باسمه تعالی

دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش پروژه کارشناسی (۱)

موضوع: Generative Adversarial Networks

نگارش:

امیرحسین جوادی

۹۷۱۰۱۴۸۹

استاد راهنما:

دکتر محمدحسین یاسایی

استاد درس پروژه:

دکتر ترانه اقلیدس

تیر ۱۴۰۱

چکیده

در دهه گذشته، افزایش عظیم داده‌های بزرگ [۱] و تکامل مداوم قدرت محاسباتی، هوش مصنوعی [۲] را قادر به انجام هر چه بیشتر وظایف انسانی کرده است. اگر ادعا کنیم هدف هوش مصنوعی شبیه‌سازی هوش انسانی است، چالش اصلی خلاقیت این مدل‌ها است. برای مقابله با این چالش، مدل‌های مولد [۳] شکل گرفته‌اند و یکی از محبوب‌ترین مدل‌های مولد امروزی شبکه‌ی GANs [۴] است. در این پروژه، قصد داریم با مقدمات تئوری شبکه‌های تخاصمی آشنا شویم و و با چالش‌های پیاده‌سازی عملی این شبکه‌ها روبرو شویم.

[۱] Big Data
[۲] Artificial Intelligence
[۳] Generative models
[۴] Generative Adversarial Networks

فهرست مطالب

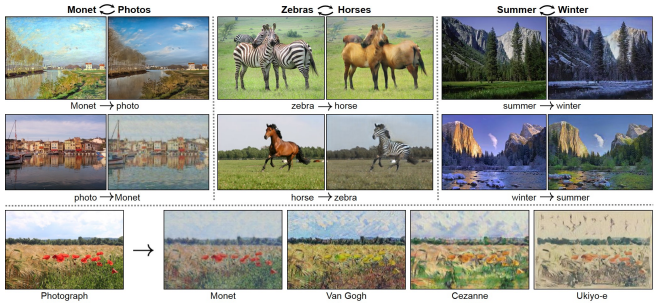
فصل ۱

مقدّمه

۱.۱ کارهای پیشین

یادگیری توزیع احتمالی یک متغیر تصادفی مورد علاقه‌ی آماردانان است. پاسخ کلاسیک به این مسئله، مشخص کردن خانواده‌ی پارامتری از توزیع‌ها، $(P_\theta)_{\theta \in R^d}$، و پیدا کردن توزیعی از این خانواده که با داده‌های ما سازگاری بیشتر دارد، است. شبکه‌های مولد تخاصمی یا Generative Adversarial Networks شاخه‌ای از مسائل یادگیری ماشین است که هدف ابتدایی‌اش، مدل‌سازی توزیع‌های پیچیده در ابعاد بالا بوده‌است. داده‌افزایی، تولید تصاویر صورت انسان، تولید داده برای ساخت دیتاست‌های تصویری و تبدیل تصویر به تصویر [1] مثال‌هایی از کاربردهای این شبکه‌ها است.

(الف) تولید تصاویر صورت انسان



(ب) تبدیل تصویر به تصویر

شبکه‌های GAN در چهارچوب دو مدل در یک فرآینده تخاصمی تعریف می‌شود [2]. این دو شبکه، شبکه‌ی مولد [۱] و شبکه‌ی تفکیک کننده [۲] است. شبکه‌ی Generator تلاش می‌کند که توزیع داده‌های ورودی را یاد بگیرد و نمونه‌های متنوع و نزدیک به نمونه‌های واقعی بسازد. شبکه‌ی Discriminator در مقابل سعی می‌کند داده‌های واقعی را از داده‌هایی که شبکه‌ی Generator تولید کرده است جدا کند. این شبکه این کار را با اعلام احتمال واقعی بودن نمونه انجام می‌دهد. هدف شبکه‌ی Generator ماکزیمم کردن احتمال خطای شبکه‌ی Discriminator است. برد هر شبکه باعث باخت شبکه‌ی دیگر می‌شود و هر شبکه سعی می‌کند پارامترهای خود را به نحوی تغییر دهد که شبکه‌ی مقابلش را شکست دهد.

