〈데이터분석과기계학습 10주차〉
군집분석 / 인공신경망

인공지능융합공학부 데이터사이언스전공
곽찬희

군집분석

11.1. k-means

- k-means
  - ✓ 프로토타입 기반 군집(prototype-based clustering)
    - 각 클러스터가 하나의 프로토타입으로 표현
    - 프로토타입? 각 클러스터를 대표하는 속성을 도출함
    - 센트로이드(centroid, 평균) 혹은 메도이드(medoid, 최빈 포인트 혹은 대표 포인트)

  - ✓ k-means 알고리즘
    1. 샘플 포인트에서 랜덤하게 k개의 센트로이드를 초기 클러스터 중심으로 선택
    2. 각 샘플의 가장 가까운 센트로이드 $\mu^{(j)}, j \in \{1, \ldots, k\}$에 할당
    3. 할당된 샘플들의 중심으로 센트로이드를 이동
    4. 클러스터의 할당이 변하지 않거나, 사용자가 지정한 허용 오차 혹은 최대 반복 횟수에 도달할 때까지 단계 2/3 반복

11.1. k-means

- 샘플 간 유사도는?
  - ✓ 유클리디안 거리의 제곱 (squared Euclidean distance)

$$d(x,y)^2 = \sum_{j=1}^{m}\left(x_j - y_j\right)^2 = \|x-y\|_2^2$$

  - ✓ 이를 이용해 k-means를 간단하게 표현하면 SSE 혹은 클러스터 관성(cluster inertia)을 반복적으로 최소화하는 방법임

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)}\left\|x^{(i)} - \mu^{(i)}\right\|_2^2$$

    - 여기서 $\mu^{(i)}$는 클러스터 j의 센트로이드
    - $x^{(i)}$ 가 클러스터 j 안에 있다면, $w^{(i,j)} = 1$, 아니면 $w^{(i,j)} = 0$

11.1. k-means

- k-means의 단점
  - ✓ k 를 정해야 함
    - 시각화가 되는 2~3차원은 그렇다 치고, 그 이상은?
  - ✓ 처음 시작을 잘못하면 계산이 매우 길어짐
    - Centroid 선정의 중요성
    - k-means ++ !
  - ✓ 결과 해석이 어려울 수 있음
  - ✓ Scale!
    - 특성을 표준화하지 않으면 특정 특성이 모든 것을 삼킴

11.1.2. k-means++

- k-means ++
  1. 선택한 k 개의 센트로이드를 저장할 빈 집합 M을 초기화
  2. 입력 샘플에서 첫 번째 센트로이드 $\mu^{(i)}$를 랜덤하게 선택하고, M에 할당
  3. M 에 있지 않은 각 샘플 $x^{(i)}$에 대해, M에 있는 센트로이드까지 최소 제곱 거리 $d\left(x^{(i)}, M\right)^2$을 계산
  4. 다음 식과 같은 가중치가 적용된 확률 분포를 사용해, 센트로이드 $\mu^{(p)}$를 랜덤하게 선택

$$\frac{d\left(\mu^{(p)}, M\right)^2}{\sum_i d(x^{(i)}, M)^2}$$

  5. K개의 센트로이드를 선택할 때까지 2/3반복
  6. k-means 수행

11.1.2. k-means++

그림 11-1 랜덤하게 생성된 데이터셋

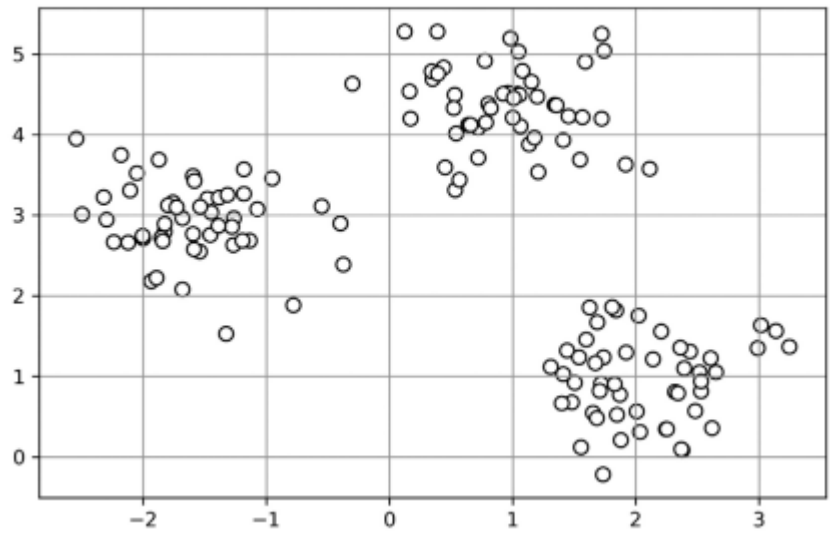
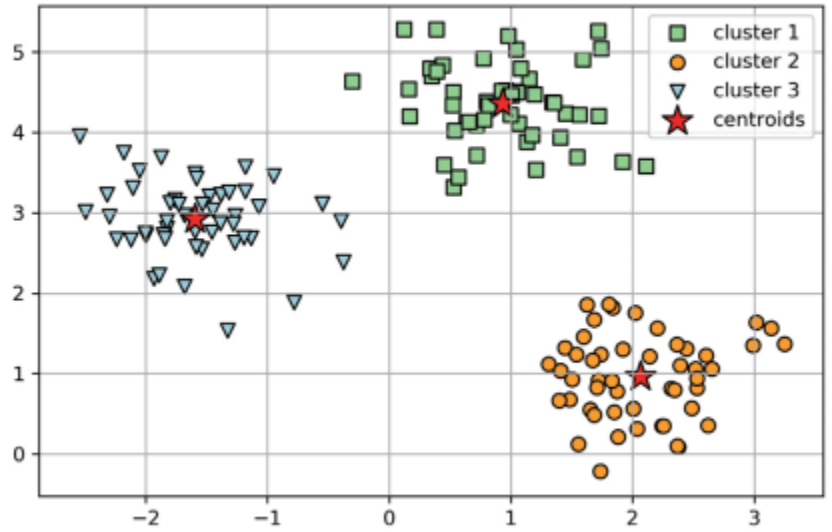


