Generative Adversarial Networks

CMU 11-785 Introduction to Deep Learning, Fall 2020
lecture 24

Presenter : Changdae Oh
bnormal16@naver.com

TAVE Research DL001

2021.05.30

Topics

1. Recap
2. GAN Optimization Issues
3. GAN Training & Stabilization
4. Conclusion

Topics

1. Recap
2. GAN Optimization Issues
3. GAN Training & Stabilization
4. Conclusion

Recap

Adversarial Learning

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

$$L = -(t\log(y) + (1-t)\log(1-y))$$

Discriminator <u>calculates a divergence</u> between generated and target distribution.

acts as a loss function.

Generator tries to minimize the divergence.

- Powerful tool for generative modeling
- Lots of potential
- Limited by pragmatic issues (stability)

Recap

```python
class Discriminator(nn.Module):

    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 256, kernel_size=8, stride=2),
            nn.BatchNorm2d(256),
            nn.GELU(),

            nn.Conv2d(256, 256, kernel_size=8, stride=2),
            nn.BatchNorm2d(256),
            nn.GELU(),

            nn.Conv2d(256, 3, kernel_size=8, stride=2),
            nn.GELU(),

            View(3*10*10),
            nn.Linear(3*10*10, 1),
            nn.Sigmoid()
        )

        self.loss_function = nn.BCELoss()
        self.optimizer = torch.optim.Adam(self.parameters(), lr=0.0001)
```

```python
    def train(self, inputs, targets):
        outputs = self.forward(inputs)
        loss = self.loss_function(outputs, targets)

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
```

```python
class Generator(nn.Module):
    def __init__(self):
        super().__init__()

        self.model = nn.Sequential(
            nn.Linear(100, 3*11*11),
            nn.GELU(),
            View((1, 3, 11, 11)),

            nn.ConvTranspose2d(3, 256, kernel_size=8, stride=2),
            nn.BatchNorm2d(256),
            nn.GELU(),

            nn.ConvTranspose2d(256, 256, kernel_size=8, stride=2),
            nn.BatchNorm2d(256),
            nn.GELU(),

            nn.ConvTranspose2d(256, 3, kernel_size=8, stride=2, padding=1),
            nn.BatchNorm2d(3),
            nn.Sigmoid()
        )

        self.optimizer = torch.optim.Adam(self.parameters(), lr=0.0001)
```

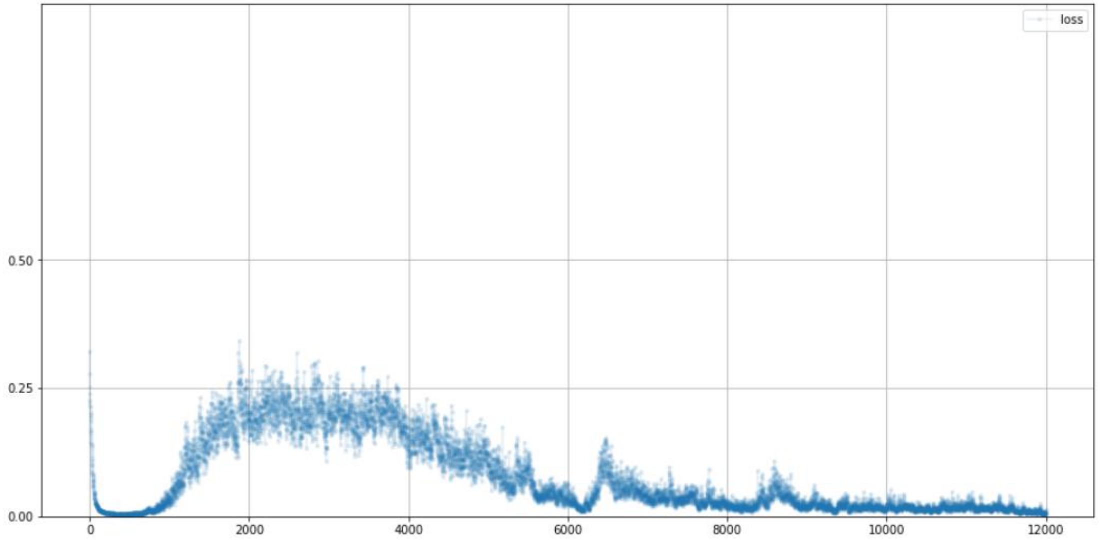
```python
    def train(self, D, inputs, targets):
        g_output = self.forward(inputs)
        d_output = D.forward(g_output)
        loss = D.loss_function(d_output, targets)

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
```

