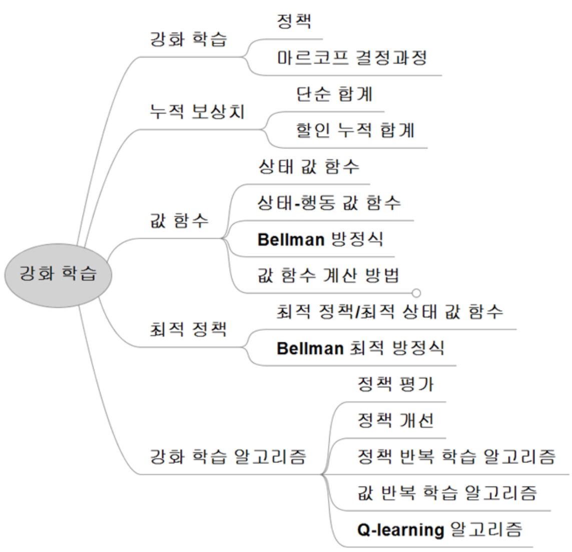
기계 학습

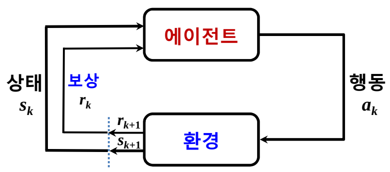
Part V. 강화 학습

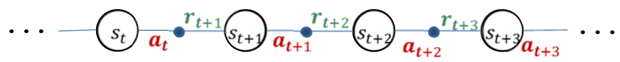
충북대학교 소프트웨어학과
이건명



11.1 강화 학습

❖ **강화 학습**(reinforcement learning)
- **어떤 모르는 환경**에서 동작하는 **에이전트**가 있을 때, 에이전트가 **현재 상태(state)**에서 향후 기대되는 **누적 보상값(reward)**이 **최대**가 되도록 **행동(action)**을 선택하는 **정책(policy)**을 찾는 것

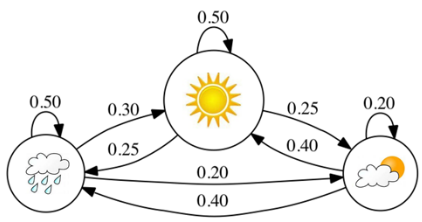




- **강화학습 문제 표현**
  - **마르코프 결정 과정(MDP)** 사용 표현
    - 불확실성 반영

강화 학습

❖ **마르코프 결정 과정(Markov Decision Process, MDP)**
- **상태 전이**(state transition)가 **현재 상태** S_t와 **입력** (또는 **행동**) A_t에 의해서 **확률적**으로 결정되는 **마르코프 모델**(Markov model)
  - **마르코프 모델**
    - **미래의 상태** S_{t+1}는 **현재 상태** S_t에 영향을 받고 과거 상태 $S_{t-1}, S_{t-2}, \ldots$에는 영향을 받지 않는 시스템에 대한 **확률 모델** (stochastic model)
    - $P(S_{t+1} | S_t, S_{t-1}, \ldots, S_0) = P(S_{t+1} | S_t)$



  - **마르코프 결정과정**
    - $P(S_{t+1} | S_t, S_{t-1}, \ldots, S_0, A_t) = P(S_{t+1} | S_t, A_t)$

강화 학습

❖ **마르코프 결정 과정(Markov Decision Process, MDP)** – cont.

- **상태**의 집합 $S = \{s_1, s_2, \ldots, s_N\}$
- **행동**의 집합 $A = \{a_1, a_2, \ldots, a_M\}$
- **상태 전이(state transition)** 결정 확률분포
  - t 시점의 상태 S_t에서 행동 A_t를 취할 때 도달하는 다음 상태 S_{t+1}를 결정하는 것 (s_t, a_t, s_{t+1})
    - $P(S_{t+1} = s' | S_t = s, A_t = a) = T(s, a, s')$
- 상태 전이가 일어날 때 **즉시 보상값(immediate reward)**
  - 상태 전이 (s_t, a_t, s_{t+1})에서 받는 즉시 보상값 r_{t+1}
    - $R(s_t, a_t, s_{t+1}) = R(s_{t+1}) = r_{t+1}$

