

Image Classification

Image Classification

: 미리 지정해둔 카테고리들중에 인풋 이미지의 라벨을 예측하는 작업
사람과 달리 컴퓨터가 이미지를 인식하는 방법은 $H \times W \times C$ 의 3차원 행렬

Semantic Gap

- 사람이 이해하는 고수준의 개념이나 의미와 컴퓨터가 이해하고 처리할 수 있는 저수준의 데이터나 명령어 사이의 괴리

이미지 분류에서의 Challenges

Viewpoint variation

- 물체를 담는 각도 차이

Illumination

- 장면의 조도 차이

Deformation

- 물체가 다른 형태로 존재할 때

Occlusion

- 물체의 일부만 비쳐질 때

Background clutter

- 배경과 물체가 구분하기 어려울 때
- 같은 라벨의 물체에도 차이가 존재할 때

Image classifier

단순히 하드코딩을 통해서 이미지 분류를 완벽하게 할 수 있는 명확한 알고리즘이 없음

이전의 접근 방식: 이미지의 edges를 사용해서 corners를 찾기
→ 까다로움. 매번 학습해야하는 번거로움. 고비용

데이터 기반 접근

1. 이미지와 라벨 데이터를 수집
 2. 머신러닝을 사용해 분류기를 학습
 3. 새로운 이미지로 분류기 성능 평가
- ⇒ 학습, 예측의 2가지 함수 사용
- 학습(train): 이미지와 라벨을 입력받아 학습

```
def train(images, labels):
    # Machine learning
    return model
```

- 예측(predict): 이미지 입력받아 라벨 예측

```
def predict(model, test_images):
    # Use model to predict labels
```

```
    return test_labels
```

Nearest Neighbor

Nearest Neighbor 알고리즘 code

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.Ytr = Y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the rearest traing image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

- train: 모든 데이터와 라벨을 기억
- predict: 새로운 이미지를 가장 비슷한 훈련이미지의 라벨로 예측
- Distance Metric
 - L1 distance
- Nearest Neighbor 알고리즘의 계산비용
 - train O(1), predict O(N)
 - 일반적으로 좋은 분류기를 만들기 위해 훈련비용을 늘리고 예측비용을 줄여 속도 확보. Nearest Neighbor은 예측비용이 높아 비효율적
- Nearest Neighbor 알고리즘 시각화

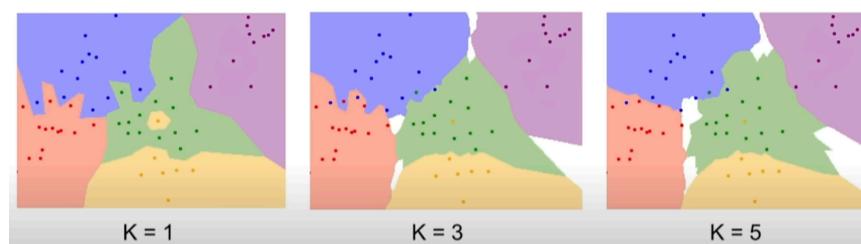
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



노이즈 때문에 각 클래스의 결정경계가 매끄럽게 만들어지지 못하는 모습

K-Nearest Neighbors

- 데이터로부터 거리가 가까운 K개의 데이터의 레이블을 참조해서 (다수결의 방식으로) 결정경계를 형성하는 알고리즘
 - 이미지 분류에서 단순히 Nearest Neighbor을 사용했을 때보다 더 나은 성능을 보임
⇒ 더 강건한(Robust) 알고리즘
- 시각화



Nearest Neighbor(K=1)보다 K에 비례해 결정경계가 매끄럽게 생성

Distance Metric

distance metric을 선택하는 것은 공간에서 기하학이나 위상학에 대한 다른 가정을 만들기에 과제에 적절한 metric을 선택하는 것이 중요
⇒ 즉, “
데이터 간 차이를 어떻게 정의할 것”인가 정하는 일

L1 (Manhattan) distance

- 각 축의 차이의 절댓값의 합

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

- 회전 민감(rotation sensitive)
: 좌표계를 회전시키면 두 점 사이의 거리의 크기가 바뀔 수 있음
 - 마름모꼴로 동거리선을 형성하므로, 특정 방향으로 더 예민하게 반응

