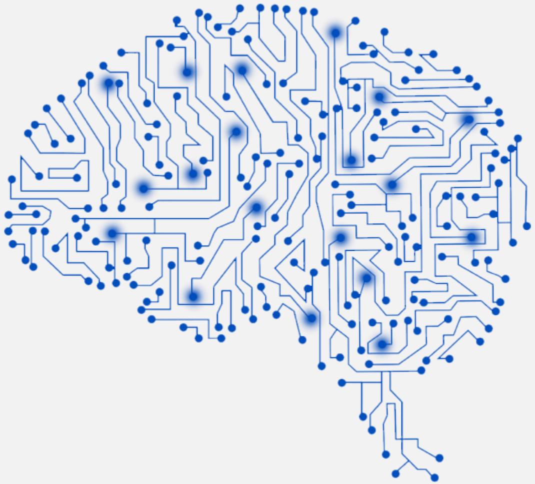


Deep Reinforcement Learning

Prof. Joongheon Kim
Korea University, Seoul, Korea
https://joongheon.github.io/
joongheon@korea.ac.kr

Lecture Roadmap

Introduction and Preliminaries

Deep Reinforcement Learning Theory

- **Introduction and Motivation**
- Deep Neural Network Summary
- Deep Q-Network (DQN)
- Performance Improvement on DQN

Deep Reinforcement Learning Implementation

Imitation Learning and Autonomous Driving

Deep Reinforcement Learning
DRL Theory

- **Introduction and Motivation**
- Deep Neural Network Summary
- Deep Q-Network (DQN)
- Performance Improvement on DQN

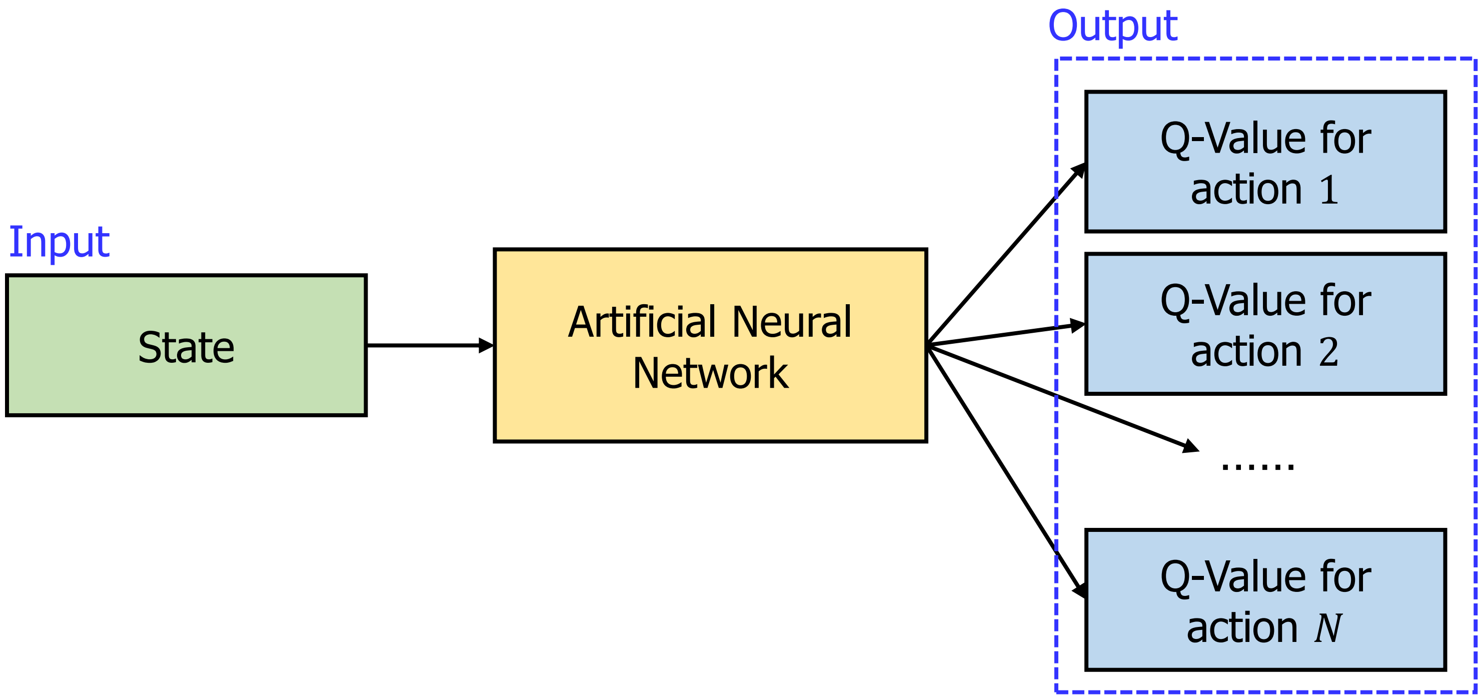
- Small-Scale Q-Values

$$Q = \begin{array}{cc} & \overbrace{\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}}^{\text{Action}} \\ \text{State}\left\{\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\right. & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{cc} & \overbrace{\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}}^{\text{Action}} \\ \text{State}\left\{\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\right. & \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

Q-table update example

- Large-Scale Q-Values
  - It is inefficient to make the Q-table for each state-action pair.
    → ANN is used to **approximate the Q-function**.



