**딥러닝을 활용한 이미지처리\_CNN**

학사 대부분의 딥러닝 취업처 : 데이터 전처리, 필터링

실제 딥러닝/머신러닝 : 통계학과(석.박사)가 분석

학사는 java, python을 가지고 크롤링, 파싱, 필터링 (pandas / numpy) 데이터 수집/저장

경진대회 많다. 많이 다녀보자.

딥러닝은 답이 없다. (예측)

\* 조사 분석 + 마케팅

- 닥터앤서(한국형 왓슨)

R -> Python 으로 분위기가 넘어가는 중 (빠른 속도 / C, C++ lib 연동이 쉽다.)

R : 통계학자, 전문 분석가 (직관적이다.) // 분석

Python : 개발자 // 프로그램 능력 + 분석

- Flask (웹) + mongoDB (대용량 데이터) + Keras (분석 라이브러리)

Tensorflow : 32bit 지원 안함 (windows 에서 설치 어려움).

- Python과 Anaconda version 충돌이 빈번하다. Python부터 설치하지 말고 Anaconda로 한번에 설치하자.

<관련 개발도구>

jupyter : 분석가

pycharm (유료)

vs code

eclipse

<머신러닝 학습방법>

-지도학습 : 문제도 알고 답도 앎. 데이터 유형에 따라 알고리즘 선택.

- 회귀 : 연속적 데이터 (평균값 중요)

// 수학을 몇시간 공부하면 점수가 몇점 오를까? (수치)

- 분류 : 불연속적인 데이터 (갯수-count-가 중요)

// 자격증을 몇시간 공부하면 합격할 수 있을까? (확률)

-비지도학습 : 문제는 아는데 답을 모름 (학습 전단계. 뭘 학습해야할지 모름)

- 'K-means'(가장 유명)

-강화학습 : 상과 벌을 주면서 학습(가중치)

\*데이터가 많아야 한다.

DeepLearning : 인간의 신경세포가 학습하는 형태를 참조. 신경망.

<명령어>

[Anaconda Prompt - 관리자 권한으로 실행]

python --version

conda -V

pip install tensorflow

pip install keras

import keras

keras.\_\_version\_\_

\*.ipynb : jupyter notebook 확장자

vs code - anaconda extension pack 설치

numpy library로 ndarray를 사용해야 행렬 연산 가능

- ndarray : numpy가 가진 array를 사용한 행렬

a=[[1,2,3],[4,5,6]] // 연산이 어렵다.

a=np.array([[1,2,3],[4,5,6]])

- ndarray를 복사할 때는 .copy() 사용!

b = a

b = a.copy()

- 행렬의 크기 변경

.reshape(n, m) : n \* m 행렬로 변경

- np.save : ndarray타입을 파일에 저장 \*.npy (확장자 변경 불가)

matplotlib : 분석 단계에서의 그래프. 이쁘지 않다.

(시각화로 사용하지 않는다. 시각화는 다른 lib 사용 - D3.js or cmon?)

matplotlib tutorails>

<https://matplotlib.org/3.1.0/tutorials/index.html>

\* 꼭 알고 넘어가야 하는 함수들

- 지수함수 :

- 시그모이드 함수 :

= x값이 커지면 1, x값이 작아지면 0에 한없이 가까워진다.

- 선형 회귀분석 : 평균값 (직선이나 로그함수 그래프)

- 로지스틱 회귀분석 : 특정 상황(s자 그래프) 0 or 1 -이진분류 / '분류(둘 이상)' -> 시그모이드 함수 -다중분류

- 음에서 양의 실수를 0에서 1까지의 사이로 변환. 확률로

\* 시그모이드 함수는 완만하기 때문에, 일정 확률 이상이 되면 경사도가 가파라지는 ReLU 함수 사용한다.

- 정밀도가 좋아짐 (판단하기 쉬워짐)

- 소프트맥스 함수 : 3개 이상의 수.

- 가우스 함수 : 곡선을 근사하는 기저 함수

- vector (방향성이 있는 좌표)

-> 내적 계산하는 것이 for문보다 빠르다.

2

)

ex) 키 + 몸무게 산포도

1) 데이터 전처리 (예외적인 값들 제거)

2) 오차 평균이 작은 값을 찾는다. (최소제곱 오차법 사용)

선형회귀분석

분석 : cost (비용) 계산할 때 수학이 필요하다.

미분 (함수의 기울기 도출)

- 함수의 미분은 기울기를 나타낸다.

- 편미분 : 필요없는 변수를 상수화.

머신러닝

-> 명시적인 프로그래밍 없이도 컴퓨터가 스스로 학습할 수 있도록 하는 학문

AI(Artifical Intelligence)

ML(Machine Learning)

DL(Deep Learning) - '신경망'(지도학습)

사람 : input -> f(x) -> output

컴퓨터 : input -> 학습 -> output // 인공지능이 스스로 학습, 결과를 도출. 우리는 학습 방법(알고리즘)만 알려주면 된다.

- 회귀 / 분류 를 구분하는 법만 알면, 알고리즘은 통계학자가 만들어 놨다.

- 데이터 유형을 찾아야 한다. 평균(회귀) / 갯수(분류)

- 알고리즘은 정답이 없다.

- 데이터들을 -> 학습(분석/분류)해서 -> 모델(결과물)을 뽑아낸다.