[۱] Generator
[۲] Discriminator

۲.۱ فرمول بندی ریاضی مسئله

برای یادگیری توزیع داده‌های واقعی، توزیع $p_Z(z)$ را روی ورودی شبکه‌ی Generator تعیین می‌کنیم. این توزیع می‌تواند نویز باشد. نگاشت فضای ورودی به خروجی را با $G(z, \theta_g)$ نشان می‌دهیم که G یک تابع معمولا مشتق‌پذیر است که پارامترهای این تابع θ_g هستند. شبکه‌ی Discriminator در ورودی نمونه‌هایی می‌گیرد که داده‌های واقعی یا داده‌های تولید شده توسط Generator است. این شبکه در خروجی باید احتمال واقعی بودن نمونه ورودی را گزارش کند. نگاشت فضای نمونه‌های ورودی به احتمال خروجی را با $D(x, \theta_d)$ نشان می‌دهیم که D یک تابع معمولا مشتق‌پذیر است که پارامترهای این تابع θ_d هستند. برای بررسی عملکرد هر یک از مدل‌ها باید متری برای سنجش معرفی کنیم. برای شبکه‌ی Discriminator تابع سنجش می‌تواند به شکل زیر باشد.

$$\frac{1}{m}\sum_{i=1}^{m}\Big[\log(D(x)) + \log(1 - G(z))\Big] \tag{1.1}$$

شبکه‌ی Discriminator باید برای داده‌های واقعی احتمال ۱ خروجی بدهد و برای داده‌های تولیدی شبکه‌ی Generator احتمال ۰ اعلام دهد. اگر Discriminator به خوبی عمل کند، در هر دو صورت گفته شده خروجی تابع سنجش صفر است. در غیر این صورت مقدار منفی به عنوان جریمه به تابع سنجش اضافه می‌شود. پس هدف شبکه‌ی Discriminator ماکزیمم کردن این تابع سنجش است. در مقابل، هدف تابع Generator تولید داده‌ای است که شبکه‌ی Discriminator را به اشتباه بیندازد. برای این شبکه تابع سنجش می‌تواند به شکل زیر باشد.

$$\frac{1}{m}\sum_{i=1}^{m}\log(1 - G(z)) \tag{1.2}$$

در نگاه اول، این تابع سنجش با تابع سنجش شبکه‌ی Discriminator همسو به نظر می‌رسد؛ اما باید دقت کرد که هدف شبکه‌ی Generator قصد دارد این تابع سنجش را تا جایی که ممکن است کم کند چون قصد دارد خروجی‌هایش توسط Discriminator به اشتباه برچسب ۱ بخورد. این چهارچوب با مفهوم minimax two-player game در نظریه بازی مطابقت دارد. به عبارت دیگر، D و G بازی minimax با تابع هدف $V(G, D)$ را برقرار می‌کنند.

$$\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_Z(z)}[\log(1 - D(G(z)))] \tag{1.3}$$

۳.۱ چالش‌ها

در فرآیند یادگیری شبکه‌های GAN سه مشکل شناخته شده وجود دارد.

۱. فرآیند یادگیری شبکه‌های GAN ناپایدار است. مسئله یادگیری میتواند پارامتر یا هایپرپارامترهای زیادی داشته باشد و نتیجه نهایی به مقدار این پارامترها حساسیت زیادی میتواند داشته باشد. حتی نقطه ی شروع الگوریتم هم میتواند اثر قابل توجهی روی خروجی داشته باشد.

۲. شبکه‌های GAN معمولا به هدف یادگرفتن یک توزیع احتمالاتی مورد استفاده قرار می‌گیرد. سنجش این که یک شبکه چه قدر توانسته توزیع را یاد بگیرد معمولا سخت است. این موضوع باعث میشود که ارزیابی این شبکه‌ها سخت باشد.

۳. مورد سوم پدیده‌ای به اسم mode collapse است. mode collapse حالتی است که شبکه‌های ما در یکی از دو حالت زیر گیر افتاده باشند.

(الف) حالت اول این است که شبکه به سمت تولید کردن داده‌های خاصی برود و به طبع داده‌های دیگر را تولید نمی‌کند. مثلا اگر هدف تولید عکس اعداد باشد شبکه Generator داده‌های خاصی را بیشتر تولید می‌کند. به طور مثال، اعداد ۱ و ۷ را با فرکانس بیشتری از اعدادی مثل ۵ تولید کند. این اتفاق احتمالا به این دلیل است که شبکه‌ی Generator با این نمونه‌ها توانسته است بهتر شبکه Discriminator را گول بزند. طبعا این مورد مطلوب ما نیست چون هدف ما یادگیری کل توزیع بود.

