Image Classification

Image Classification

1. Challenges
    a. Illumination
        i. 빛에 따른 물체의 변화 (명암) 때문에 같은 종류의 물체도 다르다고 판단할 수 있음
    b. Deformation
        i. 물체가 놓여져있는 방식 또는 포즈에 따라 같은 종류의 물체도 다르다고 판단할 수 있음
        ii. e.g., 서있는 고양이와 앉아있는 고양이
    c. Occlusion
        i. 물체가 일부분만 보일 때 해당 물체가 어떤 종류인지 알기 어려움
        ii. e.g., 이불에 덮여져서 꼬리만 보이는 고양이
    d. Background clutter
        i. 배경과 색이 비슷하여 물체와 배경이 잘 구분되지 않음
    e. Intraclass variation
        i. 같은 종류이지만 색, 모양 등이 다른 여러 물체가 있을 때 이들을 모두 같은 종류라고 판단하기 어려움
        ii. e.g., 나이, 종, 색 등이 다른 여러 고양이
2. Image Classifier
    a. There is no obvious way to hard-code the algorithm for recognizing an object
    b. Previous attempts
        i. Find edges and corners (boundaries)
            - not work very well
            - 새로운 종류의 물체에 대해서 분류를 진행하기 위해서는 다시 찾아야함
        ii. **Data-driven approach**
            - Step 1. Collect a dataset of images and labels

- Step 2. Use machine learning to train a classifier
- Step 3. Evaluate the classifier on new images

c. Nearest neighbor
    i. Training
        - Memorize all data and labels
    ii. Prediction
        - Predict the label of a new image as the label of the most similar training data
    iii. Distance metric
        - L1 distance
    iv. Problem
        - Fast at training (O(1)), slow for prediction (O(n))

d. KNN (k-nearest neighbor)
    i. Take majority vote from k closest points, instead of copying label from nearest neighbor
    ii. The value of K
        - Bigger K → smooth out the decision boundary
    iii. Distance metric
        - L1 distance (= Manhattan distance)
            - coordinate frame 에 영향을 받는 방식이므로 각 feature vector의 element 가 의미를 가질 때 유용함
            - e.g., feature vector = (height, weight, age)
        - L2 distance (= Euclidean distance)
            - coordinate frame 에 영향을 받지 않으므로 feature 가 어떤 의미를 갖는지 모를 때 유용함
    iv. Problem
        - Very slow at test time
        - Distance metrics on pixels are not informative

        - e.g., boxed, shifted, tinted images have same L2 distance to the original image
    - Curse of dimensionality
        - KNN이 잘 동작하기 위해서는 데이터가 dense 하게 분포해야함
        - 따라서 dimension이 증가함에 따라 데이터 포인트가 exponential 하게 증가함

e. Hyperparameters

i. Choices about the algorithm that we set rather than learn

ii. Problem-dependent

- b/c must try them all out and see what works best

iii. Setting

- Method 1. Choose hyperparameters that work best on the training data
    - 우리의 목표는 모델이 새로운 데이터셋에 대하여 잘 동작하는 것
    - 따라서 학습 데이터에 대해서 성능이 좋은 하이퍼파라미터를 선택하는 것은 좋지 않음
- Method 2. Split data into train and test and choose hyperparameters that work best on test data
    - 이것 역시 지정한 test data에 대하여 잘 동작하는 방식이므로 좋지 않음
- Method 3. Split data into train, validation, and test, choose hyperparameters on validation and evaluate on test
- Method 4. Cross-validation: split data into folds, try each fold as validation and average the results
    - Useful for small datasets
    - But not used too frequently in deep learning

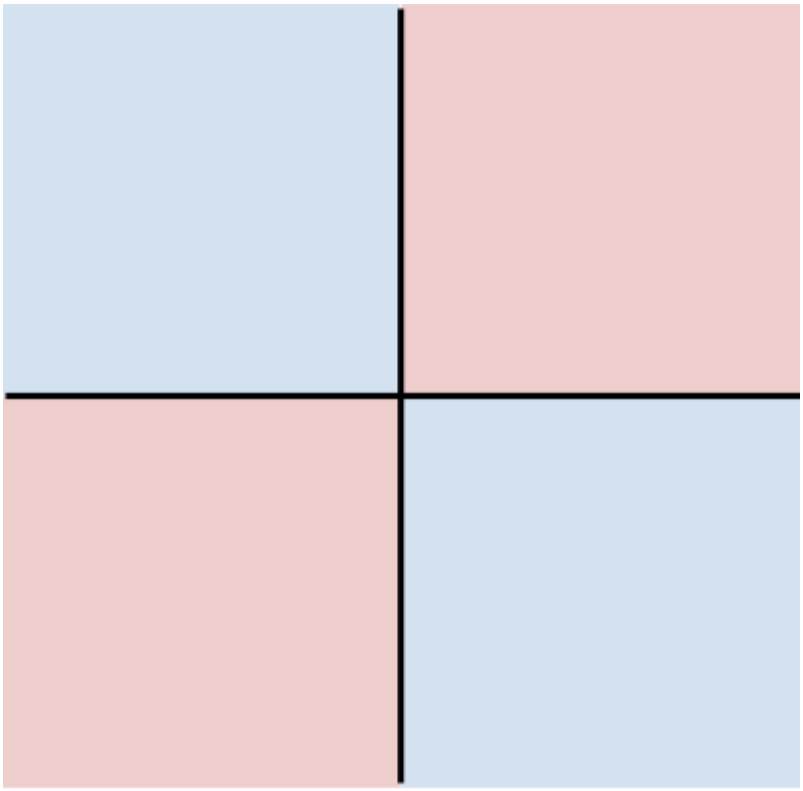
Linear Classification

1. Parametric approach

a. Input image (32 x 32 x 3)

