**이미지 지도 학습**

1. 이미지 전처리

- 이미지 불러오기

Pillow는 이미지 처리할 때 사용되는 유용한 라이브러리이다.

**from** **PIL** **import** Image

**import** **os**

작업 폴더에 마음에 드는 사진을 넣어둔다.

image\_path = 'images/monkey.jfif'

Pillow 의 Image를 통해 사진을 연다.

image = Image.open(image\_path)

image

- 변환 작업

**크기 변환**

image.resize((128, 128)) *# 픽셀 단위*

resize = image.resize((128, 128))

**저장**

resize.save('images/resized.jpg')

**회전**

resize.rotate(45) *# 반시계방향 각도*

**반전**

**상하반전**

resize.transpose(Image.FLIP\_TOP\_BOTTOM)

**좌우 반전**

resize.transpose(Image.FLIP\_LEFT\_RIGHT)

**90도로 돌리기**

resize.transpose(Image.ROTATE\_90)

- 이미지 필터

**from** **PIL** **import** ImageFilter

**블러 처리**

resize.filter(ImageFilter.BLUR)

**엠보싱**

resize.filter(ImageFilter.EMBOSS)

**윤곽**

resize.filter(ImageFilter.CONTOUR)

**자세히**

resize.filter(ImageFilter.DETAIL)

**날카롭게**

resize.filter(ImageFilter.EDGE\_ENHANCE)

**부드럽게**

resize.filter(ImageFilter.SMOOTH)

- Convert

.convert()에 색상 표현 방식을 넘겨줘서 이미지를 원하는 색상표현으로 변경할 수 있다.

자주 사용되는 방식들은 다음과 같다

* L : 흑백
* RGB : Red, Green, Blue 삼원색을 이용한 색 표현
* RGBA : Red, Green, Blue 삼원색을 이용한 색 표현 방법 + 투명도
* CMYK : Cyan, Magenta, Yellow, Black 을 이용한 색 표현 방법

이미지에 .convert() 함수에 L, RGB, RGBA 등을 넘겨줘서 원하는 색상표현 방식으로 변환할 수 있다.

**흑백**

.convert()에 'L'을 넘겨줘서 8 비트 그레이스케일로 변환할 수 있다.

resize.convert('L') *# ITU-R 601-2 luma transform*

**행렬로 변환**

행렬로 변환하는 것은 numpy.asarray를 사용한다.

**import** **numpy**

m = numpy.asarray(resize)

형태를 보면 가로 128, 세로 128인 행렬 3개가 겹쳐진 형태가 된다.

m.shape

흑백 이미지를 행렬로 바꿔보자.

bw = numpy.asarray(resize.convert('L'))

가로 128, 세로 128인 행렬 1개가 된다.

bw.shape

2. 이미지 기계 학습

**- 이미지 다운로드 및 압축 해제**

실습에서 쓸 데이터는 캐글에서 강아지와 고양이를 구분하는 경연용 데이터(www.kaggle.com/c/dogs-vs-cats/data) 에서 다운로드.

국가 선택 82(KR) 선택 후 전화번호 인증 다운가능

학습용 25,000 이미지(개 12,500개, 고양이 12,500개), 테스트 12,000개

학습용 이미지는 파일명으로 개와 고양이가 라벨링 되어져 있음

각 1,000개씩 폴더로 구분해 놓음

**- 이미지 불러오기**

이미지의 픽셀 값은 0~255 범위인데 값이 너무 커서 학습이 잘 이뤄지지 않는다. 학습률을 낮추는 방법도 있지만 픽셀 값을 255로 나눠 0~1 범위로 바꿔준다.

정규화와 같은 역할 해줌.

또한 학습용 데이터의 갯수가 적기 때문에 여러 가지 변환을 해서 이미지의 갯수를 불려준다. 이를 데이터 증강(data augmentation)이라 한다.

