**2강**

이미지 classification이 가능하면 Detection, Segmentation, Captioning 수월해짐.

이미지: height \* width \* channel

보는 시각에 따라 달라지거나,

Illumination(조명 차이), Deformation(형태의 변형), Occlusion(가려진, 은닉), Background clutter(배경하고 객체가 너무 비슷), Intraclass variation(같은 고양이에서 더 세분화된 구분의 경우)

의 다양한 챌린지가 존재

지금하려는 것은 Data-driven approach, 데이터 기반이다.

L1 Distance = 맨허튼 Distance

L2 Distance = Euclidean Distance

어떤 Distance를 쓸지는 사용자 선택

NeareastNeighbor는 모든 train image와 L1 Distance가 가장 적은 것으로 예측하는 것임. = 약간 노가다 느낌 = 비논리적, 허점 많음.

또한, trainset 크기에 따라 분류 속도가 엄청 느려짐, = linear하게 증가한다(일정하게 비율 증가, 직선으로 증가, 선형으로 증가), train 속도는 메모리에 할당하고 끝남.

k-Neareast Neighbor: 가장 비슷한 k개의 이미지 찾고 그 중 많이 뽑힌 레이블로 결정.

만약 trainset이 testset으로 쓰이면 NN의 경우는 100%의 정확도, k-NN은 뽑힌 것에 따라 달라진다.

어떤 Distance? k는 몇? = 하이퍼 파라미터 = 문제에 따라 항상 다르게 설정, 주어진 환경을 잘 파악하고, 실험해보고 경험해보는 것이 중요.

하지만 test data에 대해 하이퍼 파라미터를 맞추는 것은 아님, fold로 나누고 valid data를 갖춘 후 파라미터를 조정, 이 tuning 시 cross-validation이 자주 쓰임. 특히, train data가 적은 경우 더 효과적.

현실에서 NN은 절대 사용안됨.

1. 테스트 속도 느림
2. 이미지 거리 유사도가, 이미지 변형에 매우 둔감하다. (옮겨짐, 가려짐, 조명 차이), 에도 불구하고 3개의 경우 원본 이미지와 같은 거리를 나타낸다면 동일하게 판단된다.

**Linear Classfication**

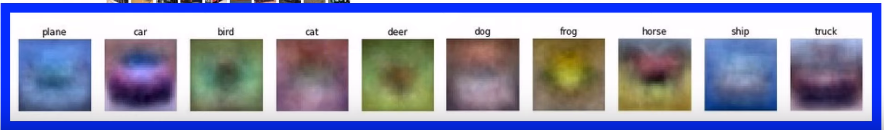
파라미터 기반의 접근 방식, NN 에서는 파라미터가 존재하지 않았음.

말로 풀어서 설명하면, 이미지 내의 모든 픽셀에 대해 가중치를 곱하여 처리를 한 결과의 합이다.

= just a weighted sum of all the pixel values in the image

또한, 각각 다른 공간적 위치에 있는 컬러들을 카운팅한 것이다.

= classifer의 weight를 시각화 했을 때 다른 색깔을 보여서 이렇게 말함



또한, 샘플의 특성도 많이 담김

frog의 경우 노란색 배경이 많은데, 노란색 차가 데이터로 올 경우 잘못 예측할 수도 있다. = 단순 Linear Classification의 한계 = Neural Net(이제부터 NN은 Neareast Neighbor 아니다.)

선형으로 단지 linear하게 이터레이션 마다 파라미터를 바꾸어 선을 바꾸고 데이터를 classification 한다.

선형 분류의 한계 = 다양하게 학습을 못함?(내 생각)

컬러에 따라 다르게 예측할 수도 있다.

흑백의 경우 텍스쳐와 디테일에 따라 구분하는데 여기에 약한 모습을 보인다.

텍스쳐가 다르지만 색상이 다른 경우