(ب) نشانه‌ی دیگر این پدیده این است که دو نمونه از فضای z که فاصله زیادی از هم دارند به نمونه های مشابهی نگاشت شوند.

فصل ۲

Mode Collapse

۱.۲ مقدّمه

هدف شبکه‌های GAN یادگیری توزیع‌ها در ابعاد بالا مانند تصاویر است. بنابراین مطلوب آن است که خروجی شبکه‌ی Generator متنوع باشد. به طور مثال در مسئله‌ی تولید اعداد رندوم، انتظار دارید که با ورودی تصادفی مختلف اعداد متفاوتی ساخته شود. با این حال، اگر یک Generator خروجی قابل قبولی تولید کند، ممکن است یاد بگیرد که فقط آن خروجی را تولید کند. در نتیجه Generator مجموعه کوچکی از فضای خروجی را تولید می‌کند. به این مشکل در شبکه‌های GAN، Mode Collapse نامیده می‌شود.

برای مقابله با این مشکل راهکارهای قابل توجهی ارائه شده است که یکی از آن‌ها ایده‌ی PacGAN [3] است. پیش از توضیح این الگوریتم چند تعریف ارائه می‌کنیم.

۱. توزیع p و q از (ϵ, δ)-Mode Collapse رنج میبرند ($0 \leq \epsilon < \delta \leq 1$) در صورتی که وجود داشته باشد مجموعه‌ی $S \in X$ که احتمال این مجموعه در توزیع p از δ بیشتر باشد و در توزیع q از ϵ کمتر باشد. هر چه δ بزرگ‌تر و ϵ کوچک‌تر شود Mode Collapse شدیدتری داریم. معنای این عبارت این است که قسمت قابل توجهی از توزیع هدف $p(A) \geq \delta$ توسط Generator نادیده گرفته شده است و احتمال تولید همچنین سمپل‌هایی کم است. $(q(A) \leq \epsilon)$

۲. در تئوری آمار، total variation distance یک متر برای فاصله‌ی بین دو توزیع است که به اختصار فاصله‌ی TV هم نام دارد. برای دو توزیع احتمال p و q برابر است با

$$TV(p, q) = \sup_A |p(A) - Q(A)| = \sup_A \int_{x \in A} |p(x) - Q(x)| \qquad (۱.۲)$$

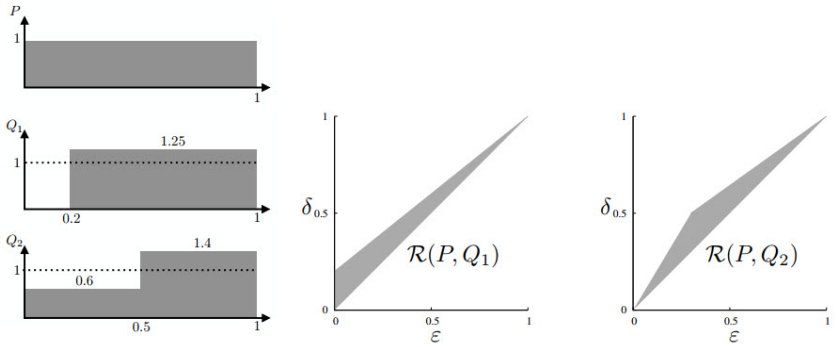
که A هر زیر مجموعه از فضای نمونه است.

۳. Mode Collapse region پوش محدب مجموعه‌ی هر زوج مرتبی مثل (x, y) است که دو توزیع P و Q از (x, y)-Mode Collapse رنج ببرند.

$$R(P, Q) = conv\big(\{(\epsilon, \delta) | \delta > \epsilon \ \& \ (P, Q) \text{ has } (\epsilon, \delta) - \text{Mode Collapse}\}\big) \qquad (۲.۲)$$

نکته اصلی این است که دو توزیع ممکن است فاصله TV یکسانی با توزیع هدف داشته باشد ولی Mode Collapse متفاوتی تجربه کنند.

