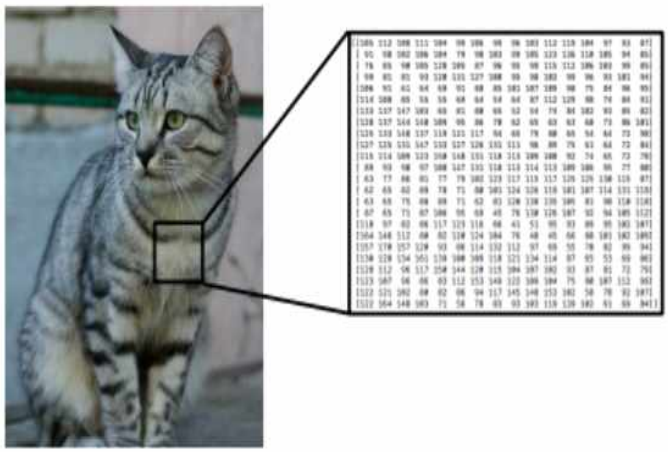
cs231n lecture 2

1. Image Classification

- 정의 : 입력 이미지가 이미 정해져 있는 category에서 어떤 category에 속할지 고르는 것

- computer가 image classification 이 어려운 이유:

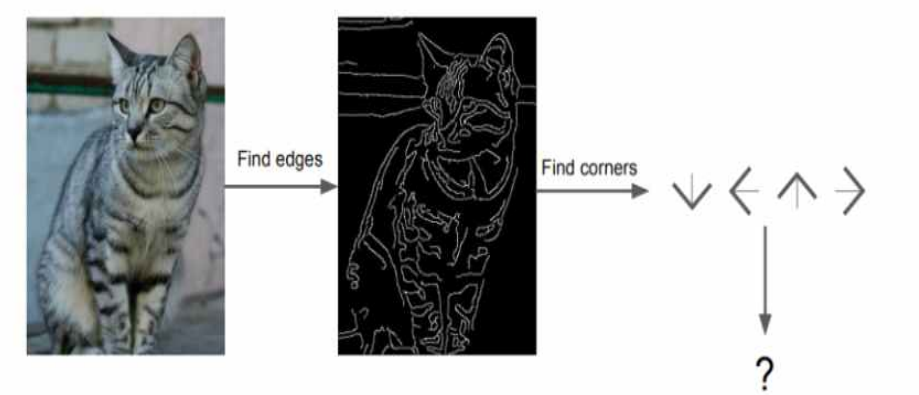
  * 사람은 고양이 이미지를 있는 그대로 받아들이지만
    computer 는 이미지를 숫자의 집합 (아래의 우측 사진)으로 봄



  * viewpoint variation / illumination / deformation / occlusion / background clutter / intraclass variation으로 인해
    image classification이 어려움

- image classification 의 방식으로 2가지를 들 수 있음

* image 의 feature을 추출하고 이용하여 규칙을 만드는 방식



변화에 강하지 않아 결과가 잘 나오지 않는다.
다른 객체를 인식할 때 그 클래스에 맞는 집합을 또 따로 하나하나 만들어야 해서 비효율적

* Data-Driven Approcach (데이터 중심 접근방법)

방대한 고양이 사진, 강아지 사진을 컴퓨터에 제시하고,
Machine Learning Classifier을 학습

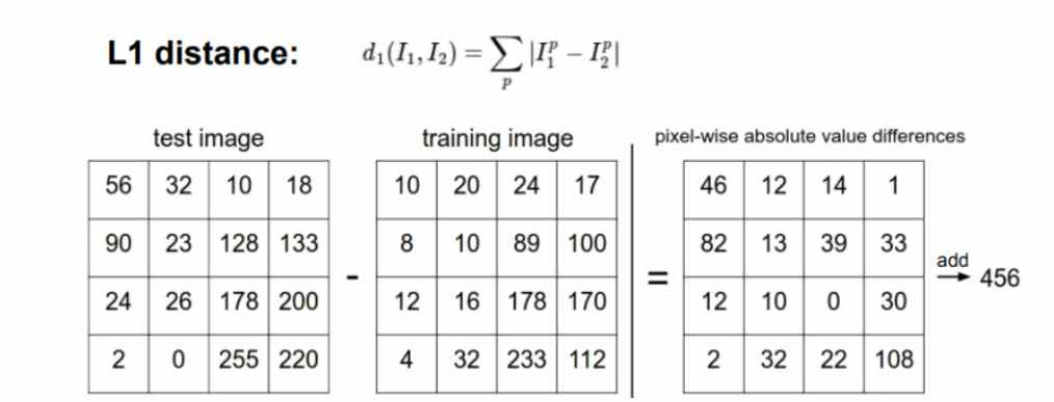
2. Nearest Neighbor Algorithm

- 총 2가지 과정을 거침

    * Train : 모든 train data를 기억
    * Predict : input 데이터를 train data와 비교하며 어떤 label 값을 가질지 예측