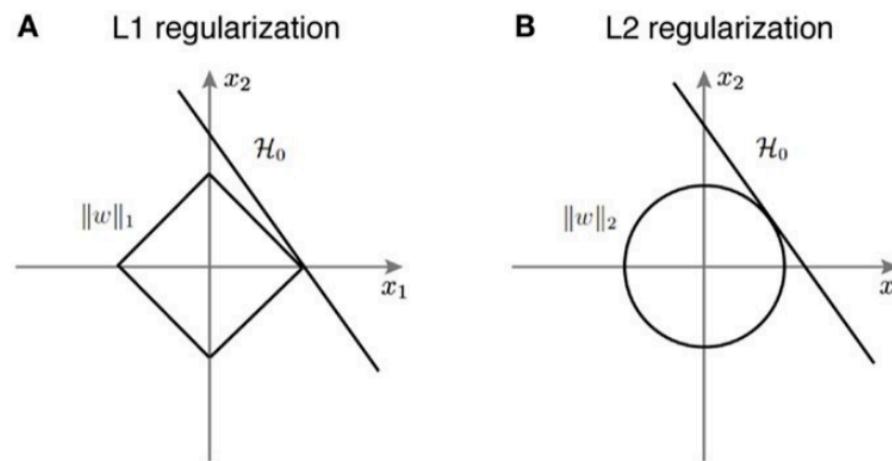
L2 (Euclidean) distance

- 두 점 사이의 직선거리

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

- 회전 불변(rotation invariant)
 - : 좌표계를 회전시켜도 두 점 사이의 거리가 변하지 않음
 - 원으로 표현되기에 모든 방향에 대해 동일한 거리 감각 유지

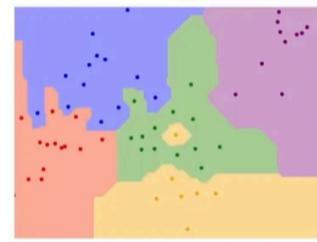
▼ L1, L2 distance



▼ 이전 Nearest Neighbor 알고리즘에서 L1과 L2의 차이 시작화

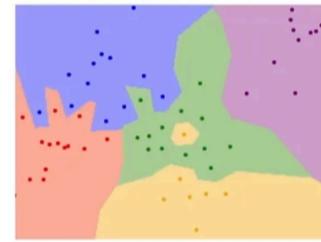
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



L1에 비해 L2가 더 부드러운 곡선형의 결정경계 생성

다른 distance metrics를 사용하면 (벡터나 이미지 이외에도) 더 많은 유형의 데이터에 KNN classifier를 일반화할 수 있다.

ex. 텍스트 데이터

L1, L2 정리 (+ L1, L2 Regularization)

Hyperparameters

알고리즘을 설정할 때 선택하는 파라미터. 학습 전 지정하고 훈련시 고정
→ K-NN 알고리즘에서의 K, distance metric 등

- 직면한 과제에 따라 hyperparameter 설정이 달라짐
- 일반적으로 일정한 범위 내에 다양한 상수로 테스트해보고 성능이 좋은 방향으로 설정

Hyperparameter 선택

idea #1. 훈련 데이터

- 전체 데이터에서 성능이 가장 나오는 hyperparameter 선택
⇒ 새로운 데이터에 일반화 가능성이 낮음

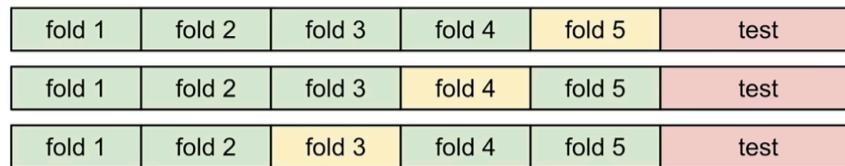
idea #2. 훈련/테스트 데이터

- 훈련 데이터에서 학습시켜 테스트 데이터에서 성능이 가장 나오는 hyperparameter 선택
⇒ 새로운 데이터에서도 성능이 잘 나올지 알 수 있음

