CH8

8.1 성능 최적화

- 데이터를 사용하자 → 최대한 많은 데이터 수집/생성/범위 조정 (정규화 표준화 규제화)
- 알고리즘을 이용하자
- 알고리즘 튜닝 → 하이퍼파라미터 다양하게 시도
    - 문제 진단 - 모델 평가 : 오버피팅 ? 언더피팅? 조기종료
    - 가중치, 학습률, 활성화함수(데이터 타입에 따라), 배치와 에포크, 옵티마이저와 손실함수(SGD, Adam, RMSProp 등)
    - 네트워크 토폴로지 : 은닉층이나 뉴런 개수 등
- 앙상블 : 모델 여러 개 섞어서

8.2 하드웨어를 이용한 성능 최적화

- GPU를 이용하자
    - CPU : 명령어가 입력되는 순서대로 데이터를 처리하는 직렬 처리 방식
        - 연산을 담당하는 ALU와 명령어를 해석하고 실행하는 컨트롤(control), 그리고 데이터를 담아 두는 캐시(cache)로 구성되어 있습니다.
        - 한 번에 하나의 명령어만 처리하기 때문에 연산을 담당하는 ALU 개수가 많을 필요가 없음
    - GPU : 여러 명령을 동시에 처리하는 병렬 처리 방식에 특화
        - 캐시 메모리 비중은 낮고, 연산을 수행하는 많은 ALU로 구성
        - GPU는 서로 다른 명령어를 동시에 병렬적으로 처리하도록 설계되었기 때문에 성능에 부담이 없습니다.
        - 또한, 하나의 코어에 ALU 수백~수천 개가 장착되어 있기 때문에 CPU로는 시간이 많이 걸리는 3D 그래픽 작업 등을 빠르게 수행할 수 있습니다.
    - 개별적 코어 속도는 CPU가 GPU보다 훨씬 빠름

- 파이썬이나 매트랩(MATLAB)처럼 행렬 연산을 많이 사용하는 재귀 연산이 대표적인 '직렬' 연산 수행
- 역전파처럼 복잡한 계산은 병렬 연산이 속도 빠름 → 딥러닝에서 시간 단축 많이

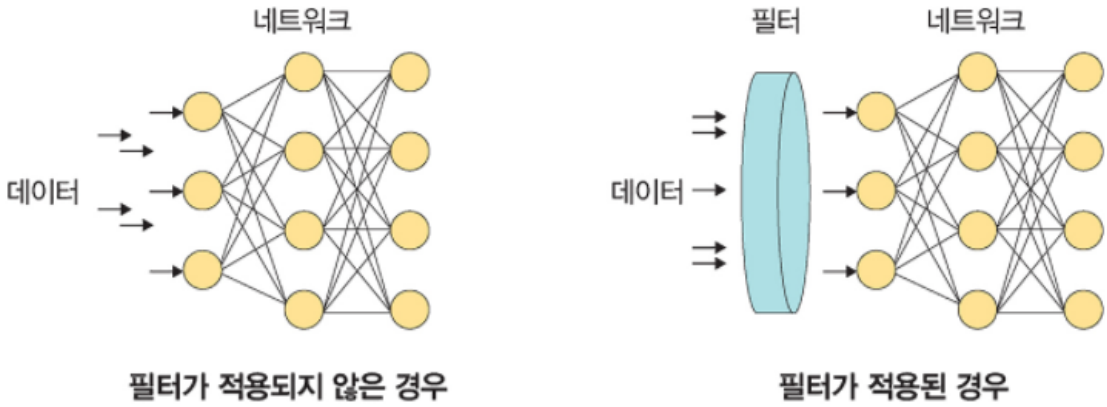
8.3 하이퍼파라미터를 이용한 성능 최적화

1. 정규화
   - EX) 이미지 픽셀 정보 0~255 → 0~1사이 값을 갖게
   - 특성 스케일링

$$\frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

(x : 입력 데이터)