- Distance metric / L1 distance

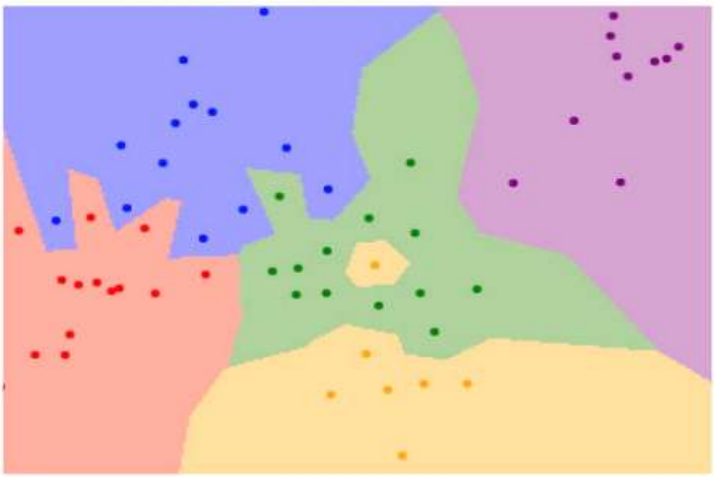
    각 픽셀값을 비교하고, 그 차이의 절댓값을 모두 합하여 하나의 지표로 설정