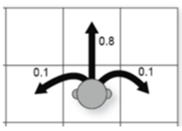
❖ **강화학습의 목적**

- **기대 누적 보상값**(expected accumulated reward)이 최대가 되도록 하는 **정책(policy)**을 찾는 것
  - 정책 : 각 상태에서 선택할 행동 지정

강화 학습

❖ **예. 강화 학습 문제**

- $S = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3)$
  $(3,1), (3,2), (3,3), (4,1), (4,2), (4,3)\}$
- $A = \{east, west, south, north\}$
- **상태 전이확률**



$T\big((3,1), north, (3,2)\big) = 0.8$

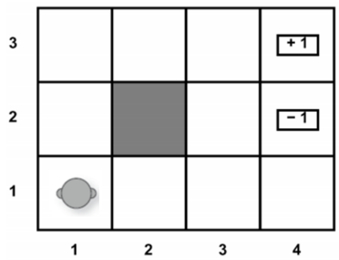
$T\big((3,1), north, (2,1)\big) = 0.1$

$T\big((3,1), north, (4,1)\big) = 0.1$

- **보상**(Reward)

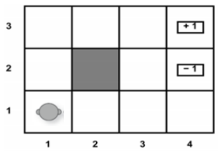
$R((4,3)) = +1,\ R\big((4,2)\big) = -1$

$R\big((x, y)\big) = c \quad (x, y) \neq (4,1)\ or\ (4,3)$



강화 학습

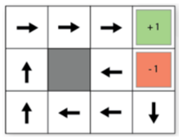
❖ 예. 강화 학습 문제



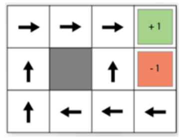
- **정책**(policy)
  - 각 상태 s에서 취할 행동 a을 결정해 둔 것

$$R((4,3)) = +1,\ R((4,2)) = -1,\ R((x,y)) = c \quad (x,y) \neq (4,1)\ or\ (4,3)$$

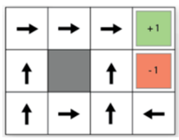
$c = -0.01$



$c = -0.04$



$c = -0.09$



$c = -2.00$

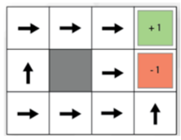


image : Richard S. Sutton

11.2 누적 보상치

❖ 누적 보상치의 계산 방법

- **단순 합계**
  - $V(s_0, s_1, \ldots) = r(s_0) + r(s_1) + r(s_2) + \ldots$
  - 연속해서 보상치가 더해지면 지속적으로 커질 수 있음

- **할인 누적 합계(sum of discounted reward)**
  - $V(s_0, s_1, \ldots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots$
    - 할인율 (discount factor) γ : $0 < \gamma < 1$
    - 가까운 보상이 먼 미래의 보상보다 가치가 있음

- 행동에 대한 **보상**이 **지연**(delay)되어 주어지는 경우
  - 많은 경우 언제 보상이 주어지는지 알 수 없음

11.3 값 함수

❖ **값 함수 (value function)**

- **상태 값 함수(state value function)** $V^{\pi}(s)$
  - **상태 s**에서 시작하여 **정책 π**에 따라 행동을 할 때 얻게 되는 **기대 보상**(expected reward)

$$V^{\pi}(s) = \mathrm{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s, \pi]$$
$$= \mathrm{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi]$$

- **상태-행동 값 함수(state-action value function)** $Q^{\pi}(s,a)$
  - **상태 s**에서 **행동 a**를 한 다음, **정책 π**에 따라 행동을 할 때 얻게 되는 **기대 보상**

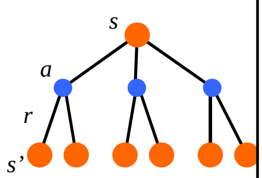
$$Q^{\pi}(s,a) = \mathrm{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s, a_t = a, \pi]$$
$$= \mathrm{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a, \pi]]$$