특징(feature) -> 특성, 변수

\*실무에서는 지도학습에서의 적절한 특성을 찾아내기 위한 전처리 방법으로 비지도 학습을 쓰기도 한다.

k-means

연관규칙(apriori)

산포도를 이용하여 주어진 데이터를 가지고 직선 긋기

\*정규 분표로 이루어지지 않은 데이터셋은 에측 불가 => 거의 대부분은 정규분표로 이루어져 있음

1. 데이터 구분(범주형이냐 비범주형이냐?)

- 기본적인 데이터에 대한 조사, 평균, 최대/최소값, box plot

- 이상치(잘못되거나 범위가 지나치게 벗어나서 평균값에 지대한 영향을 미치는 요소)는 버리거나 중위값 등으로 대체

- 평균 : 왜곡의 가능성이 있다.(지나치게 높거나 낮은 값에 영향을 받음)

- 중위수 : 중간에 위치한 값 [5 3 6 4 7 8 100 12] => 정렬 후 중간 값. (6.5)

2. 학습 데이터랑 테스트 데이터를 나눈다. (7:3 or 6:4)

- 과소/과대 적합은 좋지 않다. 특정 데이터셋에서만 정확도가 높아지면 문제가 생김.

3. 학습 알고리즘을 사용해서 학습을 한다. (모델 구성)

4. 정확도 체크를 한다.

5. 예측을 한다.

예측이 가능한 이유는 데이터가 쌓이기 때문

오버피팅 문제 : 학습 데이터와 테스트 데이터의 비율 중요!

- 적중률이 100%인 것이 오히려 더 위험할 수 있다.

- 학습을 많이 할수록, 데이터가 많을 수록 오버피팅 문제가 점점 사라진다.

- 여러개를 가지고 각각의 모델들을 평가분석하며 작업을 진행해야 한다.

<Tensorflow>

- 기계 학습과 딥러닝을 위해 만든 오픈소스. 그래프!

- Tensor가 flow한다.

\* Tensorflow보다는 keras를 사용하는 것이 현재 추세.

<분류>

~목표 데이터는 '클래스' 입니다. = 명목형 자료. 범주형 자료. category. factor 타입(R). class(Python)

ex) 선호도 : 1, 2, 3, 4, 5 // 이외의 값은 금지. 순서의 의미가 없음.

x : 데이터, 수치, 이미지, 음성 ...

y : 범주형 데이터 타입이어야 한다.

- 분류는 확률로 개념을 나타내어야 한다.

ex) 강낭콩(0.8) 완두콩(0.1) 서리태(0.1)

-> 강낭콩[1, 0, 0] : one hot encoding

\* 기본적인 딥러닝도 분류!

Deep Learning : 뉴런 모델. 심층 신경망 DNN(Deep Neural Network)

Perceptron : 어느정도의 자극까지는 무시, 이후는 출력 (역치). 0 or 1 // sigmoid 함수

- 상수(더미값)를 이용하여 자극을 제어 (상수도 입력으로 생각!)

= 가중치를 상수로 사용하기 위해, 자극을 항상 1로 고정

- 자극 3개(x) 에 대해...

y = x0w0 + x1w1 + x2w2 + b[x3w3(w3을 상수로 사용하기 위해 x3 = 1 // = b(bias) 편향)]

// 학습을 통해 w0, w1, w2를 얻기 원하는 것!

- weight가 클수록 해당 신호가 중요. 처음엔 weight를 임의로 설정, 분류가 가능할 때까지 weight를 찾는 것.

높은 확률을 찾는 것 = 학습

- Perceptron은 선형에만. (비선형 사용 불가)

\* 과적합이 되면, 다른 데이터셋이 입력되었을 때 맞지 않을 확률이 높음.

- overfitting : 과대적합. 학습한 모델에만 잘 맞는 부작용

- underfitting : 과소적합. 학습이 제대로 되지 않음 (사용할 수 없음)

multi-layer perceptron : xor, flipflop 형태로..

활성화 함수

- 계단 함수

- 시그모이드 함수

- ReLU 함수 (최근엔 거의 ReLU 사용)

\* 선형 함수를 사용하면 학습이 되지 않는다! (학습의 의미가 없다)

p.302 가중치를 변경, 새로 시작 -> 오차의 역전파 (신경망에서 해준다.)

- 한번 완료하는 것 : 1 epoch 주기 (epoch가 증가할수록 가중치 업데이트)

손실 함수 (Loss Function)

- 평균제곱오차 MeanSquaredError : 예측값과 실측값을 제곱하여 평균

- 교차 엔트로피 함수 : 신경망에서는 교차 엔트로피 함수가 적합.

<지도학습 방법>

1) 데이터 탐색, 전처리

2) 데이터셋 구성(훈련용, 테스트용)

3) 알고리즘 선택, 모델

4) 훈련용 데이터셋으로 모델 학습

5) 모델 검증 진행

6) 모델 배치

\* training data 그래프가 최적 그래프, test data 그래프가 training data 그래프에 근접할수록 최적화가 잘 된 학습모델.

<CNN>

합성곱 신경망(Convolutional Neural Network)

- 기존 인공신경망 : 2차원을 1차원으로 변환하여 사용. cnn은 2차원 그대로 사용.