Machine Learning

13장 Cross Domain GAN

고려대학교 통계학과
박유성

Contents

01 CycleGAN

02 CIFAR10 데이터를 이용한 CycleGAN

03 MNIST와 SVHN 데이터를 이용한 CycleGAN

04 모네그림과 사진데이터를 이용한 CycleGAN

CycleGAN

- 이미지 처리과정에서 한 영역(domain)에 있는 이미지를 다른 영역의 이미지로 전환하는 것은 GAN모형의 응용분야 중 하나이다.
- 그동안 딥러닝에서는 한 영역의 이미지와 다른 영역의 이미지를 짝지어서(align) 이러한 이미지 처리과정을 실행하여 왔다.
- 원천도메인(source domain)과 목표도메인(target domain)의 짝짓기는 어렵거나 불가능한 경우가 많다.
- 자율자동차의 주행을 위해서 가능한 한 많은 도로 이미지가 필요하다.
- 같은 도로일지라도 기후조건, 계절, 시간 등에 따른 도로 이미지가 다르므로 이를 다양한 조합으로 짝짓기하면 자료수집에 엄청난 비용이 들어가게 될 것이다.

CycleGAN

- 만약 이미지의 짝짓기가 없더라도 영역간 이동이 가능하다면 특정도로의 겨울이미지와 다른 도로의 여름이미지를 통해 특정도로의 여름이미지를 생성할 수 있으므로 자료수집을 획기적으로 줄여주는 효과를 가져 오게 될 것이다.
- 이처럼 cross-domain GAN은 원천도메인과 목표도메인을 짝짓기없이 서로간의 이미지를 생성하는 GAN모형으로 이 장에서는 CycleGAN 모형을 통해 cross-domain GAN 모형을 구현하고자 한다.
- CycleGAN 모형의 대표적인 응용영역은 위성사진을 지도로, 얼굴이미지를 이모티콘이나 커리컬처로, 몸(body)이미지를 아바타로, 흑색이미지를 컬러이미지로, 의료스캔 이미지자료를 실제 사진이미지로, 사진이미지를 화가의 그림으로의 전환 등이 있다. 이러한 cross-domain GAN 모형은 단지 이미지자료 뿐만 아니라 글쓰기, 작곡 등에도 응용될 수 있는 분야이다.

CycleGAN

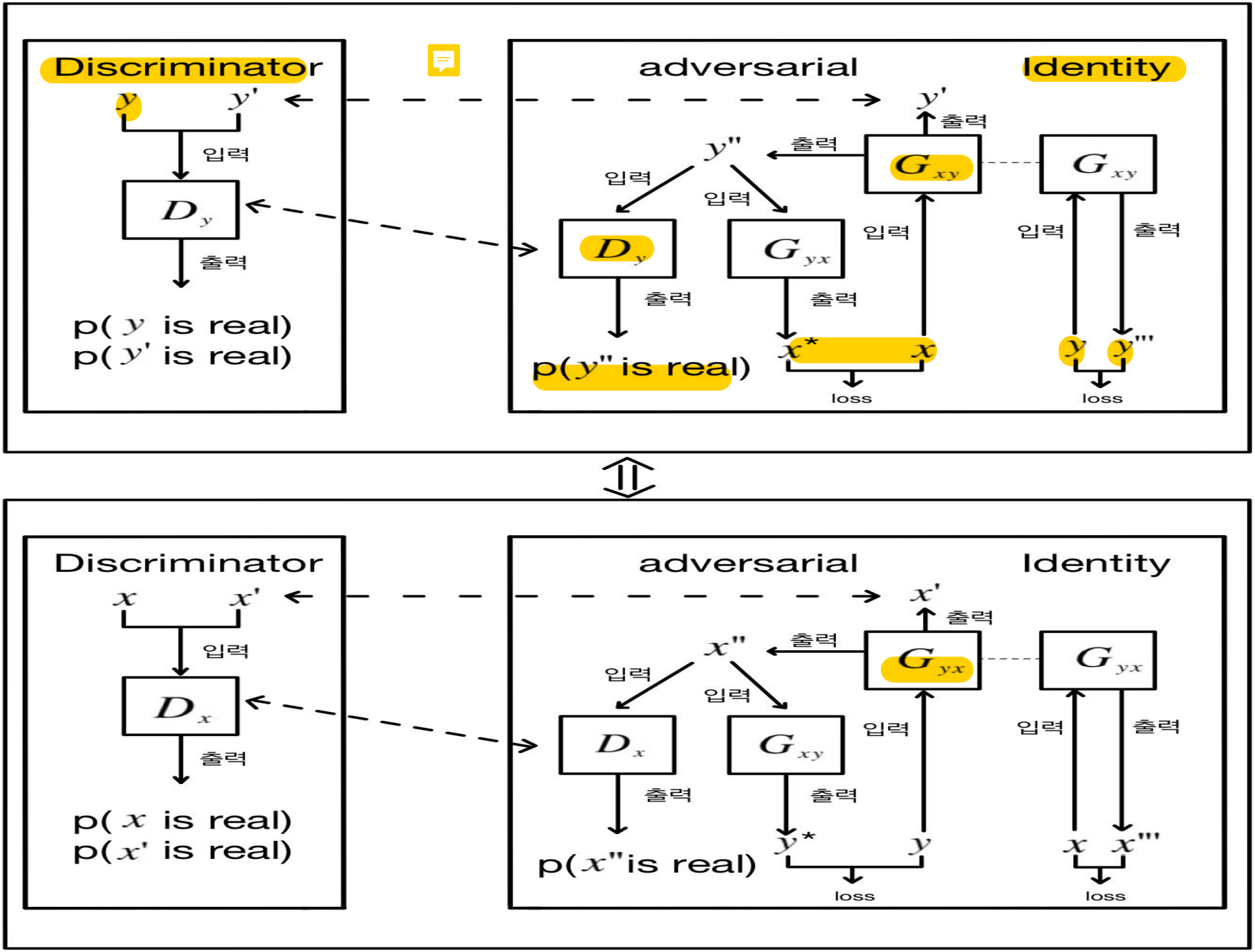
- CycleGAN은 두 개의 generator와 두 개의 discriminator로 구성됨.
- 도메인 A데이터: x, 도메인 B데이터:y