برای فهم بیشتر مثال زیر را ببینید. فرض کنید که ما قصد داریم توزیع $P = U([0,1])$ را یاد بگیریم. دو توزیع کاندید $Q_1 = U([0.2,1])$ و $Q_2 = 0.6U([0,0.5]) + 1.4U([0.5,1])$ داریم که توزیع‌های هر سه در سمت چپ کشیده شده است. هر دو کاندید فاصله‌ی TV یکسانی با P دارند اما نمودار Mode Collapse region این دو کاندید با هم متفاوت است.



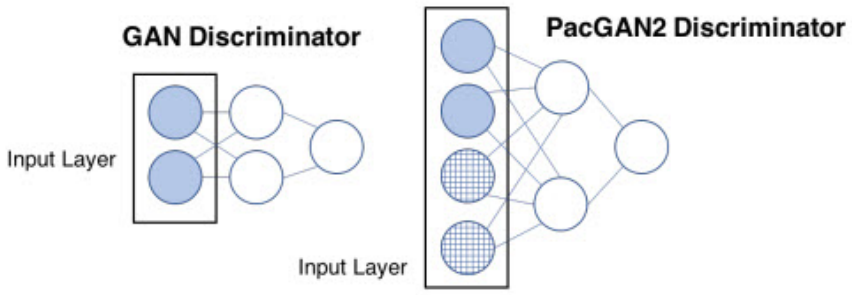
شکل ۱.۲: Mode collapse region

۲.۲ PacGAN

یک شبکه‌ی GAN شامل یک معماری مولد [۱]، معماری تفکیک‌کننده [۲] و یک تابع ضرر [۳] است. به این سه عامل ساختار مادر [۴] شبکه‌ی GAN نام دارند. چهارچوب PacGAN معماری مولد و تابع ضرر و هایپر پارامترهای ساختار مادر را حفظ می‌کند . تفاوت این شبکه در این است که به جای داشتن یک Discriminator مانند $D(x)$ که به یک نمونه یک احتمال را نسبت می‌دهد Discriminator مانند $D(X_1, X_2, \ldots, X_m)$ دارد که m نمونه میگیرد و یک احتمال به عنوان خروجی مشخص می‌کند. این m نمونه به صورت iid از یک توزیع تولید شده است که این توزیع می‌تواند توزیع داده‌های واقعی $(\mathrm{Y} = 1)$ یا داده‌های تولید شده توسط Generator $(\mathrm{Y} = 0)$ باشد. این چهارچوب روی هر شبکه‌ی GAN با Discriminator به شکل $D(x)$ قابل پیاده‌سازی است. از برچسب "Pac(X)(m)" برای شبکه‌ی PacGAN که ساختار مادر (X) و درجه‌ی پکینگ m دارد استفاده می کنیم.

یک شبکه‌ی PacGAN با m برابر شدن گره‌های ورودی و ثابت ماندن گره‌های پنهان نسبت به ساختار مادر شکل می‌گیرد. برای مثال در شکل 2.2 مثالی از شبکه‌ی پکینگ مشخص است. PacGAN2 تعداد گره‌های لایه‌ی ورودی m برابر شده است و یال‌های بین لایه‌ی ورودی و لایه‌ی پنهان به نحوی تغییر کرده است که ساختار fully-connected شبکه‌ی مادر حفظ شود.

[۱] generator architecture
[۲] discriminator architecture
[۳] loss function
[۴] mother architecture



شکل ۲.۲: ساختار PacGAN و GAN متناظر

۳.۲ تحلیل نظری شبکه‌های PacGAN

Discriminator سعی می‌کند از تفاوت توزیع داده‌ی اصلی و داده‌ی تولید شده توسط Generator متوجه شود که داده اصلی یا تقلبی است. در عوض Generator سعی می‌کند فاصله‌ی $d_{TV}(P,Q)$ را کاهش دهد و توزیع خروجی خود را به توزیع داده‌ی اصلی نزدیک کند. دلیل استفاده از total variation distance به عنوان متر سنجش فاصلهِ‌ی دو توزیع این است که تاثیر پکینگ روی این متر راحت‌تر و قابل فهم‌تر از مترهای دیگر مثل Jensen-Shannon divergence است.

