Generative Adversarial Networks (GANs)

Yu-Guan Hsieh

January 2018

Plan

Problem Setting

Given a set of real data examples $\{x^{(i)}\}_{i=1}^{m} \in \mathcal{X}^m$, we suppose they come from some real data distribution $\mathbb{P}_r$ and we would like to 'learn' this probability distribution.

Maximum Likelihood

- A generative model parameterized by some vector $\theta \in \mathbb{R}^d$.
- Note $\mathbb{P}_g$ the generator's distribution, supposed to be continuous, and P_g its density. Since the model is parameterized by θ here, they'll also be noted $\mathbb{P}_\theta$ and P_θ for clarity.
- We want to find solution of the problem

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \log P_\theta(x^{(i)}).$$

- This amounts to minimize the **Kullback-Leibler (KL) divergence** between $\mathbb{P}_r$ and $\mathbb{P}_\theta$:

$$KL(\mathbb{P}_r \| \mathbb{P}_\theta) = \int_{\mathcal{X}} P_r(x) \log \frac{P_r(x)}{P_\theta(x)} dx,$$

where P_r is the density of $\mathbb{P}_r$ supposing it's continuous.

Generative Adversarial Network - Setting

- First appears in "Ian J. Goodfellow et al. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2014".
- Define a random variable Z with a fixed distribution $\mathbb{P}_z$ taking values in $\mathcal{Z}$ (ex: a multivariate gaussian) and some **generator** $G : \mathcal{Z} \to \mathcal{X}$ that directly generates samples following a certain distribution $\mathbb{P}_g$.
- The generator is trained adversarially using a discriminator $D : \mathcal{X} \to [0, 1]$ which indicates the **probability** that a certain example x comes from the data rather than $\mathbb{P}_g$.

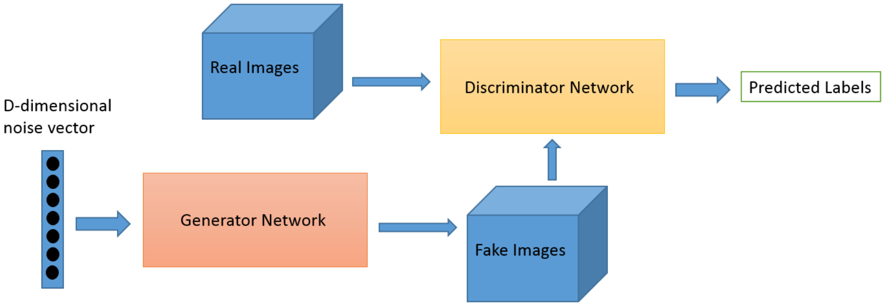
Generative Adversarial Network - Objectif

- Define the function

$$V(G, D) = \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D(G(z)))].$$

  The goal is to solve $\min_G \max_D V(G, D)$.
- In practice, D and G are two neural networks so alternatively we attempt to solve $\max_D V(G, D)$ fixing G and $\min_G V(G, D)$ fixing D by computing the gradients.
- Instead of minimizing $\mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D(G(z)))]$ we would rather maximize $\mathbb{E}_{z\sim\mathbb{P}_z}[\log D(G(z))]$.

Generative Adversarial Network - Illustration



Algorithm 1 Training of classic GAN (D and G are respectively parameterized by w and θ and are noted D_w and G_θ).

for number of training iterations **do**
    for k steps **do**
        Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from real data.
        Sample $\{z^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_z$ a batch of prior samples.
        Update the discriminator by ascending its stochastic gradient:

$$\nabla_w \frac{1}{m} \sum_{i=1}^{m} [\log D_w(x^{(i)}) + \log(1 - D_w(G_\theta(z^{(i)})))].$$

    end for
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_z$ a batch of prior samples.
    Update the generator by ascending its stochastic gradient:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log D_w(G_\theta(z^{(i)})).$$

end for

Generative Adversarial Network - Theory

- If D and G are arbitrary functions, when D is trained to its optimum for some fixed G, we have

$$D_G^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}.$$

- Note $C(G) = \max_D V(G, D)$ and $\mathbb{P}_m = (\mathbb{P}_r + \mathbb{P}_g)/2$ (mixture with densities $(P_r + P_g)/2$), we can show that

$$C(G) = -\log(4) + KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m).$$

- We recognize in the previous expression the **Jensen-Shannon (JS) divergence** between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JS(\mathbb{P}_r \| \mathbb{P}_g).$$

- Minimize the JS divergence between $\mathbb{P}_g$ and $\mathbb{P}_r$.

Generative Adversarial Network - Defects

- Need of synchronization between D and G – we **shouldn't** train the discriminator till convergence.
- Training is very unstable. Choice of architectures rely on heuristics that are extremely sensitive to modifications (batch normalization, dropout, ...).
- In "Alec Radford et al. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv preprint arXiv:1511.06434*, 2015", the authors propose some architectures of GANs that perform quite well in general.