Lecture Roadmap

Introduction and Preliminaries

Deep Reinforcement Learning Theory

- Introduction and Motivation
- **Deep Neural Network Summary**
- Deep Q-Network (DQN)
- Performance Improvement on DQN

Deep Reinforcement Learning Implementation

DDPG-based Vehicular Caching

Imitation Learning and Autonomous Driving

Deep Reinforcement Learning

DRL Theory

- Introduction and Motivation
- **<u>Deep Neural Network Summary</u>**
- Deep Q-Network (DQN)
- Performance Improvement on DQN

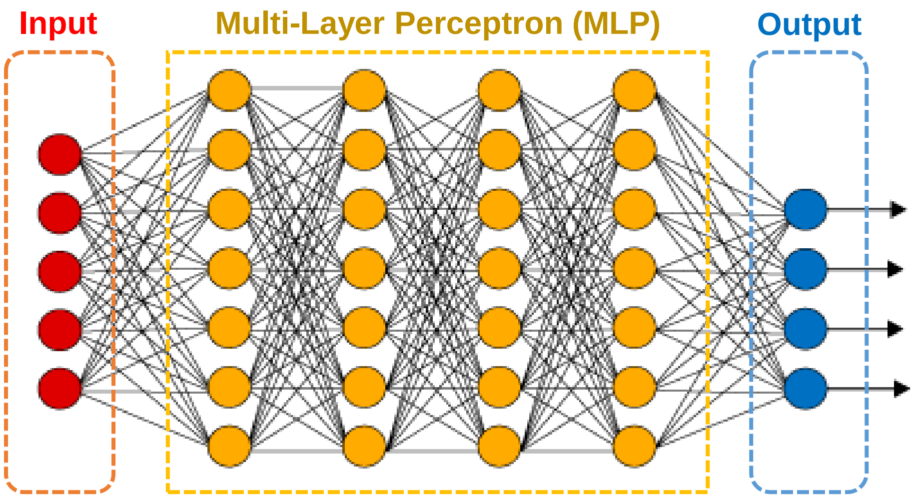
- **Introduction and Basics**
- Linear Classifiers
  - Linear Regression
  - Binary Classification (Logistic Regression)
  - Softmax Classification
- Artificial Neural Network (ANN)

- How Deep Learning Works?
  - Deep Learning Computation Procedure

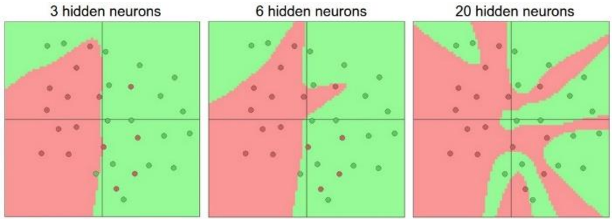
Deep Learning Model Setup
- MLP, CNN, RNN, GAN, or Customized
- # Hidden Layers, # Units, Input/Output, …
- Cost Function / Optimizer Selection

Training (with Large-Scale Dataset)
- Input: Data, Output: Labels
- Learning → Weights Updates for Cost Function Minimization

Inference / Testing (Real-Word Execution)
- Input: Real-World Input Data
- Output: Interference Results based on Updated Weights in Deep Neural Networks



Non-Linear Training (Weights Updates) for Cost Minimization: GD, SGD, Adam, etc.



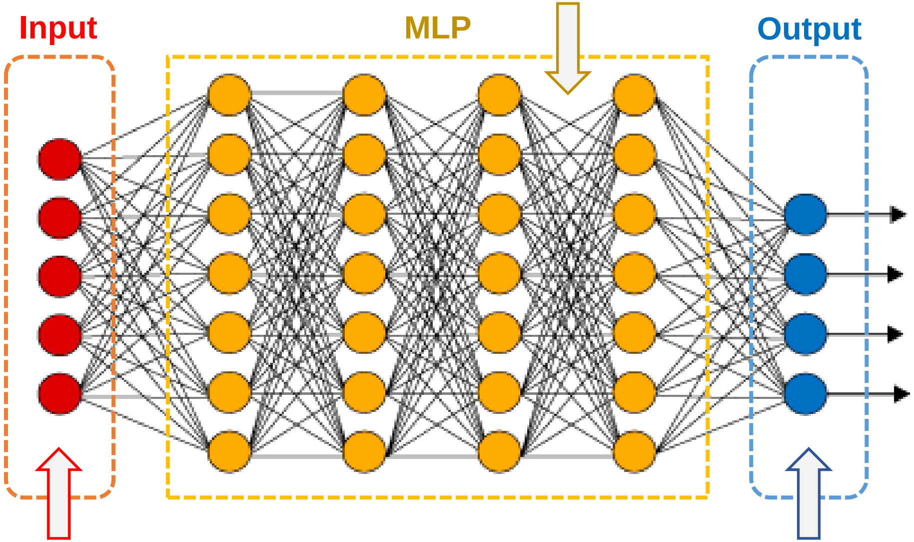
- How Deep Learning Works?
  - Deep Learning Computation Procedure

Deep Learning Model Setup
- MLP, CNN, RNN, GAN, or Customized
- # Hidden Layers, # Units, Input/Output, …
- Cost Function / Optimizer Selection

Training (with Large-Scale Dataset)
- Input: Data, Output: Labels
- Learning → Weights Updates for Cost Function Minimization

Inference / Testing (Real-Word Execution)
- Input: Real-World Input Data
- Output: Interference Results based on Updated Weights in Deep Neural Networks

All weights in units are trained/set (under cost minimization)



INPUT: Data
- One-Dimension Vector

OUTPUT: Labels
- One-Hot Encoding

We need a lot of training data for generality (otherwise, we will suffer from overfitting problem).

