Sharif University of Technology
Department of Computer Engineering
July 2022

Hadis Ahmadian
400211524
ML for Bioinformatics Assignment #5

Question ONE ……

الف)

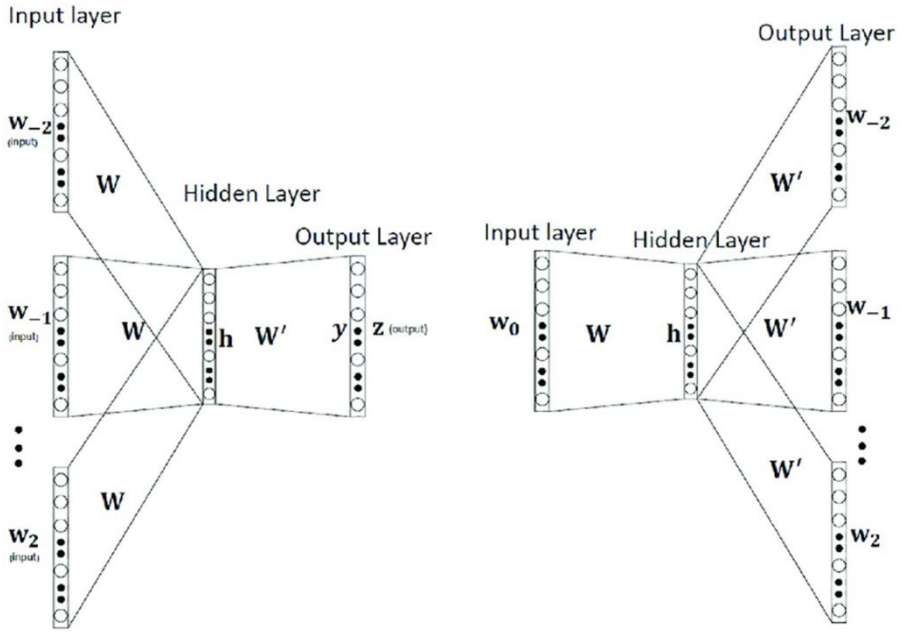
به طور کلی هدف **embedding** کلمه به وکتور در **word2vec** تبدیل کلمات به وکتور است به گونه ای که کلماتی که از لحاظ مفهومی واقعا به هم نزدیک هستند، وکتورهایشان در فضای خروجی نیز به هم نزدیک باشند. ایده ی اصلی در طراحی چنین شبکه ای این است که کلماتی که در یک متن به دفعات زیادی نزدیک هم قرار دارند مفاهیم مشابه و حتی یکسان دارند و با استفاده از این فرض شبکه متنی را به عنوان ورودی دریفات میکند و با توجه به کلمات همسایه ی کلمات آموزش میبینند.

Word2vec دو معماری دارد که تفاوت آنها در زیر مشهود است.

در **CBOW** معماری به این گونه است که پنجره ای از کلمات همسایه دریافت میشود و باید خود کلمه به عنوان خروجی داده شود. هم چنین ترتیب کلمات ورودی تاثیری در خروجی ندارد و صرفا خود کلمات مهم اند نه ترتیبشان (**bag of words**). این معماری

در **Skip-gram** معماری به این گونه است که کلمه به عنوان ورودی داده میشود و کلمات همسایه به عنوان خروجی دریافت میشود. هم چنین وزنی که به کلمات نزدیک تر نسبت داده میشود بیشتر از کلمات دور است در واقع فاصله ی کلمات در پنجره از کلمه ی فعلی مهم است.

هر دو معماری فقط یه لایه بین لایه های ورودی و خروجی دارند.



(a) CBOW (b) Skip-Gram

سایر تفاوت هایی که نویسندگان مقاله به آن اشاره کرده اند :

۱. به نظر میرسید **CBOW** سریع تر است در حالی که **skip-gram** برای کلمات **infrequent** بهتر عمل میکند.

۲. **CBOW** به دلیل داشتن تنها یک خروجی بیشتر مستعد این است که کلمات نزدیک از نظر دستور زبانی را بازگرداند مانند تخت : تخت ها. در حالی که **skip-gram** به دلیل داشتن چند خروجی کلماتی که از نزدیک معنایی نزدیک هستند را باز میگرداند مانند تخت : پتو.

ب)

گفتیم که ایده ی اصلی **word2vec** این است که کلماتی که در متن نزدیک به هم هستند احتمالا از نظر معنایی نیز به هم نزدیک هستند و با این فرضیه کلمات نزدیک به هم را به وکتور های نزدیک به هم **embed** میکند.

