial w_2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \cdot 2 \cdot w_1 \\ \frac{1}{2} \cdot 10 \cdot 2 \cdot w_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ 10\, w_2 \end{bmatrix}$$

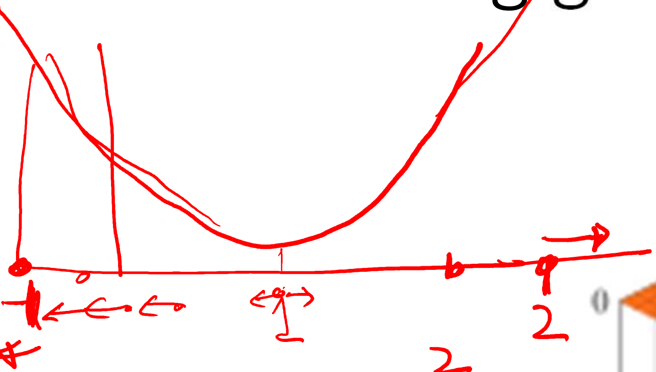
$$\nabla F(w) \Big|_{w = \underline{[1,\ 3]'}} = \begin{bmatrix} 1 \\ 10 \times 3 \end{bmatrix}.$$

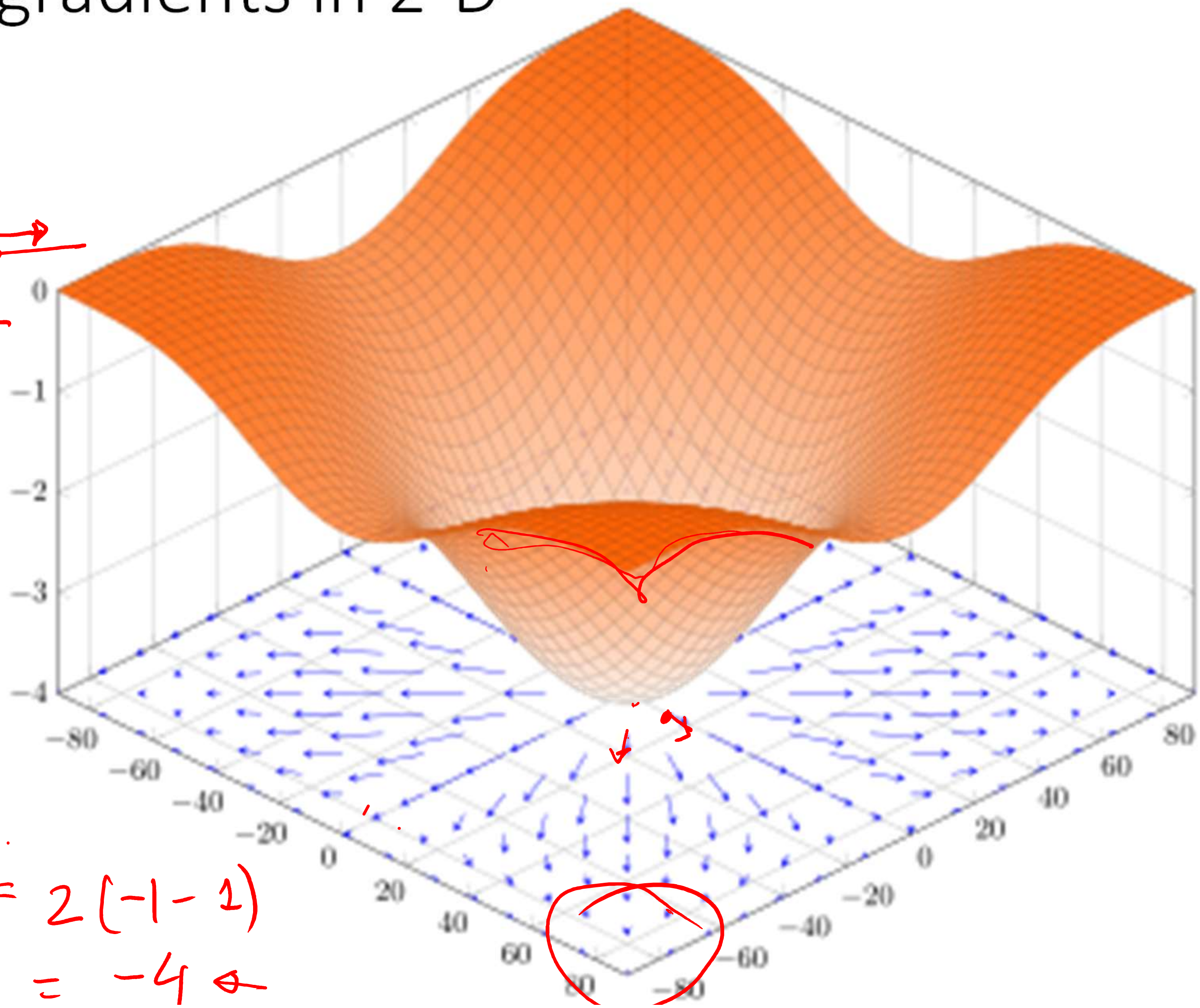
# Minima and gradients

- **Theorem**: If F(w) is differentiable and if $w^*$ is a local minima of F(w), then $\nabla F(w^*) = 0$

- Theorem: For convex functions a $w^*$ is a global minima if and only if $\nabla F(w^*) = 0$. That is the local minima is the global minima.

# Visualizing gradients in 2-D




$$F(w_1) = (w_1 - 1)^2$$

$$\left.\frac{\partial F(w_1)}{\partial w_1}\right|_{w_1=2} = 2(w_1 - 1)$$

$$= 2(2-1)$$

$$= 2$$

$$\left.\frac{\partial F(w_1)}{\partial w_1}\right|_{w_1=-1} = 2(-1-1)$$

$$= -4$$

# Iterative optimization algorithms

- Most often we will not be able to solve for $\nabla F(w) = 0$ in closed form
  - E.g. MLE for the logistic loss. (Show)
- Iterative algorithms (General template)
  - $w^0$ = Choose an initial point.
  - For t = 1 to stopping criteria (local minima, maximum iterations, etc)