- 한계

```python
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 nearest training 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 보다 predict에서가 더 시간이 많이 걸림
decision boundary가 Robust 하지 않음



점은 학습데이터, 점의 색은 클래스 레이블
녹색 내부의 노란색 영역 / 초록색 영역에서 파란색 영역 침범 구간을 통해 decision boundary 가 robust 하지 않음을 알 수 있음

3. K-Nearest Neighbor Classifier

- 일반화한 KNN(k-nearest neighbors)을 도입

- 제일 가까운 이웃을 찾는 것 X.

- 가까운 이웃을 K개를 찾고, 투표를 통해 가장 많은 레이블을 예측.

- 영향을 미치는 parameter :

    * K값

    * Distance Metric(거리 척도)

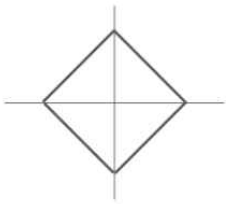
K가 증가함에 따라 점점 경계가 부드러워짐

- Distance metric / L2 distance

    * L1 distance : 특정 벡터가 개별적인 의미를 가지고 있을 때
      L2 distance : 일반적인 벡터 요소들의 의미를 모르거나
      의미가 별로 없을 때

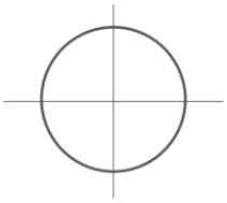
L1 (Manhattan) distance

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



L2 (Euclidean) distance

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



L1 (Manhattan) distance

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

K = 1

L2 (Euclidean) distance

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

K = 1

L1 distance : region도 좌표축에 영향

L2 distance : decision boundary가 더 부드러움

4. 하이퍼파라미터 (Hyperparameter)

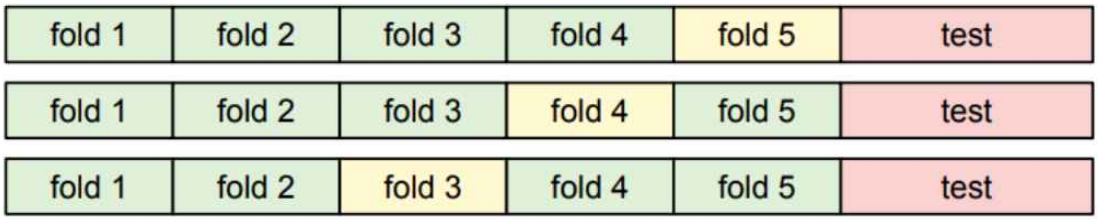
- K값, Distance Metric(거리 척도)와 같은 Parameter 가 Hyperparameter

- Hyperparameter는 학습을 하기 전 선택하고, 학습을 하는 데 영향을 미치는 parameter

- Hyperparameter를 정하는 방법 (학습데이터의 정확도와 성능을 최대화하는 방법)

    * 데이터를 train, validation, test 세 개의 셋으로 나눔