حال در **prod2vec** ایده این بوده است در ایمیل رسید های خریدهای اینترنی کالاهایی که در کنار هم خریده میشوند در متن رسید ایمیل نیز کنار هم آمده اند پس این متن به عنوان داده ی آموزشی به **word2vec** داده میشود (معماری **skip-gram**). سپس وقتی به شبکه ی آموزش داده شده نام یک کالا داده شود، کالاهای نزدیک به آن در خروجی داده میشود. (کاربرد برای پیشنهاد کالا در آنلاین شاپ ها و ...)

اشکال **prod2vec** این است که مستعد **overfit** شدن است به این علت که وقتی شخصی برای مثال یک مایوی شنای خاص خریداری کرده است و در کنار آن نیز یک ضدآفتاب خاص خریده لزوما به این معنا نیست که همان مایوی شنا همیشه در کنار همان کرم ضدآفتاب خریداری میشود. در یک دید کلی تر ارتباط بین این کالا ها به این علت است که اصولا لوازم شنا در کنار کرم لوازم مراقبت پوستی خریداری میشود. ولی **prod2vec** به این موضوع توجهی نمیکند و به طور جزیی روی نوع دقیق و خاص کالاها متمرکز میشود. و مثلا به مواردی مانند اینکه کالاهای بعدی ممکن است در کتگوری کالای فعلی باشند یا اینکه ارتباط کتگوری های کالای نزدیک به هم چیست دقتی نمیکند.

MetaProd2vec برای حل این مشکل ایجاد شده به این صورت که به جای در نظر گرفتن **"فقط" خود کالاها یک سری metadata** نیز از آن ها استخراج میکند و آن ها را نیز در پیش بینی خود دخیل میکند. برای مثال اگر کالا یک پیراهن خاص باشد متا دیتای آن میتواند رنگ آن سایز آن تولید کننده دسته بندی لباس(مثلا پیراهن مردانه ی خانگی) و جنس آن باشد. به این صورت دیگر روی خود کالاها به طور خاص فیت نخواهد شد.

با جزییات بیشتر اگر بخواهیم **objective function** این روش ها را مقایسه کنیم میبینم که در تابع هدف **Metaprod2vec** یک جمله ی **regularization** با ضریب λ اضافه شده است که متا دیتا را نیز در نظر میگیرد. در **prod2vec** فقط درستی پیش بینی کالای **J** را با توجه به کالای **I** در نظر میگیرد اما **metaprod2vec** در بخش رگیولاریزیشنش چهار مورد دیگر را در نظر میگیرد:

$$L_{\text{P2V}} = L_{J|I}(\theta)$$

$$L_{MP2V} = L_{J|I} + \lambda \times (L_{M|I} + L_{J|M} + L_{M|M} + L_{I|M})$$

LI | M : weighted cross-entropy between the observed conditional probability of input product ids given their metadata and the predicted conditional probability.

LJ | M :weighted cross-entropy between the observed conditional probability of surrounding product ids given the input products' metadata and the predicted conditional probability.

LM | I : weighted cross-entropy between the observed conditional probability of surrounding products' metadata values given input products and the predicted conditional probability.

LM | M : weighted cross-entropy between the observed conditional probability of surrounding products' metadata values given input products metadata and the predicted conditional probability.

مقایسه ی معماری p2v و mp2v را نیز در شکل زیر میبینیم که در mp2v کتگوری کالاها در ورودی و خروجی در نظر گرفته شده.

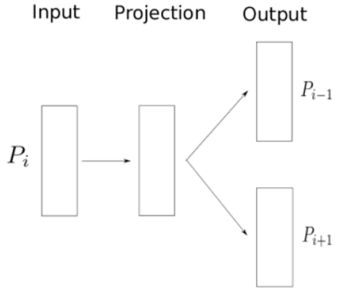


Figure 1: Prod2Vec Neural Net Architecture.

