Machine learning 10

Byung Chang Chung

Gyeongsang National University

bcchung@gnu.ac.kr

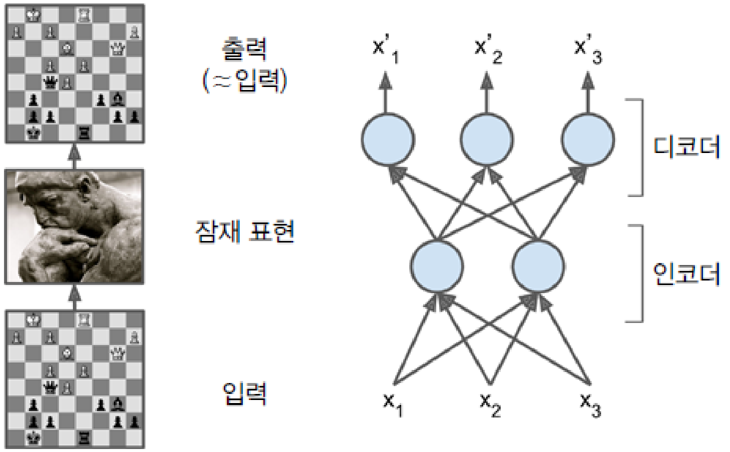
Gyeongsang
National
University
경상국립대학교

Contents

- Autoencoder and reinforcement learning
  - basic structure of autoencoder
  - types of autoencoder
  - applications of autoencoder
  - basic theory of reinforcement learning
  - applications of reinforcement learning

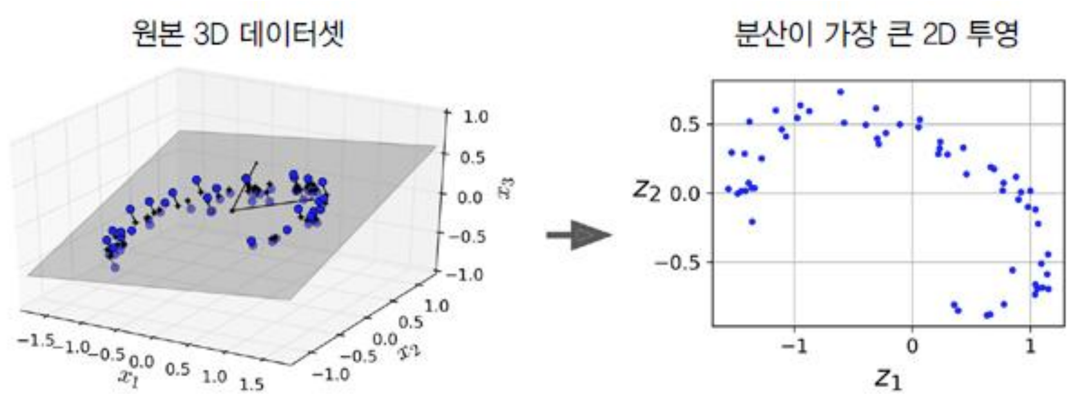
What is autoencoder

- Encoder and decoder
  - the process of encoding converts information from a source into symbols for communication or storage
  - decoding is the reverse process, converting code symbols back into a form that the recipient understands



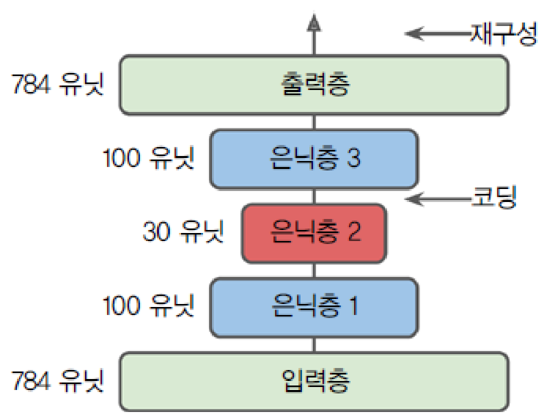
What is autoencoder

- Autoencoder in linear process
  - like PCA
  - encoded results can be viewed similar with the results of dimension reduction



What is autoencoder

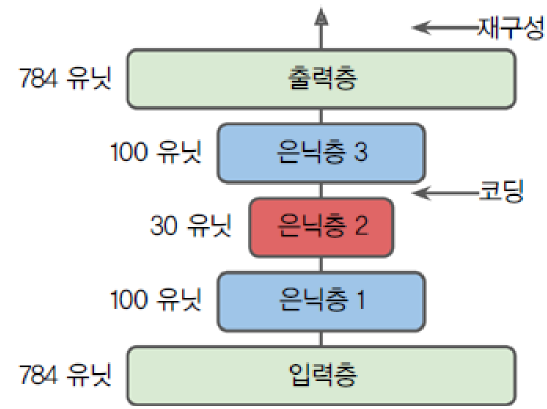
- Stacked encoder
  - an autoencoder which has multiple hidden layers
  - compare inputs and outputs to ensure that the autoencoder is properly trained