그림 11-2 k-평균 알고리즘으로 찾은 클러스터



11.1.3. 직접 군집 vs. 간접 군집

- 직접 군집(hard clustering)
  - ✓ 하나의 샘플이 하나의 클러스터에 할당
- 간접 군집(soft clustering 혹은 fuzzy clustering)
  - ✓ 하나의 샘플이 하나 이상의 클러스터에 할당
  - ✓ 대표적으로 퍼지 C 평균 (Fuzzy C-means 혹은 soft k-means 혹은 fuzzy k-means) 이 있음
  - ✓ FCM의 목적 함수

$$J_m = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{m(i,j)} \left\| x^{(i)} - \mu^{(j)} \right\|_2^2$$

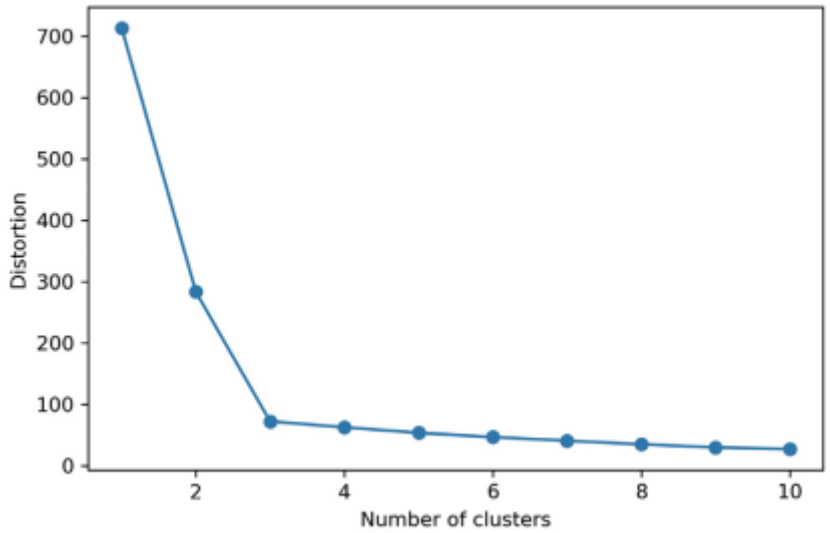
- 여기서 $w^{m(i,j)}$ 는 각 클러스터 j 에 포함될 확률을 가지고 있음

$$w^{m(i,j)} = \left[\sum_{p=1}^{k} \left(\frac{\left\| x^{(i)} - \mu^{(j)} \right\|_2}{\left\| x^{(i)} - \mu^{(p)} \right\|_2} \right)^{\frac{2}{m-1}} \right]^{-1}, \qquad \mu^{(j)} = \frac{\sum_{i=1}^{n} w^{m(i,j)} x^{(i)}}{\sum_{i=1}^{n} w^{m(i,j)}}$$

1.1.4. 엘보우 방법

- 비지도 학습인 클러스터링이기에, 알고리즘 평가에 자체 지표 사용
- 클래스 내에 SSE(혹은 왜곡, distortion)가 갑자기 튀는 구간을 찾음
- 팔꿈치 모양