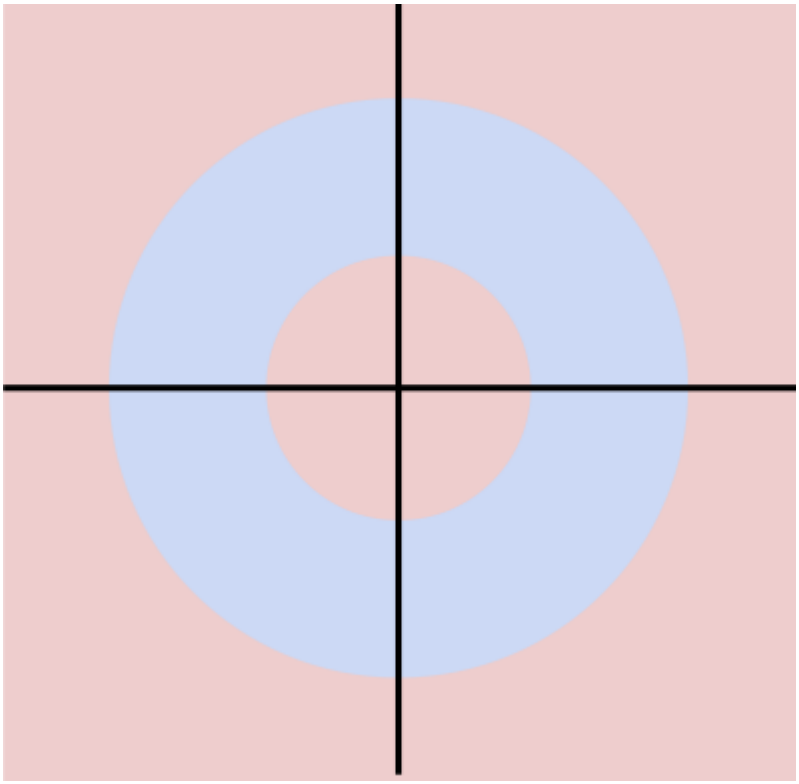
b. f(x, W) = Wx + b
    i. W is a weight parameter and b is a bias term
    ii. Choosing different function F makes different architecture
    iii. The result of the function F ends up the class scores
        - larger score means the larger probability to be classified to the category

2. Linear Classifier
    a. Generate linear decision boundary
        i. 하나의 클래스에 대하여 하나의 선이 decision boundary가 됨
        ii. 이 선을 이용하여 하나의 클래스와 나머지를 구분함
    b. Hard cases
        i. Situation 1.
            - Class 1: number of pixels > 0 is odd
            - Class 2: number of pixels > 1 is even



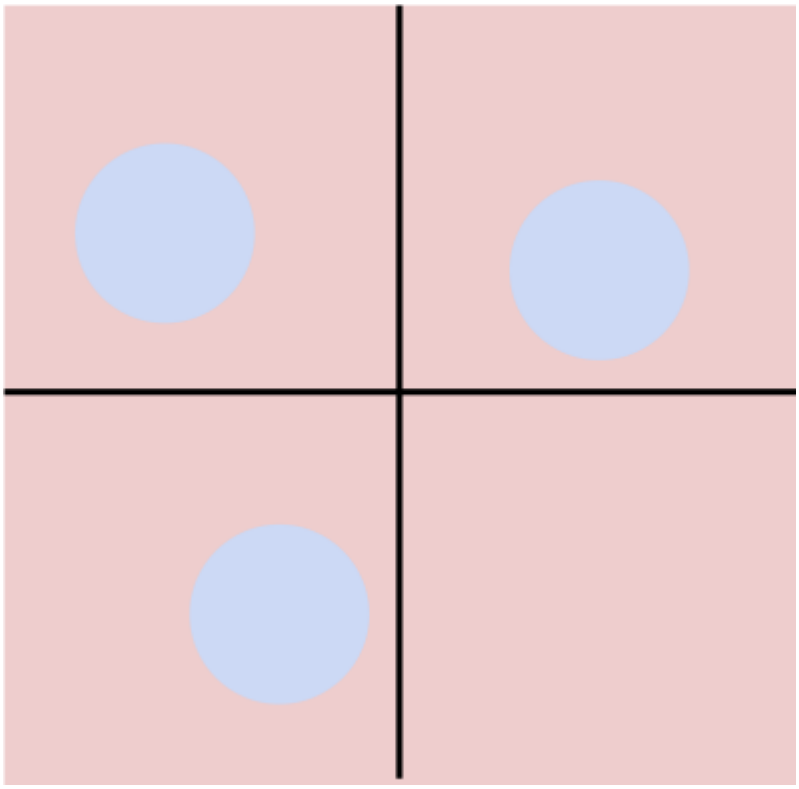
        ii. Situation 2.

- Class 1: 1 ≤ L2 norm ≤ 2
- Class 2: Everything else



iii. Situation 3.

- Class 1: Multi modal (e.g., three modes)
- Class 2: Everything else



3. KNN vs Linear Classifier
    a. KNN keeps whole training set
    b. Linear classifier summarizes and sticks the knowledge from training data to the weight W
        i. i.e., only need parameter W at the test time

Q&A

1. Training data vs Validation data
    a. Training
        - set of images with labels to be memorized by model in the training step
        - able to see labels
    b. Validation
        - used to calculate the accuracy and set hyperparameters
        - cannot directly access to the labels b/c labels are only used to check the performance of the algorithm
2. What if the test set is not representative?

a. randomly split training and test data

3. After choosing the best hyperparameters, can we train the model again?
    a. yes, this will be the trick you can use