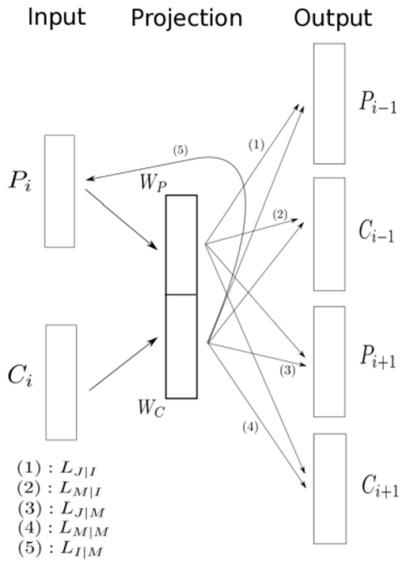
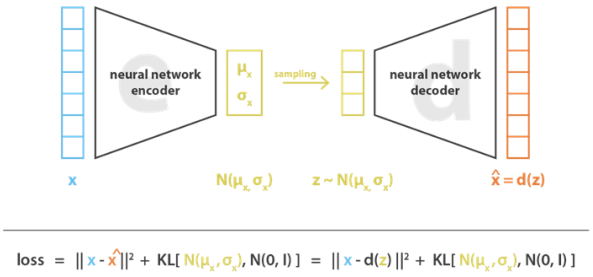


Figure 2: Meta-Prod2Vec Neural Net Architecture.

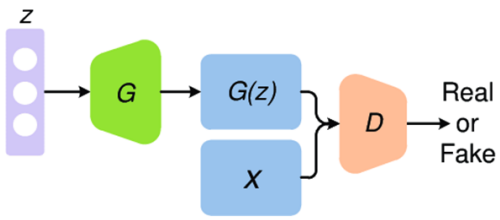
■

Question TWO.....

(آ)



VAE



GAN

VAE با بردن توزیعی از ورودی ها به یک فضای latent و بعد بازگرداندن آن ها به فضای اصلی و reconstruct کردن آن ها آموزش میبیند (یک معماری انکودر-دیکودر) به طوری که بعدا اگر یک نویز به جای فضای latent برای reconstruct کردن به دیکودر داده شود بتواند تصویری تولید کند. اما GAN یک ساختار تولیدکننده-تمیز دهنده دارد که تولید کننده تلاش میکند تصاویری تولید کند که تمیز دهنده نتواند آن را از تصاویر اصل تشخیص دهد.

تصاویر تولید شده توسط GAN عموما کیفیت بهتری دارند. اما دیتای آموزش بیشتری نیاز دارند و آموزش دادن آن ها سخت تر و کند تر است.

VAE چون برای آموزش داده ها را به فضای latent میبرد باعث میشود ویژگی های مهم داده و توزیع آن ها در فضای latent را به دست بیاوریم و هم چنین چون آن فضای latent کوچک تر است و در عین حال represent کننده ی داده ها هست، میتوان از آن ها به عنوان کاهش ابعاد استفاده کرد و نیاز به فضای ذخیره سازی کمتر داشت.

تفاوت در objective function : VAE دارای تابع هدفی است که میزان خوبی خروجی نهایی را میسنجد اما GAN تابعی که سعی در بهینه کردن آن دارد درواقع این است که تا چه مقدار میتواند تمیز دهنده را فریب دهد.

و به همین علت مقایسه ی خروجی دو VAE با محاسبات ریاضیاتی راحت تر است زیرا میتوان به طور ساده loss خروجی آن ها را با هم مقایسه کرد ولی مقایسه ی GAN ها سخت تر است و باید خروجی های نهایی آن ها را با چشم یا روش هایی مانند بینایی ماشین و ... با هم مقایسه کرد.

(ب)

در قسمت قبل گفتیم که کیفیت تصاویر GAN بهتر است یک علت آن میتواند وجود فضای latent در VAE باشد چون ممکن است امکان represent کردن تمام جزییات در آن فضا وجود نداشته باشد و برای همین هم تصاویر نهایی تولید شده جزییات کمتری دارند و دیگر اینکه VAE در اکثر مواقع فرض میکند توزیع داده ها گوسی است در حالی که در موارد زیادی میتواند اینگونه

نباشد و باز جزییاتی از توزیع اصلی داده هایمان از دست برود. یک دلیل نهایی نیز این است که VAE سعی در کمینه کردن تابع loss ساده تری دارد در حالی که همانطور که در قسمت قبل گفتیم GAN برای خود خروجی ها تابع loss مشخصی ندارد(تابع هدفش رو ی میزان فریب خوردن تمیز دهنده است) بنابراین میتواند ورای یک تابع لاس که برای آن چارچوب مشخص کند، ویژگی های پیچیده تری را یاد بگیرد.