Introduction

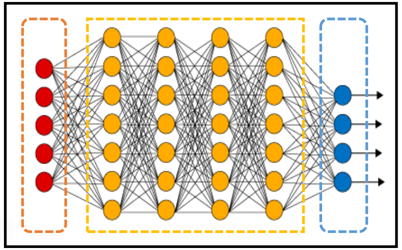
- How Deep Learning Works?
  - Deep Learning Computation Procedure

Deep Learning Model Setup
- MLP, CNN, RNN, GAN, or Customized
- # Hidden Layers, # Units, Input/Output, …
- Cost Function / Optimizer Selection

Training (with Large-Scale Dataset)
- Input: Data, Output: Labels
- Learning → Weights Updates for Cost Function Minimization

Inference / Testing (Real-Word Execution)
- Input: Real-World Input Data
- Output: Interference Results based on Updated Weights in Deep Neural Networks

Trained Model



Intelligent Surveillance Platforms

INPUT: Real-Time Arrivals

OUTPUT: Inference
- Computation Results based on (i) INPUT and (ii) trained weights in units (trained model).

- How Deep Learning Works?
  - Issue - **Overfitting**

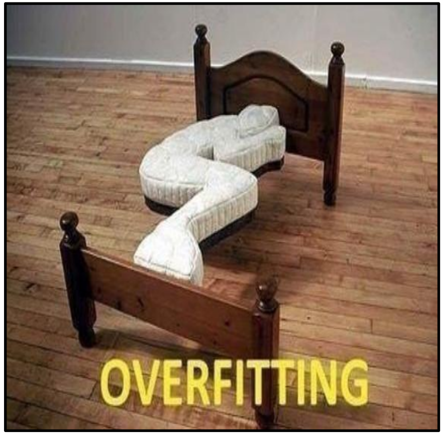
Deep Learning Model Setup
- MLP, CNN, RNN, GAN, or Customiz
- # Hidden Layers, # Units, Input/Output, …
- Cost Function / Optimizer Selection

Training (with Large-Scale Dataset)
- Input: Data, Output: Labels
- Learning → Weights Updates for Cost Function Minimization

Inference / Testing (Real-Word Execution)
- Input: Real-World Input Data
- Output: Interference Results based on Updated Weights in Deep Neural Networks

What if we do not have enough data for training (not enough to derive Gaussian/normal distribution)?

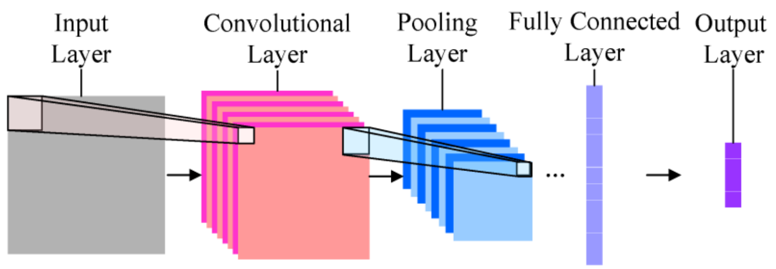
Situation becomes worse when the model (with insufficient training data) accurately fits on training data.



To Combat the Overfitting
- More training data
- Autoencoding (or variational auto-encoder (VAE))
- Droupout
- Regularization

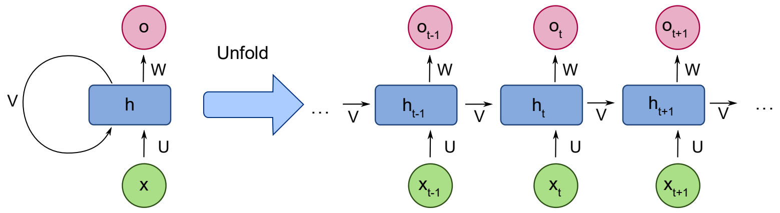
- Two Major Deep Learning Models → CNN vs. RNN

Convolutional Neural Network (CNN)



- In conventional neural network architectures, the input should be one-dimensional vector.
- In many applications, the input should be multi-dimensional (e.g., 2D for images). Thus, we need architectures in order to recognize the features in high-dimensional data.
- Mainly used for **visual information learning**