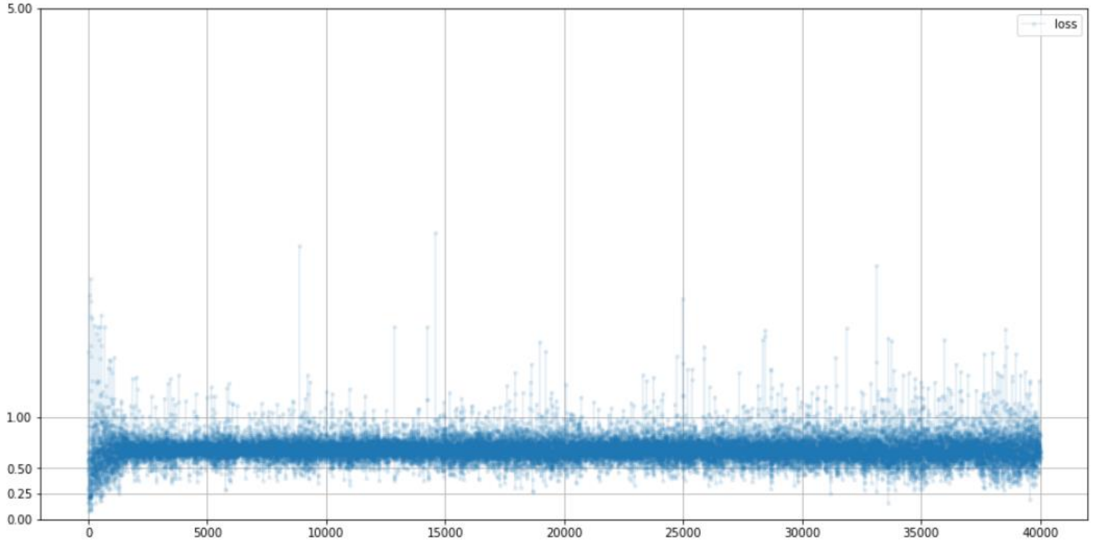
Train Loop

```python
%%time
epochs = 10
D = Discriminator() ; D.to(device)
G = Generator() ; G.to(device)

for epoch in range(epochs):
    print ("epoch = ", epoch + 1)

    for image_data_tensor in celeba_dataset:
        D.train(image_data_tensor, torch.cuda.FloatTensor([1.0]))
        D.train(G.forward(generate_random_seed(100)).detach(), torch.cuda.FloatTensor([0.0]))
        G.train(D, generate_random_seed(100), torch.cuda.FloatTensor([1.0]))
```

