1시간만에 GAN 완전 정복하기(Generative Adversarial Network)

https://www.youtube.com/watch?v=odpjk7_tGY0&ab_channel=naverd2

① Branches of ML

Supervised Learning	UnSupervised Learning	Semi-supervised Learning	Reinforcement Learning
The discriminative model learns how to classify input to its class. → Fully labelled = Discriminative Model	The generative learns the distribution of training data. → No labelled = Generative Model	When there is not enough labeled data, the performance of supervised learning can be further improved by learning using unlabeled data. = Supervised Learning + Unsupervised Learning	Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

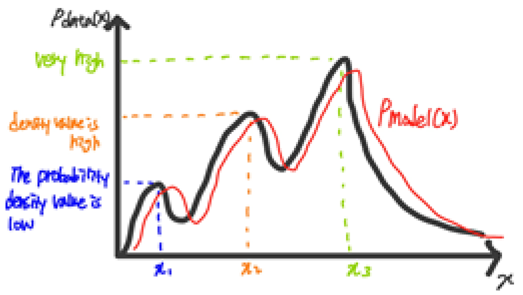
+) Weakly-supervised(self-training): Start with a small number of samples, create a classifier, predict a positive example, label it, and retrain to grow the classifier.(Unlabeled)

https://medium.com/@kabbi159/semi-supervised-learning-%EC%A0%95%EB%A6%AC-a7ed58a8f023

https://hoya012.github.io/blog/Self-Supervised-Learning-Overview/

https://blog.lunit.io/2019/04/25/ficklenetweakly-and-semi-supervised-semantic-image-segmentation-using-stochastic-inference/

② Probability Distribution



- **Pdata(x)** represents the distribution of natural images.

- The probability density value is low(), if there is a few images of a class.

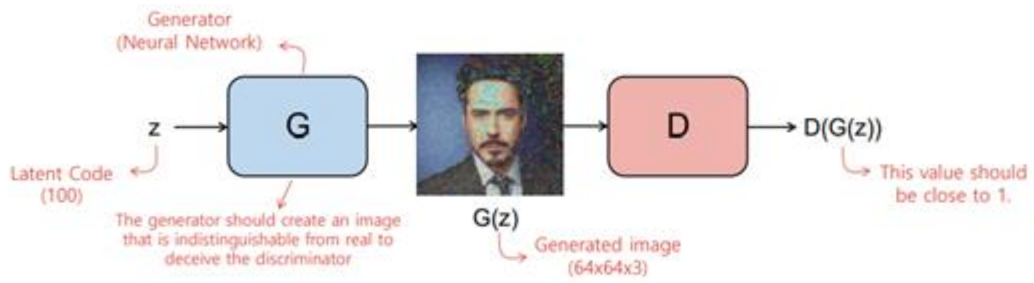
- The probability density value is high(), if there are lots of images of the other class.

- Pmodel(x): The goal for the generative model is to find Pmodel(x) that approximates Pdata(x) well.

③ Intuition GAN

1) First, the discriminator model trains a real image. The D(x) has a value, which is the probability of that x came from the real data(0~1).
2) The discriminator(Neural Network) should classify a real image as real. The value of D(x) should be close to 1.

3) When a fake image is generated by the generator, the discriminator should classify a fake image as fake. The value of D(x) should be close to 0.



4) The generator should create an image that is indistinguishable from real to deceive the discriminator. It is a goal that makes value close to 1 like above.

https://m.blog.naver.com/euleekwon/221557899873

https://yamalab.tistory.com/98

④ Pytorch Implementation for key summary

```
import torch
import torch.nn as nn

# Discriminator
D = nn.Sequential(
   nn.Linear(784, 128),
   nn.ReLU(),
   nn.Linear(128, 1),
   nn.Sigmoid())

# Generative
G = nn.Sequential(
   nn.Linear(100, 128),
   nn.ReLU(),
   nn.Linear(128, 784),
   nn.Tanh())

# BinaryCrossEntorpyLoss(h(x),y) = -ylogh(x) - (1-y)log(1-h(x))
criterion = nn.BCELoss()

# There is a conflict when d and g are learning, so they must by optimized separately
```

```
d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape(batch_size, 784)
# Assume z be random noise of shape(batch_size, 100)
while True:
  # train D
  loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
  loss.backward()              # Gradient value are calculated over all weights
  d_optimizer.step()           # Leaning the AdamOptim Gradient previously defined
                               # to minimize the loss(True is close to 1/ Fake is close to 0)

  # train G
  loss = criterion(D(G(z)), 1) # Creating fake images: When the generated fake
                               # images are put in D, loss.backward() it is learned
                               # and calculated to be close to 1.
  g_optimizer.step()           # Learn only the parameter for G for the
                               # parameters to be learned.
```

Based on tensorflow: https://m.blog.naver.com/PostView.nhn?blogId=euleekwon&logNo=221560040601&targetKeyword=&targetRecommendationCode=1

Based on keras: https://yamalab.tistory.com/98

+) Variants of GAN: DCGAN, LSGAN, SGAN, ACCGAN and so on...

+) Extension of GAN: CycleGAN, stackGAN, StyleGAn and so on...

https://www.youtube.com/watch?v=odpjk7_tGY0&ab_channel=naverd2

https://github.com/yunjey

https://medium.com/@kabbi159/semi-supervised-learning-%EC%A0%95%EB%A6%AC-a7ed58a8f023

[Additional]

⑤ Adam(Adaptive Moment Estimation) optimizer

: It is an algorithm like the combination of RMSProp and Momentum method, and has the advantage that stepsize is not affected by gradient rescalling. Even if the gradient increases, the stepsize is bound, so it is possible to stably descend for optimization no matter what objective function is used.

- Adaptive Gradient Algorithm(AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients.
- Root Mean Square Propagation(RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight(e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems(e.g. noisy).

- Benefits of using Adam on non-convex optimization problems
  - Straightforward to implement
  - Computationally efficient
  - Little memory requirement
  - Invariant to diagonal rescale of the gradients
  - Well suited for problems that are large in terms of data and/or parameters
  - Appropriate for problems with very noisy/or sparse gradients
  - Hyper-parameters have intuitive interpretation and typically require little tuning

https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Focus on equation https://mjgim.me/2018/01/22/adam.html

Paper Review https://dalpo0814.tistory.com/29

Gradient Descent Optimization Algorithms http://shuuki4.github.io/deep%20learning/2016/05/20/Gradient-Descent-Algorithm-Overview.html