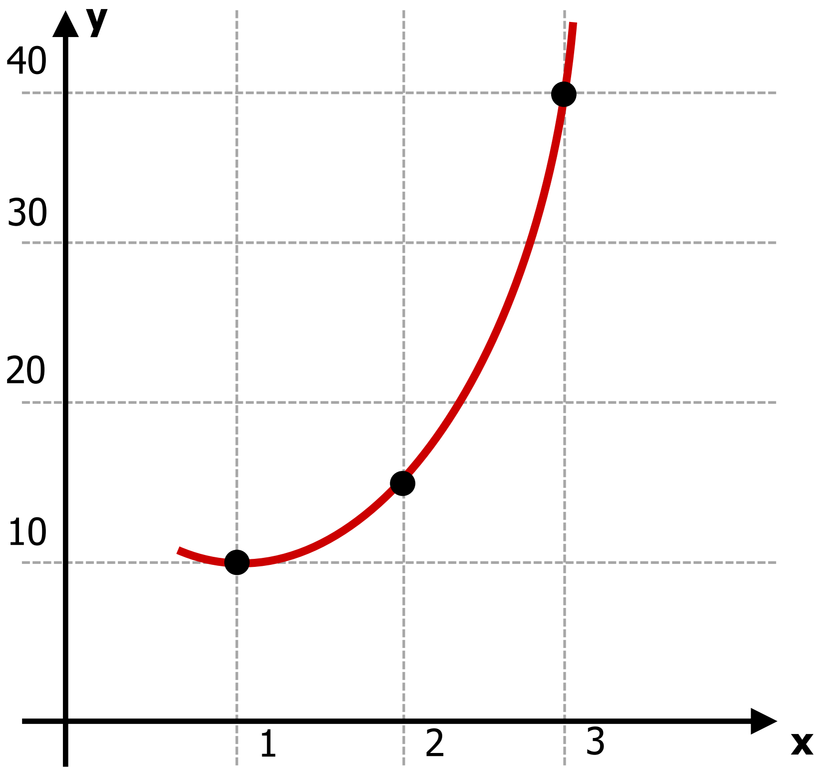
Recurrent Neural Network (RNN)



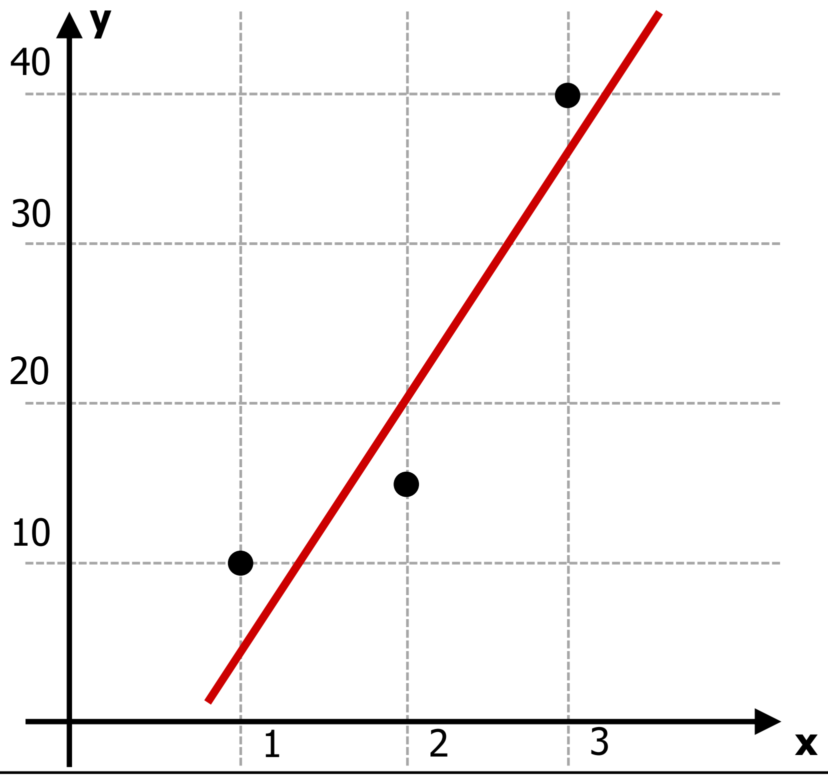
- In conventional neural network architectures, there is no way to introduce the concept of time.
- The time index can be represented as the chain of neural network models.
- The representative models are LSTM and GRU.
- Mainly used for **time-series information learning**