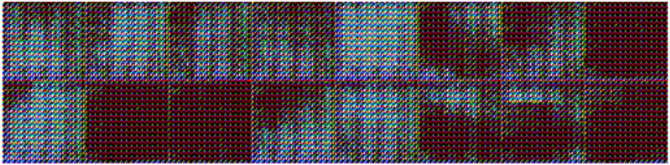
Generative Adversarial Network - Results

- DCGAN (reported in the DCGAN paper).

Generative Adversarial Network - Results

- DCGAN without batch normalization in G (reported in the LSGAN paper).

- Generator without batch normalization and constant number of filters at every layer, as opposed to duplicating them every time as for a normal DCGAN generator (reported in the WGAN paper).



Wassertein GAN - Theory

- Consider the Earth-Mover (EM) or Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_g$:
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||],$$
where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.
- When G is a feedforward neural network parameterized by some vector θ (noted G_θ in this case), $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is **continuous everywhere, and differentiable almost everywhere**. This is not the case for neither the JS nor any KLs.

Wassertein GAN - Theory

- We generally believe that the supports of $\mathbb{P}_r$ and $\mathbb{P}_g$ lie in two low dimensional manifolds that have a intersection of measure zero. In this case, the KL divergence is not defined and the JS divergence equals always $\log 2$ and doesn't provide any usable gradient information.
- The topology induced by Wasserstein distance is weaker than the one induced by the JS divergence. That is, for any $\mathbb{P}$ probability distribution on $\mathcal{X}$ and any $(\mathbb{P}_n)_{n\in\mathbb{N}}$ sequence of probability distributions over $\mathcal{X}$,

$$JS(\mathbb{P}_n, \mathbb{P}) \xrightarrow[n\to\infty]{} 0 \Longrightarrow W(\mathbb{P}_n, \mathbb{P}) \xrightarrow[n\to\infty]{} 0.$$

- The EM distance should be a sensible cost function when learning distribution supported by low dimensional manifolds.

Wassertein GAN - Training

- By the Kantorovich-Rubinstein duality, for any $K > 0$,

$$W(\mathbb{P}_r, \mathbb{P}_g) = \frac{1}{K} \left(\sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_g}[f(x)] \right)$$

  where the supremum is taken over all the K-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

- In practice, f is another neural network parameterized with weights w (thus noted f_w hereinafter) lying in a compact space $\mathcal{W}$. The fact that $\mathcal{W}$ is compact implies that all possible f will be K-Lipschitz for some K that only depends on $\mathcal{W}$ and not the individual weights.
- In order to have parameters w lie in a compact space, we can for example clamp the weights to a fixed box after each gradient update.

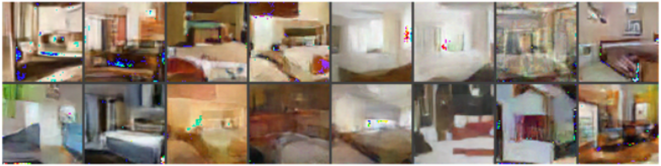
Wassertein GAN - Training

- We train the 'critic' f_w to optimality and carry out an update step of G_θ by turns.
- By using the formula of last page, we get an estimate of the quantiy $K \cdot W(\mathbb{P}_r, \mathbb{P}_g)$. **This estimate correlates well with the quality of the generated samples.**
- Such property doesn't exist for the classic GAN.
- No more mode collapse.
- Work with more architectures.
- NB: Use of momentum based optimizer such as Adam on the critic makes WGAN training unstable – use RMSProp instead.

Wassertein GAN - Results

- DCGAN architecture.

- MLP generator with 4 layers and 512 units with ReLU nonlinearities.



Least Squares GAN

- Replace the sigmoid cross entropy loss function with the least squares loss function. We fix some $a - b$ coding scheme for the discriminator and a value c that G wants D to believe for fake data:

$$V^D_{\mathrm{LSGAN}}(G, D) = \frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_r}[(D(x) - b)^2] + \frac{1}{2}\mathbb{E}_{z\sim\mathbb{P}_z}[(D(G(z)) - a)^2],$$
$$V^G_{\mathrm{LSGAN}}(G, D) = \frac{1}{2}\mathbb{E}_{z\sim\mathbb{P}_z}[(D(G(z)) - c)^2].$$

- Minimize V^D_{LSGAN} with respect to D and V^G_{LSGAN} with respect to G alternatively.

Least Squares GAN

- If $b - c = 1$ and $b - a = 2$, we are minimizing the **Pearson χ^2 divergence** between $\mathbb{P}_r + \mathbb{P}_g$ and $2\mathbb{P}_g$:

$$\chi^2_{\mathrm{Pearson}}(\mathbb{P}_r + \mathbb{P}_g \| 2\mathbb{P}_g) = \int_{\mathcal{X}} \frac{(2P_g(x) - (P_r(x) + P_g(x)))^2}{P_r(x) + P_g(x)} dx.$$

  NB: in some other papers this is rather inverse Pearson, also known as the Neyman χ^2 divergence.
- **Penalize samples lying a long way to the decision boundary.**
- Move the generated samples toward the decision boundary which should go across the manifold of real data.
- Generate more gradients when updating the generator.