값 함수

❖ **Bellman 방정식**

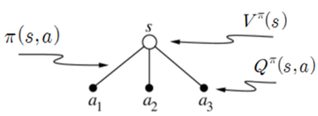
- **상태 값함수**와 **상태-행동 값함수**의 관계

$$V^{\pi}(s) = \mathrm{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi]$$
$$= \mathrm{E}\left[r_{t+1} + \gamma \sum_{s'} \gamma^k r_{t+k+2} \mid s_t = s, \pi\right]$$
$$= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma \mathrm{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right]\right]$$
$$= \sum_a \pi(s,a) \boxed{\sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^{\pi}(s')\right]}$$
$$= \sum_a \pi(s,a) Q^{\pi}(s,a)$$



- $\pi(s,a)$: 정책 π가 상태 s에서 행동 a를 선택할 확률
- $P^a_{ss'}$: 상태 s에서 행동 a를 할 때, 상태 s'이 될 확률
- $r^a_{ss'}$: 상태 s에서 행동 a를 할 때, 보상값
- γ : 할인율

$$\boxed{V^{\pi}(s) = \sum_a \pi(s,a) Q^{\pi}(s,a)}$$
$$\boxed{Q^{\pi}(s,a) = \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^{\pi}(s')\right]}$$



값 함수

❖ 값 함수 (value function)

$$\boxed{V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)}$$

$$\boxed{Q^\pi(s,a) = \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^\pi(s') \right]}$$

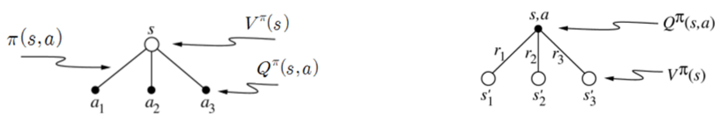


image : Richard S. Sutton

값 함수

❖ **값 함수 계산 방법**

- **동적계획법 방법** (dynamic programming, DP)
  - 모든 상태에 대한 섭렵하면서 **Bellman 최적 방정식** 성질을 이용하여 값함수 계산
  - **정책반복 학습, 값반복 학습 알고리즘**
- **몬테 카르로 방법**(Monte Carlo method)
  - 주어진 정책 π에 따라 에이전트가 행동을 하여 상태와 행동에 따른 보상값을 기록하여 상태 값함수 또는 상태-행동 값함수 추정
- **모수적 함수(parameterized function) 학습 방법**
  - **상태의 개수의 매우 많은 경우** 각 상태에 대한 보상값 관리 곤란
  - 값함수의 역할을 하는 **모수적 함수**를 학습하여 사용

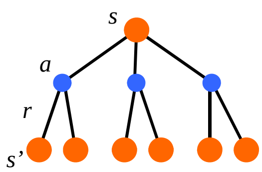
11.4 최적 정책

❖ **최적 정책**(optimal policy) π^*과 **최적 상태 값 함수** V^*

$$\pi^* = \operatorname{argmax}_{\pi} V^{\pi}(s), \ (\forall s)$$

$$V^*(s) = V^{\pi^*}(s)$$

❖ **Bellman 최적 방정식(optimality equation)**



- 최적 정책에 따른 값함수들이 만족하는 성질
- **상태 값 함수**의 경우

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^*(s') \right]$$

  - 모든 가능한 **행동 중**에서 가장 **큰 기대보상값**을 주는 행동의 값

- **상태-행동 값함수**의 경우

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^*(s') \right]$$

11.5 강화 학습 알고리즘

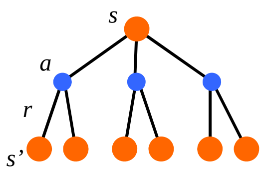
❖ **정책 평가 (policy evaluation)** $\pi \rightarrow V^{\pi}$

- **주어진 정책** π을 따를 때, 각 **상태**에서 얻게 되는 **기대보상 값** V^{π} 계산

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{k'} P^a_{ss'} \left[r^a_{ss'} + \gamma V_k(s') \right]$$