Interpolation vs. Linear Regression

Interpolation

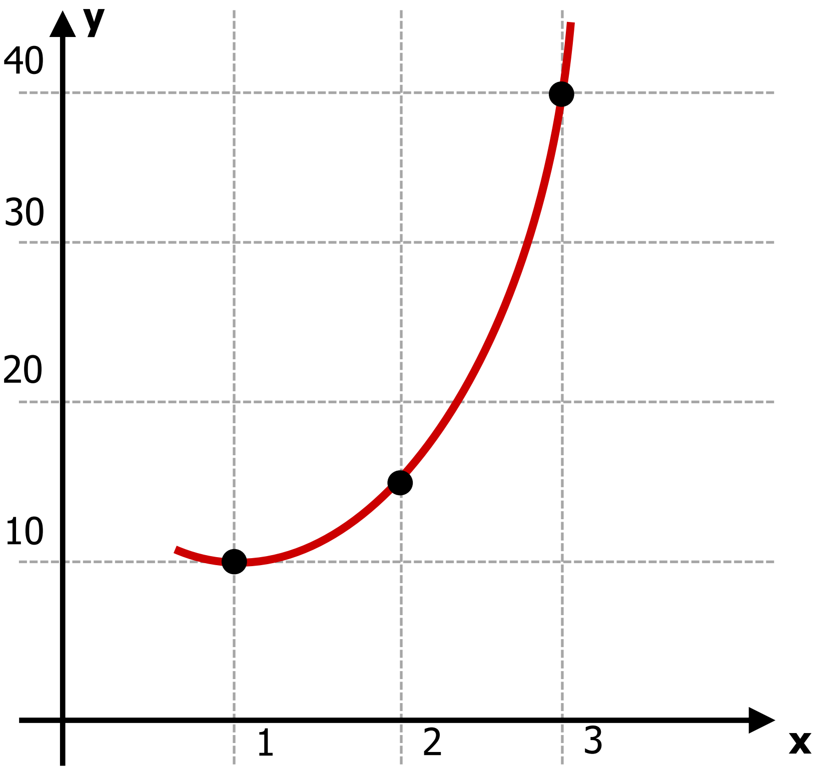


Linear Regression



Interpolation vs. Linear Regression

Interpolation



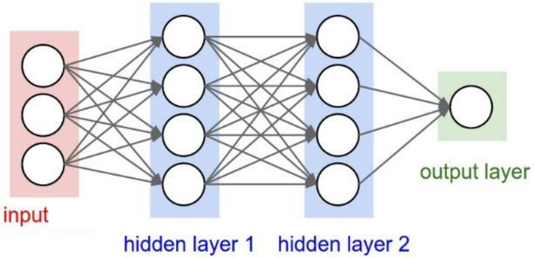
Interpolation with Polynomials

$$y = a_2 x^2 + a_1 x^1 + a_0$$

where three points are given.
→ Unique coefficients (a_0, a_1, a_2) can be calculated.

Is this related to
Neural Network Training?

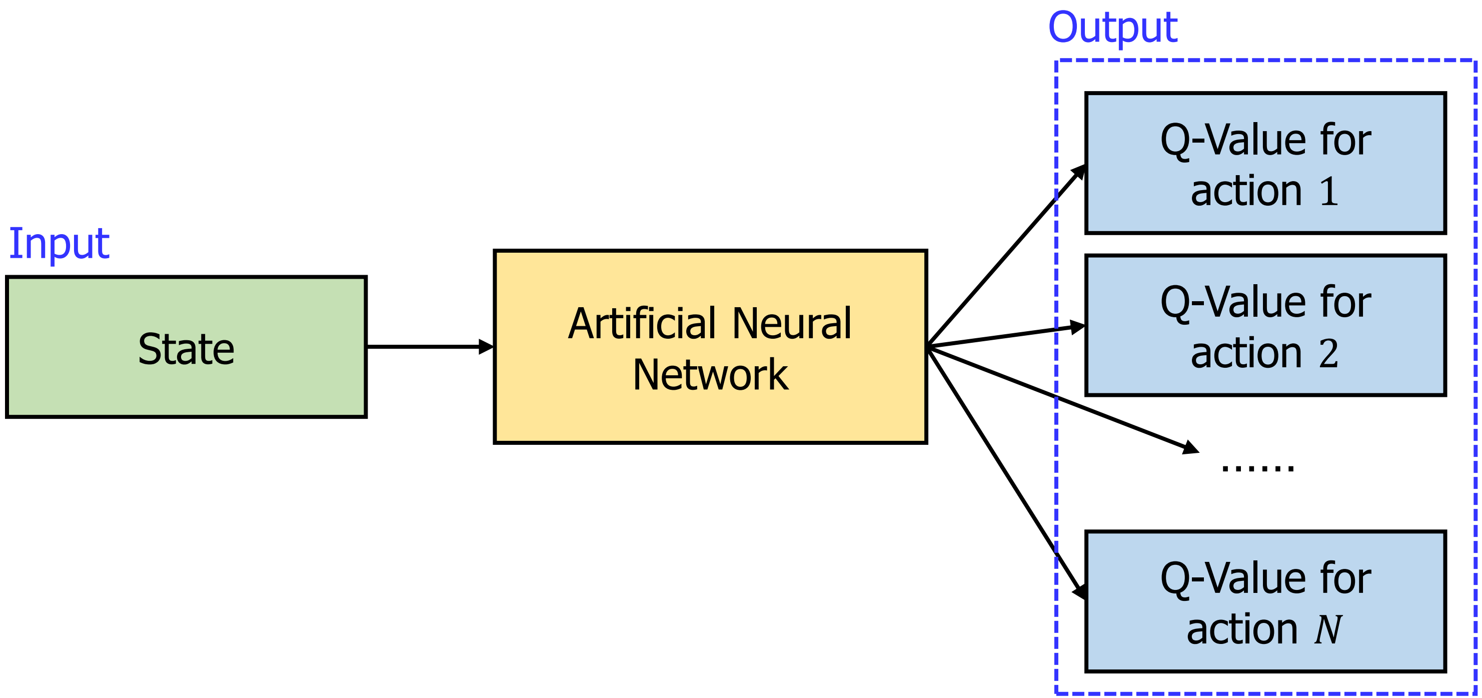
Interpolation and Neural Network Training



$$Y = a(a(a(X \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_o + b_o)$$

where training data/labels (X: data, Y: labels) are given.
→ Find $W_1, b_1, W_2, b_2, W_o, b_o$
→ This is the mathematical meaning of neural network training.
→ **Function Approximation**
→ The most well-known function approximation with neural network: **Deep Reinforcement Learning**

- It is inefficient to make the Q-table for each state-action pair.
  → ANN is used to **approximate the Q-function**.



