12주차 예습과제

▼ 8.1 성능 최적화

▼ 8.1.1 데이터를 사용한 성능 최적화

- 최대한 많은 데이터 수집하기
- 데이터 생성하기
- 데이터 범위 조정하기
- 정규화(minmax)
- 규제화
- 표준화(standard)

▼ 8.1.2 알고리즘을 이용한 성능 최적화

- svm, k-최근접 이웃
- 시계열 : RNN, LSTM, GRU

▼ 8.1.3 알고리즘 튜닝을 위한 성능 최적화

- 진단 : 과적합인지 고려해보기.
  - 훈련 성능이 검증보다 눈에 띄게 좋다면 과적합 의심. 규제화로 성능향상 시킬 수 있음
  - 둘다 좋지 않으면 과소적합 의심. 네트워크 구조를 변경하거나 훈련을 늘리기 위해 에포크 수 조정
  - 훈련 성능이 검증을 넘어서는 변곡점이 있으면 조기종료 고려
- 가중치 : 초깃값은 작은 난수를 사용하는데, 애매하다면 오토인코더와 같은 비지도 학습을 이용하여 사전훈련 후 지도학습을 진행하는 것도 방법
- 학습률 : 난수를 선택하여 학습결과를 보고 조금씩 변경. 네트워크 계층과 비례하도록
- 활성화 함수 : 신중해야. 일반적으로 활성화함수로 시그모이드나 하이퍼볼릭 탄젠트를 사용했다면 출력층에서는 소프트맥스나 시그모이드함수를 선택
- 배치와 에포크 : 큰 에포크과 작은 배치를 사용하는 것이 트렌드
- 옵티마이저 및 손실함수 : 아담과 알엠에스프롭 주로 사용. 여러가지 사용해보고 성능 좋은 것 선택

▼ 8.1.4 앙상블을 이용한 성능 최적화

▼ 8.2 하드웨어를 이용한 성능최적화

▼ 8.2.1 CPU와 GPU 사용의 차이

- CPU : 직렬 처리 방식, 파이썬이나 매트랩처럼 행렬 연산을 많이 사용하는 재귀 연산이 대표적인 직렬 연산 수행
- GPU : 병렬 연산 방식, 역전파처럼 복잡한 미적분은 병렬 연산을 해야 속도가 빨라짐.
- 개별적 코어 속도는 CPU가 GPU보다 훨씬 빠름

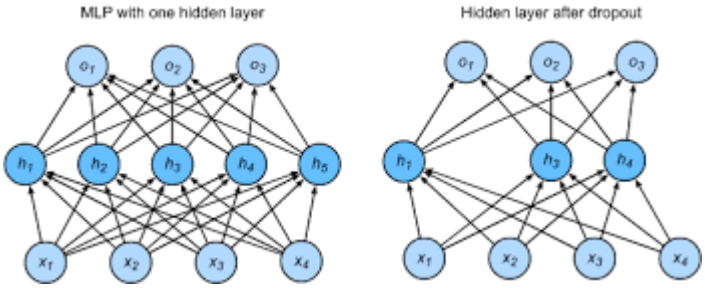
▼ 8.3 하이퍼 파라미터를 이용한 성능 최적화

▼ 8.3.1 배치 정규화를 이용한 성능 최적화

- 정규화 : 0~1 사이의 값으로 . Minmax
- 규제화 : 필터 적용
  - 드롭아웃
  - 조기종료
- 표준화 : 평균은 0 표준편차는 1로. z-score normalization
- 배치 정규화 : 기울기 소멸이나 폭발 같은 문제를 해결하기 위한 방법
  - 손실함수로 렐루를 사용하거나 초깃값 튜닝, 학습률 조정
  - 기울기 소멸이나 폭발의 원인은 공변량 변화 때문
  - → 정규분포로 만들기 위해 표준화와 유사한 방식을 미니배치에 적용하여 평균은 0으로, 표준편차는 1로 유지하게 함

▼ 8.3.2 드롭아웃을 이용한 성능 최적화

- 드롭아웃 : 일정비율의 뉴런만 사용하고, 나머지 뉴런에 해당하는 가중치는 업데이트 하지 않는 방법
- 어떤 노드를 비활성화할지는 학습할 때마다 무작위로 선저오디며, 테스트 데이터로 평가할 때는 노드를 모두 사용하여 출력하되 노드 삭제 비율을 곱해서 성능을 평가
-



▼ 8.3.4 조기종료를 이용한 성능 최적화

- 조기종료 : 뉴럴 네트워크가 과적합을 회피하는 규제방법. 훈련데이터와 별도로 검증데이터를 준비하고 매 에포크마다 검증데이터에 대한 오차를 측정하여 모델의 종료시점을 제어. 오차가 증가하는 시점에서 학습을 멈추도록 조정