قسمت امتیازی :

(آ)

دو تصویر بالا را در نظر بگیرید این دو تصویر به جز رنگ شیشه های عینک، کاملا شبیه هم هستند. حال اگر یک رویکرد داشته باشیم که این دو تصویر را به یک فضای latent کوچک تر embed کند وکتور های نهایی ممکن است بسیار از یک دیگر متفاوت باشد زیرا ویژگی های متفاوت در داده ی اولیه در یکدیگر entangled بوده اند و با تغییر یکی از آن ها چون تمام درایه های وکتور به نحوی به آن وابسته بوده، تغییر خواهد کرد. هدف از disentanglement این است که بتوانیم ویژگی های تصویر را در مستقل از یکدیگر در فضای کوچک تر داشته باشیم یعنی مثلا یک مولفه از وکتور embed شده رنگ شیشه ی عینک باشد و فقط همان یک مولفه در فضای embed شده تغییر کند و مولفه های دیگری به طور مستقل برای ویژگی های دیگر باشند مثل رنگ مو، رنگ پوست و... و آن ها با تغییر رنگ شیشه ی عینک، درایه ی نظیر بقیه ویژگی ها در وکتور فضای latent، تغییری نکنند.

کاربرد : این باعث میشود ۱. فضای latent ما تفسیرپذیر تر باشد. ۲. به مشکل ذکر شده در VAE ها به این نحو کمک میکند که اگر مثلا تمام داده های ورودی ما انسان هایی با پوست تیره بودند، چون ویژگی ها در فضای latent از هم مستقل شدند، ما بتوانیم با وکتور در فضای latent که درایه ی مربوط به رنگ پوستش را تغییر دادیم، انسان هایی با پوست روشن تر در داده های خروجی تولید کنیم.

(ب) تابع loss برای VAE در حالت عادی به صورت $L_{\mathsf{VAE}} = -\log p_\theta(\mathbf{x}) + D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$ میباشد، روش دیگری که برای رسیدن به هدف نهایی همین تابع هدف هست نوشت آن به صورت یک مساله ی بیشینه سازی است به صورتی که در آن احتمال تولید دیتای واقعی بیشینه شود در حالی که تا حد ممکن توزیع دیتای واقعی و approximate posterior به یکدیگر نزدیک باشند (از یک مقدار مثبت دلتا کوچکتر باشد) :

$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})]$$
$$\text{subject to } D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) < \delta$$

حال اگر این مساله ی بیشینه سازی را با روش لاگرانژ بنویسیم به عبارت زیر میرسیم که در آن بتا یک ضریب مثبت است :

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta(D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) - \delta)$$

اگر شکل باز شده ی ضرب ها را بنویسیم به عبارت زیر میرسیم و چون بتا و دلتا هر دو طبق فرض ما ضرایب مثبتی بودند حذف حاصل ضربشان عبارت را کوچکتر میکند :

$$\begin{aligned}&=\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) + \beta\delta \\ &\geq\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))\end{aligned}$$

پس در نهایت به تابع هدف زیر رسیدیم که در آن ضریب بتا موجود است و به این شکل است که ضریب بتا در تابع ظاهر میشود:

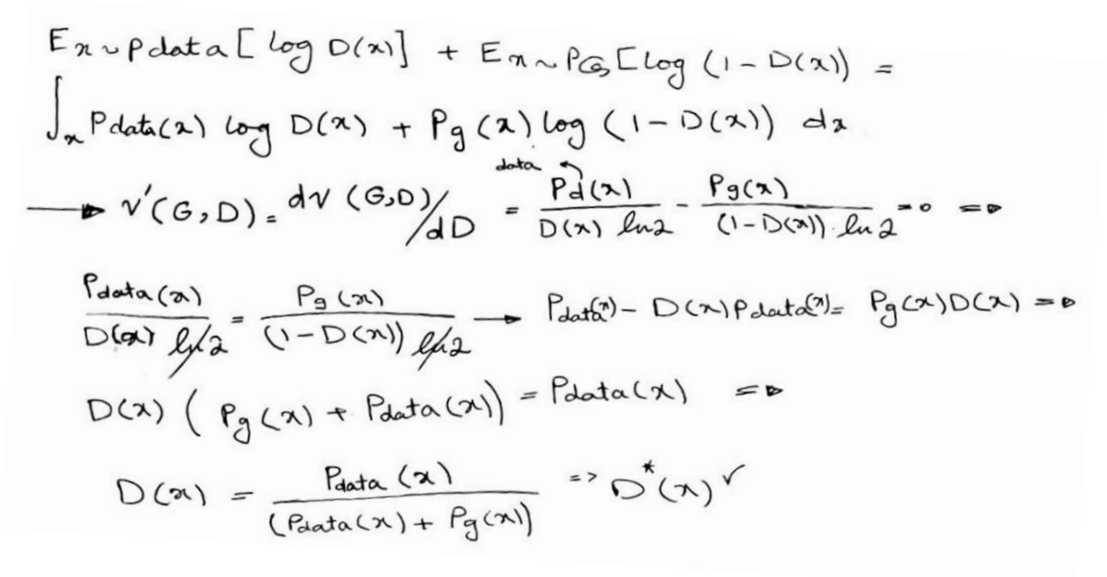
$$L_{\mathsf{BETA}}(\phi,\beta) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$

واضح است که اگر بتا را برابر با ۱ قراردهیم همان تابع هدف VAE ساده را خواهیم داشت. نکته ی مورد توجه این است که با بزرگ کردن بتا تابع هدف ما باعث میشود **disentanglement** بهتری داشته باشیم.