Deep Reinforcement Learning
DRL Theory

- Introduction and Motivation
- Deep Neural Network Summary
- **Deep Q-Network (DQN)**
- Performance Improvement on DQN

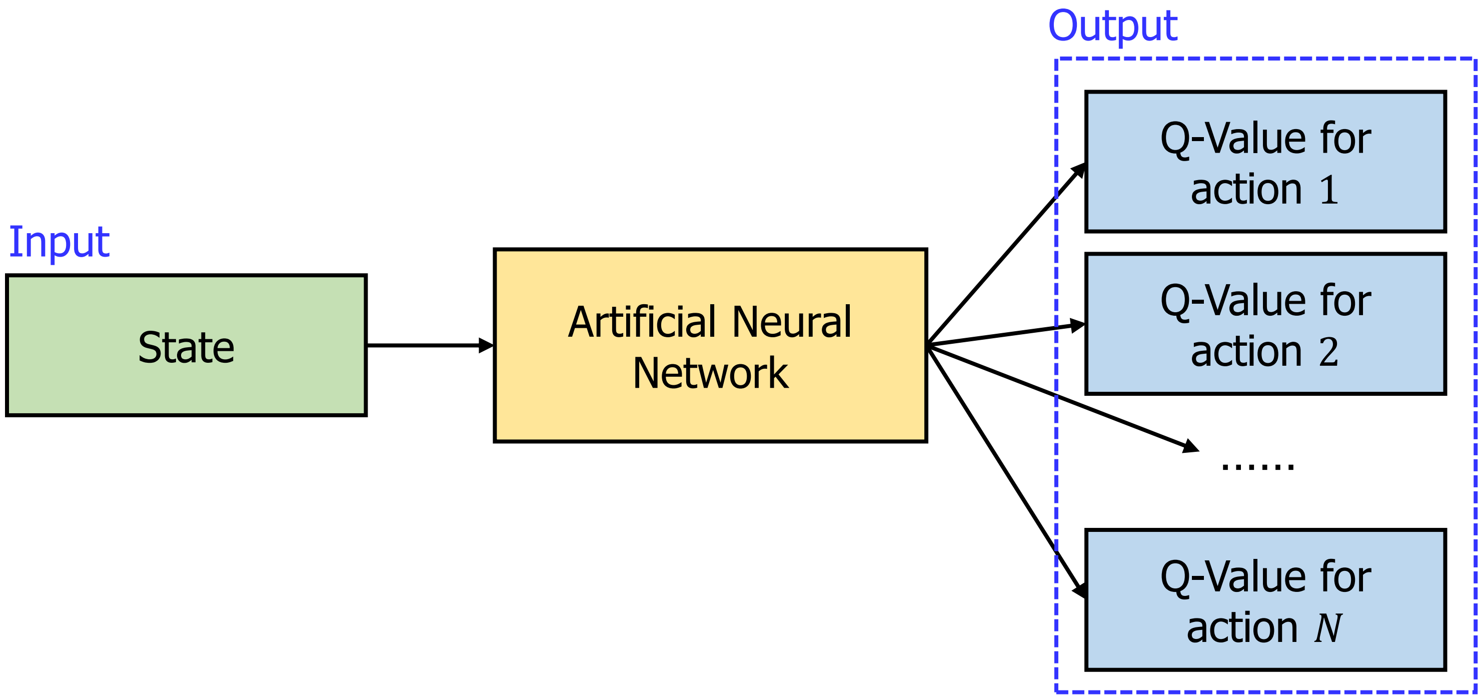
Q-Network

- Small-Scale Q-Values

$$Q = \begin{array}{c c} & \overbrace{\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}}^{\text{Action}} \\ \text{State} \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \right. & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \Rightarrow Q = \begin{array}{c c} & \overbrace{\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}}^{\text{Action}} \\ \text{State} \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \right. & \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

Q-table update example

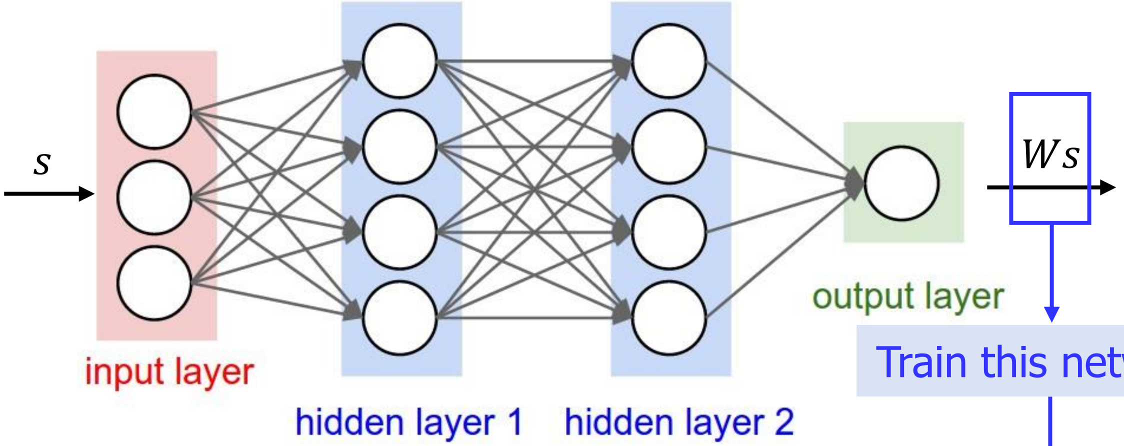
- Large-Scale Q-Values
  - It is inefficient to make the Q-table for each state-action pair.
    → ANN is used to **approximate the Q-function**.