- 임의의 값 함수 V_0에서 시작하여, V_k가 **수렴할 때까지 반복**

Input : 평가할 정책 π
$V(s) \leftarrow 0$ for each $s \in S$
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$
    $temp \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(s,a) \sum_{k'} P^a_{ss'} \left[r^a_{ss'} + \gamma V(s') \right]$
    $\Delta \leftarrow \max(\Delta, |temp - V(s)|$
until $\Delta < \theta$ (작은 양수)
Output : $V \approx V^{\pi}$



강화 학습 알고리즘

❖ **정책 개선 (policy improvement)** $V^{\pi} \rightarrow \pi$

- 상태 값 함수 $V(s)$ 값으로 부터 정책 π 결정

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$
$$= \arg\max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^{\pi}(s') \right]$$

Input : 상태값 함수 V

for each $s \in S$

$\pi(s) \leftarrow \arg\max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^{\pi}(s') \right]$

Output : 정책 π

정책 반복 학습 알고리즘

❖ **정책 반복(policy iteration) 학습**

$$\pi_0 \xrightarrow{\text{정책평가}} V^{\pi_0} \xrightarrow{\text{정책개선}} \pi_1 \xrightarrow{\text{정책평가}} V^{\pi_1} \xrightarrow{\text{정책개선}} \cdots V^{\pi_*} \xrightarrow{\text{정책개선}} \pi_*$$

- **임의의 정책 π에서 시작**하여, π에 대해서 **Bellman 방정식을 수렴**할 때까지(즉, 바뀌지 않을 때까지) **적용**하여 **V^{π}를 계산**하고, V^{π}를 사용하여 **π를 개선하는 과정**을 정책 π가 수렴할 때까지 **반복**

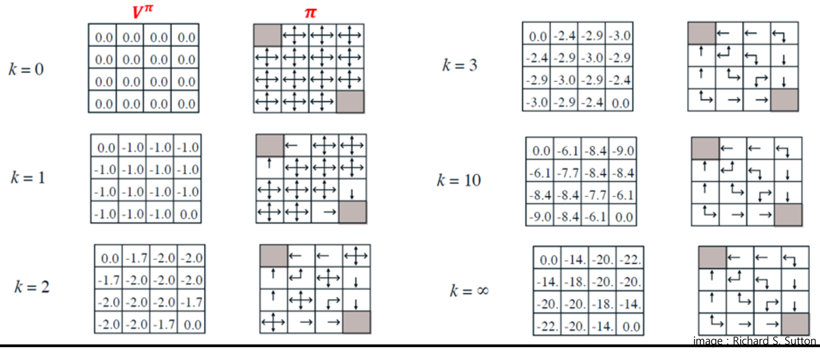


image : Richard S. Sutton

값 반복 학습 알고리즘

❖ **값 반복(value iteration) 학습**

$$V_{k+1}(s) = \max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V_k(s') \right]$$

- 임의의 값함수 V_0에서 시작하여 정책은 계산하지 않고 **값함수가 수렴할 때까지 반복**
- 수렴한 값함수 V^*를 사용하여 **정책 π를 결정**

```
V(s) ← 0 for each s ∈ S
repeat
   Δ ← 0
   for each s ∈ S
         temp ← V(s)
         V(s) ← max_a Σ_{s'} P^a_{ss'} [r^a_{ss'} + γV_k(s')]
         Δ ← max(Δ, |temp − V(s)|
until Δ < θ (작은 양수)

정책결정
for each s ∈ S
     π(s) = arg max_a Σ_{s'} P^a_{ss'} [r^a_{ss'} + γV^π(s')]
```

Q-learning 알고리즘

❖ **정책 반복, 값 반복 학습 알고리즘**
- **정확한 MDP 모델**이 필요
- 실제 상황에서는 정확한 MDP 모델을 모르는 경우가 많음

❖ **Q-learning 알고리즘**
- 모델이 없이 학습하는 강화학습 알고리즘

```
for each s and a
      Q̂(s,a) ← 0
현재 상태 s 관찰
repeat forever
     행동 a를 선택하여 수행
     즉시보상값 r를 관측
     새로운 상태 s'관찰
     Q̂(s,a) ← r + γmax_{a'} Q̂(s',a')
     s ← s'
```