■

Question THREE .....

الف) برای یافتن مقدار بهینه باید مشتق گرفته و برابر صفر قرار دهیم:



ب)

$$JSD(P\|Q) = \tfrac{1}{2} D_{KL}(P\|M) + \tfrac{1}{2} D_{KL}(Q\|M) \quad M = \frac{P+Q}{2}$$

KL Divergence $\Rightarrow D_{KL}(P\|Q) = \int P \log\left(\frac{P}{Q}\right) dH$

$$V(G,D^*) = \int_x P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x)+P_g(x)} dx + \int_x P_g(x) \log\left(1 - \frac{P_{data}(x)}{P_{data}(x)+P_g(x)}\right) dx$$

$$= \frac{P_g(x)}{P_{data}(x)+P_g(x)}$$

$$= \int_x P_{data}(x) \log \frac{\frac{1}{2} \times P_{data}(x)\, dx}{\frac{1}{2}(P_{data}(x)+P_g(x))} + \int_x P_g(x) \log \frac{\frac{1}{2} P_g(x)}{\frac{1}{2}(P_g(x)+P_{data}(x))} dx$$

$$= \int_x P_{data}(x) \log \frac{P_{data}(x)}{(P_{data}(x)+P_g(x))/2} + \int_x P_g(x) \log \frac{P_g(x)}{(P_{data}(x)+P_g(x))/2} dx + \int_x -\log 2 \,(P_{data}(x)+P_g(x))$$

$$E = E_{P_{data}}[-\log r] + E_{P_g}[-\log r]$$

$$= D_{KL}\left(P_{data}(x) \,\|\, \frac{P_{data}(x)+P_g(x)}{2}\right) + D_{KL}\left(P_g(x) \,\|\, \frac{P_{data}(x)+P_g(x)}{2}\right) + E$$

$$= \boxed{2\, JSD(P_{data}(x) \| P_g(x)) - \log(4)}$$

توضیح: ما مقادیر GAN بهینه را وقتی $P_{data} = P_g$ در حالت انتقال می‌دانیم و در آن نقطه، E برابر $-\log(4)$ می‌شود

ج) در مراحل اولیه که **generator** هنوز آموزش دیده نشده و تصاویر رندوم تولید میکند، تشخیص این تصاویر برای **discriminator** بسیار راحت است و به طور کلی تسک جنریتور تسک سخت تری نسبت به دیسکریمینیتور است و این موضوع در مراحل اولیه به مراتب بیشتر است. بنابراین میتوان نتیجه گرفت که در مراحل اولیه بهتر است میزان گرادیان جنریتور بیشتر باشد. همانطور که میتوان دید شیب تابع $\log(-D(x))$ نسبت به تابع $\log(1-D(x))$ در مراحل اولیه بیشتر است و این گرادیان بیشتر برای مراحل اولیه، مشکل گفته شده را حل میکند بنابراین $\log(-D(x))$ برای مشکل ذکر شده انتخاب بهتری است.

■

Question FOUR .....

الف)

$D(x) = \sigma(a)$ a is logits $\rightarrow$ logits نسبت به $\log(1-D(x))$ گرادیان $\Rightarrow$

$$\rightarrow \frac{d \log(1-\sigma(a))}{da} = \frac{d(1-\sigma(a))}{da} \times \frac{1}{1-\sigma(a)} =$$

$$\left(\frac{-d\sigma(a)}{a}\right)\left(\frac{1}{1-\sigma(a)}\right) = -(1-\sigma(a))\sigma(a) \times \frac{1}{1-\sigma(a)} = -\sigma(a)$$

ب)

همانطور که در سوال گفته شده دامنه ی عکس های اصلی و شبکه ی مولد با هم همپوشانی ندارد و تمیز دهنده نیز تقریبا دارد بی نقص عمل میکند. این به این معناست که تمیز دهنده خطایی ندارد پس مقدار $\sigma(a)$ در عبارت $\log(1-\sigma(a))$ تقریبا خواهد بود و از آنجا که گرادیان این عبارت نیز محاسبه شد و برابر با $-\sigma(a)$ بود، مقدار گرادیان نیز تقریبا صفر میشود و وقتی در طی **back propagate** گرادیان صفر به شبکه ی**G** برسد، دیگر آپدیت نمیشود و عملا آموزش نمیبینند.

