Generative Adversarial Networks.

CAP 5415, FALL 2019.

(BASED ON LECTURES BY HUNG-YI LEE, MIKE MOZER, JAMES ENGELMANN)
Generative Adversarial Network

- GAN was first introduced by Ian Goodfellow et al in 2014
- Have been used in generating images, videos, poems, some simple conversation.
- Note, image processing is easy (all animals can do it), NLP is hard (only human can do it).
- This co-evolution approach might have far-reaching implications. Bengio: this may hold the key to making computers a lot more intelligent.

- Ian Goodfellow:
  [https://www.youtube.com/watch?v=YpdP_0-lEOw](https://www.youtube.com/watch?v=YpdP_0-lEOw)
- Radford, (generate voices also here)
  [https://www.youtube.com/watch?v=KeJINHjyzOU](https://www.youtube.com/watch?v=KeJINHjyzOU)
- Tips for training GAN: [https://github.com/soumith/gan_hacks](https://github.com/soumith/gan_hacks)
Autoencoder

As close as possible

Randomly generate a vector as code
Autoencoder with 3 fully connected layers

Training: model.fit(X,X)
Cost function: $\Sigma_{k=1..N} (x_k - x'_k)^2$
Auto-encoder

NN Decoder

2D code

code

NN Decoder

<table>
<thead>
<tr>
<th>1.5</th>
</tr>
</thead>
</table>

-1.5

NN Decoder

| -1.5 |
Auto-encoder

-1.5 1.5
Auto-encoder

From a normal distribution


c = \exp(\sigma) e + m

\Sigma_{i=1..3} \left[ \exp(\sigma_i) - (1+\sigma_i) + (m_i)^2 \right]

This constrains $\sigma_i$ approaching 0 is good.

Problems of VAE

- It does not really try to simulate real images

**code**

![Diagram showing the process of VAE](image)

- One pixel difference to the target
- Also one pixel difference to the target

VAE treats these the same
Gradual and step-wise generation

NN Generator v1 ➔ Discriminator v1 ➔ Generated images

Real images: 2399 0000

These are Binary classifiers

NN Generator v2 ➔ Discriminator v2 ➔ Generated images

Real images: 5041

NN Generator v3 ➔ Discriminator v3 ➔ Generated images
GAN – Learn a discriminator

Randomly sample a vector

Something like Decoder in VAE

Real images Sampled from DB:

image

NN Generator v1

Discriminator v1

1/0 (real or fake)
GAN – Learn a generator

Updating the parameters of generator

The output be classified as “real” (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator

They have Opposite objectives

Randomly sample a vector

Train this

Do not Train This

1.0

0.13
Generating 2\textsuperscript{nd} element figures

Source of images: [https://zhuanlan.zhihu.com/p/24767059](https://zhuanlan.zhihu.com/p/24767059)
From Dr. HY Lee’s notes.

DCGAN: [https://github.com/carpedm20/DCGAN-tensorflow](https://github.com/carpedm20/DCGAN-tensorflow)
GAN – generating 2\textsuperscript{nd} element figures

100 rounds

This is fast, I think you can use your CPU
GAN – generating 2\textsuperscript{nd} element figures

1000 rounds
GAN – generating 2nd element figures

2000 rounds
GAN – generating 2\textsuperscript{nd} element figures

5000 rounds
GAN – generating 2\textsuperscript{nd} element figures

10,000 rounds
GAN – generating 2nd element figures

20,000 rounds
GAN – generating 2\textsuperscript{nd} element figures

50,000 rounds
Next few images from Goodfellow lecture

Next Video Frame Prediction

Ground Truth  MSE  Adversarial

Traditional mean-squared Error, averaged, blurry

(Lotter et al 2016)
Single Image Super-Resolution

(Ledig et al 2016)

Last 2 are by deep learning approaches.
Image to Image Translation

(Isola et al 2016)
DCGANs for LSUN Bedrooms

(Radford et al 2015)
Similar to word embedding (DCGAN paper)

**Vector Space Arithmetic**

- Man with glasses
- Man
- Woman

= Woman with Glasses

(Radford et al, 2015)
256x256 high resolution pictures by Plug and Play generative network