با توجه به توضیحات قبل، نمونه‌های پک‌شده به صورت iid تشکیل می‌شوند و P^m توزیع نمونه‌های واقعی و Q^m توزیع نمونه‌های تولید شده توسط Generator است. مقدار خطای شبکه‌ی Discriminator هم متناسب با متر $d_{TV}(P^n,Q^n)$ تغییر می‌کند. شهود کارایی این عبارت این است که فاصله‌ی $d_{TV}(P^n,Q^n)$ به نحوی تغییر می‌کند که با Mode Collapse region در ارتباط است.

ما تغییرات total variation در حالت پکینگ برای هر جفت توزیع (P,Q) که فاصله‌ی total variation ثابتی برابر τ در حالت آنپک دارند و از مشکل (ϵ,δ)-Mode Collapse رنج می‌برند را بررسی میکنیم.

$$\min_{P,Q}\max_{P,Q} d_{TV}(P^m,Q^m) \tag{2.3}$$

$$\text{Subject to}\begin{cases} d_{TV}(P,Q)=\tau \\ (P,Q)\ \text{has}\ (\epsilon,\delta)-\text{mode collapse}\end{cases} \tag{2.4}$$

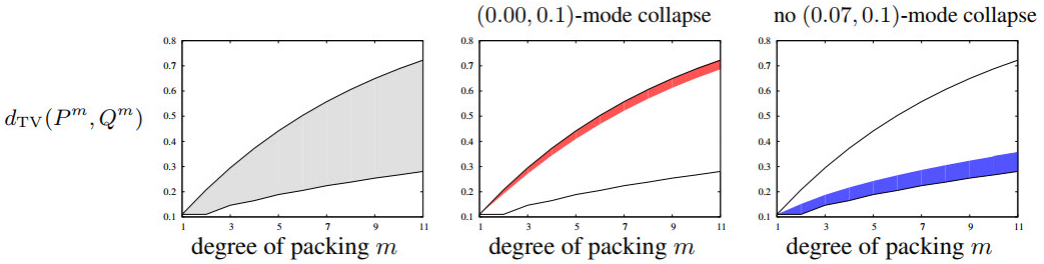
برای هر $0 \le \epsilon < \delta \le 1$ و عدد طبیعی m، اگر $1 \ge \tau \ge \delta-\epsilon$ باشد جواب ماکزیمم 2.3 برابر است با $1-(1-\tau)^m$ و جواب مینیمم برابر است با

$$\min\left\{\min_{0\le\alpha\le 1-\frac{\tau\delta}{\delta-\epsilon}} d_{TV}\left(P_{inner1}(\alpha)^m, Q_{inner1}(\alpha)^m\right), \min_{1-\frac{\tau\delta}{\delta-\epsilon}\le\alpha\le 1-\tau} d_{TV}\left(P_{inner2}(\alpha)^m, Q_{inner2}(\alpha)^m\right)\right\}$$

که

$$\begin{cases} P_{inner۱}(\alpha) = [\delta, ۱ - \alpha - \delta, \alpha] \\ Q_{inner۱}(\alpha) = [\epsilon, ۱ - \alpha - \tau - \epsilon, \alpha + \tau] \\ P_{inner۲}(\alpha) = [۱ - \alpha, \alpha] \\ Q_{inner۲}(\alpha) = [۱ - \alpha - \tau, \alpha + \tau] \end{cases}$$

متغیر تصادفی به این شکل است و m نشان دهندهٔ توزیع product است. اگر $\tau < \delta - \epsilon$ باشد، مسئلهٔ بهینه‌سازی 2.3 هیچ جوابی ندارد و مسئله غیر قابل حل است. به شکل 2.3 دقت کنید. جفت توزیع (P, Q) که بیشترین Mode Collapse را دارند در قسمت بالای نمودار مشخص شده‌اند (با رنگ قرمز) که باند بالای $d_{TV}(P^n, Q^n)$ را مشخص می‌کنند و جفت توزیع (P, Q) که بیشترین Mode Collapse را دارند در قسمت پایین نمودار مشخص شده‌اند (با رنگ آبی) که باند پایین $d_{TV}(P^n, Q^n)$ را مشخص می‌کنند. توزیع‌هایی با Mode Collapse قوی‌تر به خطای بیشتری می‌انجامند و به همین دلیل Generator خطای $d_{TV}(P^n, Q^n)$ را مینیمم می‌کند و به سمت توزیع‌هایی میرود که Mode Collapse کمتری دارد.