    - $w^{t+1} \leftarrow$ Move to a near-by point $w^t$ such that F(w) reduces.
- Many iterative algorithms have been proposed for such cases
  - Zero-th order algorithm (line-search) for optimizing 1-d convex functions.
  - First-order or gradient-based algorithms
    - Gradient descent
    - Conjugate gradient descent,
    - .
  - Second-order algorithms
  - Semi or pseudo second order algorithms

# Example of solving in closed form

$$F(w_1, w_2) = (w_1 - 1)^2 + 3(w_2 - 10)^2$$

$$\nabla F(w) = \begin{bmatrix} 2(w_1 - 1) \\ 6(w_2 - 10) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \quad 2w_1^* - 2 = 0 \quad \Rightarrow \quad w_1^* = 1$$

$$6w_2^* - 60 = 0 \quad \Rightarrow \quad w_2^* = 10$$

$$F(w_1, w_2) = \log\left(e^{-w_1 + 2w_2}\right) + \log\left(e^{w_1 + 7w_2}\right)$$

# Iterative Optimization using gradients

- Choose an arbitrary initial point : $w^0 \Leftarrow [0 \ldots 0]^T$
- $\lambda$ = Chosen learning rate ← has to be small but not too small.
- Epoch t = 0 —
- While stopping criteria not reached $(t)$

$\nabla F(w)\big|_{w=w^t}$ $\nabla F(w^t) \equiv$ Compute gradient of function at current $w^t$.

if $\|\nabla F(w^t)\| \approx 0$, then return $w^t$ as minima.

