Machine learning & deep learning basics

Lecture 2

Changho Suh

January 22, 2024

Gradient descent and DNNs

Logistic regression

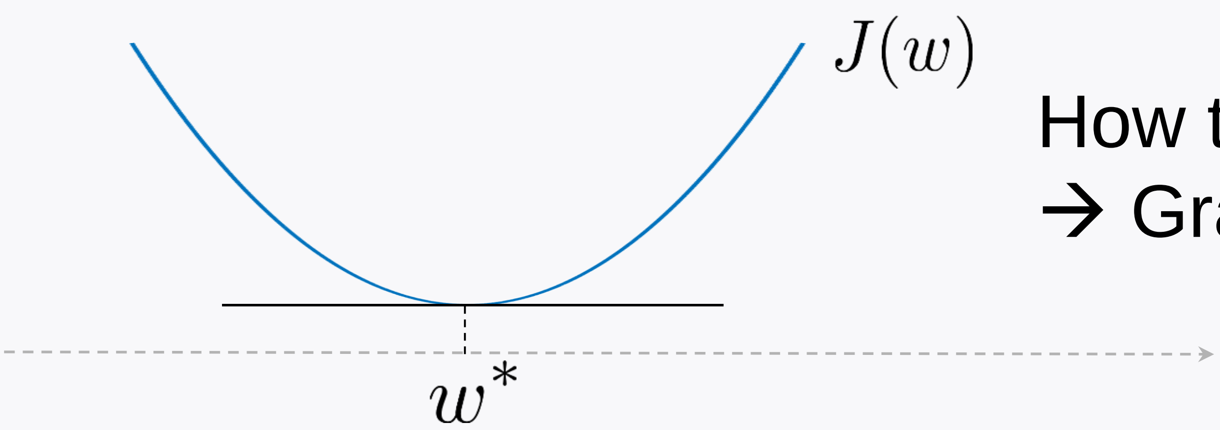
$$\min_{w} \boxed{\sum_{i=1}^{m} -y^{(i)} \log \frac{1}{1+e^{-w^T x^{(i)}}} - (1-y^{(i)}) \log \left(1 - \frac{1}{1+e^{-w^T x^{(i)}}}\right)}$$

