CNN

1. 완전 연결 계층(Fully Connected Layer)

- 정의 : 한 층의 모든 뉴런이 그 다음층 모든 뉴런과 연결된 상태
  -> 1차원 배열의 형태로 평탄화된 행렬을 통해 이미지를 분류하는데 사용되는 계층
- 방법 : 흑백 이미지, 2차원 벡터 행렬을 1차원 배열로 평탄화 -> Relu 함수로 뉴런 활성화
  -> softmax 함수로 이미지를 분류
- 문제점 : 3차원 컬러 이미지의 정보 손실
  -> 3차원 이미지를 1차원으로 평탄화하면 공간 정보가 손실됨
  -> 정보 부족으로 이미지를 분류하는데 한계가 생김
  -> 정확도를 높이는데 한계가 있음
- 해결방법 : CNN

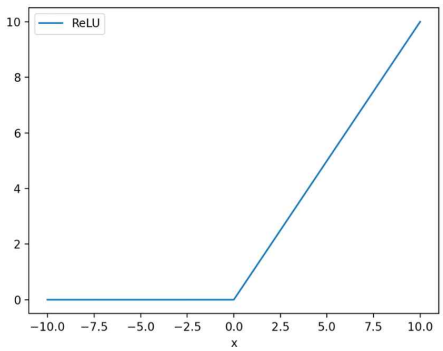
추가 개념

1. 렐루 함수(Relu) : $ReLU(x) = \max(0, x)$

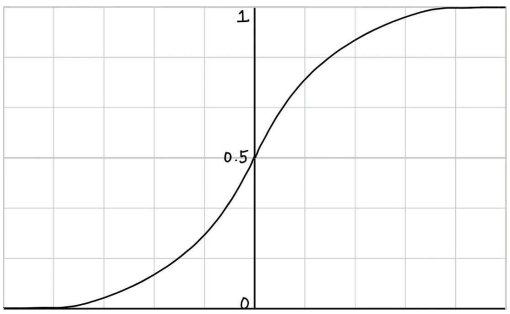
- 딥러닝에서 사용하는 활성화 함수로 시그모이드보다 렐루를 더 많이 사용
- 양수면 입력값을 출력하고 아니면 0을 출력
  ->양수의 기울기 = 1, 음수의 기울기 = 0
- 선호 이유 : 기울기 소실 방지(시그모이드 함수의 문제), 포화 문제 방지(심층 네트워크에 적합), 계산 효율성이 높음, 희소 활성화를 생성
  + 희소 활성화 : 뉴런이 0을 출력하게 해서 활성화를 적게 함

2. 소프트맥스 함수(softmax) : $softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$

- 입력받은 값의 출력을 0 ~ 1 사이의 값으로 모두 정규화하며 출력 값들의 총합 = 1
- 분류하고 싶은 클래스의 수 만큼 출력으로 구성
  -> 가장 큰 출력 값을 부여받은 클래스가 확률이 가장 높은 것으로 이용



렐루 함수



소프트맥스 함수

2. CNN

- 정의 : 이미지의 공간 정보를 유지한 상태로 학습 가능
    -> CNN 구조 : 합성곱층(covolutional layer), 풀링층(pooling layer)
- 파라미터의 수를 줄일 수 있음 -> 복잡한 기능 구현 가능
- 장점
    1. 고차원 데이터를 오버피팅 되지 않고 다룸
    2. 위치마다 동일한 feature을 추출 -> translation invariant 문제 해결
    -> translation invariant : 이미지의 위치가 바뀌어도 해석은 변하지 않음
    3. local realationship 추출
- Local Connectivity : 이미지 내의 각기 다른 영역의 가중치와 같은 동일한 정보 공유
- 가중치 공유(Weight Sharing) : 필터로 가중치를 공유하고 학습해야 할 가중치를 줄임
    -> 장점
        1. 파라미터 수 감소 : 메모리 사용량과 계산 복잡도 감소
        2. 학습 개선 : 더 많은 데이터를 활용하여 패턴 학습
        3. 일반화 : 유사한 패턴을 학습하여 일반화 능력 향상
        4. 도메인 지식의 효율적 활용
    -> 적용 가능 조건
        1. 동일한 계층 구조 : 동일한 형태와 크기
        2. 동일한 연산
        3. 동일한 손실 함수 및 역전파
- 단점
    1. 연산량이 많음
    2. 전체 데이터를 보고 특성을 추출할 수 없음
    3. Pooling 층을 거치면서 작은 local 정보를 잃을 수 있음

3. 합성곱

- 합성곱 : $(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$

1. 연속함수 g를 반전시킨다
2. 반전시킨 함수 g를 t만큼 이동시킴
3. 이동시킨 함수 g를 함수 f와 곱하여 결과를 하나씩 기록
-> 변수 타우를 변화시키며 결과를 기록하는 것
- 사용 : 딥러닝, 신호의 필터(샘플링), 라플라스 변환, 푸리에 변환 etc...
+ f = 본래의 신호, 행렬, 이미지 / g = 필터 가중치로 가정함
두 함수를 합성곱 하면 주어진 신호, 행렬, 이미지를 원하는 함수로 만들 수 있음
-> 목적에 따라 함수 g를 선정하여 분해, 변환, 필터링 할 수 있음
- 문제점 : 이미지의 크기가 줄어든다
    -> 해결을 위해 Padding 사용