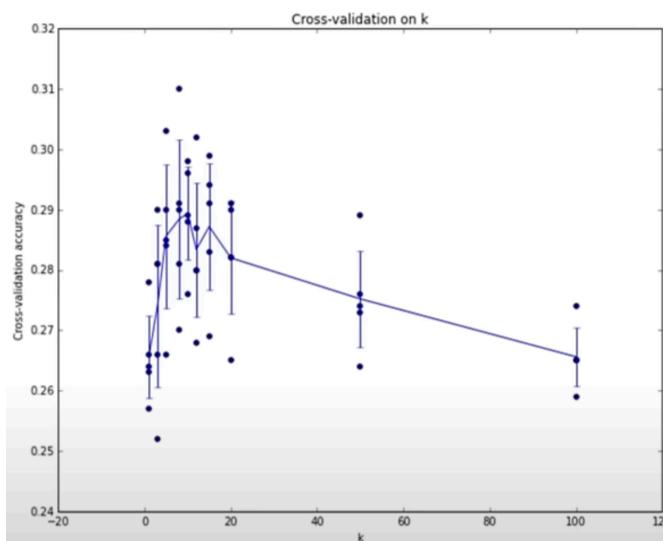
idea #3. 훈련/검증/테스트 데이터

- 훈련 데이터에서 학습시켜 검증 데이터으로 하이퍼파라미터를 최적화한 후 테스트 데이터에서 가장 성능이 좋은 것 선택
⇒ #1,2에 비해서 좋은 방법

idea #4. 교차검정



- 데이터를 다수의 fold로 나눠 각 fold를 검증 데이터로 시도해서 모든 결과를 평균
⇒ 더 robust한 방법이기에 선택한 hyperparameter에 더 높은 신뢰도를 보여줄 수 있음
- 데이터 규모가 적을 때 유용하지만 (높은 비용 문제로) 자주 사용하지 않음
- 교차검정 결과 예시



평균으로 계산하기에 각 hyperparameter 성능지표의 분산 또한 확인 가능

하지만, K-Nearest Neighbor은 이미지 데이터에 전혀 사용되지 않음

1. 테스트 시간이 너무 느리며
2. 픽셀에서 vectorial distance function은 유용하지 않기 때문



같은 사진에서 다양한 효과를 주더라도 결국 L2 distance는 같음(같은 이미지에서 변형한 모습이기 때문). 즉 L2는 사진의 유사성을 판단하기 적절치 않음

Curse of Dimension

K-NN 알고리즘은 매니폴드 가정을 하지 않기에 훈련 포인트들의 dense 샘플이 있어야 제대로 작동

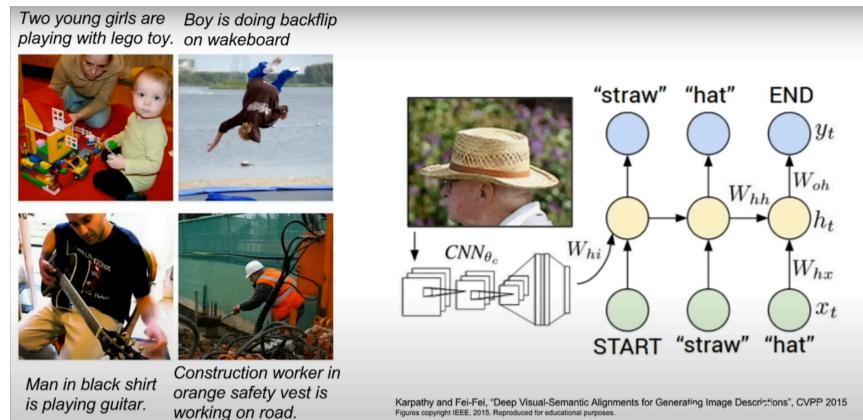
이 방식은 이미지에서 다루는 차원의 수가 많아질수록 계산량이 지수함수대로 기하급수적으로 늘어남
→ 1차원일 때