Gyeongsang National University 경상국립대학교

Applications of autoencoder

- Fashion MNIST dataset
  - usage of stacked autoencoder
  - structure is the same as previous image




```
stacked_encoder = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(100, activation="selu"),
    keras.layers.Dense(30, activation="selu"),
])
stacked_decoder = keras.models.Sequential([
    keras.layers.Dense(100, activation="selu", input_shape=[30]),
    keras.layers.Dense(28 * 28, activation="sigmoid"),
    keras.layers.Reshape([28, 28])
])
stacked_ae = keras.models.Sequential([stacked_encoder, stacked_decoder])
stacked_ae.compile(loss="binary_crossentropy",
                   optimizer=keras.optimizers.SGD(learning_rate=1.5), metrics=[rounded_accuracy])
history = stacked_ae.fit(X_train, X_train, epochs=20,
                         validation_data=(X_valid, X_valid))
```

GNU Gyeongsang National University 경상국립대학교

Applications of autoencoder

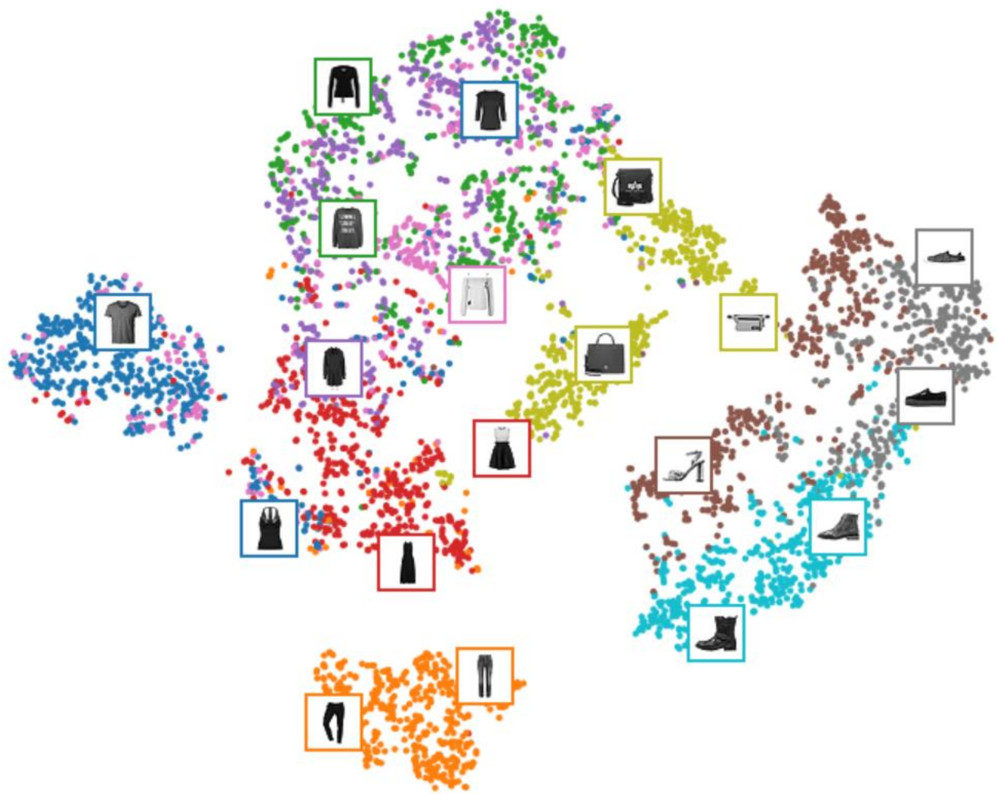
- Fashion MNIST dataset
  - by comparing input and output, the training can be validated

Applications of autoencoder

- Fashion MNIST dataset
  - dimension reduction
    - middle hidden layer has 30 dimensions
    - after 30-D, use t-SNE to reduce the dimension to 2