그림 11-3 클래스 내 SSE를 사용한 엘보우 그래프



11.1.5. 실루엣 그래프

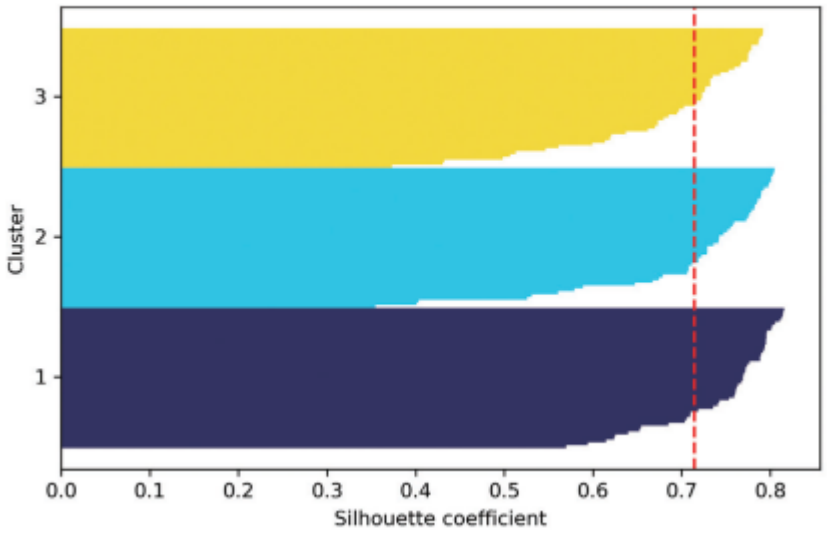
- 실루엣 분석 (silhouette analysis)
- 실루엣 계수는 다음과 같이 계산
  1. 클러스터 응집력 $a^{(i)}$:샘플 $x^{(i)}$ 와 동일한 클러스터 내 모든 다른 포인트 사이의 평균 거리
     - 클러스터 내 다른 샘플과 얼마나 비슷한지
  2. 최근접 클러스터 분리도 $b^{(i)}$: 샘플 $x^{(i)}$와 가장 가까운 클러스터의 모든 샘플 간 평균 거리
     - 샘플이 다른 클러스터와 얼마나 다른지
  3. 실루엣 $s^{(i)}$

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\{b^{(i)}, a^{(i)}\}}$$

실루엣 계수는 -1~1의 값을 가짐

이상적인 실루엣 계수 : 1 ($b^{(i)} \gg a^{(i)}$일 때)

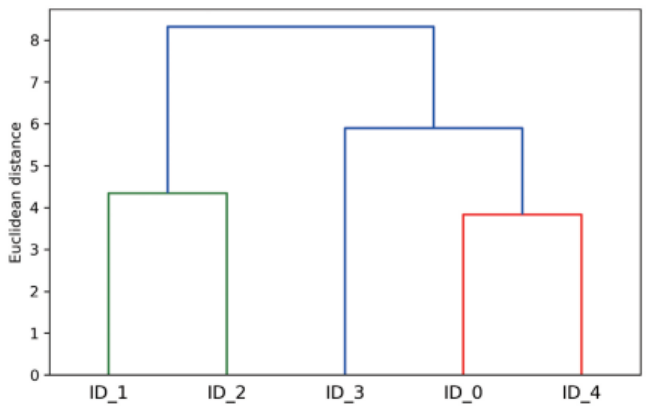
그림 11-4 k=3인 k-평균의 실루엣 그래프



11.2. 계층 군집

- 계층 군집(Hierarchical Clustering)
  - ✓ 병합 계층 군집(Agglomerative Hierarchical Clustering)
    - 전체 샘플을 포함하는 하나의 클러스터에서 시작해 작은 클러스터로 반복적으로 나눔
  - ✓ 분할 계층 군집(Divisive Hierarchical Clustering)
    - 각 샘플이 독립적인 클러스터가 되어, 클러스터가 1개가 될 때까지 합침
  - ✓ 덴드로그램(Dendrogram)
    - 계층 군집의 시각화 방법

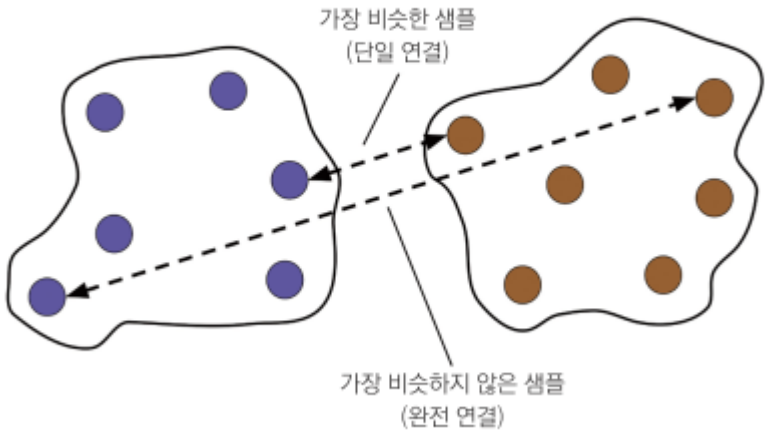
그림 11-11 덴드로그램



11.2.1. 클러스터 묶기

- 단일 연결 (single linkage)
  - ✓ 클러스터 간에 가장 가까운(가장 유사한) 두 개를 묶기
- 완전 연결 (complete linkage)
  - ✓ 클러스터 간에 가장 먼(가장 유사하지 않은) 두 개를 묶기

그림 11-7 단일 연결과 완전 연결



- 완전연결의 단계
  1. 모든 샘플의 거리 행렬 계산
  2. 모든 데이터 포인트를 단일 클러스터로 표현
  3. 가장 비슷하지 않은 샘플 사이 거리에 기초해 가장 가까운 두 클러스터 병합
  4. 유사도 행렬 업데이트
  5. 하나의 클러스터가 될 때까지 2~4를 반복

11.3. DBSCAN

- DBSCAN(Density-Based Spartial Clustering of Applications with Noise)
  - ✓ 특정 밀집도의 반경 안에 있는 샘플 개수를 정하고, 이에 따라 밀도 기반으로 클러스터링을 완성