■

Question FIVE .....

(آ) اگر جنس داده های ما به این گونه باشند که چند **mode** داشته باشند به این معنا که **distribiution** داده ها چندین قله داشته باشد دچار این مشکل میشویم. مثلا داده ی ارقام دست نویس را تصور کنید این داده ها دارای ۱۰ **mode** هستند (اعداد ۰ تا ۹) که ما انتظار داریم اگر **GAN** ما واقعا خوب باشد بتواند تمام این ارقام را تولید کند اما مشکلی که پیش میاید به این شرح است : فرض کنید در قدم های ابتدایی ما **generator** دارد ۱ های خوبی تولید میکند اما سایر ارقام را به خوبی تولید نمیکند، حال تمیزدهنده برای ۱ های تولیدی فریب میخورد و جنریتور فیدبک خوبی دریافت میکند اما برای سایر اعداد تمیزدهنده به خوبی تصاویر را تشخیص میدهد و جنریتور فیدبک بدی برای آن ها دریافت میکند. این باعث میشود جنریتور به این سمت بایاس شود که فقط ۱ تولید کند و سایر **mode** های داده ی ما را تولید نکند تا برای آن ها **loss** دریافت نکند و این گونه تنها یکی از **mode** های دیتاسیت ما پوشش داده میشود وسایر **Mode** ها تولید نمیشوند که به این مشکل **Mode collapse** گفته میشود.

برای اینکه **mode-collapse** بهبود یابد ما باید تمیزدهنده را به حدی خوب آموزش داده باشیم که نمونه های تکراری جنریتور که یک **Mode** تکراری تولید میکند را بفهمد و در صورت دیدن آن ها فیک بودنشان را اعلام کند . درواقع متوجه تفاوت داده های اصلی که چندین **Mode** دارد با داده های جنریتور که فقط یک **mode** دارد بشود و نگذارد جنریتور تنها با تولید یک **Mode** میزان **loss** کمی داشته باشد و به عبارتی به یک بالانس خوب بین قدرت جنریتور و تمیزدهنده شده باشیم. اما میدانیم که با تابع هزینه فعلی **GAN** اگر تمیزدهنده خیلی خوب شود و به حالت اپتیمال نزدیک شود دچار **vanishing gradiant** میشویم (در سوالات قبل درمورد آن بحث شد) به همین علت طراحی شبکه ای که تابع هزینه اش متفاوت باشد و این مشکل را نداشته باشد. در قسمت ب در این مورد بیشتر توضیح داده میشود.

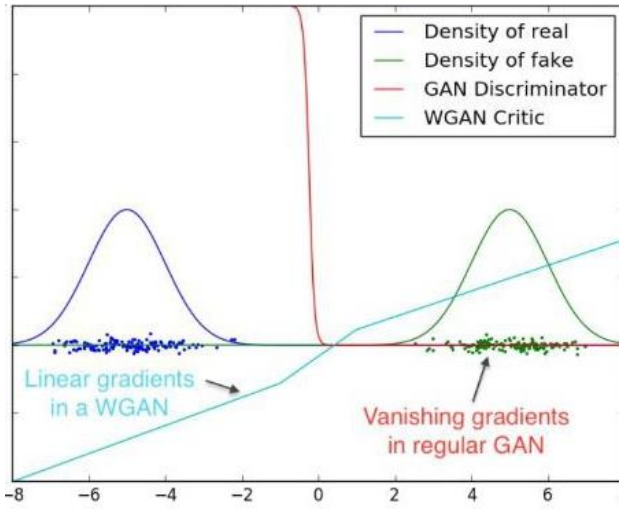
(ب)

تفاوت **W-GAN** با **GAN** در تابع **loss** آن و استفاده از **critic** به جای **discriminator** است. در **GAN** ما یک تمیزدهنده داشتیم که خروجی آن به این صورت بود که آیا تصویر تولیدی اصل است یا فیک اما **critic** یا منتقد در **W_GAN** تصویر تولیدی را دریافت میکند و یک امتیاز به آن میدهد. در واقع خروجیش میتواند هر امتیاز عددی در سرتاسر دامنه ی اعداد حقیقی باشد که هرچقدر تصویر تولیدی بهتر باشد امتیاز بهتری به آن میدهد.