D_x : 데이터가 x 진짜인지 아닌지를 구분하는 discriminator

D_y : 데이터가 y 진짜인지 아닌지를 구분하는 discriminator

G_{xy} : 도메인 A 데이터를 입력하여 도메인 B 데이터를 생성하는 generator

G_{yx} : 도메인 B 데이터를 입력하여 도메인 A데이터를 생성하는 generator


Discriminator
adversarial
Identity
입력
출력
p(y is real)
p(y' is real)
p(y" is real)
loss
Discriminator
adversarial
Identity
입력
출력
p(x is real)
p(x' is real)
p(x" is real)
loss

CycleGAN

- Discriminator 손실

$$l_{D_y} = E_Y(D_Y(y) - 1)^2 + E_Y(D_Y(y'))^2$$

$$l_{D_x} = E_X(D_X(x) - 1)^2 + E_X(D_X(x'))^2$$

$$l_D = \frac{1}{2}(l_{D_y} + l_{D_x})$$

- Generator 손실: validity+reconstruction+identity손실

validity: $l_{D_y} = E_X(D_y\left(G_{Xy}(x)\right) - 1)^2 = E_X(D_y(y'') - 1)^2$

$$l_{D_x} = E_Y(D_x\left(G_{Yx}(y)\right) - 1)^2 = E_Y(D_x(x'') - 1)^2$$

CycleGAN

reconstruction loss

$$l_{xx} = E_X|G_{yx}\left(G_{Xy}(x)\right) - X| = E_X\left|G_{yx}(y'') - X\right| = E_X|X^* - X|$$

$$l_{yy} = E_Y|G_{xy}\left(G_{Yx}(y)\right) - Y| = E_Y\left|G_{xy}(x'') - Y\right| = E_Y|Y^* - Y|$$

Identity loss

$$l_{Iy} = E_Y|G_{xY}(y) - Y| = E_Y|Y''' - Y|$$

$$l_{Ix} = E_X\left|G_{yX}(x) - X\right| = E_X|X''' - X|$$

잔차연결(Residual Connection)

- 딥러닝의 은닉층이 많을수록 손실함수의 최소화를 위한 backpropagation과정에서 소위 미분값의 실종현상(vanishing gradients)이 발생할 가능성이 커진다.
- 특히 backpropagation과정에서 가장 멀리 떨어진 은닉층(즉, 입력층에 가장 가까운 은닉층)의 미분값 실종현상은 두드러지게 나타난다.
- 이러한 현상을 완화하기 위해 dropout나 정규화 등을 사용하고 있으나, 보다 적극적인 방법은 입력층이나 입력층에 가까운 은닉층의 출력을 후반부의 입력으로 직접 연결하는 방법이 잔차연결(residual connection)이다.