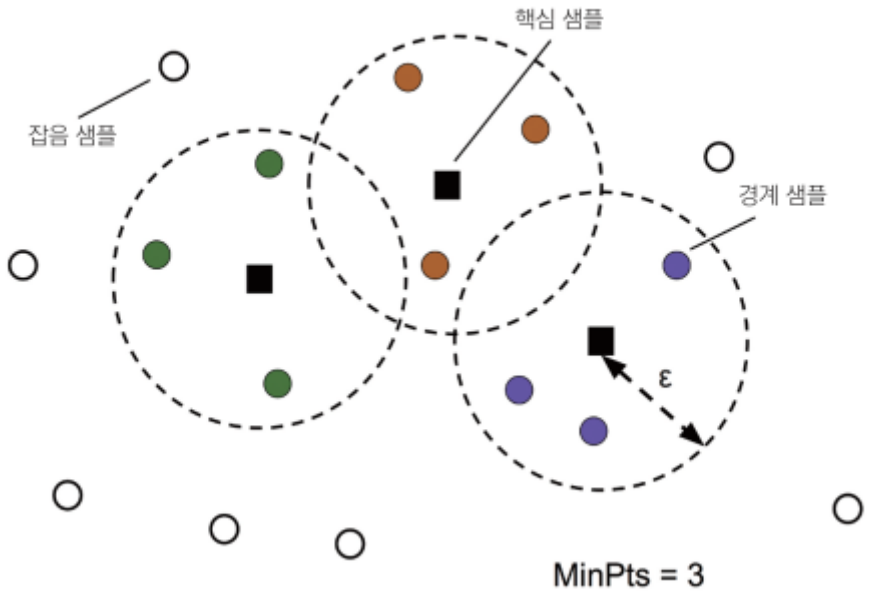
  - ✓ DBSCAN의 레이블 할당 방법
    - 어떤 샘플의 특정 반경 ε 안에 있는 이웃 샘플이 지정된 개수(MinPts) 이상이면 핵심 샘플(core point)
    - ε 이내에 MinPts 보다 이웃이 적지만 다른 핵심 샘플의 반경 ε에 있으면 경계 샘플(border point)
    - 핵심/경계가 아닌 모든 샘플은 잡음 샘플(noise point)

  - ✓ 레이블 할당 후 DBSCAN은
    1. 개별 핵심 샘플이나 ε 이내에 있는 핵심 샘플을 연결한 핵심 샘플 그룹을 클러스터로 만듦
    2. 경계 샘플을 해당 핵심 샘플의 클러스터에 할당

11.3. DBSCAN

- DBSCAN(Density-Based Spartial Clustering of Applications with Noise)

그림 11-13 DBSCAN의 핵심 샘플, 경계 샘플, 잡음 샘플



11.3. DBSCAN

- DBSCAN(Density-Based Spartial Clustering of Applications with Noise)

그림 11-15 반달 모양 데이터셋에 적용한 k-평균과 계층 군집

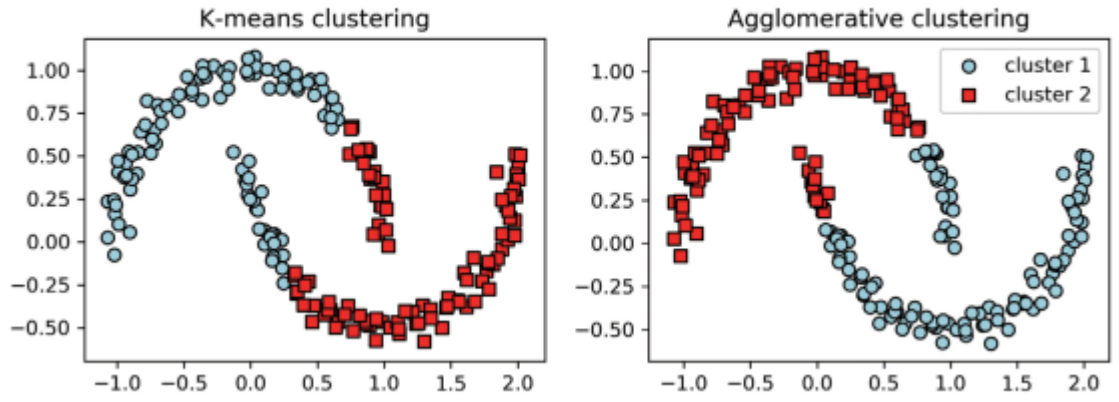
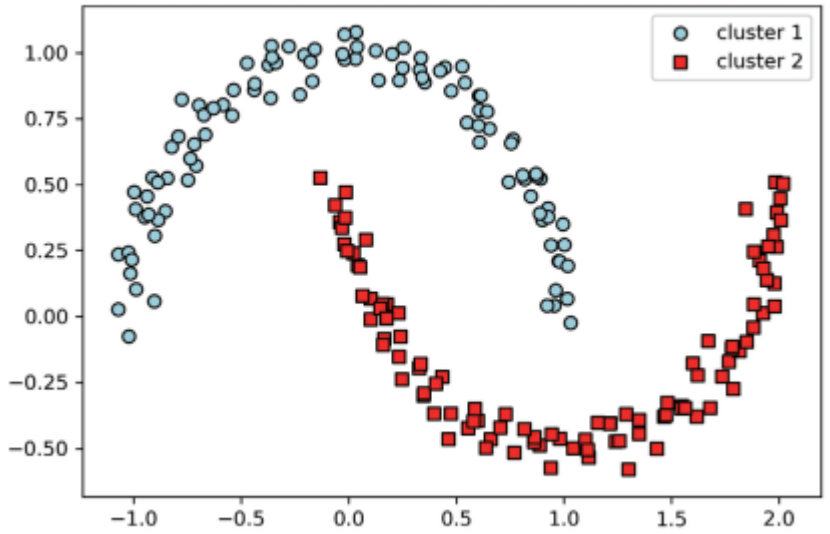