شکل ۳.۲: بازه‌ی تغییرات $d_{TV}(P^n, Q^n)$ حصول شده توسط توزیع‌هایی که $d_{TV}(P, Q) = \tau$ برای $\tau = ۰/۱۱$

فصل ۳

دیگر شبکه‌های GAN

۱.۳ Wasserstein GANs

برای بررسی فاصله بین دو توزیع مترهای مختلفی در تئوری اطلاعات وجود دارد. چهار متر معروف در این زمیه عبارت اند از

1. The Total Variation (TV) distance

$$\delta(P_r, P_g) = \sup_{A \in \Sigma} |P_r(A) - P_g(A)| \tag{3.1}$$

2. The Kullback-Leibler (KL) divergence

$$KL(P_r||P_g) = \int \log(\frac{P_r(x)}{P_g(x)}) P_r(x) d\mu(x) \tag{3.2}$$

3. The Jensen-Shannon (JS) divergence

$$JS(P_r, P_g) = KL(P_r||P_m) + KL(P_g||P_m) \qquad (P_m = (P_r + P_g)/2) \tag{3.3}$$

4. The Earth-Mover (EM) distance or Wasserstein-1

$$W(P_r, P_g) = \inf_{\lambda \in \prod(P_r, P_g)} \mathbb{E}_{x,y \in \lambda} \big[|x - y|\big] \tag{3.4}$$

که $\prod(P_r, P_g)$ توزیع جوینت تمام $\lambda(x, y)$ است که توزیع مارجینال آن به ترتیب P_r و P_g است. برای این که نشان دهیم که فاصله‌ی Wasserstein-1 می‌تواند نتایج بهتری در مسائل خاصی بدهد به مثال زیر توجه کنید.
فرض کنید $Z \sim U[۰, ۱]$ یک متغیر تصادفی یکنواخت است. توزیع روی $(۰, Z) \in R^۲$ را با $P_۰$ مشخص می‌کنیم. حال فرض کنید که $g_\theta(z) = (\theta, z)$ توزیع خروجی Generator است. فاصله‌ی بین این دو توزیع با مترهای ارائه شده به شکل زیر است.

1. The Total Variation (TV) distance

$$\delta(P_0, P_\theta) = \begin{cases} 1 & \theta \neq 0 \\ 0 & \theta = 0 \end{cases} \tag{3.5}$$

2. The Kullback-Leibler (KL) divergence

$$KL(P_0, P_\theta) = KL(P_\theta, P_0) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases} \tag{3.6}$$

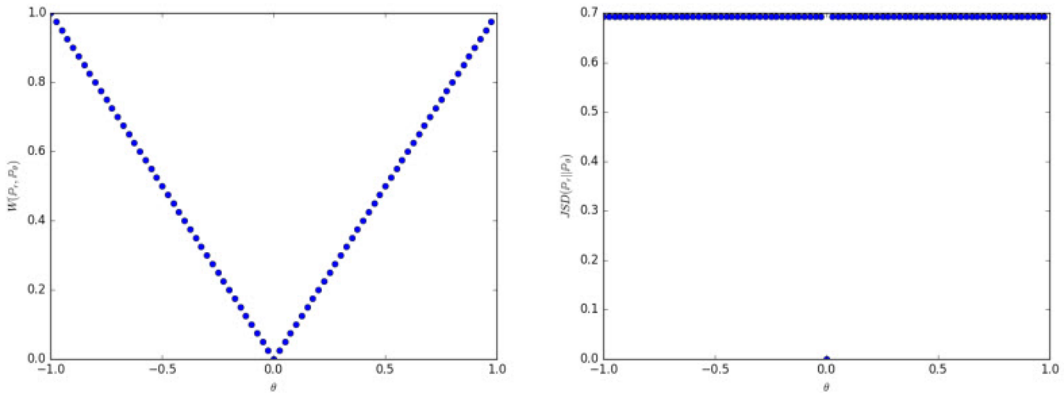
3. The Jensen-Shannon (JS) divergence

$$JS(P_0, P_\theta) = \begin{cases} \log 2 & \theta \neq 0 \\ 0 & \theta = 0 \end{cases} \tag{3.7}$$