گفتیم که برای نداشتن **mode-collapse** ما میخواهیم تمیزدهنده به حدی قوی باشد و نزدیک به اپتیمال باشد که نمونه های تکراری جنریتور که یک **Mode** تکراری تولید میکند را بفهمد و در صورت دیدن آن ها فیک بودنشان را اعلام کند. ولی اگر تمیز

دهنده به حالت اپتیمال نزدیک باشد برای آموزش دادن جنریتور به دلیل داشتن **vanishing gradiant** دچار مشکل میشویم، در ادامه تفاوت های تابع هدف **W-GAN** و تاثیر آن را توضیح میدهیم.

میدانیم تابع هدف برای **GAN** به شکل $\mathbb{E}_{x\sim q_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$ است در حالی که **W_GAN** از **Wasserstein-loss** استفاده میکند که به صورت $\mathbb{E}_{x\sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim \mathbb{P}_\theta}[f(x)]$ تعریف میشود. تفاوت این دو تابع را میتوانید در شکل زیر ببینید :



همانطور که مشهود است تابع هزینه تمیز دهنده **GAN** در دو انتهای خود رفتار مجانبی دارد ولی **w-loss** به دلیل نداشتن **log** در تعریف خود این رفتار را ندارد. وقتی توزیع تصاویر تولیدی جنریتور از تصاویر اصلی بسیار دور باشد (مثل دو توزیع سبز و آبی) تمیز دهنده میتواند بسیار خوب عمل کند و همواره تصویر درست از حقیقی را تشخیص دهد (این مورد در اوایل آموزش **GAN** بسیار اتفاق میافتد) وقتی چنین اتفاقی بیوفتد، میتوان دید هنگام این تفاوت زیاد بین توزیع واقعی و توزیع تولید شده، تابع تمیزدهنده ی **GAN** در ناحیه ای از خود قرار دارد که بسیار به خط افقی نزدیک شده است یعنی گرادیان آن تقریبا ۰ خواهد بود و وقتی این گرادیان به جنریتور برسد دیگر آموزش نمیبیند چون گرادیان تقریبا ۰ بوده و عملا مشکل گرادیان ونیشینگ را خواهیم داشت. در حالی که برای منتقد یا همان **critic** این مشکل را نداریم و در نقاط فاصله ی زیاد توزیع ها نیز گرادیان قابل توجهی داریم و گرادیانی که به جنریتور میرسد ۰ نخواهد بود و مشکل ونیشینگ گرادیان را نداریم پس میتوانیم بدون نگرانی از ونیش شدن گرادیان، اول منتقد را خیلی قوی کنیم که داده های تکراری را نیز تشخیص دهد و مشکل **mode collapse** بهبود یابد.

■

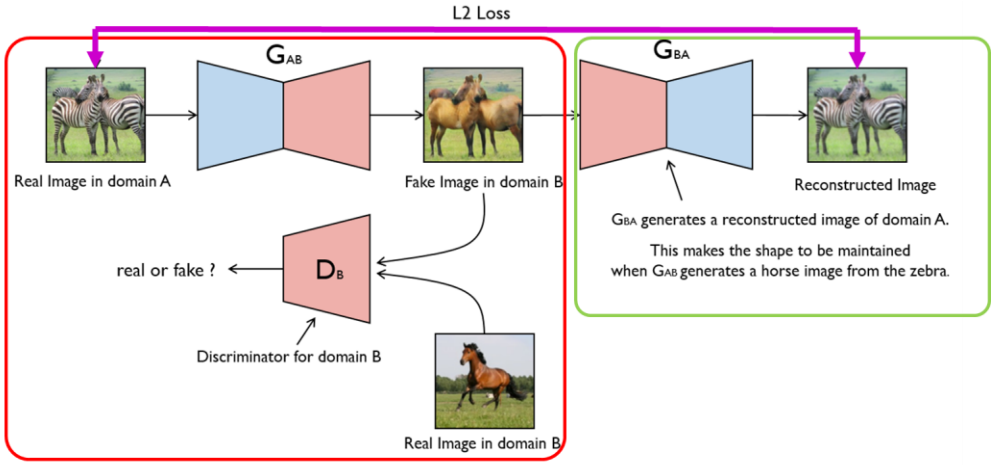
Question SIX .....

به ظور کلی مسئله ی **image2image translation** به این معناست که ما تصویری را از یک دامنه به دامنه ی دیگری ببریم برای مثال تصویری از یک منظره که در تابستان است (دامنه ی **A**) به همان تصویر در زمستان تبدیل کنیم (دامنه ی **B**).