$$=: J(w)$$

Employ: Perceptron w/ logistic function

Cross Entropy (CE) loss

Convex optimization

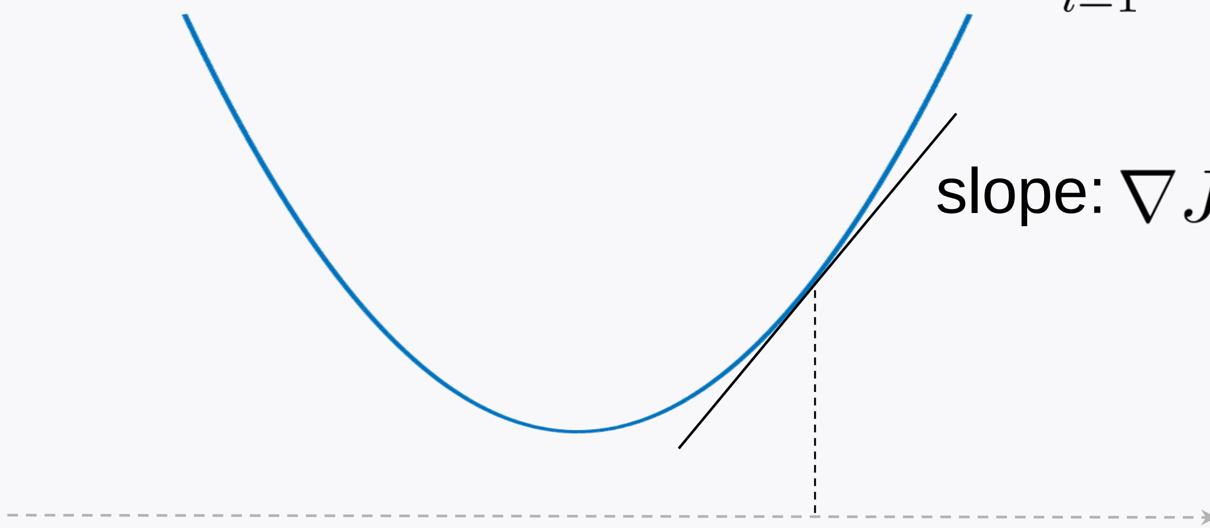


How to find w^*?
→ Gradient descent

Gradient descent

iterative algorithm

$$J(w^{(t)}) := \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$



slope: $\nabla J(w^{(t)})$

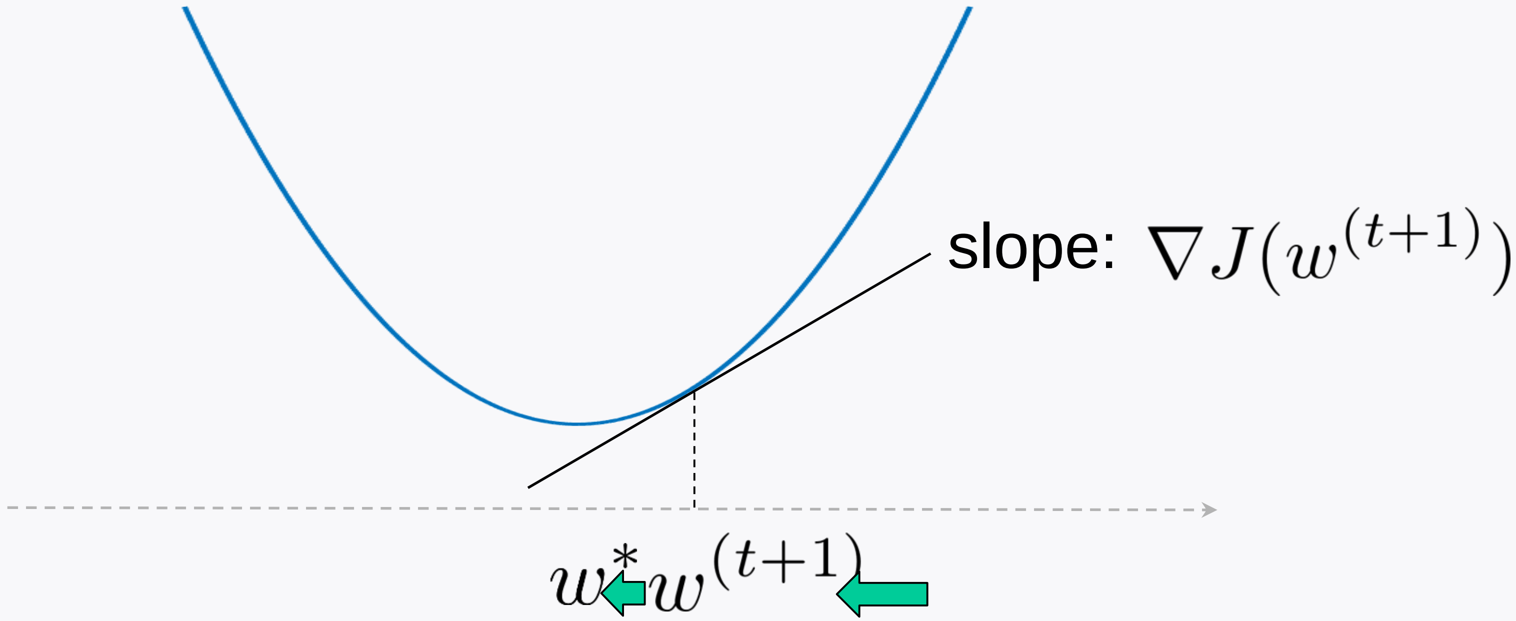
w^* $\quad$ $w^{(t)}$ *t-th* estimate

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha^{(t)} \nabla J(w^{(t)})$$

move to the left! $\quad$ learning rate (>0)

Gradient descent

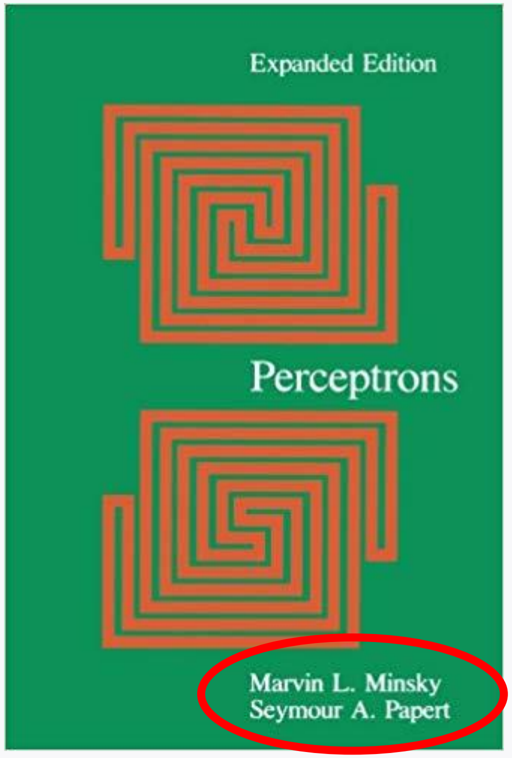
(t+1)-*th* step:

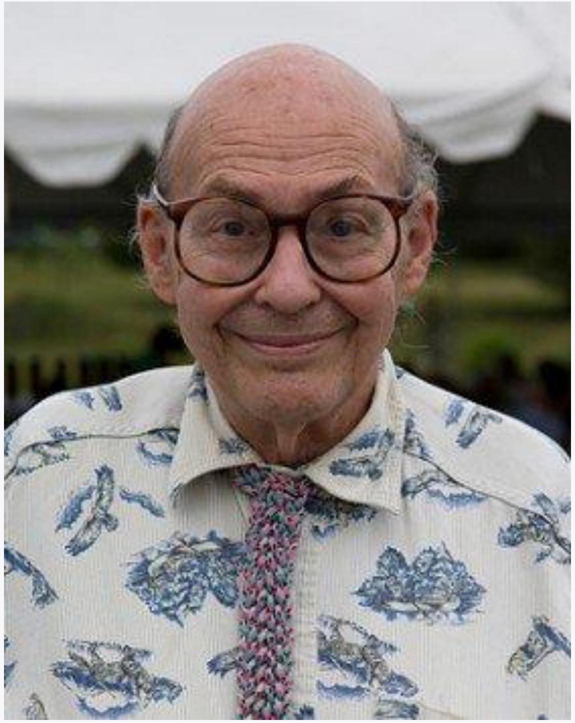


$$w^{(t+2)} \leftarrow w^{(t+1)} - \alpha^{(t+1)} \nabla J(w^{(t+1)})$$

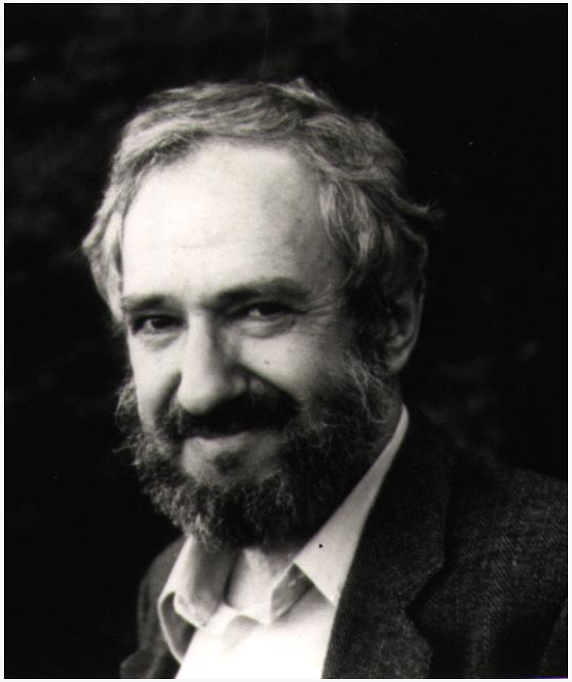
Turns out: $w^{(t)} \longrightarrow w^* \quad \text{as } t \to \infty$

AI boomed in 1960s but …





Marvin Minsky



Seymour Papert '69

Demonstrated limitations of the Perceptron architecture.

→ Led to the AI winter!

AI revived in 2012

Alex Krizhevsky

Ilya Sutskever

Geoffrey Hinton

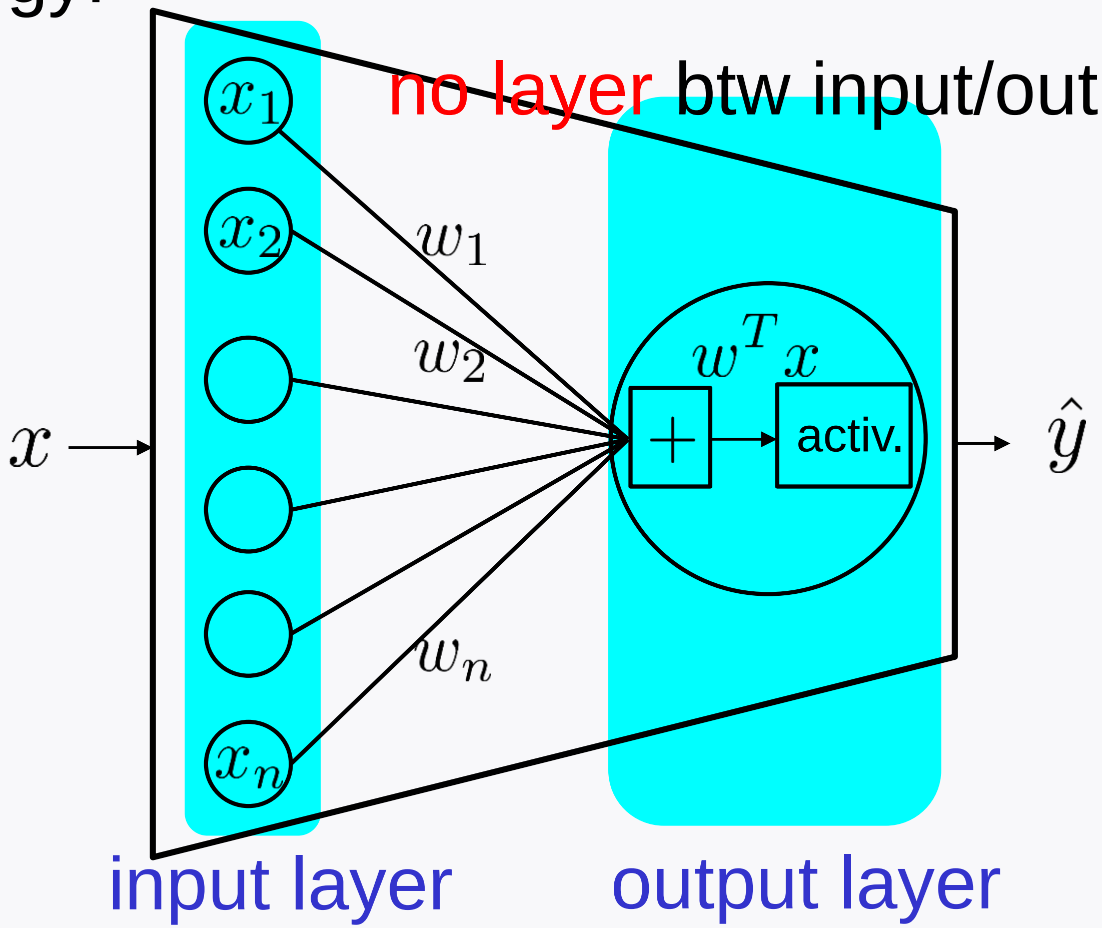
Developed a deep neural network (DNN), named **AlexNet**, that can achieve *human-level recognition performances*!

Anchored the start of deep learning revolution!

DNN architecture

Perceptron architecture

terminology:



→ called: A shallow neural network

Deep neural networks (DNN)

Say: A neural network is deep if it has **at least one layer** btw input/output layers.