- Q-Network Training (Linear Regression)

$$H(x) = Wx$$

$$\text{Cost}(W) = \frac{1}{m}\sum_{i=1}^{m}\left(Wx^i - y^i\right)^2$$



Train this network to approximate optimal Q, i.e., Q^*

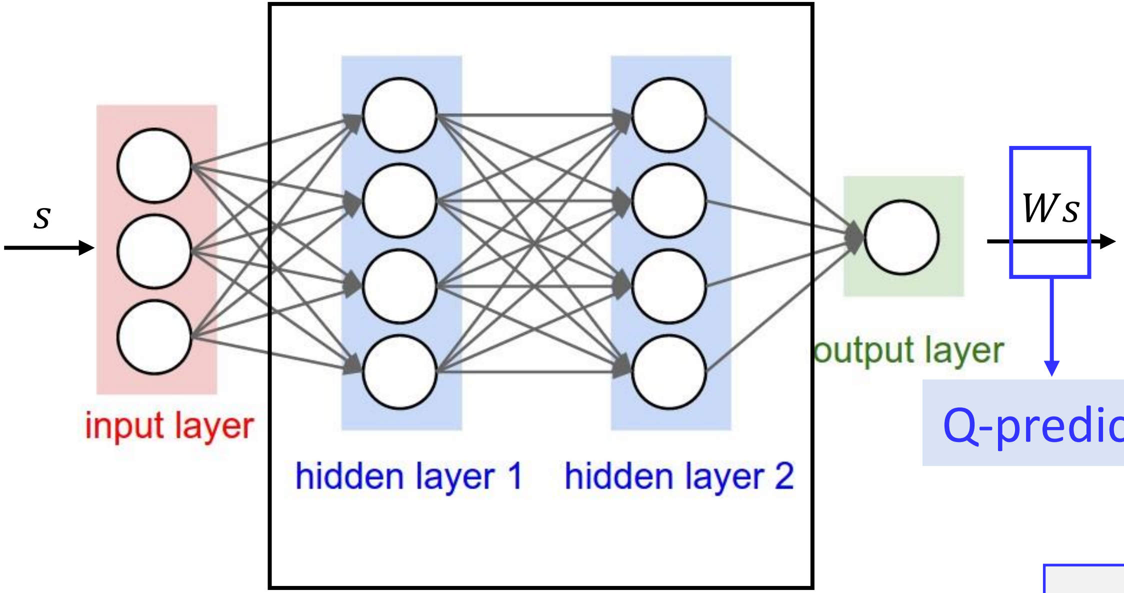
$$\boxed{Ws \approx Q^*}$$

- Q-Network Training (Linear Regression)

$$\text{Cost}(W) = (Ws - y)^2$$

$$y = r + \gamma \max Q(s')$$

Q^*



Q-predict: $\hat{Q}$

Weights, θ

Approximating Q^* using θ

$$Ws = \hat{Q}(s, a|\theta) \approx Q^*(s, a)$$

- Q-Network Training (Linear Regression)

$$\boxed{Ws = \hat{Q}(s,a|\theta) \approx Q^*(s,a)}$$

Approximating Q^* using θ

$$\min_{\theta} \sum_{t=0}^{T} \left[\underbrace{\hat{Q}(s_t, a_t|\theta)}_{\hat{Q}(s,a|\theta)} - \underbrace{\left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta) \right)}_{Q^*} \right]^2$$

Deep Reinforcement Learning
DRL Theory

- Introduction and Motivation
- Deep Neural Network Summary
- Deep Q-Network (DQN)
- **Performance Improvement on DQN**

Q-Network

Algorithm 1 Deep Q-learning

Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

If preprocessing is not needed, $\emptyset(s) = s$

ε-greedy

Learning

$$\text{Set } y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Play Atari with Deep Reinforcement Learning

Q-Network

y-label

$$\text{Set } \boxed{y_j} = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$$

Perform a gradient descent step on $\boxed{\left(y_j - Q(\phi_j, a_j; \theta)\right)^2}$ according to equation 3

Cost function

Target/Label $\rightarrow Q^*$

$Ws \rightarrow \hat{Q}(s, a|\theta)$

Deep Q-Network (DQN)