Q-learning 알고리즘

for each s and a
$\hat{Q}(s,a) \leftarrow 0$
현재 상태 s 관찰
repeat forever
행동 a를 선택하여 수행
즉시보상값 r를 관측
새로운 상태 s'관찰
$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
$s \leftarrow s'$

상태 $s = 1$에서 시작, $\gamma = 0.8$
행동 $a = 5$ 선택 수행
새로운 상태 $s' = 5$ 관측
$\hat{Q}(1,5) \leftarrow$
$r(1,5)+0.8 * \max\{\hat{Q}(5,1), \hat{Q}(5,4), \hat{Q}(5,5)\}$
$=100+0.8*0 = 100$

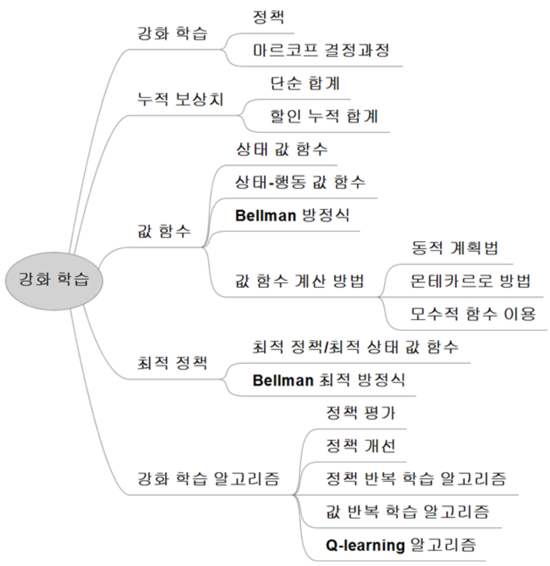
$$\hat{Q}(s,a) \quad \begin{array}{c|c} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0\\1\\2\\3\\4\\5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$r \quad \begin{array}{c|c} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0\\1\\2\\3\\4\\5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$\hat{Q}(s,a) \quad \begin{array}{c|c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0\\1\\2\\3\\4\\5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

강화 학습 알고리즘

❖ **강화 학습 알고리즘**
- **몬테카를로 방법**(Monte Carlo method)
- **시간 차이 학습**(temporal difference learning, TD-learning)
- **정책 그레이언트 알고리즘**(policy gradient algorithm)
  - 연속구간 행동을 갖는 강화학습



13. 기계학습 환경 및 도구

❖ 기계학습 환경 및 도구
- Public libraries in C, C++, Java, etc.
- **WEKA**
- R package
- Python
- MatLab or Octave

Weka 소개

❖ **Weka**
- Waikato Environment for Knowledge Analysis
- 뉴질랜드 Waikato 대학에서 개발된 Data Mining/Machine Learning 소프트웨어
- SAS, E-Miner 등의 소프트웨어보다 쉽고 무료로 제공
- 전처리, 분류, 군집화, 연관규칙, 속성 선택, 시각화에 대한 도구들을 포함
- ARFF, CSV, XRFF, binary와 같은 다양한 포맷의 파일 지원
- URL이나, JDBC를 사용한 SQL 데이터베이스로부터도 데이터 import 가능
- Java에서 import해서 사용 가능

Weka

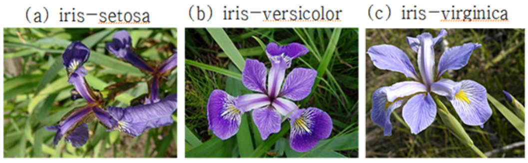
❖ **ARFF**(Attribute-Relation File Format) 데이터 형식
- 붓꽃(iris) 데이터