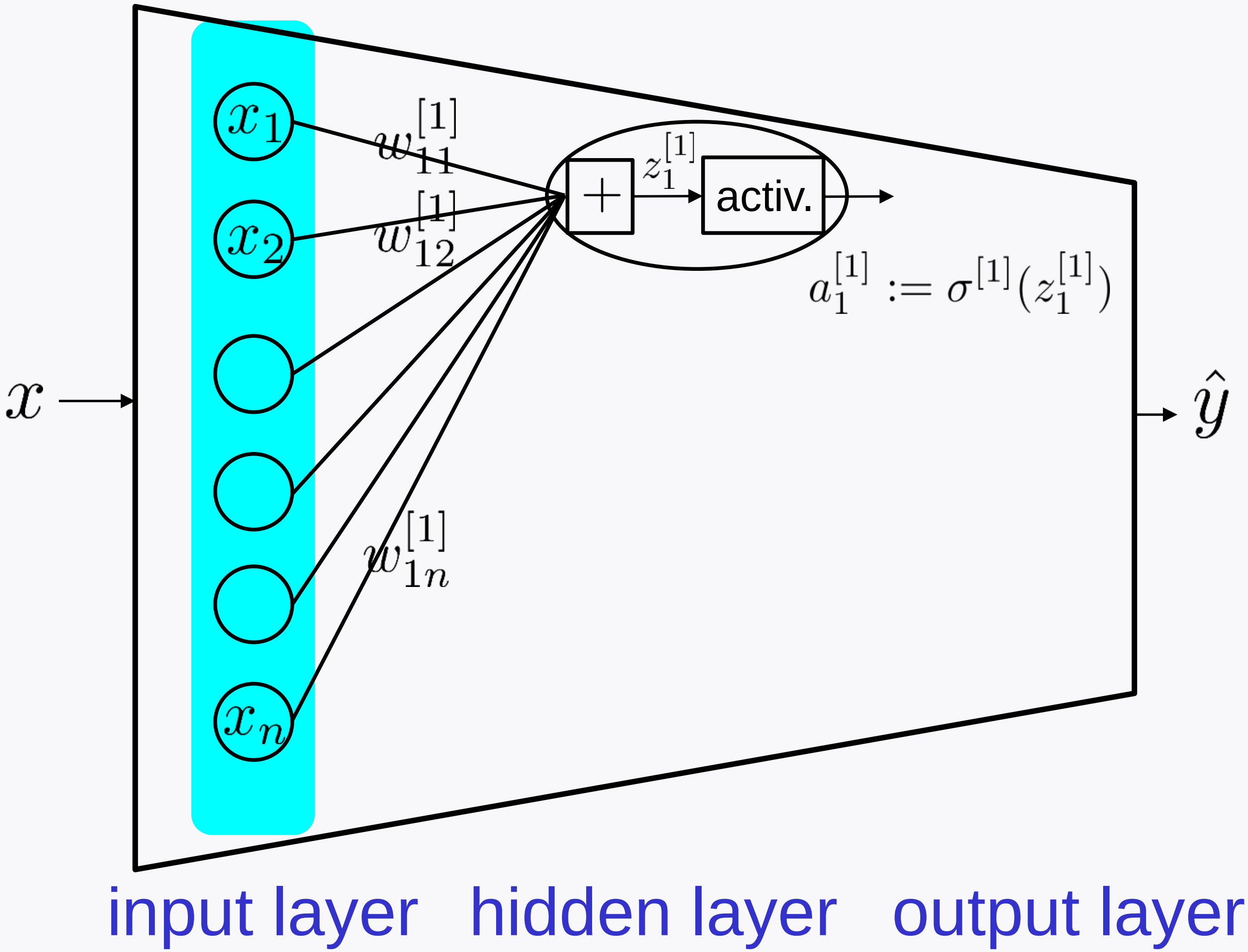
hidden layer

Definition: A deep neural network is a network that contains hidden layer(s).