4. The Earth-Mover (EM) distance or Wasserstein-1

$$W(P_0, P_\theta) = |\theta| \tag{3.8}$$

با $\theta_t \to 0$ دنباله‌ی $(P_{\theta_t})_{t \in N}$ به توزیع P. طبق EM distance همگرا می‌شود اما از طرف دیگر طبق مترهای JS، KL، reverse KL و TV divergences همگرایی نداریم. Wasserstein GAN یک شبکه‌ی GAN است که از



شکل ۱.۳: تفاوت متر Em با متر JS divergence

این متر در فرآیند یادگیری شبکه استفاده می‌کند [4]. الگوریتم یادگیری این شبکه به شکل زیر است.

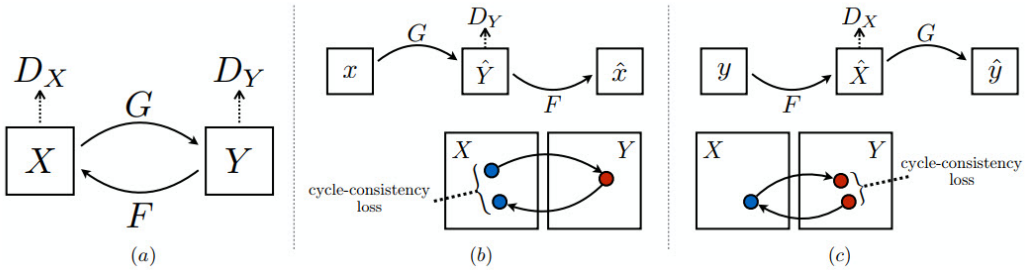
۲.۳ Cycle GANs

تبدیل تصویر به تصویر یکی از کلاس‌های مسائل در بینایی کامپیوتر است که هدف آن نگاشت تصویر ورودی به تصویر خروجی است. هدف شبکه‌ی Cycle GANs ایجاد یک Generator است که با گرفتن عکس ورودی (X) آن را به

نحوی به تصویر خروجی (Y) نگاشت کند که Discriminator متوجه غیر واقعی بودن این عکس‌ها نشود. برای مثال به عکس 3.2 توجه کنید.

شکل ۲.۳: مثال‌هایی از شبکه‌ی Cycle GANs

ایده‌ی اصلی در این شبکه، استفاده از مفهوم Cycle Consistency است. در این شبکه، بر خلاف حالت‌های معمول دو شبکه‌ی Generator و دو شبکه‌ی Discriminator داریم. شبکه‌ی Generator اول (G) سعی می‌کند نگاشتی از فضای حالت اول به حالت دوم یاد بگیرد و Discriminator اول در این بازی در مقابل Generator اول قرار می‌گیرد. شبکه‌ی Generator دوم (F) سعی می‌کند تصویر را از فضای دوم به تصویر اول بیاورد و Discriminator دوم در این بازی در مقابل Generator دوم قرار می‌گیرد.



شکل ۳.۳: Cycle Consistency

Cycle Consistency از ما می‌خواهد که $F(G(x)) \approx x$ و $G(F(y)) \approx y$ باشد. پس بنابراین، تابع خطای ما شامل دو ترم است. ترم اول خطای تخاصمی است که برای یاد گرفتن توزیع فضای دوم در نظر گرفته شده است.

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x))]$$

ترم دوم خطای cycle consistency است که مدل‌های G و F وارون هم دیگر باشند.

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1]$$

پس خطای نهایی به شکل زیر خواهد بود.

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F)$$

که λ پارامتر مسئله است. شبکه‌های نهایی از حل این مسئله به دست می‌آیند.