**from** **keras.preprocessing.image** **import** ImageDataGenerator

img\_gen = ImageDataGenerator(

rescale=1./255, *# 픽셀 값을 0~1 범위로 변환*

rotation\_range=40, *# 40도까지 회전*

width\_shift\_range=0.2, *# 20%까지 좌우 이동*

height\_shift\_range=0.2, *# 20%까지 상하 이동*

shear\_range=0.2, *# 20%까지 기울임*

zoom\_range=0.2, *# 20%까지 확대*

horizontal\_flip=True, *# 좌우 뒤집기*

)

train = img\_gen.flow\_from\_directory(

'dog-vs-cat/train', *# 이미지 디렉토리*

target\_size=(100, 100), *# 변환할 크기는 가로 100, 세로 100*

color\_mode='rgb', *# 컬러는 rgb, 흑백은 grayscale. 생략하면 컬러로 처리한다*

class\_mode='binary') *# 고양이 vs. 개로 binary 분류*

학습시킨 모형의 성능 검증을 위한 검증용 데이터도 불러오자. 방법은 동일하다. 검증용 데이터는 증강시킬 필요가 없다.

valid = ImageDataGenerator(rescale=1.0/255).flow\_from\_directory(

'dog-vs-cat/validation',

target\_size=(100, 100),

class\_mode='binary',

shuffle=False)

- 로지스틱 회귀 모형과 동일한 케라스 모형 만들기

**from** **keras** **import** Sequential

**from** **keras.layers** **import** Dense, Flatten

먼저 모형을 만든다.

model = Sequential()

데이터가 가로 세로 크기 100인 컬러 이미지이므로 입력 형태는 100×100×3100×100×3가 된다. 모형의 앞단에 이 입력을 하나의 벡터로 만들어주는 레이어(layer)를 넣어준다.

model.add(Flatten(input\_shape=(100, 100, 3)))

하나의 예측을 내놓는 dense 레이어를 추가한다. 활성화 함수(activation)를 시그모이드(sigmoid)로 설정하면 0~1범위의 출력을 내어놓는다. 이것을 "입력된 사진이 강아지(dog)일 확률"로 해석할 수 있다. 이것은 로지스틱 회귀분석과 동일한 모형이 된다.

model.add(Dense(1, activation='sigmoid'))

**요약 정보**

만들어진 모형의 요약 정보를 보자.

model.summary()

모형 설정

**from** **keras.optimizers** **import** Adam

model.compile(loss='binary\_crossentropy', metrics=['accuracy'], optimizer=Adam(lr=0.001))

이제 전체 데이터를 30회 학습시켜보자.

model.fit\_generator(train, validation\_data=valid, epochs=30)

3. LIME으로 설명

LIME으로 이미지에 대한 설명을 만든다. 먼저 이미지를 (100, 100) 사이즈로 불러온다.

**from** **keras.preprocessing.image** **import** load\_img

img = load\_img('dog-vs-cat/validation/cat/cat.1000.jpg', target\_size=(100, 100))

img

이미지를 넘파이 어레이로 바꾸고 255로 나누어 픽셀값을 0~1 범위로 바꾼다.

**import** **numpy**

img\_array = numpy.array(img) / 255

img\_array.shape

이 이미지는 강아지일 확률이 매우 낮으므로 고양이일 확률이 높다.

하위 폴더의 알파벳 순(디폴트) {'cats': 0, 'dogs': 1}

model.predict(numpy.array([img\_array]))

검증(steps -> 샘플 테스트 수 지정)

scores = model.evaluate\_generator(valid, steps=None)

print("%s: %.2f%%" %(model.metrics\_names[1], scores[1]\*100))

LIME을 이용해서 이 이미지가 고양이로 분류되는 이유를 알아보자.

**!pip install lime**

**from** **lime.lime\_image** **import** LimeImageExplainer

explainer = LimeImageExplainer()

이 이미지의 일부를 지운 30개의 샘플을 만들어서 고양이로 분류되는 이유를 찾아본다.

explanation = explainer.explain\_instance(img\_array, model.predict, num\_samples=30)

고양이로 인식하게 만드는 부분만 보이도록 마스크를 만든다.

temp, mask = explanation.get\_image\_and\_mask(label=0, hide\_rest=True)

마스크를 시각화한다.

**from** **skimage.segmentation** **import** mark\_boundaries

**import** **matplotlib.pyplot** **as** **plt**

%matplotlib inline

plt.imshow(mark\_boundaries(temp, mask))