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa, Iris-versicolor, Iris-virginica}
```

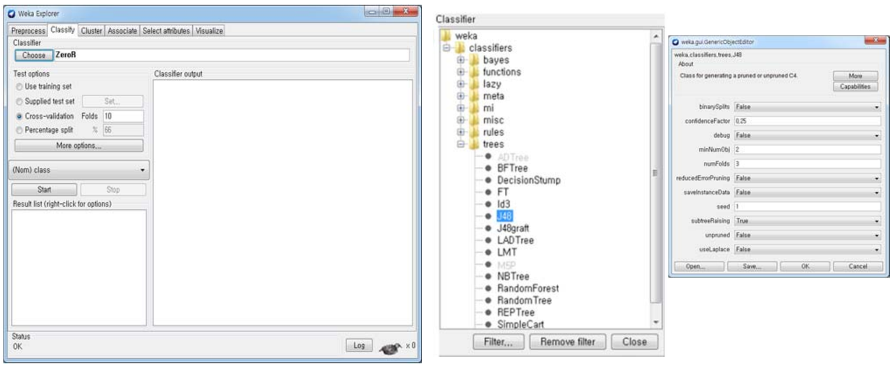
```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

(a) iris−setosa (b) iris−versicolor (c) iris−virginica

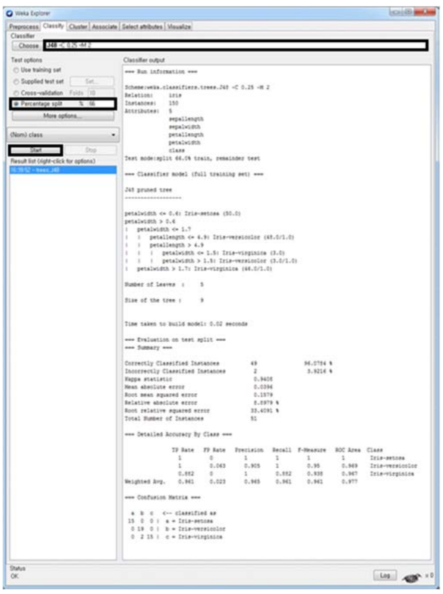


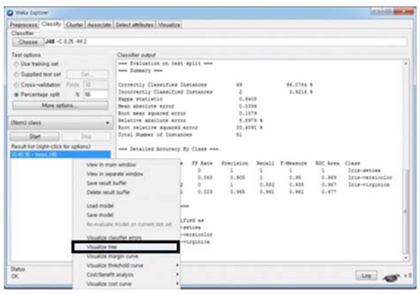
Weka

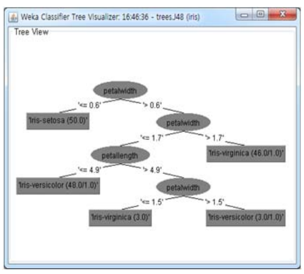
❖ 결정트리 알고리즘 J48



Weka







Java에서 Weka 사용

```java
import java.io.BufferedReader;
import java.io.FileReader;

import weka.classifiers.Evaluation;
import weka.classifiers.trees.J48;
import weka.core.Instances;

public class DecisionTreeTest {
    public static void main(String[] args) throws Exception {
        BufferedReader decisionReader = null;
        decisionReader =
            new BufferedReader(new FileReader("iris.arff"));
        Instances decision = new Instances(decisionReader); // ❶
        decision.setClassIndex(decision.numAttributes()-1);   // ❷

        J48 decisionTree = new J48();        // ❸
        decisionTree.setBinarySplits(false);    // ❹
        decisionTree.buildClassifier(decision); // ❺

        Evaluation eval = new Evaluation(decision); // ❻
        eval.evaluateModel(decisionTree, decision);   // ❼

        // ❽
        System.out.println(decisionTree.toString());
        System.out.println("=== Summary ===");
        System.out.println(eval.toSurmmaryString());
        System.out.println(eval.toClassDetailsString());
        System.out.println(eval.toMatrixString());
    }
}
```

```
J48 pruned tree
------------------

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9

=== Evaluation on test split ===

Correctly Classified Instances         147               98      %
Incorrectly Classified Instances         3                2      %
Kappa statistic                          0.97
Mean absolute error                      0.0233
Root mean squared error                  0.108
Relative absolute error                  5.2482 %
Root relative squared error             22.9089 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1          1         1          1       Iris-setosa
                 0.98      0.02       0.961      0.98      0.97       0.99    Iris-versicolor
                 0.96      0.01       0.98       0.96      0.97       0.99    Iris-virginica
Weighted Avg.    0.98      0.01       0.98       0.98      0.98       0.993

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 49  1 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```