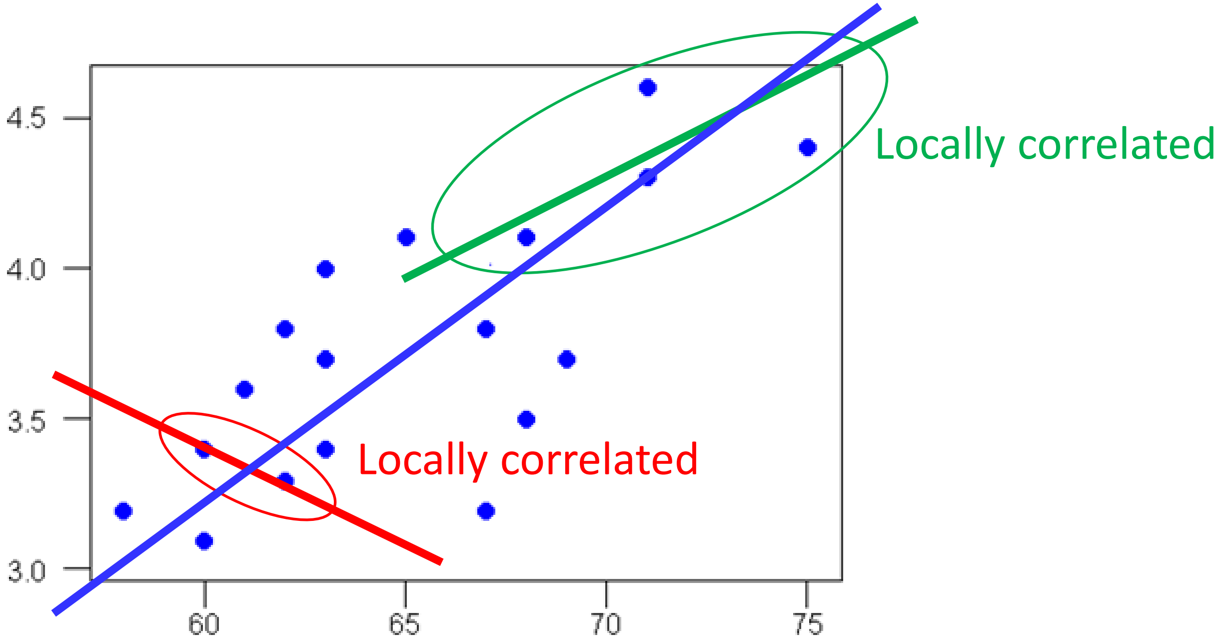
$$\min_{\theta} \sum_{t=0}^{T} \left[\hat{Q}(s_t, a_t|\theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta) \right) \right]^2$$

- Converges to Q^* using table lookup representation
- However, **diverges** using neural networks due to
  - Correlations between samples → [Issue #1]
  - Non-stationary targets → [Issue #2]

Tutorial by Google DeepMind: Deep Reinforcement Learning

Deep Q-Network (DQN)

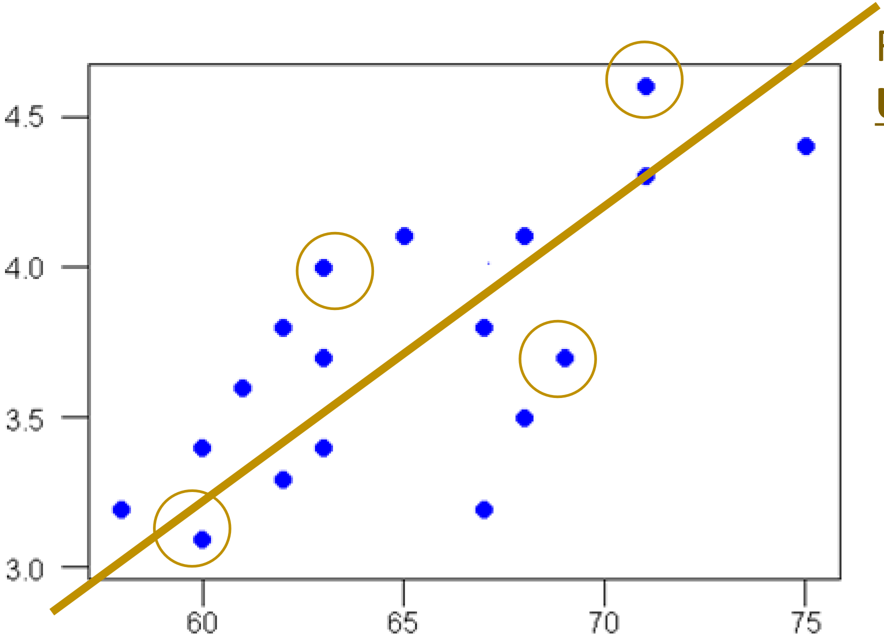
- [Issue #1] Correlations between Samples



- **Solution) Capture and Replay**
  - Store learning states in buffers → random sampling and learning

Deep Q-Network (DQN)

- [Issue #1] Correlations between Samples
  - **Capture and Replay →** Experience Replay
    - Store learning states in buffers → random sampling and learning



Random Sampling Results are **Uniformed Distributed**.

- [Issue #2] Non-Stationary Targets

Target

$$\min_{\theta} \sum_{t=0}^{T} \left[\hat{Q}(s_t, a_t|\theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta) \right) \right]^2$$

- Both sides uses same network θ.
  Thus, if our Q_predict is trained, our target is consequently updated.
  → **Non-stationary targets**.
- **Solution) Separate Networks** → create a target network

- [Issue #2] Non-Stationary Targets

Target

$$\min_{\theta} \sum_{t=0}^{T} \left[\hat{Q}(s_t, a_t | \theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \theta) \right) \right]^2$$

$$\min_{\theta} \sum_{t=0}^{T} \left[\hat{Q}(s_t, a_t | \theta) - \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a' | \bar{\theta}) \right) \right]^2$$

And periodic update!

References

- V. Mnih, *et. al.*, "**Playing Atari with Deep Reinforcement Learning**," NIPS Deep Learning Workshop (2013).
  - https://arxiv.org/abs/1312.5602
  - Citation: 2561+ (as of today)

- V. Mnih, *et. al.*, "**Human-Level Control through Deep Reinforcement Learning**," Nature (2015).
  - https://www.nature.com/articles/nature14236
  - Citation: 6066+ (as of today)