그림 11-16 반달 모양 데이터셋에 적용한 DBSCAN



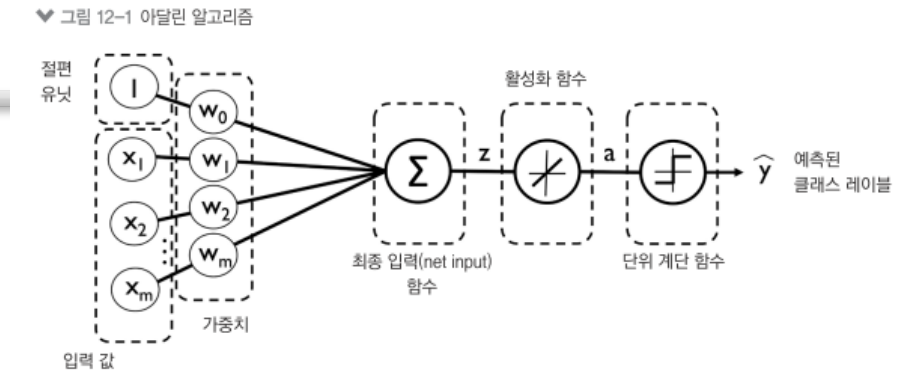
11.3. DBSCAN

- DBSCAN의 단점
  - ✓ 특성 개수가 늘어나면 차원의 저주
    - 사실 다른 클러스터링에도 문제가…
  - ✓ MinPts 와 ε 를 찾는 것이 쉽지 않음

- 실제로는 어떤 클러스터링 알고리즘이 최선일지 알기 어려움
  - ✓ 하이퍼파라미터 튜닝만으로 해결이 불가능한 경우가 많음(특히 고차원일 때)
  - ✓ Domain knowledge 가 더 중요한 역할을 할 수도

인공신경망

12.1.1.인공 신경망



- **ADALINE 기억나세요?**

$$w := w + \Delta w$$
$$\Delta w = -\eta \nabla J(w)$$

✓ **전체 데이터에 대해 그래디언트($\nabla J(w)$)를 계산하고 그 반대 방향으로 학습을 시킴**
✓ **목적함수(혹은 비용함수) $J(w)$ 최적화를 위해 제곱 오차합(Sum of Squared Errors, SSE) 사용**
✓ **가중치의 업데이트**

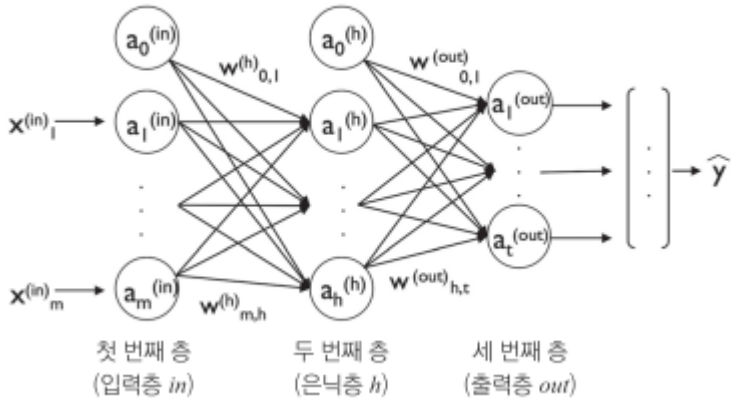
$$\frac{\partial}{\partial w_j} J(w) = -\sum_i \left(y^{(i)} - a^{(i)}\right) x_j^{(i)}$$

✓ $y^{(i)}$**는 샘플** $x_j^{(i)}$ **의 타깃 클래스 레이블이며,** $a^{(i)}$**는 뉴런의 활성화 출력(활성화 함수의 결과값)**
✓ **활성화함수** $\phi(z) = z = a$
✓ **최종 입력** $z = \sum_j w_j x_j = w^T x$

12.1.2. 다층 신경망

- ADALINE이 단층이었다면, 이 층을 늘려봅시다.
- 다층 퍼셉트론 (Multilayer Perceptron, MLP)
  - ✓ 입력층: 가장 첫 레이어
  - ✓ 은닉층: 입력-출력 사이에 있는 레이어
  - ✓ 출력층: 가장 마지막 레이어