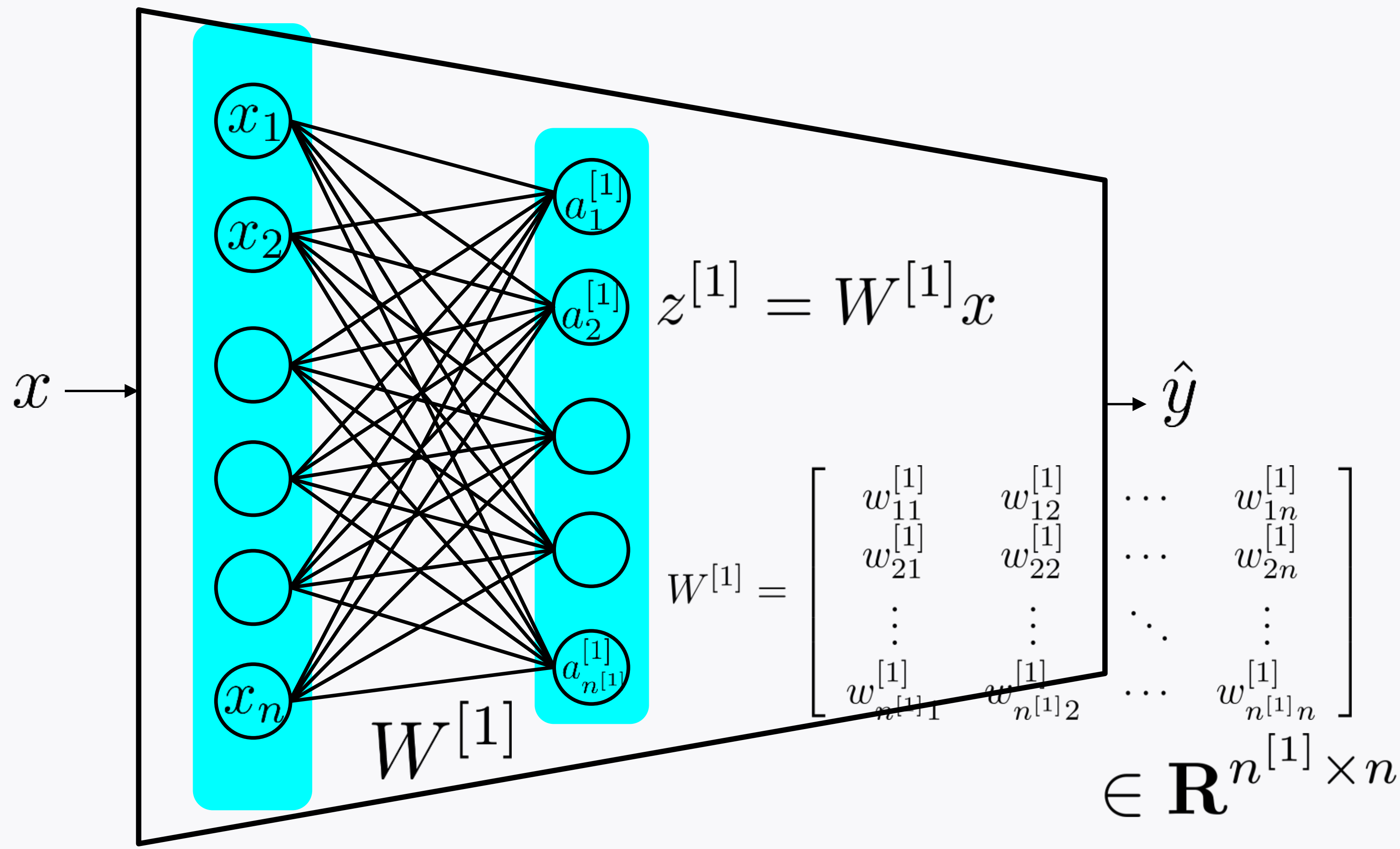
Convention: L-hidden-layer network

= (L+1)-layer network

DNN architecture

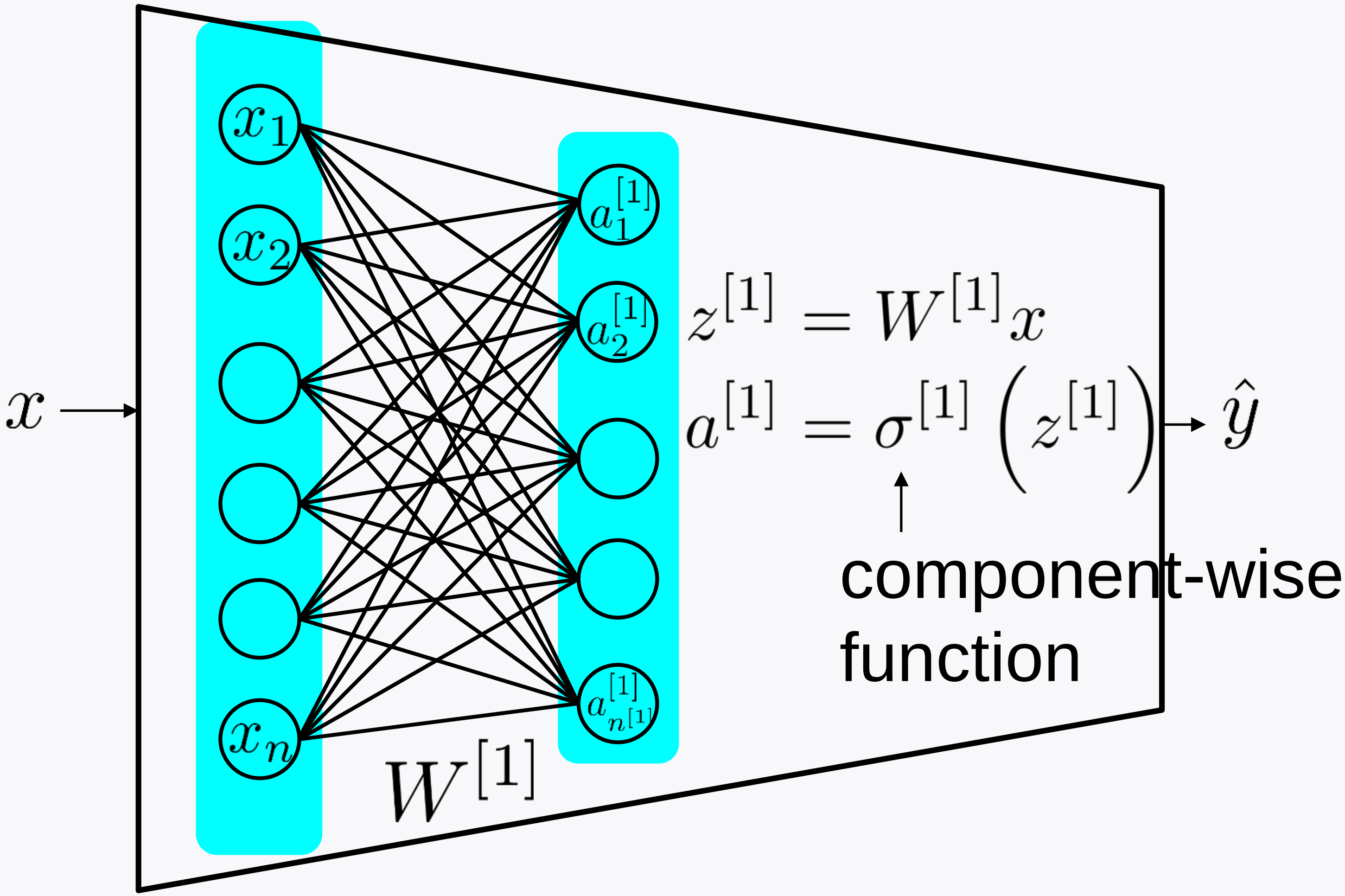


DNN architecture



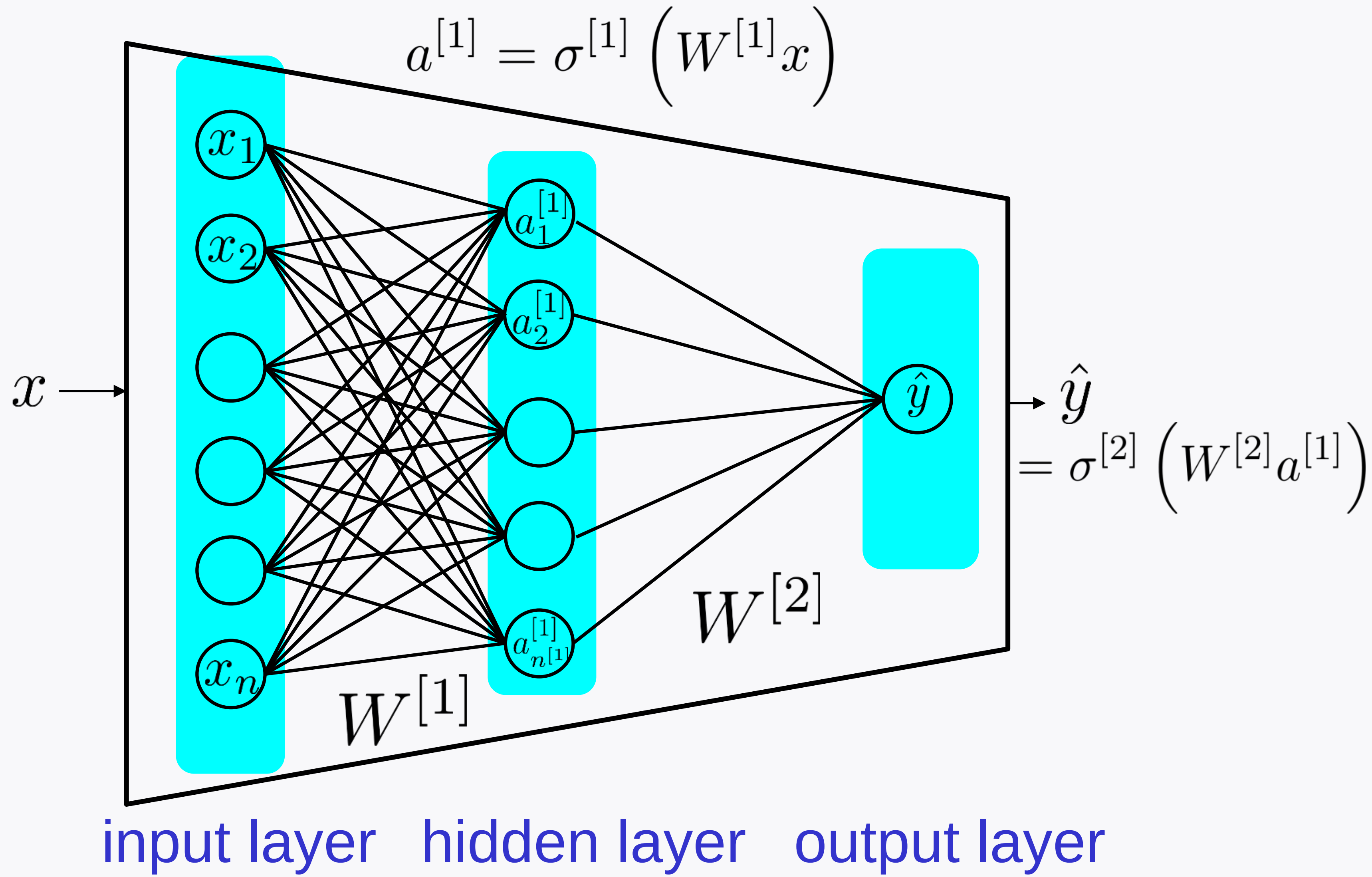
input layer hidden layer output layer

DNN architecture



input layer hidden layer output layer

DNN architecture



DNN architecture: *L* hidden layers

$$a^{[1]} = \sigma^{[1]}\left(\overset{\in \mathbf{R}^{n^{[1]}\times n}}{W^{[1]}} x\right) \qquad \in \mathbf{R}^{n^{[1]}}$$

$$a^{[2]} = \sigma^{[2]}\left(\overset{\in \mathbf{R}^{n^{[2]}\times n^{[1]}}}{W^{[2]}} a^{[1]}\right) \qquad \in \mathbf{R}^{n^{[2]}}$$

$$\vdots$$

$$a^{[L]} = \sigma^{[L]}\left(\overset{\in \mathbf{R}^{n^{[L]}\times n^{[L-1]}}}{W^{[L]}} a^{[L-1]}\right) \qquad \in \mathbf{R}^{n^{[L]}}$$

$$\hat{y} = \sigma^{[L+1]}\left(\overset{\in \mathbf{R}^{1\times n^{[L]}}}{W^{[L+1]}} a^{[L]}\right) \qquad \in \mathbf{R}$$