      여러 하이퍼파라미터 값들로 train 데이터를 학습
      validation set으로 검증
      validation set에서 가장 좋았던 하이퍼파라미터를 선택
      테스트 셋은 가장 좋은 clasifier로 "딱 한번만" 수행

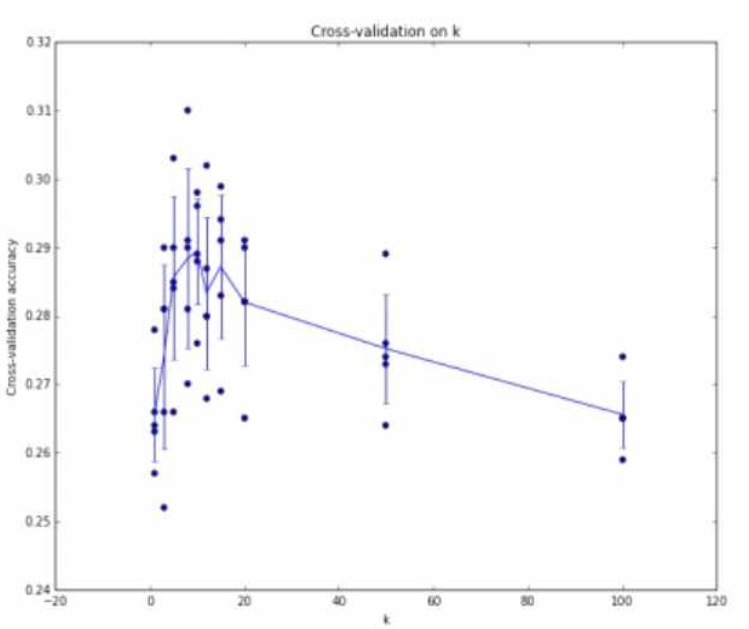
    * cross validation (교차검증)



마지막에 딱 한 번 사용할 테스트 데이터는 빼놓는다.
나머지 데이터들을 여러 부분으로 나눈다.
번갈아 validation set을 바꾸어준다.
위 그림은 5-fold Cross validation을 사용
초록색 데이터로 hyperparameter를 학습
노란색에서 평가 후 최적의 hyperparameter 결정

딥러닝은 학습 계산량이 애초에 많아서 위 방법을 많이 사용 X

- k에 따른 정확도 그래프



k마다 5번의 cross validation을 통해 알고리즘을 평가
테스트셋이 알고리즘 성능에 미치는 영향을 알아볼 때 도움이 됨
해당 하이퍼파라미터에서 성능의 분산값을 알 수 있음

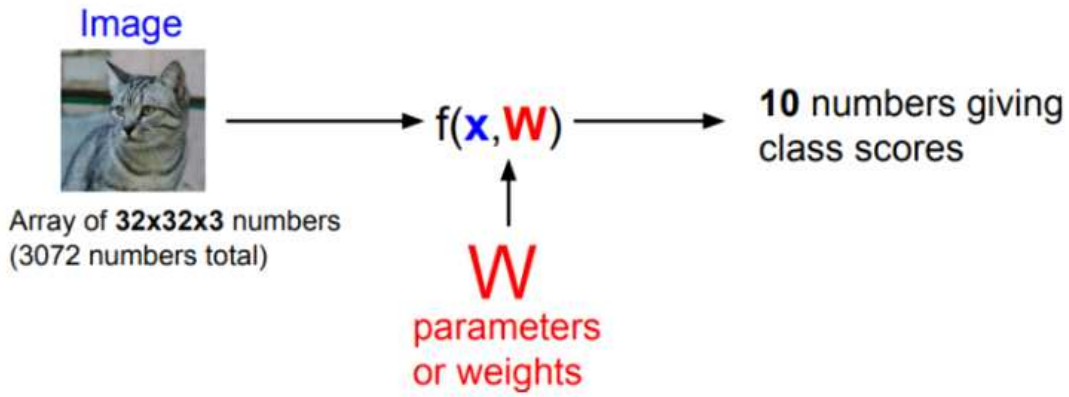
- 이미지 분류에서는 K-Nearest Neighbor Algorithm을 잘 사용하지 않음

  * test를 하는데 시간이 너무 오래 걸림

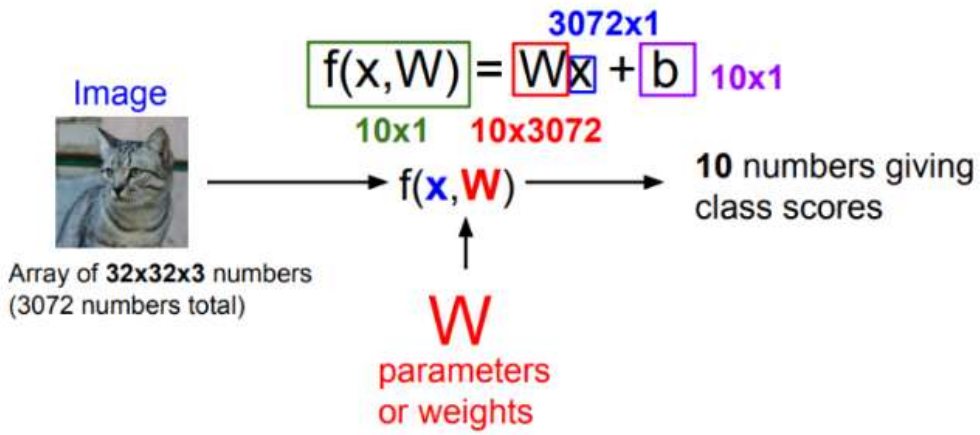
  * Distance Metric은 픽셀단위에서 유용한 정보가 아님

  * KNN이 잘 동작하기 위해서는 전체 공간을 조밀하게 커버할 만큼의 충분한 train set이 필요하며, 고차원의 이미지의 경우 모든 공간을 커버할 만큼의 데이터를 모으는 일은 현실적으로 불가능

5. Linear Classification



입력이미지: x
파라미터(가중치): W (or theta)



- 위 사진 설명

*고양이 사진 : x

*함수 f는 10개의 숫자를 출력

*10개의 숫자는 데이터셋의 각 클래스에 해당하는 스코어의 개념

* '고양이' 스코어가 높으면 '고양이'일 확률이 큰 것을 의미

- Bias

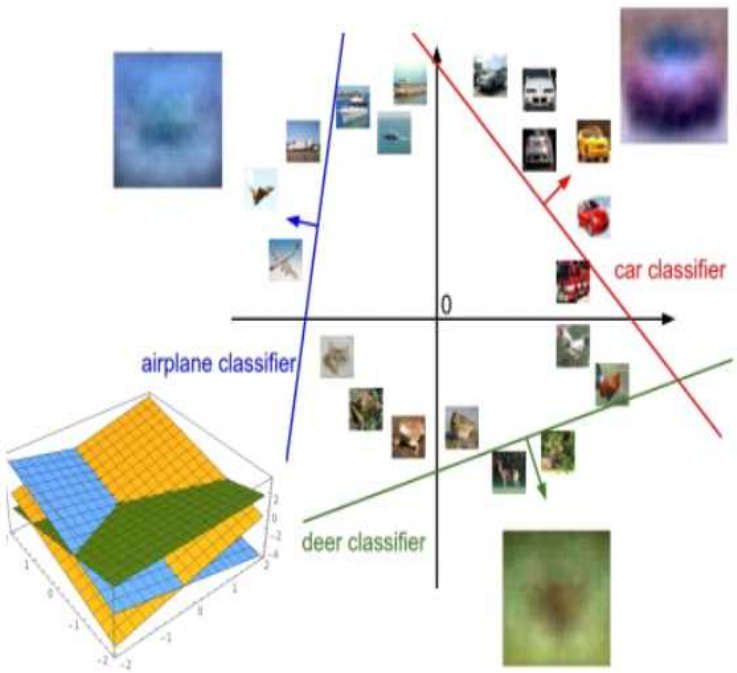
    * 데이터와 무관하게 (x와 직접 연산되지 않음) 특정 클래스에 우선권을 부여

    * 각 클래스에 scaling offsets를 추가

    * 주로 dataset이 unbalance할 때 사용한다.

    *Bias 값을 이용해 내가 원하는 선형(linear classifier)에
    잘 근사할 수 있도록 할 수 있음

- Linear classifier은 아래와 같이 각 클래스를 구분시켜주는
  선형 boundary 역할



- 영역이 반전되어 있거나, 한 클래스가 다양한 공간에 분포하면 Linear Classifier로 분류하기 어려움