n , 2차원일 때 n^2 , 3차원일 때 n^3 , ..., d 차원일 때 n^d

⇒ 고비용

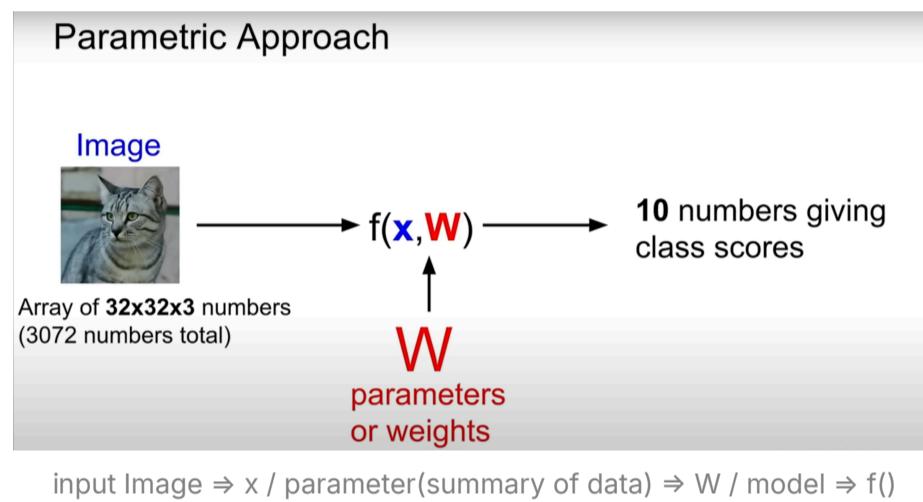
Linear Classification

Linear classifier은 Neural Network의 근간



이미지를 인풋, 이미지 설명을 라벨로 넣고 학습. 오른쪽과 같이 이미지 특성을 추출하는 CNN과 언어를 담당하는 RNN을 결합해서 모델 학습 (Multi-modal)

Parametric Approach: Linear Classification

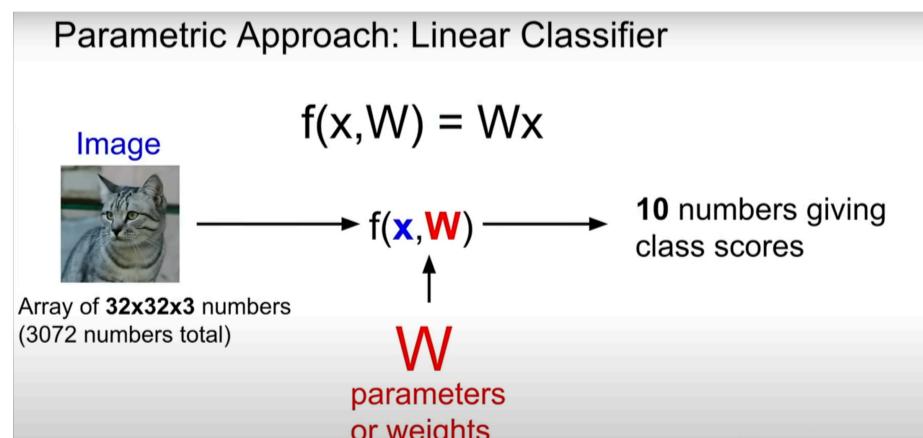


Parametric Approach

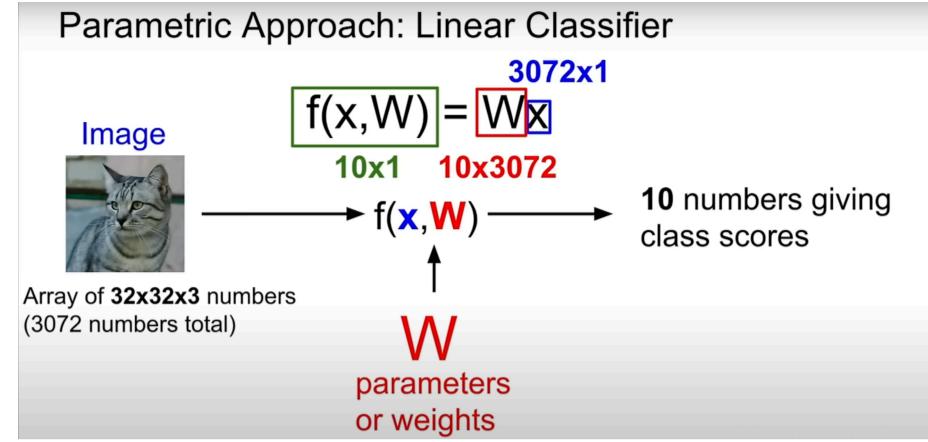
: parameter을 사용해서 훈련 데이터가 가진 지식과 정보를 요약하는 접근방식