그림 12-2 다층 퍼셉트론

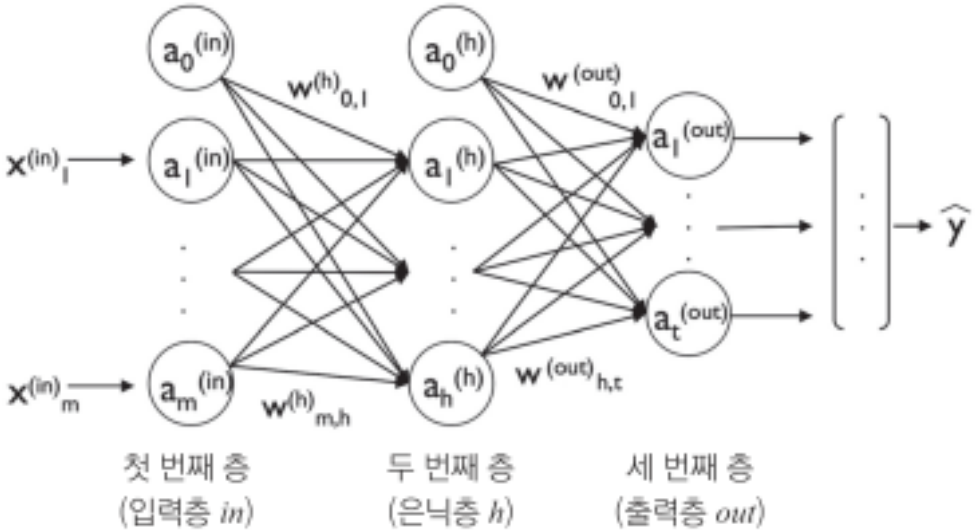


- 은닉층이 1개 이상인 경우, 딥러닝(이라고 하지만 한 개 짜리는 없…)
  - ✓ 은닉층의 숫자와 구조 또한 hyperparameter
  - ✓ 이를 연구하고 개발하는 것이 딥러닝 연구자들의 일 (혹은 미래의 여러분이 할 일)

12.1.2. 다층 신경망

- $a_i^{(l)}$: l 번째 층에 있는 i 번째 유닛의 활성화 출력
- $w_{k,j}^{(l+1)}$: l층의 k번째 유닛에서 l+1층의 j 번째 유닛으로 가능 연결
- W : 가중치 행렬
- $W^{(h)} \in \mathbb{R}^{m \times d}$: h층의 가중치 행렬은 입력 유닛의 수에 절편을 더한 m과 은닉 유닛의 수 d의 곱의 차원을 가짐

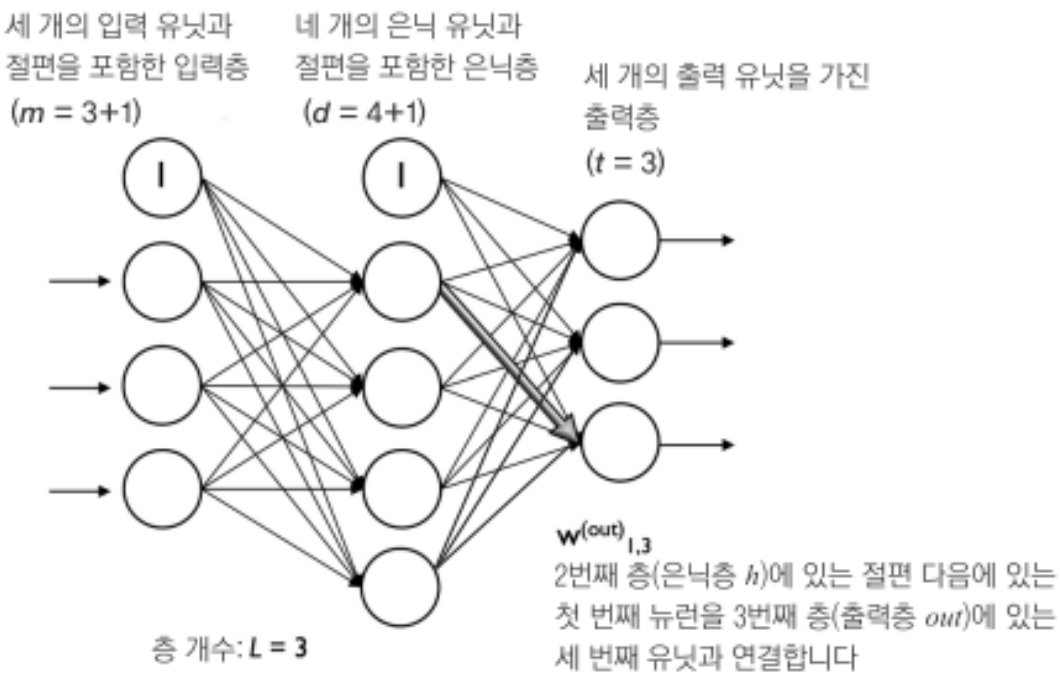
그림 12-2 다층 퍼셉트론



12.1.2. 다층 신경망

- $W^{(h)} \in \mathbb{R}^{m \times d}$ 의 의미를 계산해봅시다.