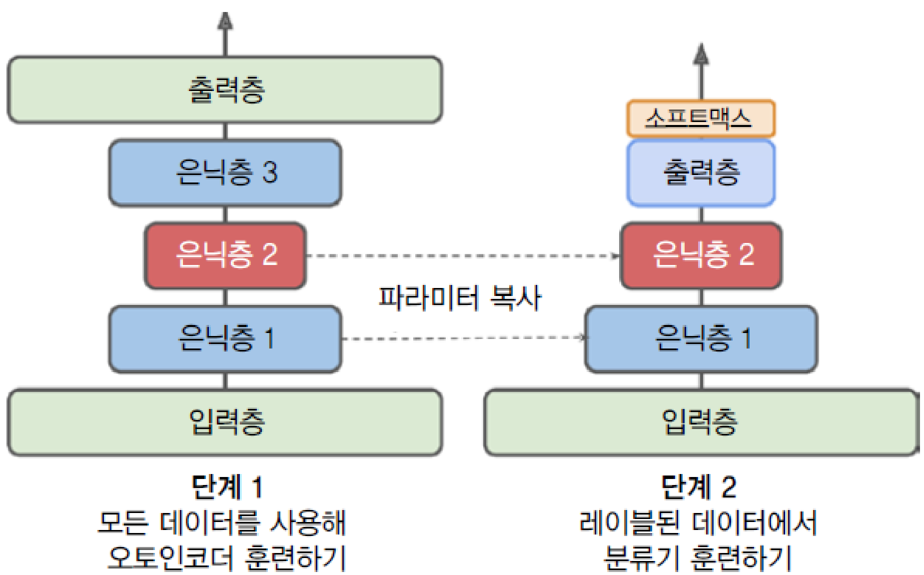
Applications of autoencoder

- Fashion MNIST dataset
  - result of dimension reduction



Applications of autoencoder

- Fashion MNIST dataset
    - unsupervised pre-training



Applications of autoencoder

- Fashion MNIST dataset
  - tied weights

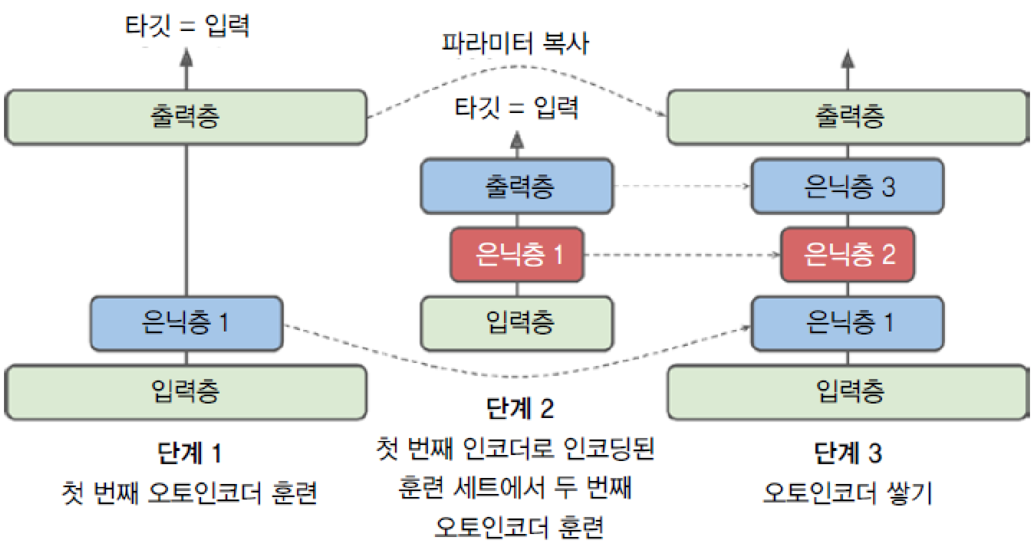
```python
class DenseTranspose(keras.layers.Layer):
    def __init__(self, dense, activation=None, **kwargs):
        self.dense = dense
        self.activation = keras.activations.get(activation)
        super().__init__(**kwargs)
    def build(self, batch_input_shape):
        self.biases = self.add_weight(name="bias",
                                      shape=[self.dense.input_shape[-1]],
                                      initializer="zeros")
        super().build(batch_input_shape)
    def call(self, inputs):
        z = tf.matmul(inputs, self.dense.weights[0], transpose_b=True)
        return self.activation(z + self.biases)
```

```python
dense_1 = keras.layers.Dense(100, activation="selu")
dense_2 = keras.layers.Dense(30, activation="selu")

tied_encoder = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    dense_1,
    dense_2
])

tied_decoder = keras.models.Sequential([
    DenseTranspose(dense_2, activation="selu"),
    DenseTranspose(dense_1, activation="sigmoid"),
    keras.layers.Reshape([28, 28])
])
```

Gyeongsang National University 경상국립대학교

Applications of autoencoder

- Fashion MNIST dataset
  - greedy layerwise training



Convolutional autoencoder

- When you treat the image dataset
  - convolutional network is better than dense network
  - the same idea applies to autoencoders

```
conv_encoder = keras.models.Sequential([
    keras.layers.Reshape([28, 28, 1], input_shape=[28, 28]),
    keras.layers.Conv2D(16, kernel_size=3, padding="SAME", activation="selu"),
    keras.layers.MaxPool2D(pool_size=2),
    keras.layers.Conv2D(32, kernel_size=3, padding="SAME", activation="selu"),
    keras.layers.MaxPool2D(pool_size=2),
    keras.layers.Conv2D(64, kernel_size=3, padding="SAME", activation="selu"),
    keras.layers.MaxPool2D(pool_size=2)
])
conv_decoder = keras.models.Sequential([
    keras.layers.Conv2DTranspose(32, kernel_size=3, strides=2, padding="VALID", activation="selu",
                                 input_shape=[3, 3, 64]),
    keras.layers.Conv2DTranspose(16, kernel_size=3, strides=2, padding="SAME", activation="selu"),
    keras.layers.Conv2DTranspose(1, kernel_size=3, strides=2, padding="SAME", activation="sigmoid"),
    keras.layers.Reshape([28, 28])
])
conv_ae = keras.models.Sequential([conv_encoder, conv_decoder])
```