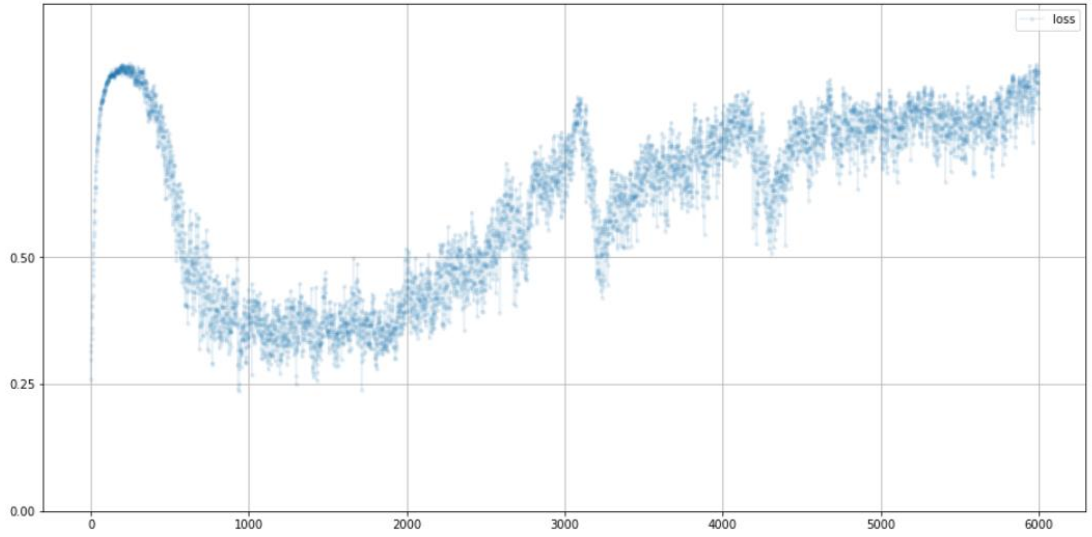
Recap

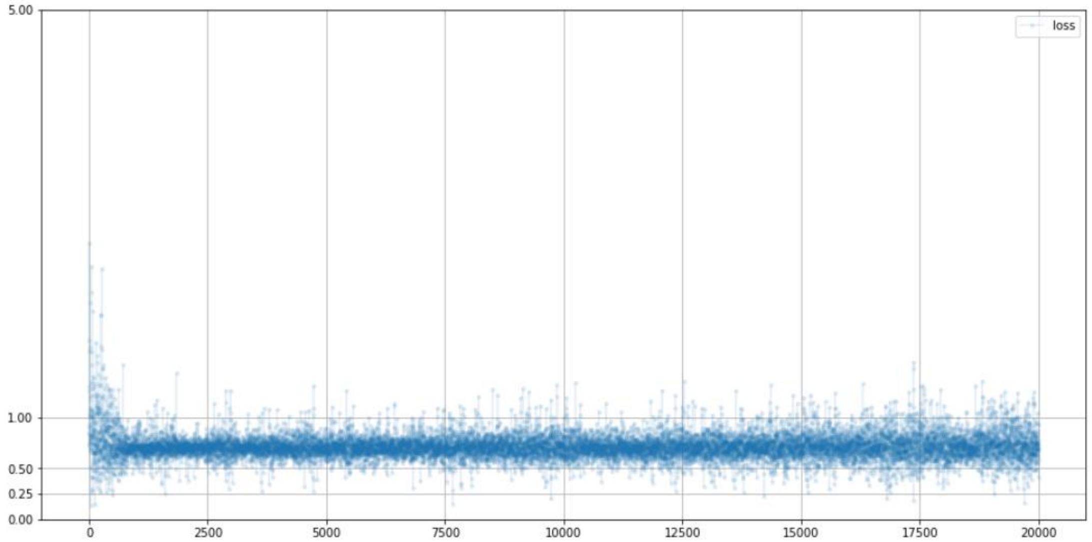
Discriminator





Generator



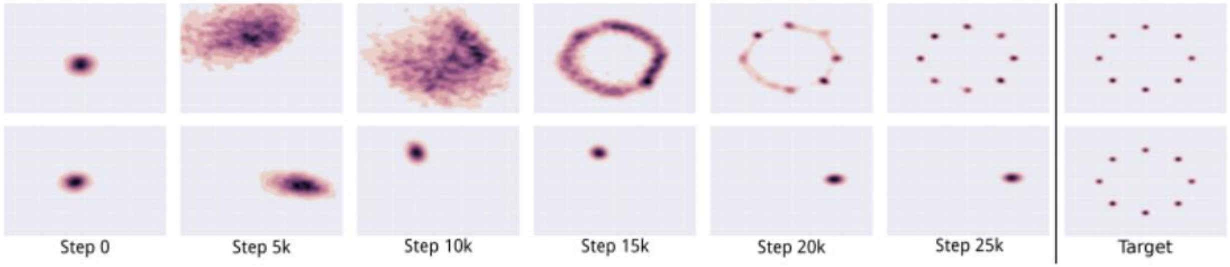


Topics

1. Recap
2. GAN Optimization Issues
3. GAN Training & Stabilization
4. Conclusion

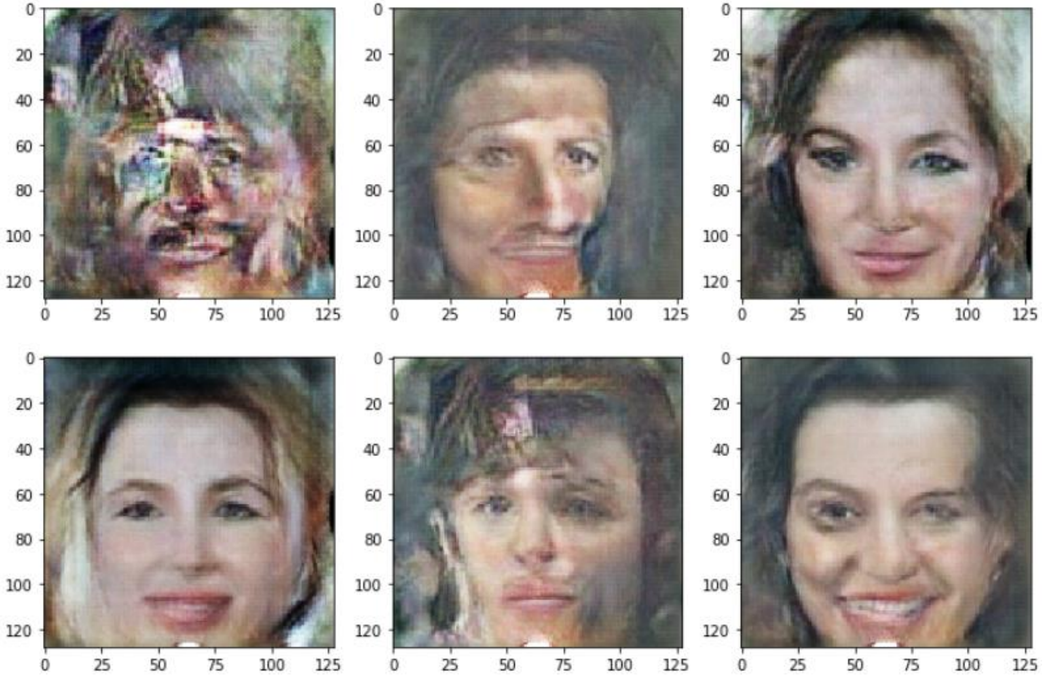
GAN Optimization Issues

Mode Collapse



- Cause can be unclear
  - Unbalanced complexity?
  - minimax <-> maximin
  - Discriminator saturation
  - Information about image diversity, quality is not reflected in the loss function.

worse



better

GAN Optimization Issues

How about just train an optimal discriminator/generator?

- The optimal discriminator emits 0.5 for all inputs…
- Optimal discriminator conditional on current generator and vice-versa.
- Cannot train generator without training discriminator first

G. and D. must be trained together!

- Simultaneous updates require a careful **balance** between players.
- There is a stationary point but no guarantee of reaching it.

GAN Optimization Issues

Factors Affecting Balance

- Different optimizers and learning rates
- Different architectures, depths, # of parameters
- Regularization
- Iterations

Convergence in two-player games

- Rock, Paper, Scissors
  - Global optimum - Both : (0.33, 0.33, 0.33)
    - Both player win, lose or draw w.p. 0.33
  - Local optimum - D : (0.4, 0.3, 0.3) , G : (0, 1, 0)
    - G wins, loses or draws w.p. (0.4, 0.3, 0.3)