그림 12-3 다층 퍼셉트론의 표기법



12.1.3. 정방향 계산 (forward propagation)

- MLP의 학습 단계
  1. (정방향 계산)입력층에서 시작해 정방향으로 훈련 데이터의 패턴을 네트워크에 전파하여 출력 생성
  2. (오차 계산)네트워크의 출력을 기반으로 비용 함수를 이용해 최소화해야 할 오차 계산
  3. (역방향 계산, 역전파)네트워크에 있는 모든 가중치에 대해 도함수를 찾고, 오차를 역전파해 모델 업데이트

12.1.3. 정방향 계산 (forward propagation)

- MLP의 계산 과정

$$z_1^{(h)} = a_0^{(in)} w_{0,1}^{(h)} + a_1^{(in)} w_{1,1}^{(h)} + \cdots + a_m^{(in)} w_{m,1}^{(h)}$$

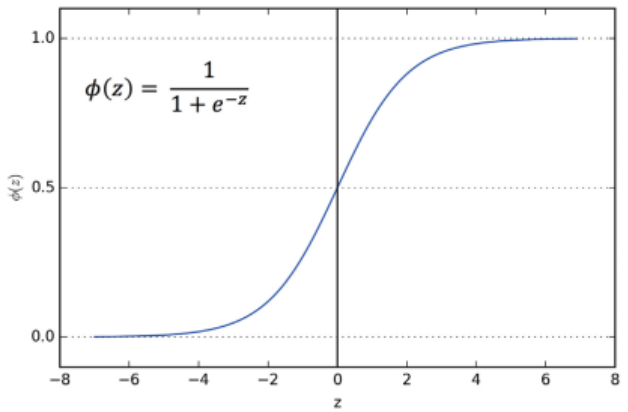
$$a_1^{(h)} = \phi(z_1^{(h)})$$

- $z_1^{(h)}$는 최종 입력
- ϕ는 활성화함수 -> 복잡한 문제를 풀기 위해서는 비선형 활성화 함수 사용 필요
  - ✓ 예를 들어 sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

  - ✓ Why?
    - 선형 함수의 조합은 선형이기 때문

그림 12-4 시그모이드 함수



12.1.3. 정방향 계산 (forward propagation)

- 간단하게 정리하면 입력층은

$$z^{(h)} = a^{(in)} W^{(h)}$$

$$a^{(h)} = \phi(z^{(h)})$$

  - ✓ $a^{(in)}$ 는 샘플 에 절편을 더한 $1 \times m$ 차원 특성 벡터
  - ✓ $W^{(h)}$는 $m \times d$ 차원의 가중치 행렬
  - ✓ 따라서 활성화 출력 $a^{(h)}$ 는 $1 \times d$ 차원을 가짐

- 일반화를 한다면,

$$Z^{(h)} = A^{(in)} W^{(h)}$$

$$A(h) = \phi\big(Z(h)\big)$$

12.1.3. 정방향 계산 (forward propagation)

- 따라서 출력층은

$$Z^{(out)} = A^{(h)} W^{(out)}$$

$$A^{(out)} = \phi\left(Z^{(out)}\right), A^{(out)} \in \mathbb{R}^{n \times t}$$

✓ $n \times d$ 차원 행렬 $A^{(h)}$와 $d \times t$ 차원 행렬 $W^{(out)}$을 곱해 $n \times t$ 차원 행렬 $Z^{(out)}$ 이 도출됨

12.3.1. 로지스틱 비용 함수 계산

- **MLP의 비용함수는 로지스틱 비용 함수와 동일**

$$J(w) = -\sum_{i=1}^{n} y^{[i]} \log\left(a^{[i]}\right) + \left(1 - y^{[i]}\right) \log\left(1 - a^{[i]}\right)$$

✓ $a^{[i]}$ **는 데이터셋 I 번째 샘플의 시그모이드 활성화 출력이며, 정방향 계산으로 구할 수 있음**

$$a^{[i]} = \phi\left(z^{[i]}\right)$$

✓ $[i]$ **는 훈련 샘플의 인덱스 (층 x)**

✓ **L2 규제항을 추가하면**

$$J(w) = -\sum_{i=1}^{n} y^{[i]} \log\left(a^{[i]}\right) + \left(1 - y^{[i]}\right) \log\left(1 - a^{[i]}\right) + \frac{\lambda}{2} \|w\|_2^2$$

12.3.1. 로지스틱 비용 함수 계산

- t개의 원소를 가진 출력 벡터와 레이블이 담긴 타깃 벡터(t x 1 차원)를 비교하면, (one-hot-encoding)

$$a^{(out)} = \begin{bmatrix} 0.1 \\ 0.9 \\ \cdots \\ 0.3 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 1 \\ \cdots \\ 0 \end{bmatrix}$$

- t개의 활성화 유닛 전체에 대해 로지스틱 비용 함수를 일반화하면,

$$J(W) = -\sum_{i=1}^{n}\sum_{j=1}^{t} y_j^{[i]} \log\left(a_j^{[i]}\right) + \left(1 - y_j^{[i]}\right) \log\left(1 - a_j^{[i]}\right)$$

✓ 여기서 [i] 는 특정 샘플의 인덱스, j 는 출력층의 활성화 출력 결과를 탐색

12.3.1. 로지스틱 비용 함수 계산

- 앞의 식에 규제항을 추가하면,

$$J(W) = -\sum_{i=1}^{n}\sum_{j=1}^{t} y_j^{[i]} \log\left(a_j^{[i]}\right) + \left(1 - y_j^{[i]}\right) \log\left(1 - a_j^{[i]}\right) + \frac{\lambda}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{u_l}\sum_{j=1}^{u_{l+1}} \left(w_{j,i}^{(l)}\right)^2$$