Linear Classification은 Parametric Approach 사용

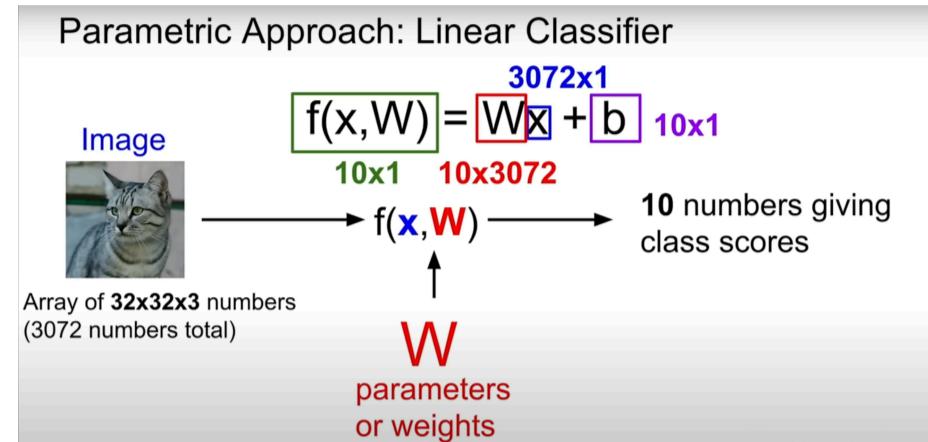
- K-NN: 모든 훈련 데이터를 기억해서 테스트 데이터로 예측
- Linear Classification: 훈련 데이터의 정보를 요약해서 그 지식을 모두 W 에 저장해서 그 파라미터 W 를 토대로 이후 테스트 데이터를 예측
→ (특히 예측할 때) 훨씬 경제적이기에 핸드폰과 같은 작은 기기에서도 사용할 수 있도록 함



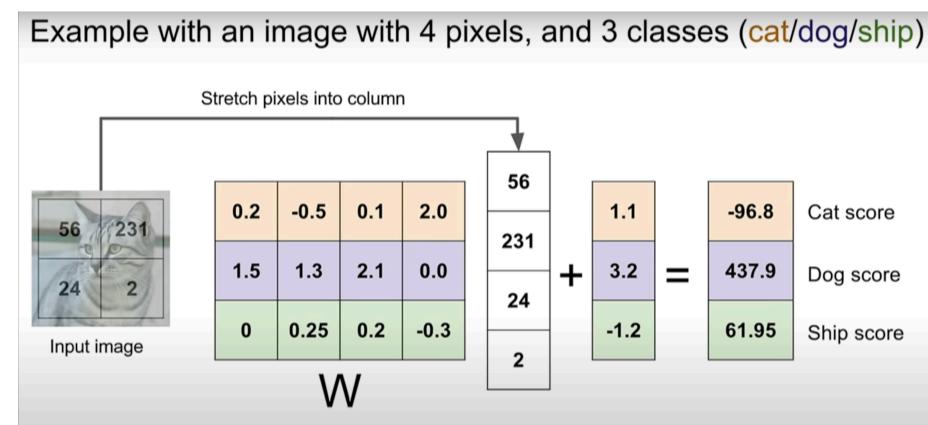
Linear Classifier은 단순히 인풋 x 와 파라미터 W 를 multiply(곱, 내적)하는 방식으로 작동 $\Rightarrow f(x, W) = Wx$



이미지의 차원이 32*32*3일 때, 이를 하나의 dense 레이어로 나타내고 10개의 라벨을 지정해 모델을 만들어 각각 라벨에 score이 나오도록 하면, 위와 같은 계산이 나옴



Bias항(b)은 훈련데이터와 상호작용하지 않지만, 데이터와는 독립적인 preference를 제공. 예를 들어 데이터 라벨 불균형이 있을 때, bias항이 불균형을 대변해 빈도 수가 높은 라벨에 반응할 가능성을 높임



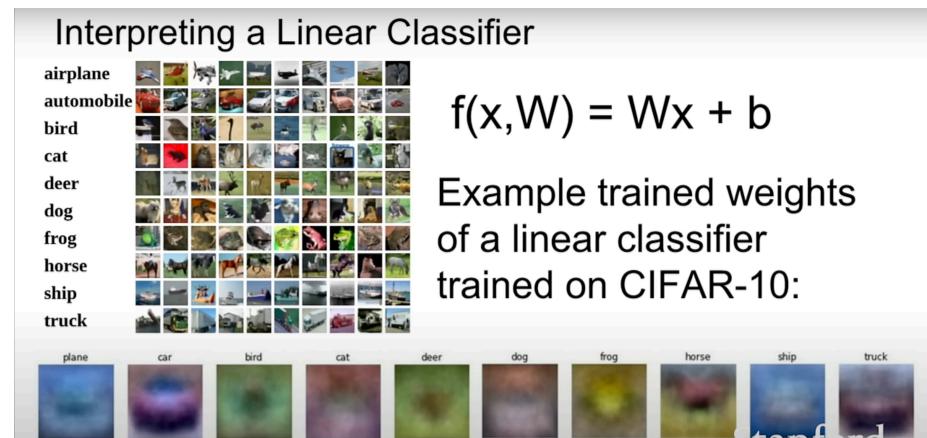
2*2 Test data 이미지를 예측하는 방법. 학습으로 template을 통해 라벨 예측