Recurrent autoencoder

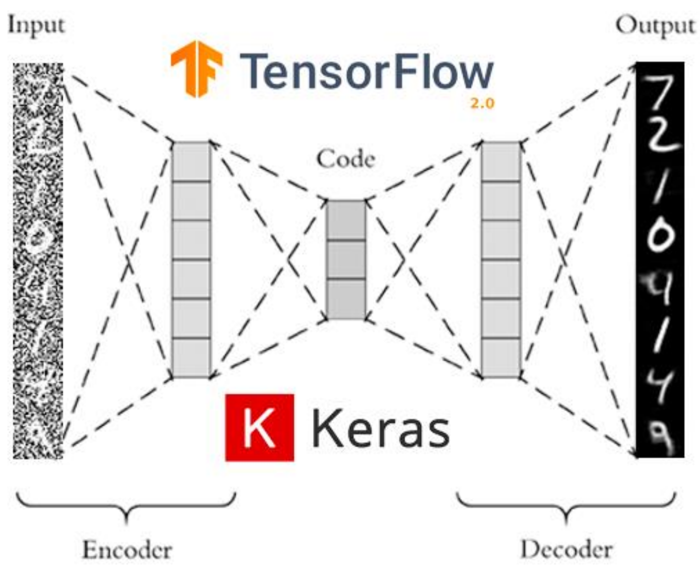
- When you treat the time-series dataset
  - recurrent network is better than dense network
  - the same idea applies to autoencoders

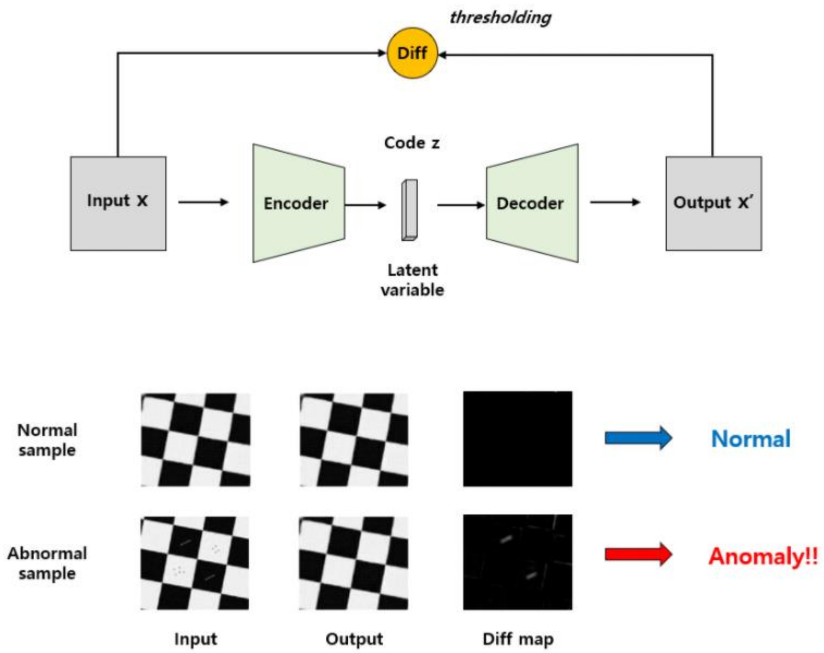
```
recurrent_encoder = keras.models.Sequential([
    keras.layers.LSTM(100, return_sequences=True, input_shape=[28, 28]),
    keras.layers.LSTM(30)
])
recurrent_decoder = keras.models.Sequential([
    keras.layers.RepeatVector(28, input_shape=[30]),
    keras.layers.LSTM(100, return_sequences=True),
    keras.layers.TimeDistributed(keras.layers.Dense(28, activation="sigmoid"))
])
recurrent_ae = keras.models.Sequential([recurrent_encoder, recurrent_decoder])
recurrent_ae.compile(loss="binary_crossentropy", optimizer=keras.optimizers.SGD(0.1),
                     metrics=[rounded_accuracy])
```