Topics

1. Recap
2. GAN Optimization Issues
3. GAN Training & Stabilization
4. Conclusion

GAN Training Techniques

Adjusted / Regularized gradient

- UGAN
- Gradient descent is locally stable
- DRAGAN
- Numerics of GANs

Modified loss function

- LSGAN
- EBGAN
- WGAN
- WGAN – GP
- SNGAN

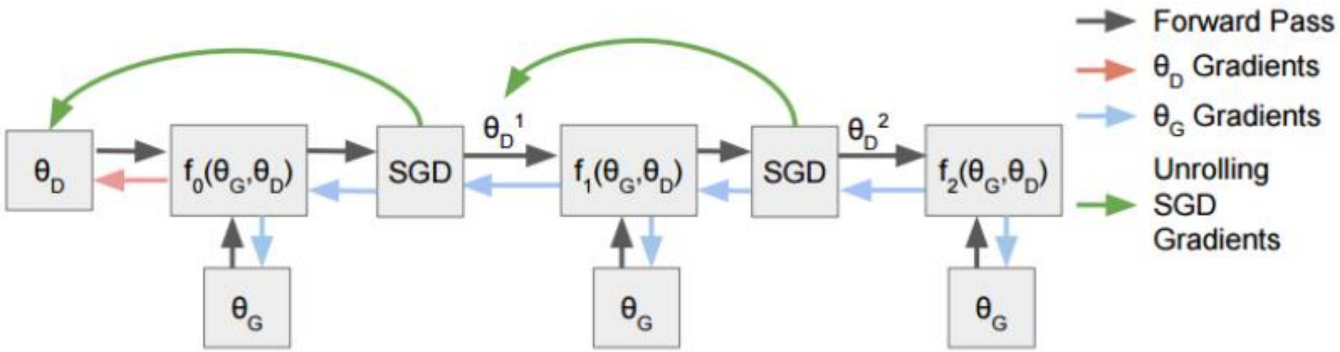
Others

- Instance Noise
- ProGAN
- Other improved techniques for GANs

It is a field that is currently being studied very actively,
and the dynamics of GANs have not been fully understood !

GAN Training & Stabilization

- Look ahead updating : optimize future loss, not current loss



$$f(\theta_G, \theta_D) = \mathbb{E}_{x\sim p_{data}(x)}[log D(x;\theta_D)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z;\theta_G);\theta_D))]$$

$$\theta_D^0 = \theta_D$$

$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{df(\theta_G, \theta_D^k)}{d\theta_D^k}$$

$$f_K(\theta_G, \theta_D) = f(\theta_G, \theta_D^K(\theta_G, \theta_D))$$

Actual update :

$$\theta_G \leftarrow \theta_G - \eta \frac{df_K(\theta_G, \theta_D)}{d\theta_G}$$

$$\theta_D \leftarrow \theta_D + \eta \frac{df(\theta_G, \theta_D)}{d\theta_D}.$$

Gradient Regularization

GD is locally stable

$$\ell_G = \ell_{G,0} + \eta \|\nabla \ell_D\|^2$$

Minimize the original objective as well as the gradient of the discriminator.

DRAGAN

$$\lambda \mathbb{E}_{X,\epsilon} \max(0, \|\nabla D(X+\epsilon)\|^2 - k)$$

Minimize the norm of the gradient in a region around real data. (gradient penalty)

The numerics of GANs

$$L = L_0 + \lambda \|\nabla L_0\|_2^2$$

G & D Mutual de-escalation

Improved Techniques

Feature matching

$$||E_{x \sim p_d ata} f(x) - E_{z \sim p_z(z)} f(G(z))||_2^2$$

Statistics of generated images
should match statistics of real images

Minibatch discrimination

Discriminator can look at multiple inputs at once and decide if those inputs come from the real or generated distribution

➡ Make robust model

Historical averaging

$$\left\| \theta - \frac{1}{t} \sum_{i}^{t} \theta_i \right\|_2^2$$

Dampen oscillations by encouraging
Updates to converge to a mean

Label smoothing

- Common and easy technique that improves performance across many domains
- Smooth the real targets but do not smooth the generated targets when training the discriminator