$$w^{t+1} = w^t - \lambda \nabla F(w^t).$$

descend gradient

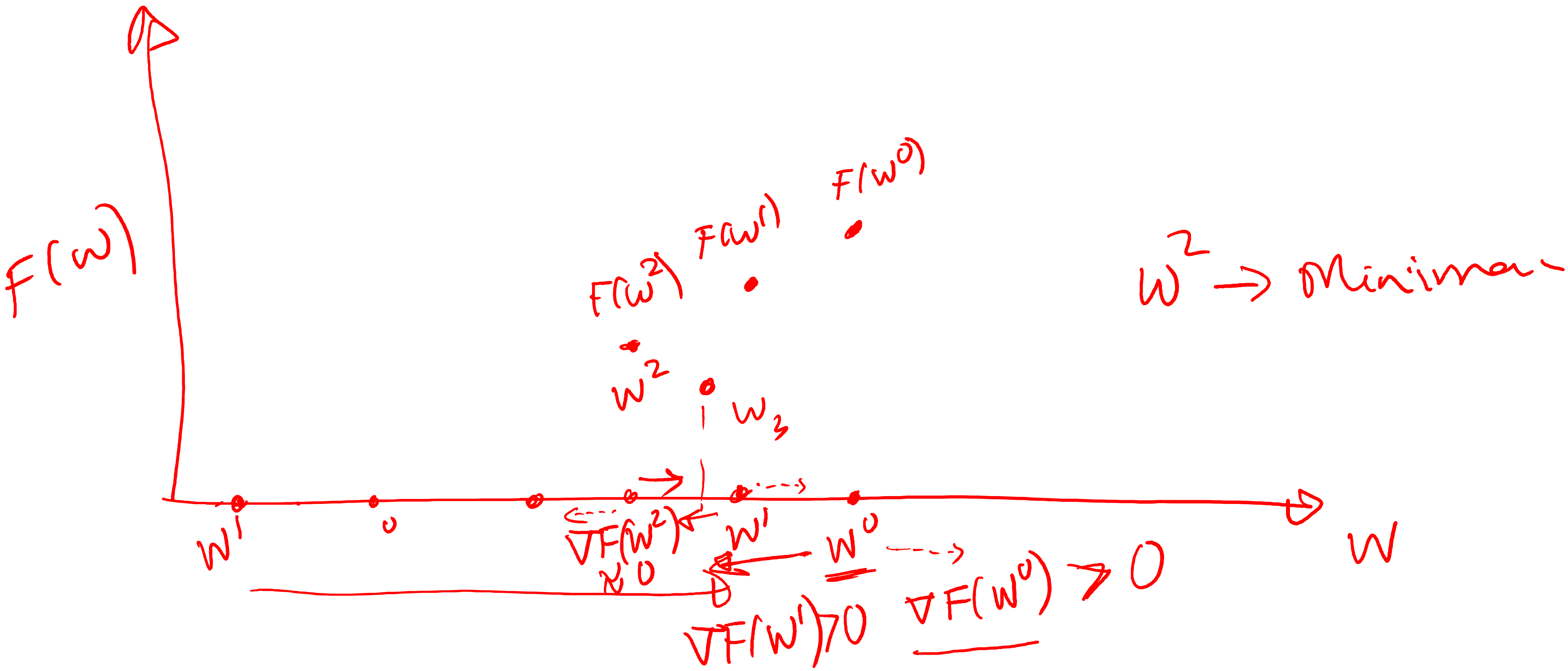
$$F(w^{t+1}) \leq F(w^t)$$

if $F(w)$ is differentiable and $\lambda$ is small.

$$t = t+1$$

# Iterative minimization of 1-d convex functions

- Convex functions ➔ double derivative (rate of change of gradients) is non-negative. Minima where derivation = 0.



# Example: gradient descent

- $F(w_1, w_2) = \frac{1}{2}(w_1^2 + 10w_2^2)$
- $w^0 = [10, 1]^T$

$\nabla F(w^0) = [10,\ 10]$ ; $\lambda = 0.1$ $\quad F(w^0) = \frac{110}{2}$

$$w^1 = \begin{bmatrix} 10 \\ 1 \end{bmatrix} - 0.1 \begin{bmatrix} 10 \\ 10 \end{bmatrix} = \begin{bmatrix} 9 \\ 0 \end{bmatrix} \quad F(w^1) = \frac{81}{2}$$

$$\nabla F(w^1) = \begin{bmatrix} 9 \\ 0 \end{bmatrix}$$

$$w^2 = \begin{bmatrix} 9 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} 9 \\ 0 \end{bmatrix} = \begin{bmatrix} 8.1 \\ 0 \end{bmatrix}$$

$\vdots$

$$w^T = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} \quad \nabla F(w^T) = \left\| \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} \right\| \approx 0 \rightarrow \text{stop}$$

# Gradient descent geometrically

- See this colab link to run the demo yourself.

# Stochastic gradient descent

- Stochastic approximation to gradient descent optimization when applied over sum of errors on several i.i.d training examples
- Typical training objective:
  - $L(w) = \frac{1}{N} \sum_{i=1} L(f(x^i; w), y_i)$
  - True gradient:
  - Stochastic approximation:
- More efficient than full batch
- Empirical found to be better at optimizing non-convex functions because of noisy nature of gradients.