GNU Gyeongsang National University 경상국립대학교

Applications of autoencoder

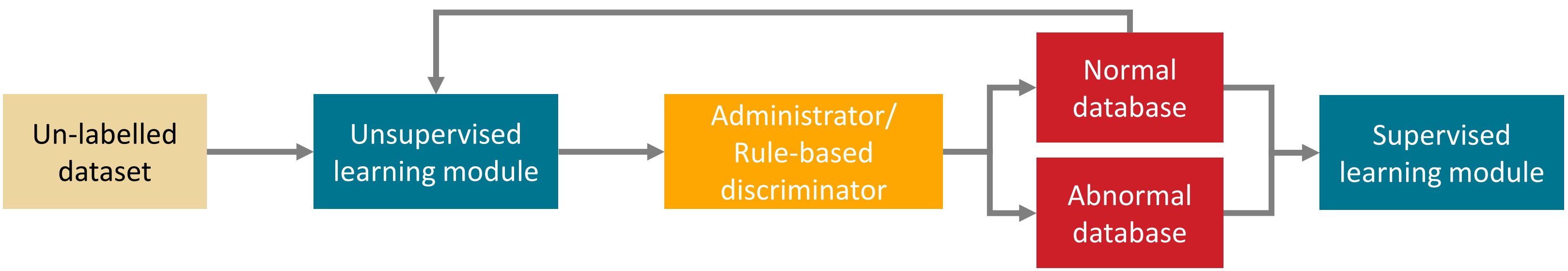
- Anomaly detection and denoising

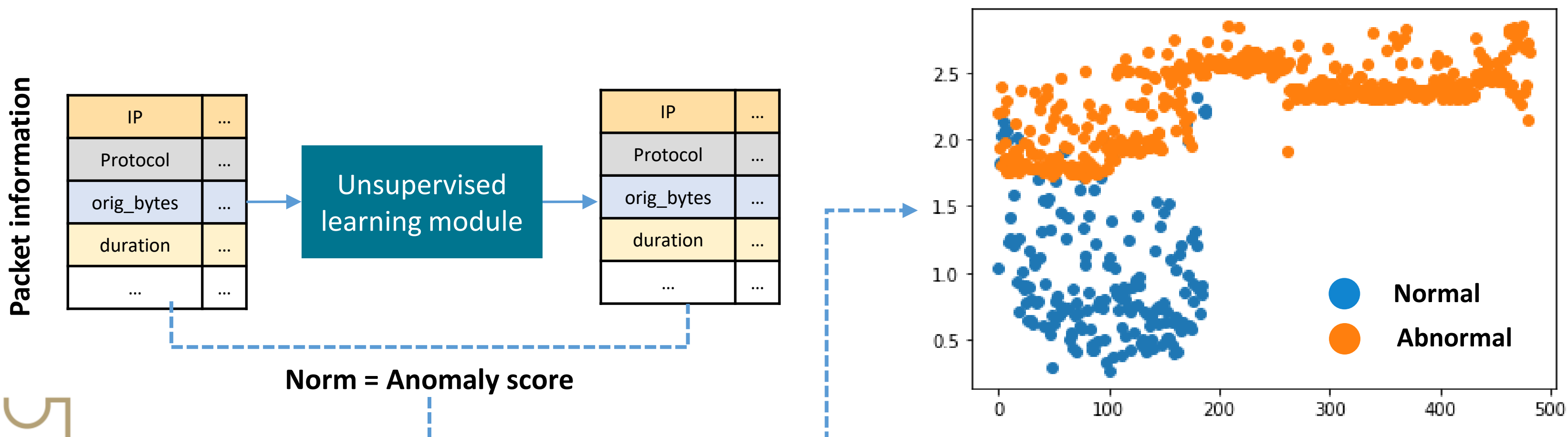




Applications of autoencoder

- Anomaly detection and denoising



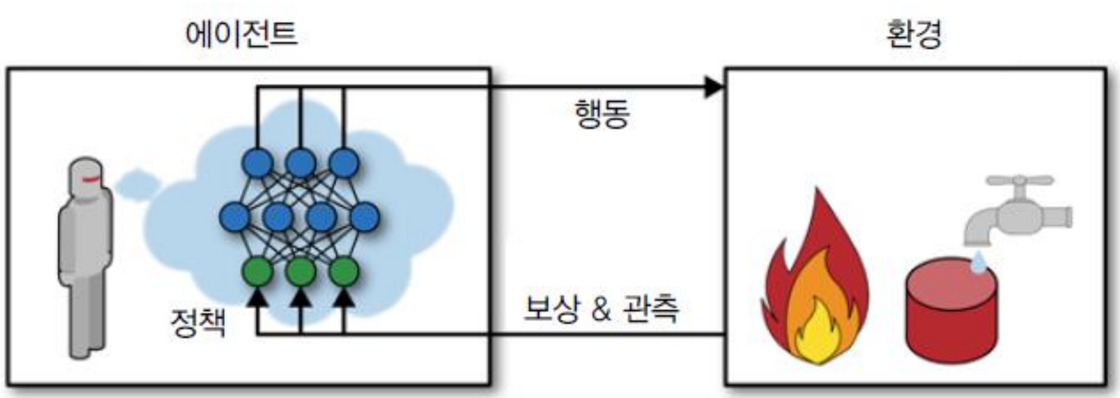


Reinforcement learning

- Definition
  - reinforcement learning is a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones

Reinforcement learning

- Basic structure
  - agent – environment
  - action – reward
  - policy

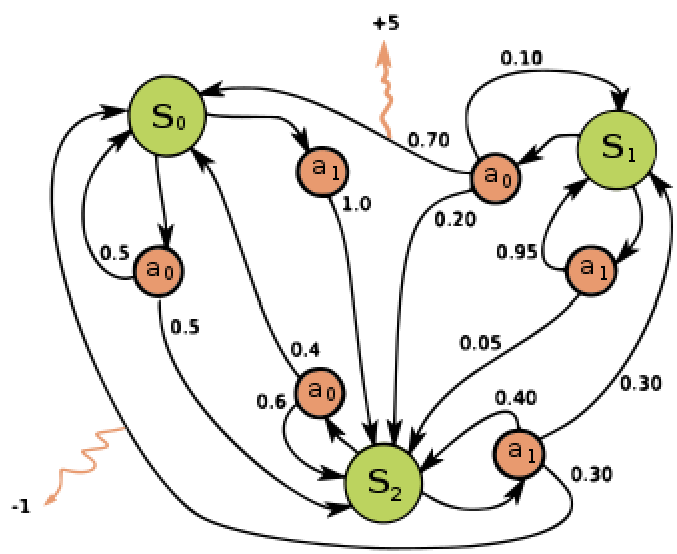


Reinforcement learning

- Sample example
  - https://youtu.be/Yr_nRnqeDp0
  - genetic algorithm is similar with reinforcement learning but they are not identical algorithm

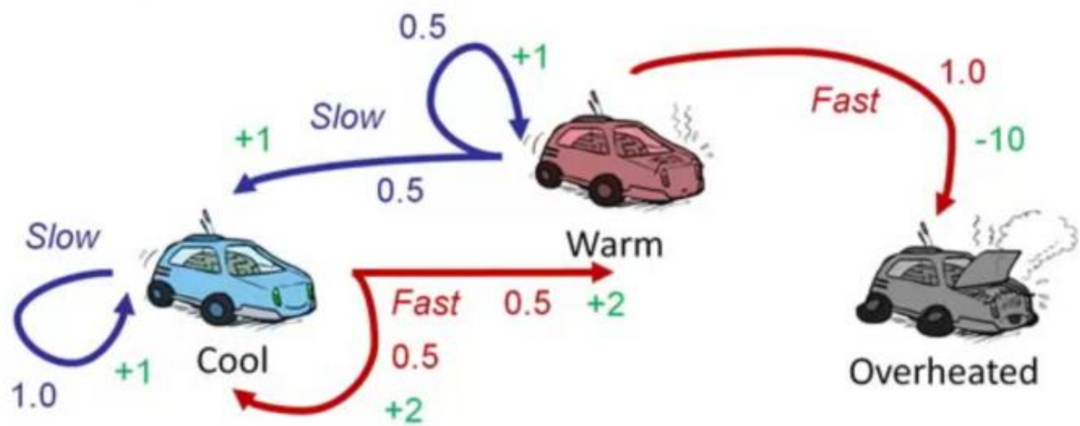
Reinforcement learning

- Markov decision process (MDP)
  - provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker



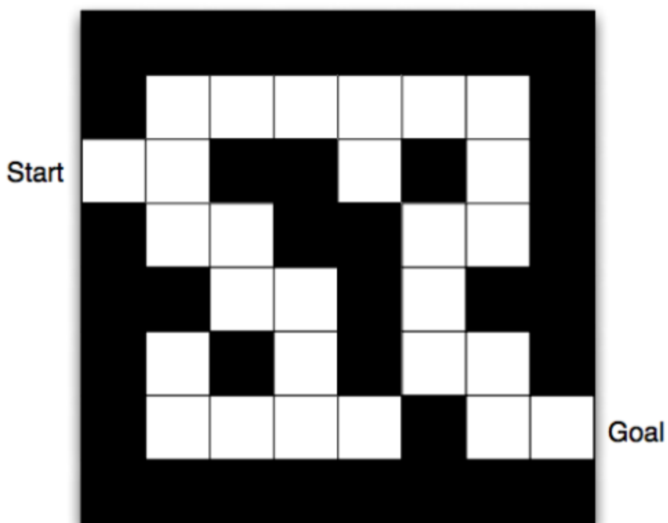
Reinforcement learning

- MDP example
  - car acceleration



Reinforcement learning

- State-reward example
  - maze example

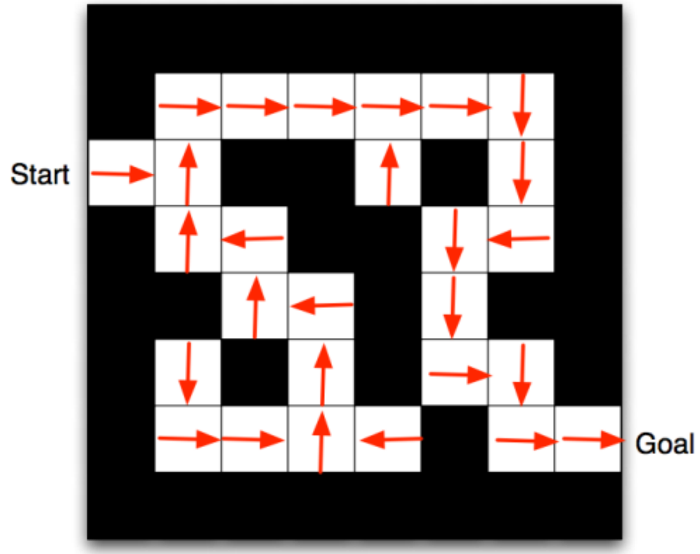


- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec21_22_handout.pdf

Reinforcement learning

- State-reward example
  - maze example

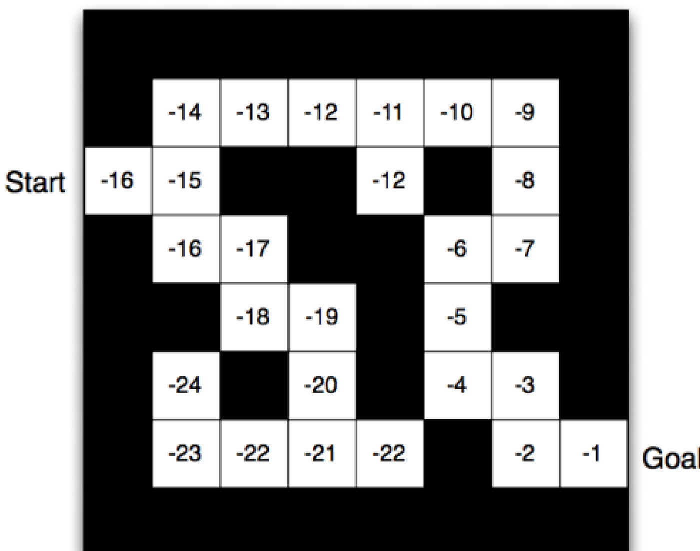


- Arrows represent policy $\pi(s)$ for each state s

https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec21_22_handout.pdf

Reinforcement learning

- State-reward example
  - maze example

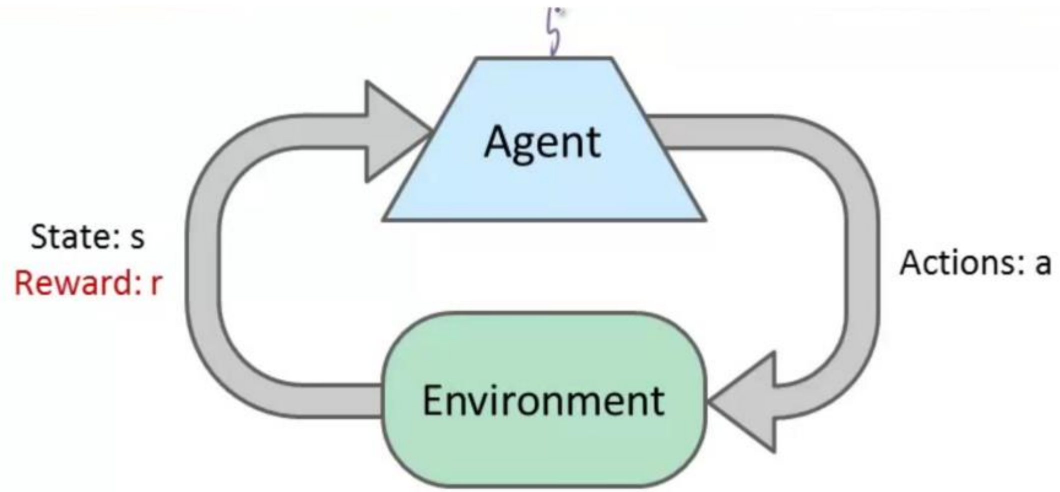


- Numbers represent value $V^{\pi}(s)$ of each state s

https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec21_22_handout.pdf

Reinforcement learning

- Q learning
  - q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state



Gyeongsang National University 경상국립대학교

https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec21_22_handout.pdf

Reinforcement learning

- Q learning
  - formulation
    - action a_t
    - reward r_t
    - state s_t
    - policy $\pi(a|s\} = \Pr[a_t = a|s_t = s]$