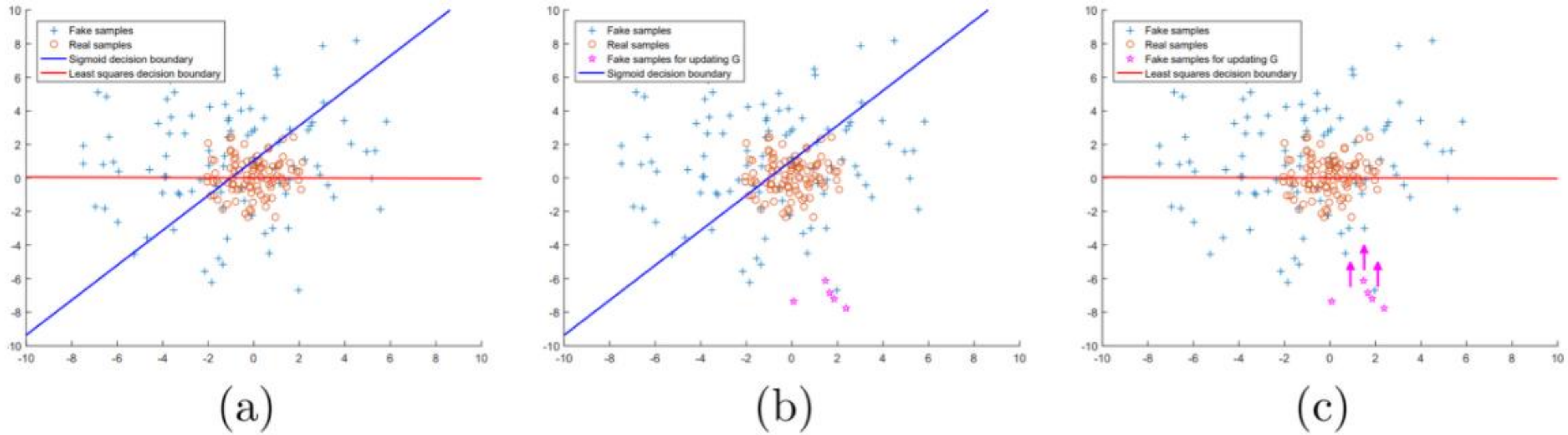
Modified loss function

Least-Squares GAN

$$\min_D V_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[\log(D(x)-b)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z))-a)^2]$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z))-c)^2]$$

Use an L2 loss instead of cross-entropy



Modified loss function

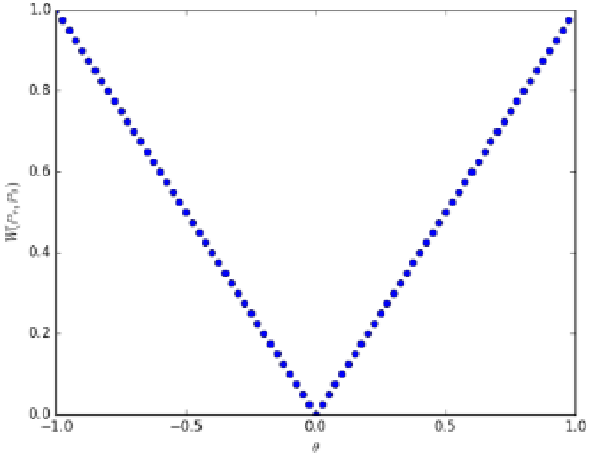
WGAN

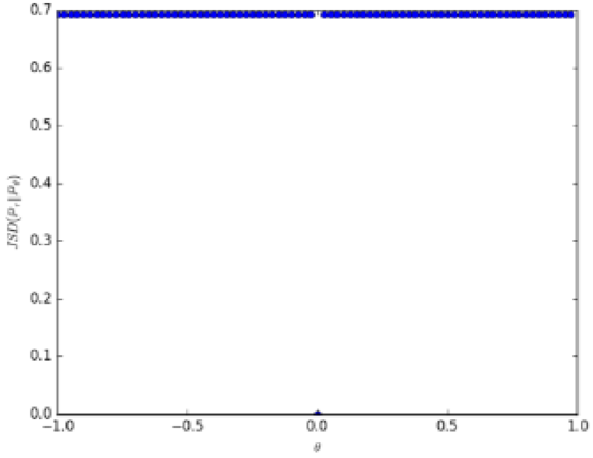
$$L_D = \mathbb{E}_X D(X) - \mathbb{E}_Z D(G(Z))$$