잔차연결(Residual Connection)

- 그러므로 잔차연결은 초반부 층의 출력을 후반부 층의 입력으로 사용하는 지름길 이라고 할 수 있다.
- 입력층과 출력층의 중간에 있는 은닉층의 출력노드수(출력 shape[-1]의 관점에서)가 작으면 backpropagation에서 소위 병목현상(bottleneck)이 발생하여 최적화에 문제를 일으키게 된다.
- 잔차연결은 이러한 병목현상을 극복하는 중요한 수단도 된다.

잔차연결(Residual Connection)

- 다음은 4D텐서인 이미지자료를 가정했을 때의 잔차연결 예제 프로그램이다. x는 4D텐서 자료로 입력층 또는 입력층에 가까운 은닉층의 출력이라고 가정한다.
- 아래에서 y1이 잔차연결이며 padding과 strides는 out=layers.add([y,y1])이 성립할 수 있도록 y와 y1의 shape을 일치시키기 위해 사용하였다.

```
from keras import layers
x=.....
y=layers.Conv2D(64,3, activation='relu', padding='same')(x)
y=layers.Conv2D(128,3, activation='relu', padding='same')(y)
y=layers.MaxPooling2D(2)(y)
y1=layers.Conv2D(128,1, strides=2, padding='same')(x)
out=layers.add([y,y1])
```

정규화(Normalization)

- 정규화는 딥러닝 모형의 수렴을 위해 자주 사용하는 기법이다.
- 입력층의 값을 정규화하였다고 해도 은닉층의 출력이 정규화되었다는 보장이 없고, 일반적으로 정규화는 배치단위가 아닌 학습데이터 전체에 정규화를 하므로
- 분석목적에 따라 다양한 자료단위의 정규화가 필요하다.
- 이미지자료인 4D텐서자료를 이용하여 정규화 기법을 설명하여 보자.
- 4D텐서는 (batch, height, width, channel)로 구성되어 있으며 이를 (i, j, k, l)로 표현하면 x_{ijkl}은 i번째 데이터의 (j, k) 픽셀, l번째 channel 값이 됨

정규화(Normalization)

- Batch Normalization(center=False이면 β=0, scale=False이면 $\alpha = 1$)

$$\mu_l = \frac{1}{n_{\cdots l}} \sum_{i,j,k} x_{ijkl}, \quad \sigma_l^2 = \frac{1}{n_{\cdots l}} \sum_{i,j,k} (x_{ijkl} - \mu_l)^2$$

$$\tilde{x}_{ijkl} = \frac{x_{ijkl} - \mu_l}{\sqrt{\sigma_l^2 + \varepsilon}}, \quad \alpha \tilde{x}_{ijkl} + \beta$$

여기에서 $n_{\cdots l} = batch \times height \times width$

- Instance Normalization

$$\mu_{il} = \frac{1}{n_{i \cdots l}} \sum_{j,k} x_{ijkl}, \quad \sigma_{il}^2 = \frac{1}{n_{i \cdots l}} \sum_{j,k} (x_{ijkl} - \mu_{il})^2$$

$$\tilde{x}_{ijkl} = \frac{x_{ijkl} - \mu_{il}}{\sqrt{\sigma_{il}^2 + \varepsilon}}, \quad \alpha \tilde{x}_{ijkl} + \beta, \quad n_{i \cdots l} = height \times width$$

정규화(Normalization)

- Layer Normalization

$$\mu_i = \frac{1}{n_{i\dots}} \sum_{j,k,l} x_{ijkl}, \quad \sigma_i^2 = \frac{1}{n_{i\dots}} \sum_{j,k,l} (x_{ijkl} - \mu_i)^2$$

$$\tilde{x}_{ijkl} = \frac{x_{ijkl} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}, \quad \alpha \tilde{x}_{ijkl} + \beta$$

여기에서 $n_{i\dots} = height \times width \times channel$

- group normalization은 평균과 분산을 각 batch와 일부 채널에 대해 표준화.
- batch normalization은 batch의 크기가 1이면 instance normalization과 동일함을 알 수 있다.

Q & A