경사 상승법

데이터 만들기

```
import tensorflow as tf
import numpy as np
```

```
x = np.arange(4, dtype=np.float32)
y = 3 * x
```

미분

```
w = tf.Variable(0.0)
```

```
with tf.GradientTape() as tape:
    yhat = w * x
    loss = tf.reduce_mean((y - yhat) ** 2)
g = tape.gradient(loss, w)
```

업데이트

```
w.assign_sub(learning_rate * g)
```

수렴할 때까지 반복

```
w = tf.Variable(0.0)
improv = np.inf
while np.abs(improv) > 0.01:
    with tf.GradientTape() as tape:
        yhat = w * x
        loss = tf.reduce_mean((y - yhat) ** 2)
    g = tape.gradient(loss, w)
    improv = (learning_rate * g).numpy()
    w.assign_sub(improv)
    print(f'w: {w.numpy():.2f}, g: {g.numpy(): >6.2f}')
```

필터

```
f = [
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1],
]
```

필터

```
f = [
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1],
]
```

합성곱 레이어 설정

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(1, 3, activation='relu'),
])
model.build((None, 100, 100, 1))

model.set_weights([np.expand_dims(f, axis=(2, 3)), np.array([0])])
```

랜덤 이미지

```
x = np.random.random((100, 100))
x = x.astype(np.float32)
x = np.expand_dims(x, axis=(0, 3))
x = tf.Variable(x)
```

시각화

```
import matplotlib.pyplot as plt
plt.imshow(x.numpy()[0, :, :, 0], cmap='gist_gray')
```

경사상승법 gradient ascent

레이어의 출력을 최대화하도록 입력 이미지를 업데이트

```
learning_rate = 1000
for _ in range(20):
    with tf.GradientTape() as t:
        y = model(x)
        activation = tf.reduce_mean(y)
    g = t.gradient(activation, x)
    x.assign_add(learning_rate * g, read_value=False)
```

이미지 보기

```
plt.imshow(x.numpy()[0, :, :, 0], cmap='gist_gray')
```

합성곱 신경망 시각화

모형 다운로드

```
model = tf.keras.applications.VGG16(
    include_top=False,
    weights='imagenet')
```

```
model.build((None, 200, 200, 3))
```

레이어

```
layer_dict = {layer.name: layer for layer in model.layers}
```

```
layer_dict.keys()
```

모형

입력에서 시각화할 레이어의 출력까지를 하나의 모형으로 설정한다

```
layer = layer_dict['block2_conv2']
submodel = tf.keras.Model(model.input, outputs=[layer.output])
filter_index = 0
```

입력 이미지 생성

```
x = np.random.random((200, 200, 3))
x = x.astype(np.float32)
x = np.expand_dims(x, axis=0)
x = tf.Variable(x)
```

경사 상승법

```
learning_rate = 10.0
for _ in range(20):
    with tf.GradientTape() as t:
        output = submodel(x)
        activation = tf.reduce_mean(output[:, :, :, filter_index])
    g = t.gradient(activation, x)
    g = tf.math.l2_normalize(g)
    x.assign_add(learning_rate * g, read_value=False)
```

이미지 시각화

```
img = x.numpy()[0]
img = (img - img.min()) / (img.max() - img.min())
plt.imshow(img)
```

스타일 트랜스퍼

모형 다운로드

```
import tensorflow_hub as hub
style_transfer = hub.load(
    'https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/1')
```

이미지 전처리

```
def load_img(path_to_img):
  max_dim = 512
  img = tf.io.read_file(path_to_img)
  img = tf.image.decode_image(img, channels=3)
  img = tf.image.convert_image_dtype(img, tf.float32)

  shape = tf.cast(tf.shape(img)[:-1], tf.float32)
  long_dim = max(shape)
  scale = max_dim / long_dim

  new_shape = tf.cast(shape * scale, tf.int32)

  img = tf.image.resize(img, new_shape)
  img = img[tf.newaxis, :]
  return img
```

이미지 불러오기

```
content_image = load_img('content.jpg')
style_image = load_img('style.jpg')
```

스타일 트랜스퍼

```
stylized_image = style_transfer(content_image, style_image)[0]
```

저장

```
tf.keras.preprocessing.image.save_img('style.jpg', stylized_image)
```

Visual Embedding

Fashion MNIST

MNIST와 동일한 형식의 의류 이미지 데이터

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
x_train, x_test = x_train / 256, x_test / 256
```

원본 이미지

flattening

```
import numpy as np
x_pos = np.reshape(x_test, (10000, 28*28))
```

시각화

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
n = 500
tsne = TSNE(n_components=2)
pos = tsne.fit_transform(x_pos[:n])
plt.scatter(pos[:, 0], pos[:, 1], c=y_test[:n], cmap='Dark2')
```

모형

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

교차 엔트로피로 임베딩 학습

```
model.compile('adam', 'sparse_categorical_crossentropy', ['accuracy'])
model.fit(x_train, y_train, epochs=3, validation_split=0.1)
```

임베딩 모형

모형 만들기

```
emb_model = tf.keras.Model(model.input, [model.layers[1].output])
```

임베딩 구하기

```
emb = emb_model(x_test)
```

임베딩 시각화

```
tsne = TSNE(n_components=2)
pos = tsne.fit_transform(emb[:n].numpy())
plt.scatter(pos[:, 0], pos[:, 1], c=y_test[:n], cmap='Dark2')
```

triplet loss를 이용한 임베딩 학습

범주별로 인덱스 모음

```
import collections
categories = collections.defaultdict(list)
for idx, cat in enumerate(y_train):
    categories[cat].append(idx)
```

triple 생성

```
import random

anchor_idxs = []
pos_idxs = []
neg_idxs = []

for _ in range(batch_size):
    pos_cat, neg_cat = random.sample(list(range(10)), k=2)
    anchor_idx, pos_idx = random.sample(categories[pos_cat], k=2)
    neg_idx = random.sample(categories[neg_cat], k=1)

    anchor_idxs.append(anchor_idx)
    pos_idxs.append(pos_idx)
    neg_idxs.append(neg_idx)
```

학습

```
optimizer = tf.keras.optimizers.Adam()
margin = 1.0

with tf.GradientTape() as t:
    anchor_emb = emb_model(x_train[anchor_idxs])
    pos_emb = emb_model(x_train[pos_idxs])
    neg_emb = emb_model(x_train[neg_idxs])

    ap_similarity = tf.reduce_sum(anchor_emb * pos_emb, axis=1)
    an_similarity = tf.reduce_sum(anchor_emb * neg_emb, axis=1)

    loss = tf.reduce_mean(tf.maximum(an_similarity + margin - ap_similarity, 0))

g = t.gradient(loss, emb_model.trainable_weights)
optimizer.apply_gradients(zip(g, emb_model.trainable_weights))
```

학습의 반복

```
losses = []
for _ in range(1000):
    # triple 생성
    # ...

    # 학습
    # ...

    losses.append(loss.numpy())
```

손실의 시각화

```
s = np.cumsum(losses)
win = 30
mv = (s[win:] - s[:-win]) / win
plt.plot(losses, 'lightgrey', mv, 'red')
```

데이터 만들기

Annotation

Google에서 "image annotation tools" 또는 "image labelling tools"를 검색

예시: https://www.makesense.ai (설치 필요 없이 웹브라우저에서 작동하는 툴)

국내 기업

https://aiworks.co.kr

https://www.crowdworks.kr

https://selectstar.ai