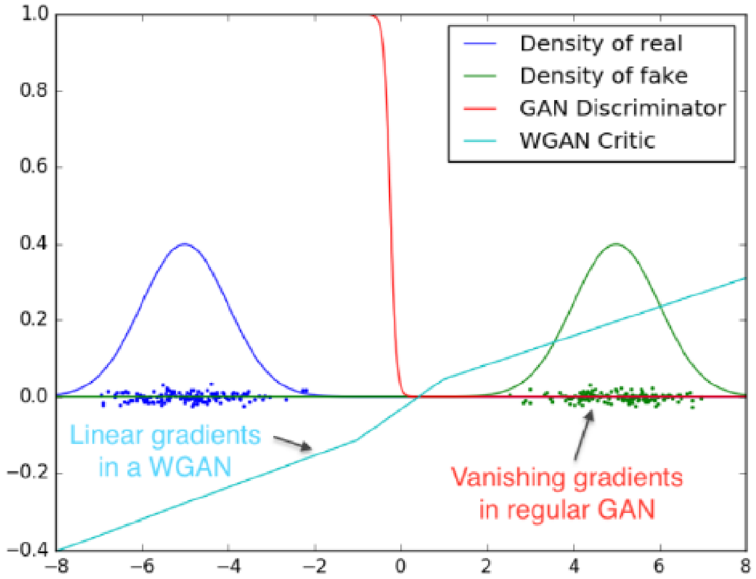
$$L_G = \mathbb{E}_Z D(G(Z))$$

- Simplify the loss function
  - Eliminate log term
  - Wasserstein Loss
- Drop sigmoid term from discriminator
- D must be 1-Lipschitz continuous function
  - Abs of the slope must be up to 1 anywhere.

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \,, \\ 0 & \text{if } \theta = 0 \,, \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \,, \\ 0 & \text{if } \theta = 0 \,, \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \,, \\ 0 & \text{if } \theta = 0 \,. \end{cases}$







Modified loss function

WGAN (compare to GAN)

- Use Wasserstein loss
- True : 1, Fake : -1
- Not use sigmoid on top of discriminator
- Clipping the weight of the discriminator after each update
  (To satisfy the Lipschitz constraint.)
- Each time we update the generator, we train the discriminator several times.

Get a stronger discriminator

And can better balance the training of discriminators and constructors.

However,

Training speed is significantly reduced because weights are clipped in the discriminator.

Modified loss function

WGAN - GP

$$L = \mathbb{E}_X(D(X)) - \mathbb{E}_Z(D(G(Z))) + \boxed{\lambda \mathbb{E}_{X'}\left(\left\|\nabla D(X')\right\|_2 - 1\right)^2}$$

- Demonstrate the flaws with gradient clipping
- Calculate the gradient of D at random samples
- Use samples that are random linear interpolations between real and fake
- Add a penalty of the mean squared distance between the gradient and 1

WGAN : bound-based – can easily learn poor local optima

WGAN GP : sampling based – can be unrealiable

Topics

1. Recap
2. GAN Optimization Issues
3. GAN Training & Stabilization
4. Conclusion

Current & Trend

- Regularize & smooth the discriminator
- Update the players towards some sort of consensus or future stable region, not greedy
- Simplify networks and losses to eliminate nonlinearities
- Constrain the complexity of the discriminator

- Lots of GPUs and tuning required
- better techniques allow larger models, always pushing the limits
- Mostly images but other modalities are in-progress

Conclusion

- How can we measure / constrain the complexity of a NN?
- Evaluation
- How can we architect a neural network to learn a meaningful loss?
  (Perceptual distance between things)

"The GAN Zoo"

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

http://cs231n.stanford.edu/schedule.html