

Week2_CS231n_2

Lecture2 | Image Classification pipeline

Image classification : A core task of computer vision

컴퓨터한테 고양이 그림 \Rightarrow RGB 픽셀값의 커다란 숫자, array with many different numbers

\Rightarrow How do we work on this image classification task?

(고양이 사진을 넣었을 때 고양이 라벨이 나오는 동작은 어떻게 이루어질까?)

Image Classification의 어려운 점

1. Viewpoint variable(시점의 다양성)
: 시점을 바꾸면 거대한 grid의 숫자들이 완전히 달라지지만 의미는 동일해야함.
2. Illumination : 밝기가 달라져도 robust하게 분류해야함.
3. Deformation : 매우 다른 모양과 자세도 같은 것으로 분류
4. Occlusion : 물체의 일부만 보여도 알아볼 수 있어야한다.
(ex, 고양이의 얼굴일부나 꼬리만 나와도 고양이로 분류)
5. Background Clutter : 뒷 배경이랑 물체를 구분하기 어려울 경우
6. Intraclass variation : 같은 범주 내에서의 다양성에 대한 구분

\Rightarrow Image classifier는 단순히 숫자를 정렬하는 알고리즘과는 다르게 이미지를 분류하는 완전한 hard-code 알고리즘이 존재하지 않는다. (Hard challenge)

방법1) Edge를 사용한 카테고라이징

: 우리는 이미지에서 edge가 이미지 인식에 중요한 역할을 하는 것을 안다.

\Rightarrow 이미지의 edge를 계산해서 각 corner와 boundary의 특징을 파악해 categorizing하는 것

결과) Bad

1. 매우 불안정하고
2. 한 물체에 대해 학습시켰다고 해도 다른 물체에 적용하기 어려움(반복작업)

방법2) Data-Driven Approach(데이터셋 많이 모으기)

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new image

⇒ 해당 접근법에서 Image classifier의 API가 주요 2가지 기능으로 바뀜

1) train

2) predict

First Classifier : (Simplest) Nearest Neighbor

Train이미지에서 가장 distance가 가까운 data의 label을 copy하는 방식

⇒ 이미지의 거리 비교에 Distance Metric 중 L1 distnac 사용

L1 distance :

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

Performance : Bad!

Train : O(1)

predict : O(N)

(Train 과정이 느린 것은 괜찮지만, Prediction 과정이 느린 것은 문제가 됨! 우리는 Prediction에서 빠른 성능을 원한다)

K-Nearest Neighbors

: 가장 가까운 neighbor의 라벨을 카피하는 대신, K개의 가장 가까운 점들로부터 label을 vote한다.



⇒ K값이 커질수록 각 class(label)의 경계가 smoothe 해지면서 안정(Robust to noise)되는 것을 확인할 수 있음

K-Nearest Neighbors : Distance Metric 로 2가지 사용가능

1. L1 distance
2. L2 distance

$$d_1(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

Distance Metric 선택 기준

: 상황에 따라 적절한 Distance Metric을 선택해야 함

- L1의 경우 coordinate frame이 바뀔 경우 distance가 바뀌지만, L2는 바뀌지 않는다.
- 각 feature값의 정확한 의미를 파악할 수 있는 경우는 L1이 좋고, 파악하기 어려운 경우는 L2가 더 좋다. ⇒ (이 부분 더 공부해보기)

Hyperparameter

- Hyperparameter : 데이터가 아닌 알고리즘에 대한 선택
(K, distance : These are hyperparameters)

⇒ K값을 선택하고, 적절한 Distance Metric 선택하는 것은 어떻게?

Setting Hyperparameters

Idea1) : Choose hyperparameters that work best on the data

⇒ terrible

(Why? $K = 1$ always works perfectly on training data)

Idea2 : Split data into train and test, choose hyperparameters that work best on test data

⇒ Bad : No idea how algorithm will perform on new data

Idea3 : Split data into train, val, and test; choose hyperparameters on val and evaluate on test ⇒ better

Idea4 : Cross-Validation

데이터를 여러 개의 folds로 나누고 각 fold에 validate한지 돌아가면서 점검

⇒ Computation이 많이 필요한 딥러닝에서는 현실적인 방법은 아님

(내 질문 : Cross Validation쓰면 folds의 개수도 새로운 hyperparameter가 되는건가? 아니면 fold-5가 표준?)

K-Nearest Neighbor의 문제점(이미지분류에 사용하지 않는 이유)

1. Very slow at test time
2. Distance metrics on pixels are not informative
3. Curse of dimensionality (exponential growth): 공간을 밀도 있게 채우기 위한 train data의 크기가 기하급수적으로 증가함

K-Nearest Neighbors Summary

- 이미지 분류에서 우리는 training set에서 시작해서 test set의 label을 predict해야함.
- K-Nearest Neighbors 분류기는 가장 가까운 training example의 라벨을 predict함
- Distance metric과 K는 hyperparameter
- Hyperparameter의 결정은 validation set에서 ⇒ test set의 시행은 가장 마지막에만!

Linear Classification

Linear Classifier들을 쌓아서 Neural Network 만듦 \Rightarrow CNN

Linear Classification 은 parametric model의 가장 심플한 버전 중 하나

Parametric model에서 Linear classifier는 두 개의 parameter를 가지고 있음

$f(x, W) \Rightarrow 10$ numbers giving class scores (CIFAR-10 기준)

x : input data

W : Weight

ex) $32 * 32 * 3$ 의 이미지

$f(x, W) = Wx$: $x : 3072 * 1$, $W : 10 * 3072 \Rightarrow$ output은 $10 * 1$ 로 10개의 label에 대한 score가 나옴

bias parameter가 추가된 경우

$f(x, W) = Wx + b$: $b = 10 * 1$

이미지 pixel을 column 벡터로 바꿔서 (Stretch pixels into column) Weight와 bias와 연산

Linear classifier의 한계 : 각 class(category)에 한 template에 대해서만 train

Hard cases for a linear classifier \Rightarrow linear하게 분류 못하는 케이스들