2. 규제화



필터가 적용되지 않은 경우 필터가 적용된 경우

모델 복잡도 줄이기 위해 제약 → 필터 적용하여 데이터를 걸러냄

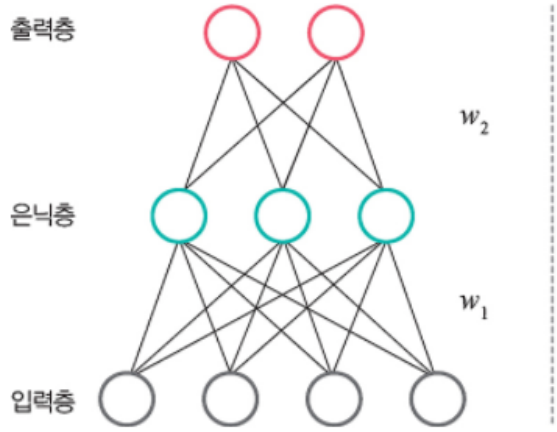
   - 드롭아웃
   - 조기 종료

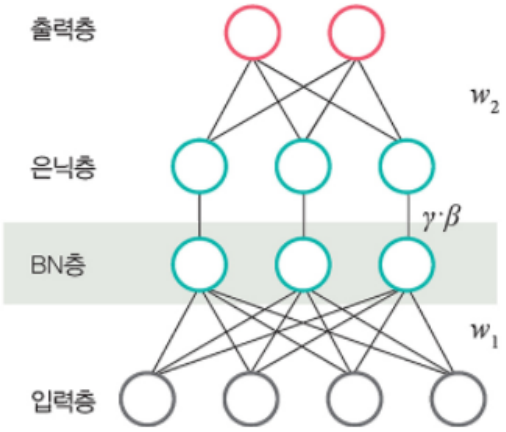
3. 표준화
   - 평균 0, 표준편차 1인 데이터로 만들기
   - 평균을 기준으로 얼마나 떨어져있나?

$$\frac{x - m}{\sigma}$$

배치 정규화

제안 논문 : Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift





▲ 그림 8-35 배치 정규화

- 데이터 분포 안정 → 학습속도 상승
- 기울기 소멸/폭발 문제 해결 가능 → 내부 공변량 변화이 원인 = 각 층마다 활성화 함수가 적용되면서 입력 값 분포가 계속 바뀌는 현상
- 정규 분포를 만들기 위해 미니 배치에 표준화 적용하는 방식
- 배치 크기가 충분히 커야 정규화 잘 됨
- 오차가 낮은 값으로 안정적이게 감소

$$\mu\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad \cdots\cdots ①$$

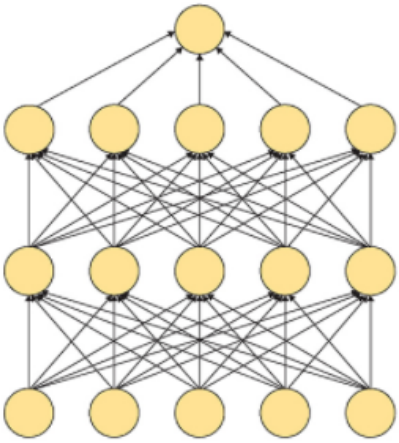
$$\sigma^2\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu\beta)^2 \quad \cdots\cdots ②$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu\beta}{\sqrt{\sigma^2\beta + \epsilon}} \quad \cdots\cdots ③$$

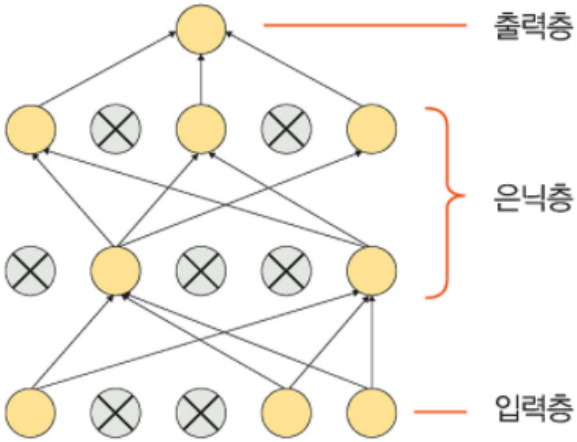
$$y_i \leftarrow \gamma\hat{x}_i + \beta \Leftrightarrow BN_{\gamma,\beta}(x_i) \quad \cdots\cdots ④$$

$$\begin{pmatrix} \text{입력: } \beta = \{x_1, x_2, \cdots, x_n\} \\ \text{학습해야 할 하이퍼파라미터: } \gamma, \beta \\ \text{출력: } y_i = BN_{\gamma,\beta}(x_i) \end{pmatrix}$$

드롭아웃



일반적인 신경망



드롭아웃이
적용된 신경망

- 훈련 시 일정 비율만 사용하고, 나머지 뉴런 가중치는 업데이트 하지 않음. 즉 은닉층 노드 일부를 일부러 끄며(무작위) 학습하여 과적합을 방지함
- 테스트 시 노드 모두 사용하되, 드롭아웃 비율을 곱해 성능 평가
- 훈련 시간 길어지나 성능 향상엔 도움

조기종료

- 학습은 언제 종료시킬지 결정. 최고 성능 보장 X. 자원의 효율화
- 학습률 스케줄러 → 오차 우상향 방지 및 성능 향상