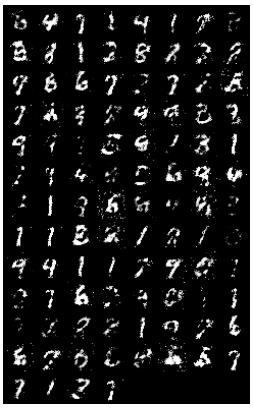
$$G^*, F^* = \min_{G,F} \max_{D_x, D_Y} L(G, F, D_X, D_Y)$$

فصل ۴

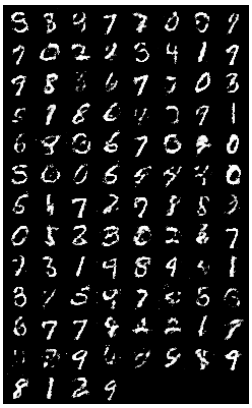
شبیه‌سازی

برای برخورد با چالش‌های شبیه‌سازی در طراحی شبکه‌های GAN من شبکه‌ای برای تولید ارقام طراحی کردم. از دیتاست Mnist برای داده‌های واقعی استفاده کردم. شبکه‌ی Generator و Discriminator ساختار یکسانی دارد. هر کدام یک شبکه‌ی fully-conected با یک لایه ورودی، دو لایه پنهان و یک لایه‌ی خروجی هستند. ورودی شبکه‌ی Generator یک بردار ۱۰۰ تایی از نویز است. فرآیند آموزش به وسیله‌ی تابع خطای Binary Cross Entropy است.

خروجی‌های رندومی از شبکه‌ی Generator در ایتریشن‌های خاصی گرفته‌ام که به صورت زیر است.



خروجی بعد از ۲۰ ایتریشن



خروجی بعد از ۶۰ ایتریشن

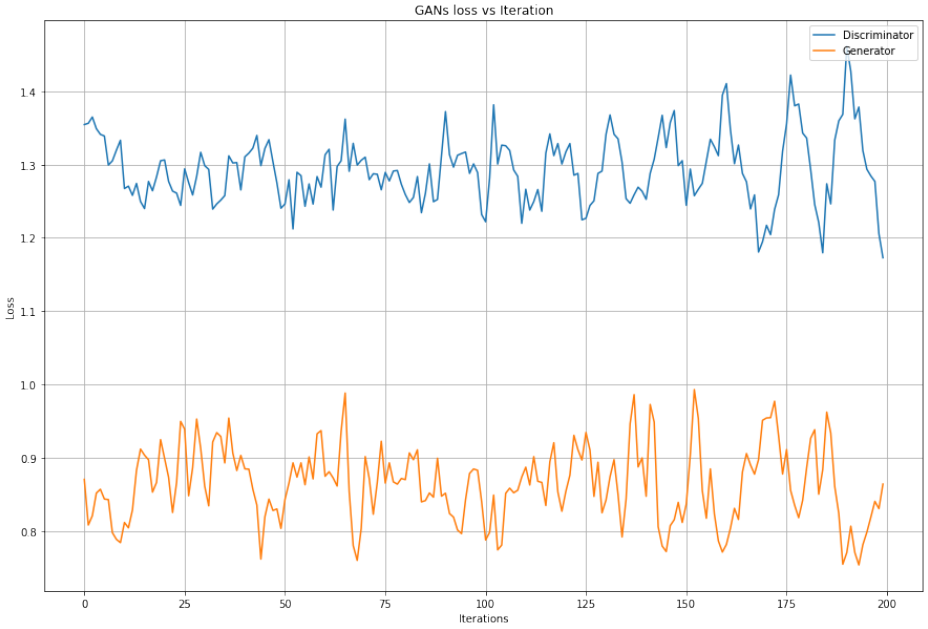
خروجی بعد از ۱۰۰ ایتریشن

خروجی بعد از ۱۴۰ ایتریشن

خروجی بعد از ۱۸۰ ایتریشن

خروجی بعد از ۲۰۰ ایتریشن

نمودار تابع ضرر روی ایتریشن‌های مختلف هم به شکل زیر در آمد.



نمودار لاس Generator و Discriminator

کتاب‌نامه

[1] Jun-Yan Zhu, T. Park, Ph. Isola A. A. Efros, *"Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"*

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, Sh. Ozair , A. Courville, Y. Bengioz, *"Generative Adversarial Nets"*

[3] Z. Lin, A. Khetan, G. Fanti, S. Oh, *"PacGAN: The power of two samples in generative adversarial networks"*

[4] M. Arjovsky, S. Chintala, and L. Bottou, *"Wasserstein GAN"*