PPGN Samples

(Nguyen et al 2016)
From natural language to pictures

PPGN for caption to image

oranges on a table next to a liquor bottle

(Nguyen et al 2016)
The Generator

- The Generator’s job is to fool the Discriminator
  - Produce fake samples, $G(z)$, from noise, $z$, the Discriminator, $D(x)$ can’t distinguish from real samples, $x$
- During training:
  - Input is random noise samples $z$ from probability distribution, $p_z$
  - Use SGD or other gradient update algorithm like Adam to update parameters, $\theta_G$ with $\nabla_{\theta_G} J_G(W, b)$

$$J_G(W, b) = \frac{1}{m} \sum_{i=1}^{m} \left\{ \log \left( 1 - D \left( G(z^{(i)}) \right) \right) - \log(D \left( G(z^{(i)}) \right)) \right\}$$

- Outputs fake samples, $G(z)$ with probability distribution $p_g$

Reference: Generative Adversarial Nets
The Discriminator

- The Discriminator’s job is detect fake samples
  - Determine probability input is real or fake

- During training:
  - Input is both fake samples, $G(z)$, and real samples, $x$, from data probability distribution, $p_{data}$
  - Update parameters $\theta_D$ by gradient ascent with $\nabla_{\theta_D} J_D(W, b)$

$$J_D(W, b) = \frac{1}{m} \sum_{i=1}^{m} \left[ \log(D(x^{(i)})) - \log(1 - D(G(z^{(i)}))) \right]$$

- Output is probability $D(x)$

Reference: Generative Adversarial Nets
The Game GANs Play

- The Generator wants \( D(G(z)) \) to be high
- The Discriminator wants \( D(G(z)) \) to be low
- Generator and Discriminator play minimax game with value function \( V(D, G) \):

\[
\begin{align*}
\min_G \max_D V(D,G) &= \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(D(x))] + \mathbb{E}_{z \sim p_z(x)}[\log(1 - D(G(z)))]
\end{align*}
\]

- GANs are trained to approximate this relationship:

\[
p_g(z) = p_{\text{data}}(x)
\]

Reference: Generative Adversarial Nets
The Algorithm

for # of training epochs do
   for k steps do
      • Sample minibatch of m noise samples \( \{z^{(1)}, z^{(2)}, ..., z^m\} \) from distribution \( p_z(z) \)
      • Sample minibatch of m data examples \( \{x^{(1)}, x^{(2)}, ..., x^m\} \) from distribution \( p_{data}(x) \)
      • Update the discriminator through gradient ascent:

\[
\nabla_{\theta_D} \left[ \frac{1}{m} \sum_{i=1}^{m} \left[ \log \left( D\left(x^{(i)}\right) \right) - \log \left( 1 - D\left(G\left(z^{(i)}\right)\right) \right) \right] \right]
\]

   end for
   • Sample minibatch of m noise samples \( \{z^{(1)}, z^{(2)}, ..., z^m\} \) from distribution \( p_z(z) \)
   • Update the generator through gradient descent:

\[
\nabla_{\theta_G} \left[ \frac{1}{m} \sum_{i=1}^{m} \left\{ \log \left( 1 - D\left(G\left(z^{(i)}\right)\right) \right) \right\} \quad (MGAN) \right]
\]
\[
\nabla_{\theta_G} \left[ \frac{1}{m} \sum_{i=1}^{m} \left\{ - \log(D\left(G\left(z^{(i)}\right)\right)) \right\} \quad (NSGAN) \right]
\]

end for

Reference: Generative Adversarial Nets

k is a hyperparameter

Updating the Discriminator before the Generator necessary to avoid Mode Collapse

Gradient updates can be done through any gradient-based learning rule
Generative Adversarial Networks

• Pros:
  • State-of-the-art Image Generation
  • Gradients calculated through Backpropagation

• Cons:
  • No explicit representation of generator distribution, $p_g(x)$
  • $G$ and $D$ need to be synchronously trained to avoid Mode Collapse
    • Mode Collapse happens when the Generator finds a “crack” in the Discriminator’s armor and continues to attack the weakness
      • Produces similar outputs with little variation between examples or between features in examples
      • Unstable/Difficult to train
Schemes to Ensure Samples Are Diverse
(Salimans et al., 2016)

Use nearest neighbor on latent representation to detect samples in a minibatch that are too similar.
Cherry Picked Results
Problems With Counting

Goodfellow (2017)
Problems With Perspective
Problems With Global Structure