DNN-based optimization

Optimization: 2-layer DNN

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, \hat{y}^{(i)})$$

$$\hat{y} = \sigma^{[2]} \left(W^{[2]} a^{[1]} \right)$$

$$a^{[1]} = \sigma^{[1]} \left(W^{[1]} x \right)$$

Optimization: 2-layer DNN

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, \hat{y}^{(i)})$$

$$\hat{y}^{(i)} = \sigma^{[2]} \left(W^{[2]} a^{[1],(i)} \right)$$

$$a^{[1],(i)} = \sigma^{[1]} \left(W^{[1]} x^{(i)} \right)$$

$$w = (W^{[1]}, W^{[2]})$$

Choice of loss function and activation functions?

How to choose a loss function?

$$\min_w \sum_{i=1}^{m} \ell(y^{(i)}, \hat{y}^{(i)})$$

$$\underline{\hat{y}^{(i)} = \sigma^{[2]}}\left(W^{[2]} a^{[1],(i)}\right)$$

$$a^{[1],(i)} = \sigma^{[1]}\left(W^{[1]} x^{(i)}\right)$$

$$w = (W^{[1]}, W^{[2]})$$

Use a logistic function for

$$\sigma^{[2]}(z) = \sigma(z) := \frac{1}{1+e^{-z}}$$

Use cross entropy loss.

How to choose $\sigma^{[1]}(\cdot)$?

$$\min_{w} \sum_{i=1}^{m} -y^{(i)} \log \hat{y}^{(i)} - (1-y^{(i)}) \log(1-\hat{y}^{(i)})$$

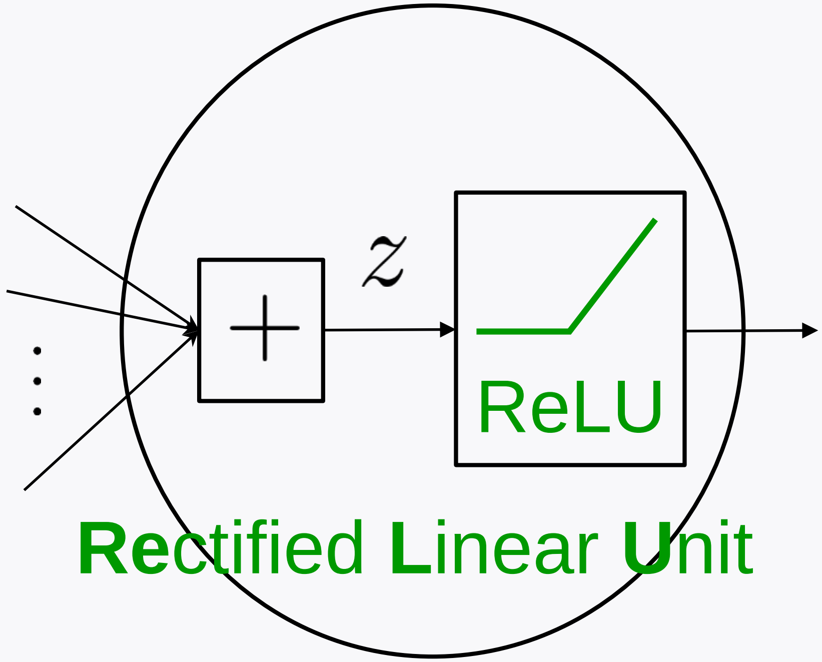
$$\hat{y}^{(i)} = \sigma\left(W^{[2]} a^{[1],(i)}\right)$$

$$a^{[1],(i)} = \underline{\sigma^{[1]}}\left(W^{[1]} x^{(i)}\right)$$

$$w = (W^{[1]}, W^{[2]})$$

Widely-used activation function

Operation at a hidden layer neuron:



$$a = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

$$a = \max(0, z)$$

DNN-based optimization

$$\boxed{\min_{w=(W^{[1]},W^{[2]})} \sum_{i=1}^{m} -y^{(i)} \log \hat{y}^{(i)} - (1-y^{(i)}) \log(1-\hat{y}^{(i)})}$$

$$\hat{y}^{(i)} = \sigma\left(W^{[2]} \max\left(0, W^{[1]} x^{(i)}\right)\right)$$