Least Squares GAN

f-GAN

- Let $\mathrm{Prob}(\mathcal{X})$ denote the space of probability measures defined on $\mathcal{X}$. For $\mathbb{P}, \mathbb{Q} \in \mathrm{Prob}(\mathcal{X})$ assumed to be absolutely continuous, $f : \mathbb{R}_+ \to \mathbb{R}$ a convex, lower-semicontinuous (i.e. its epigraph is closed) function satisfying $f(1) = 0$, we define the f-divergence,

$$D_f(\mathbb{P}\|\mathbb{Q}) = \int_{\mathcal{X}} Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx.$$

- EX: Take $f : u \mapsto u \log u$ we get the KL divergence and take $f : u \mapsto (u-1)^2$ we get the reverse Pearson (if define as presented earlier).

f-GAN

- Let $\mathcal{T}$ be an arbitrary class of functions $T : \mathcal{X} \to \mathbb{R}$, we have:

$$D_f(\mathbb{P}_r \| \mathbb{P}_\theta) \geq \sup_{T \in \mathcal{T}} (\mathbb{E}_{x \sim \mathbb{P}_r}[T(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f^*(T(x))]),$$

  where $f^* : s \mapsto \sup_{u \in \mathrm{dom}_f}(su - f(u))$ is the Fenchel conjugate of f.
- We can therefore estimate a lower bound of $D_f(\mathbb{P}_r, \mathbb{P}_\theta)$ by maximizing the above qunatity.
- Since f^* is not always defined over $\mathbb{R}$, we fix an 'output activation function' $g_f : \mathbb{R} \to \mathrm{dom}_{f^*}$ for each f and define $T_w = g_f(V_w(x))$ where $V_w : \mathcal{X} \to \mathbb{R}$ is a function parameterized by w without any range constraints on the output.

Energy-based GAN

- Output of D is regarded as **energy**. G is then trained to produce contrastive samples with minimal energies.
- Discriminator D tries to minimize

$$L_D(G, D) = \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] + \mathbb{E}_{z \sim \mathbb{P}_z}[[m - D(G(z))]^+]$$

  for some $m > 0$ and $[\cdot]^+ = max(0, \cdot)$.
- Generator G is trained to minimize

$$L_G(G, D) = \mathbb{E}_{z \sim \mathbb{P}_z}[D(G(z))].$$

- D is constrained to be non-negative.

Energy-based GAN

- Total variation (TV) distance between two probability measures $\mathbb{P}, \mathbb{Q} \in \mathrm{Prob}(\mathcal{X})$:

$$\delta(\mathbb{P}, \mathbb{Q}) = \sup_{A \in \Sigma} |\mathbb{P}(A) - \mathbb{Q}(A)| = \|\mathbb{P} - \mathbb{Q}\|_{TV},$$

  where Σ is the set of all the Borel subsets of $\mathcal{X}$ and $\|\cdot\|_{TV}$ denotes the total variation of a signed measure.
- We can show that training a EBGAN is equivalent to minimizing $\delta(\mathbb{P}_r, \mathbb{P}_\theta)$.
- JS and TV induced the same topology on $\mathrm{Prob}(\mathcal{X})$ (i.e. $\delta(\mathbb{P}_n, \mathbb{P}) \to 0 \Leftrightarrow JS(\mathbb{P}_n, \mathbb{P}) \to 0$).
- **EBGAN is not better than classical GAN.**

Conclusion

- Generaor + Descriminator/Critic/Energy function + Loss.
- WGAN: Well-done theoretical analysis and consistency between theory and practice.
- Many other extensions: InfoGAN, Conditional GAN, AdaGan, LAPGAN, BGAN, ...
- Applications: Super-resolution, Image inpainting, Style transfer, Semi-supervised learning, ...

And You Can Train Your Own Models!

- DCGAN (TensorFlow):
  https://github.com/carpedm20/DCGAN-tensorflow
- WGAN (PyTorch):
  https://github.com/martinarjovsky/WassersteinGAN
- ImprovedWGAN (TensorFlow):
  https://github.com/igul222/improved_wgan_training
- LSGAN (TensorFlow): https://github.com/xudonmao/LSGAN
- InfoGAN (TensorFlow): https://github.com/openai/InfoGAN
- And more ...

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[2] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[3] Xudong Mao et al. "Least squares generative adversarial networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 2813–2821.

[4] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. "f-gan: Training generative neural samplers using variational divergence minimization". In: *Advances in Neural Information Processing Systems*. 2016, pp. 271–279.

[5] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[6] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.

[7] Junbo Zhao, Michael Mathieu, and Yann LeCun. "Energy-based generative adversarial network". In: *arXiv preprint arXiv:1609.03126* (2016).

Thanks for your attention.