  - ✓ 규제항을 잘 보면, 네트워크에 존재하는 모든 가중치 (w)의 제곱합을 의미 (u_l 은 l 층의 유닛 개수, i와 j가 1부터이므로, 절편을 제외)

- 즉, J(W)를 최소화한다는 것은 다음을 계산해야 되는 것을 뜻함

$$\frac{\partial}{\partial w_{j,i}^{(l)}} J(W)$$

12.3.2. 역전파 알고리즘의 이해

- 연쇄 법칙(chain rule)
  - ✓ 함성함수 f(g(x))에 대한 도함수는

$$\frac{d}{dx}[f(g(x))] = \frac{df}{dg} \cdot \frac{dg}{dx}$$

  - ✓ 합성함수를 길게 만들면, F의 도함수는

$$\frac{dF}{dx} = \frac{d}{dx}F(x) = \frac{d}{dx}F\left(g\left(h\left(u(v(x))\right)\right)\right) = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

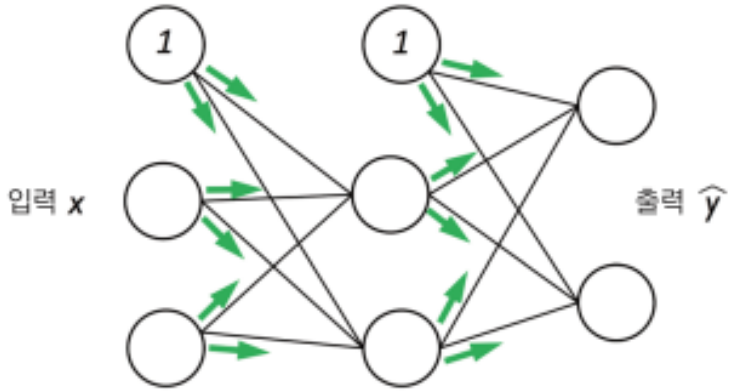
  - ✓ 컴퓨터를 이용한 컴퓨터 대수학 (computer algebra)는 이를 계산하기 위해 자동 미분 (automatic differentiation)의 여러 방법을 만듦

  - ✓ 자동 미분의 두 종류
    - 정방향: 상대적 계산 비용이 큼 (행렬 x 행렬)
    - 역방향: 상대적 계산 비용이 작음 (벡터 x 행렬)

12.3.3. 역전파 알고리즘으로 신경망 훈련

- **기본적인 공식**
  - ✓ **은닉층의 최종 입력:** $Z^{(h)} = A^{(in)}W^{(h)}$
  - ✓ **은닉층의 활성화 출력:** $A^{(h)} = \phi(Z^{(h)})$
  - ✓ **출력층의 최종 입력:** $Z^{(out)} = A^{(h)}W^{(out)}$
  - ✓ **출력층의 활성화:** $A^{(out)} = \phi\left(Z^{(out)}\right)$

그림 12-11 신경망의 정방향 계산



- **출력층의 오차 벡터**

$$\delta^{(out)} = a^{(out)} - y$$

  - ✓ **y 는 정답 클래스 레이블**

12.3.3. 역전파 알고리즘으로 신경망 훈련

- 은닉층의 오차 항 (⊙은 원소별 곱셈)

$$\delta^{(h)} = \delta^{(out)}\left(W^{(out)}\right)^T \odot \frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}}$$

✓ $\frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}}$ 는 시그모이드 활성화 함수의 도함수로 다음과 같이 정리 가능

$$\frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}} = \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right.$$

✓ 따라서, 위의 식은

$$\delta^{(h)} = \delta^{(out)}\left(W^{(out)}\right)^T \odot \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right.$$

12.3.3. 역전파 알고리즘으로 신경망 훈련

- 오차항 δ를 구한 뒤, 비용 함수의 도함수는 다음과 같이 쓸 수 있음

$$\frac{\partial}{\partial w_{i,j}^{(out)}} J(W) = a_j^{(h)} \delta_i^{(out)}$$

$$\frac{\partial}{\partial w_{i,j}^{(h)}} J(W) = a_j^{(in)} \delta_i^{(h)}$$

✓ 어디서 본듯한 형태인데? 오차를 업데이트 할 때, (틀린 정도) x (원래값)

- 이를 각 미니 배치 샘플들에 대해 계산한다면,

$$\Delta^{(h)} = \left(A^{(in)}\right)^T \delta^{(h)}$$

$$\Delta^{(out)} = \left(A^{(h)}\right)^T \delta^{(out)}$$

12.3.3. 역전파 알고리즘으로 신경망 훈련

- 여기서 편도 함수를 누적한 후, 규제항을 추가하면

$$\Delta^{(l)} := \Delta^{(l)} + \lambda^{(l)} W$$

- 마지막으로 l층에 대한 그래디언트의 반대 방향으로 가중치 업데이트

$$W^{(l)} := W^{(l)} - \eta \Delta^{(l)}$$

12.3.3. 역전파 알고리즘으로 신경망 훈련

- 한 번에 정리

그림 12-12 역전파 알고리즘