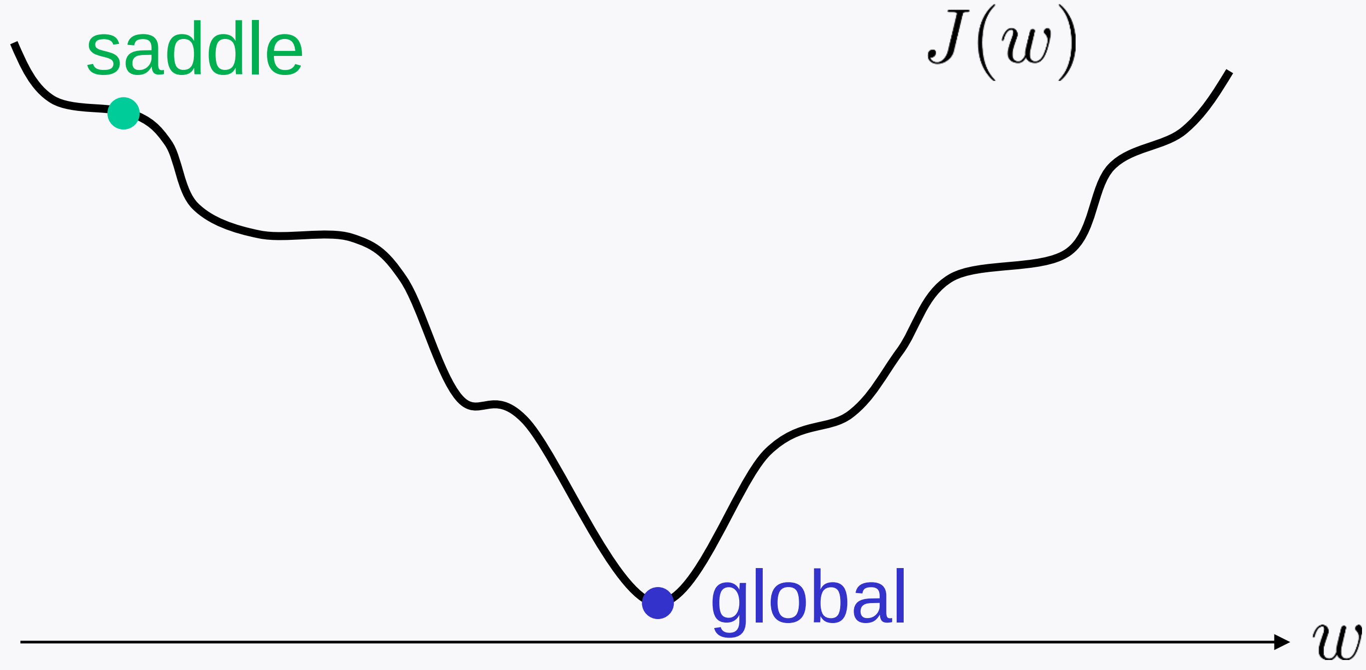
Question: Is the objective convex or non-convex?

Turns out: It is non-convex.

How to handle such a non-convex problem?

Observation (by many practitioners):

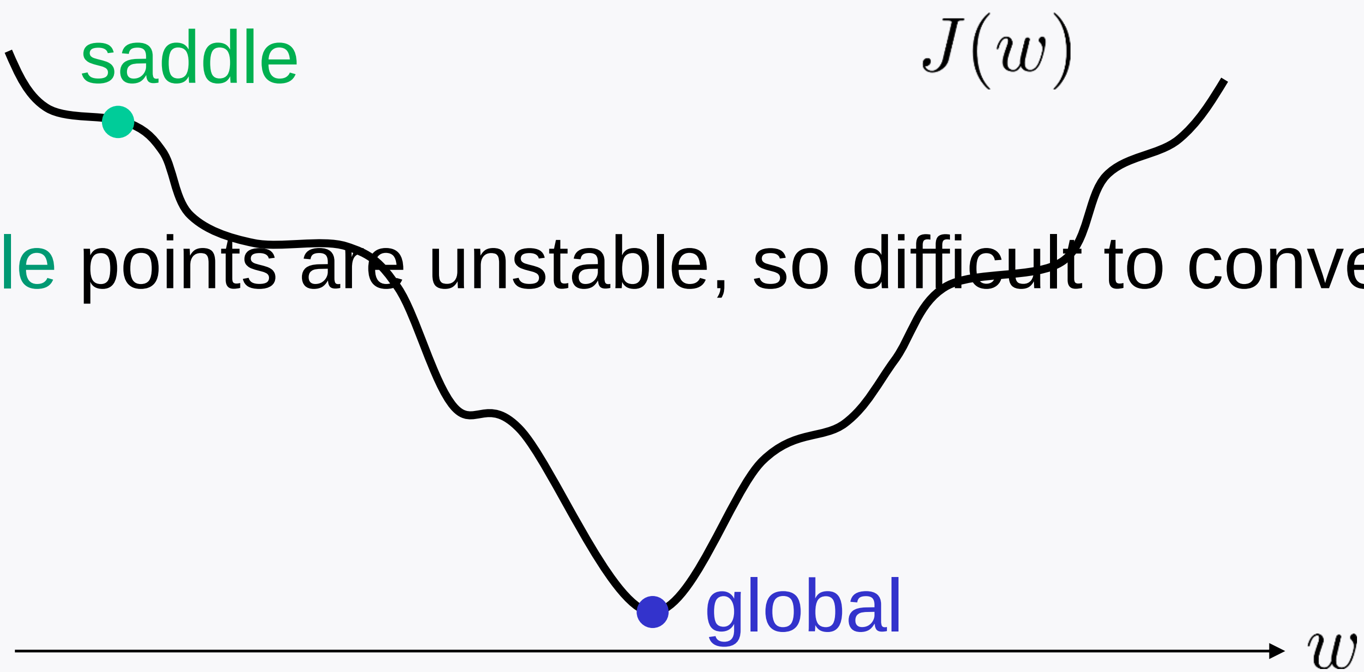
Experimental results revealed that in most cases:



How to handle such a non-convex problem?

Observation (by many practitioners):

Experimental results revealed that in most cases:



Saddle points are unstable, so difficult to converge to.

Suggests a good way

Saddle points are unstable, so difficult to converge to.

⇩

Find *any stationary point (gradient =0)* and then take it as a solution.

Hence, the algorithm is: gradient descent!

Look ahead

Turns out:

There is an efficient way for gradient descent in DNN.

Next lecture: Will explore the efficient way.