Gyeongsang National University 경상국립대학교

Reinforcement learning

- Q learning
  - optimization for
    - time-series reward $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$
    - taking expectation

$$V^\pi(s) = \underset{a_t, a_{t+i}, s_{t+i}}{\mathbb{E}} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t = s \right]$$

    - by choosing an action

$$Q^\pi(s, a) = \underset{a_{t+i}, s_{t+i}}{\mathbb{E}} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t = s, a_t = a \right]$$

Gyeongsang National University 경상국립대학교

Reinforcement learning

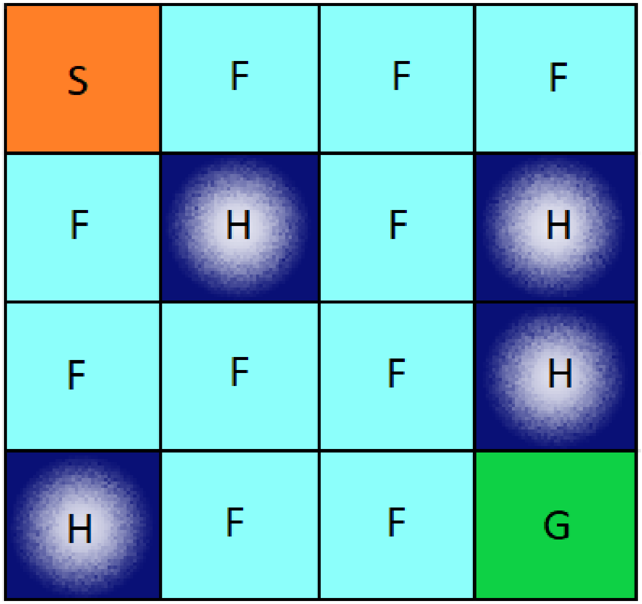
- Q learning
  - how to optimize?

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} | s_t = s, a_t = a\right] + \gamma \mathop{\mathbb{E}}_{s_{t+1}} \left[\max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\right]$$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
        $S \leftarrow S'$;
    until S is terminal

Reinforcement learning

- Q learning
  - frozen lake example



Reinforcement learning

- Q learning
  - atari game

Reinforcement learning

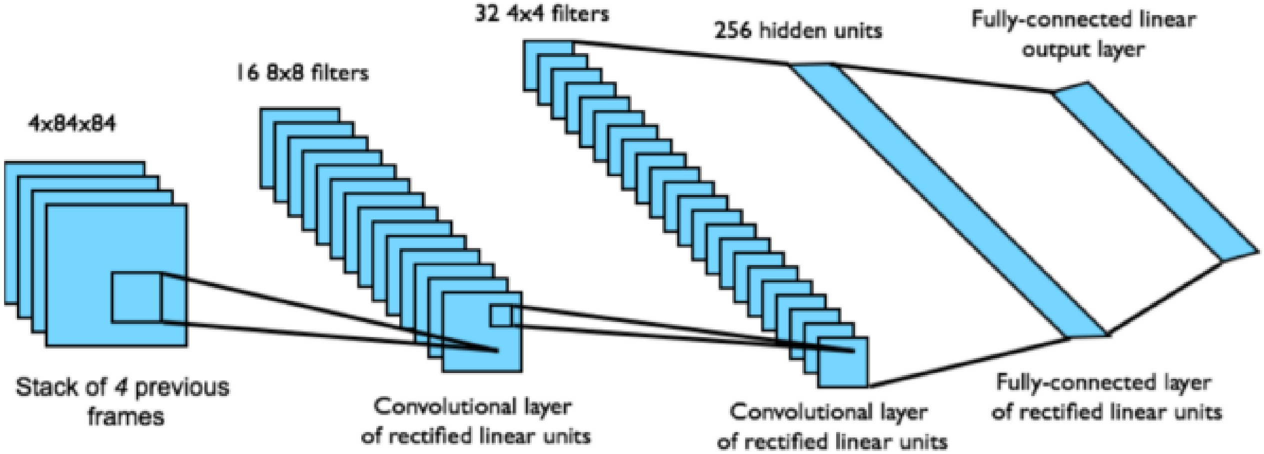
- Deep Q learning
  - expect Q value through neural network

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

Gyeongsang
National
University
경상국립대학교

Reinforcement learning

- Deep Q learning
  - expect Q value through neural network



Gyeongsang National University 경상국립대학교

Feel free to question

Through e-mail & LMS

본 자료의 연습문제는 수업의 본교재인
한빛미디어, Hands on Machine Learning(2판)에서 주로 발췌함

Gyeongsang National University
경상국립대학교