Linear Classifier의 한계

1. template을 시각화한 모습으로 사물을 확실하게 인지할 수 없음
2. 한 카테고리당 하나의 template만 존재함
→ 각도 등의 한 카테고리의 다양성이 하나의 template으로 반영될 수 없음
3. 단순히 Linear(선형)으로 분류할 수 없는 데이터 패턴이 무수함
→ 홀수와 짝수를 나누는 것
→ 멀티모달 상황(한 클래스가 공간의 다른 region에 나타나는 경우)

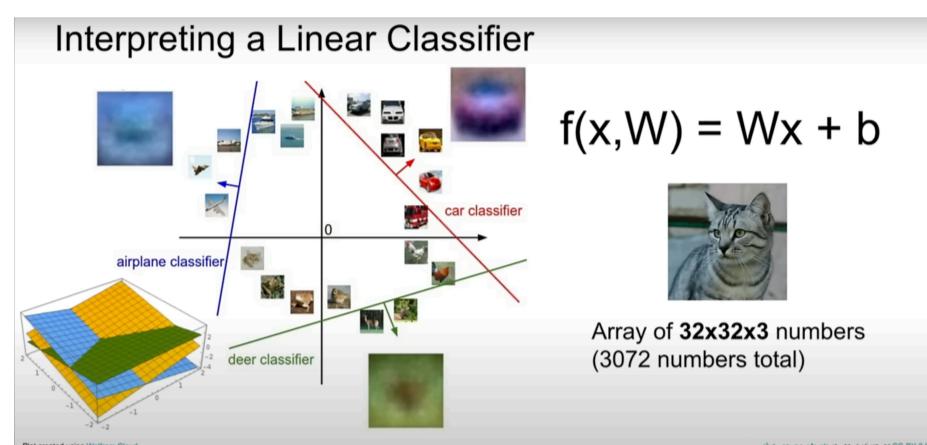
▼ Linear Classifier의 한계 이해자료

1, 2)

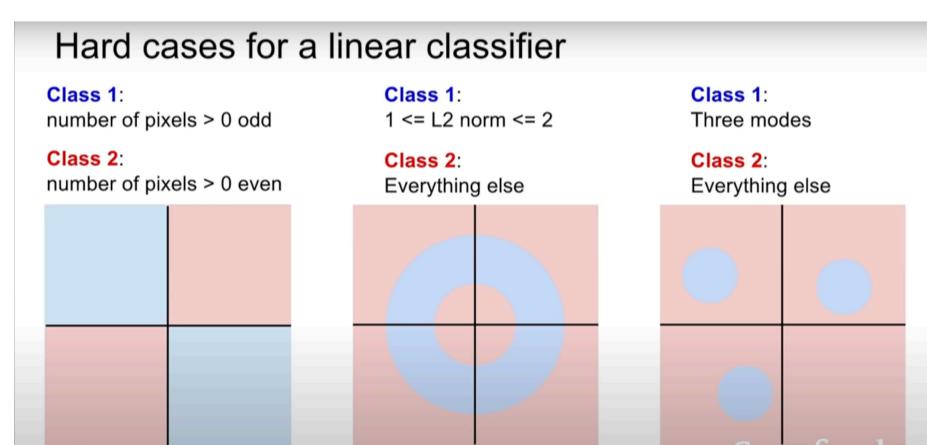


하단의 이미지가 각 라벨별 template을 시각화한 것. 각 라벨당 하나의 template 보유. 각 template이 라벨을 확실하게 대표한다고 인지하기 힘듦

3)



모든 이미지는 매우 고차원의 좌표에서 한 점으로 나타낼 수 있다고 가정함. 여기서 Linear classifier은 한 카테고리로 다른 것과 나누는 '선형'의 결정경계를 그림



고차원의 데이터를 2차원으로 나타냈을 때 나타난 다양한 패턴을 매우 단순한 '선형' 결정경계로 나타내는데 빈번히 실패. 위가 대표적인 예