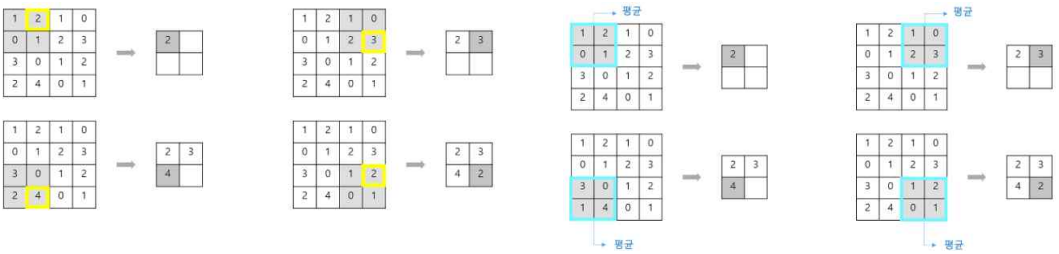
4. Padding

- 합성곱 연산 전에 입력 데이터 주변에 특정 값을 채우는 것
- 출력 데이터의 크기를 조정하기 위해 사용
- 종류
    1. 제로패딩(zero padding) : 이미지의 가장자리에 0의 값을 갖는 픽셀 추가
    2. 밸리드 패딩(valid padding) : 출력이 입력 데이터보다 작아짐
    3. 풀 패딩(full padding) : 입력데이터의 모든 원소가 합성곱 연산의 비율로 참여
    4. 세임 패딩(same padding) : 입력 크기와 출력 크기 동일
- 사용이유
    1. 이미지 데이터의 축소를 막기 위해 -> padding을 이용하여 크기 조절
    2, 윤곽자리의 이미지 정보를 보존하기 위해

+ Stride Convolution : 합성곱 신경망의 기본구성요소 -> 필터적용 시 이동 간격
ex) 필터/커널은 입력 이미지에서 1pixel씩 이동하면서 연산하는데,
stride를 주게 되면 pixel만큼씩 필터가 이동하며 연산

5. Pooling

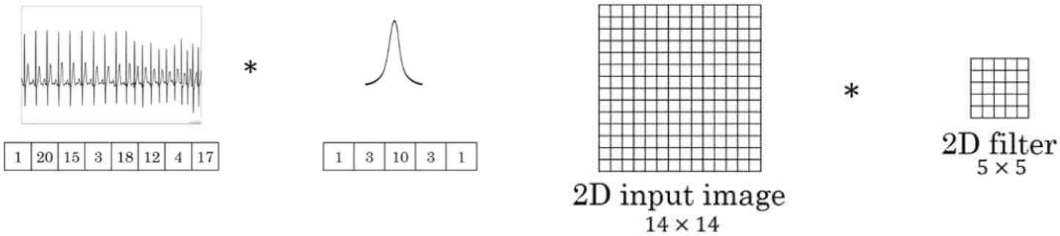
- 2차원 데이터의 세로, 가로 방향의 공간을 줄이기 위해 사용
  -> 데이터의 차원을 감소시켜 신경망의 계산의 효율을 높임
- 특징
    1. 대상 영역에서 최댓값이나 평균을 취하므로 따로 학습할 것 없음
    2. 채널 수가 변하지 않음
    -> 2차원 데이터의 크기를 줄이는 연산으로 3차원을 정하는 채널 수 건들지 않음
    3. 입력의 변화에 영향을 적게 받음
- 종류
    1. 최대 풀링(max Pooling) : 선택된 픽셀 중 제일 큰 값이 대푯값
    -> 이미지 인식 분양에서 주로 사용
    2. 평균 풀링(average Pooling) : 선택된 픽셀 값들의 평균이 대푯값



최대 풀링 평균 풀링

6. conv(n)d [n = 1, 2]

- Conv1D = 1차원 배열 데이터에 사용 / Conv2D = 2차원 배열 데이터에 사용
- Conv1D : 합성곱 진행 방향이 가로 -> 한 방향으로만 합성곱을 진행함
  활용 : sequence 모델과 자연어처리에 사용
- Conv2D : 합성곱 진행 방향이 가로, 세로 -> 두 방향으로만 합성곱을 진행함
  활용 : 컴퓨터 비전에 사용



Conv1D Conv2D

- kernel size
  1. 합성곱 필터를 커널로 부르기도 함
  2. 면적(가로x세로)을 의미함
  3. 커널 크기가 크면 더 많은 Feature 정보를 가져올 수 있음
  -> 큰 사이즈의 커널로 합성곱 연산을 하면 많은 연산량과 파라미터 필요

keras에서 MLP 구현

- MLP 구현
  1. 데이터 다운로드 : 분류할 이미지 데이터를 다운로드
  2. sequential model 구축 : sequential 클래스를 선언하고 층을 순서대로 추가하여 MLP 구현
  3. 모델 분석 : modle.summary() 활용하여 분석
  4. 모델 컴파일 : 비용함수, 최적화 방법을 설정하여 학습 진행 방법 설정
  5. 모델 학습 : fit() 메서드를 통해 학습
  -> 훈련 샘플, 레이블 에폭수, 검증세플, 레이블을 입력할 수 있음
  6. 학습 시각화 : fit() 메서드에서 리턴받은 history 사용
  7. 테스트 데이터로 평가 : evalute() 메서드 사용
  8. 예측 : predict() 메서드 사용(새로운 샘플 예측, 클래스별 확률 확인)