اگر دیتابیسی داشته باشیم که شامل تعداد زیادی تصویر جفت در دو دامنه باشد مثلا چندین تصویر از مناظر و نقاشی های نظیر آن ها، به این مسئله **paired-image2image translation** گفته میشود و با مدل هایی که تا اینجا دیدیم حل آن ها ممکن است اما چالش اصلی این است که همیشه این جفت تصویر ها در دسترس نیستند، یافتن آن ها بسیار سخت یا حتی غیر ممکن است مثلا تصور کنید تعداد نقاشی داریم که همگی از ذهن نقاش تراوش شده اند و اصلا منظره ی نظیر آن ها وجود ندارد که بخواهیم به صورتی جفتی، مناظر و نقاشی نظیر آن ها را برای آموزش به شبکه بدهیم یا میخواهیم سبک نقاشی یک نقاش را تولید کنیم و او نقاشی های خیلی کمی دارد و دیگر از دنیا رفته و نمیتواند داده های کافی برای ما تولید کند و... پس به علل زیادی داشتن چنین دیتاسیتی از جفت تصاویر در دو دامنه سخت یا غیرممکن است.

حال اینجا با مساله ی **unpaired-image2image translation** روبرو میشویم که در آن چندین تصویر از دامنه ی **A** داریم و چندین تصویر از دامنه ی **B** اما این تصاویر با یکدیگر جفت نیستند و هرکدام صرفا نمونه هایی از دامنه ی خود هستند مثلا تعداد زیادی عکس منظره و تعداد زیادی عکس نقاشی شده ی بی ارتباط به آن مناظر داشتن چنین دیتاستی به مراتب راحت تر از دیتاست جفت عکس ها است. حال میخواهیم مدلی داشته باشیم که با دیدن این دیتاست یاد بگیرد هر دامنه چیست و سپس اگر تصویری از دامنه ی **A** به آن دادیم بتواند آن را به دامنه ی **B** ببرد.

Cycle-GAN این مساله را برای ما حل کرده است که معماری آن به صورت زیر است در مثال زیر دامنه ی **A** گورخر و دامنه ی **B** اسب است :



در ابتدا قسمتی که با قرمز دور آن خط کشیده ام را توضیح میدهم:

در این بخش یک انکودر داریم که ابتدا تصاویر گورخر ها را به یک فضای **latent** میبرد حال اگر مقادیر حاصله را به یک **GAN** بدهیم که در آن تمیزدهند تصاویر خروجی **GAN** را با اسب های واقعی مقایسه میکند تا ببیند تصویر اصل است یا فیک، انتظار داریم بعد از آموزش در خروجی تصاویر اسب ببینیم اما این برای ما تضمین نمیکند که تصویر اسب خروجی دقیقا همان تصویر ورودی بوده باشد و فقط گورخر به اسب تبدیل شده باشد، ممکن است خروجی هر اسبی باشد.

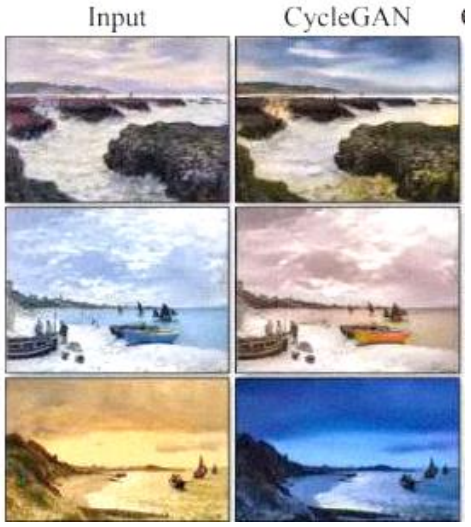
حال برای حل این موضوع قسمتی که دور آن خط سبز کشیده ام اضافه میشود:

در این قسمت اسبی که ما به عنوان خروجی از قسمت اول گرفتیم را باز با یک انکودر به یک فضای کوچکتر میبریم و مقادیر آن را به یک GAN دیگر میدهیم که این بار تلاش دارد اسب را به گورخر تبدیل کند و سپس چک میکنیم که آیا به تصویر اولیه ی خود بازگشتیم یا نه. (با استفاده از L2-loss خروجی گن دوم و ورودی گن اول را مقایسه میکنیم (خط بنفش رنگ)). به این شکل میتوانیم مطمئن شویم تصویر اسبی که در وسط شبکه ساخته شده همان تصویر گورخر ورودی است که به دامنه ی اسب برده شده.

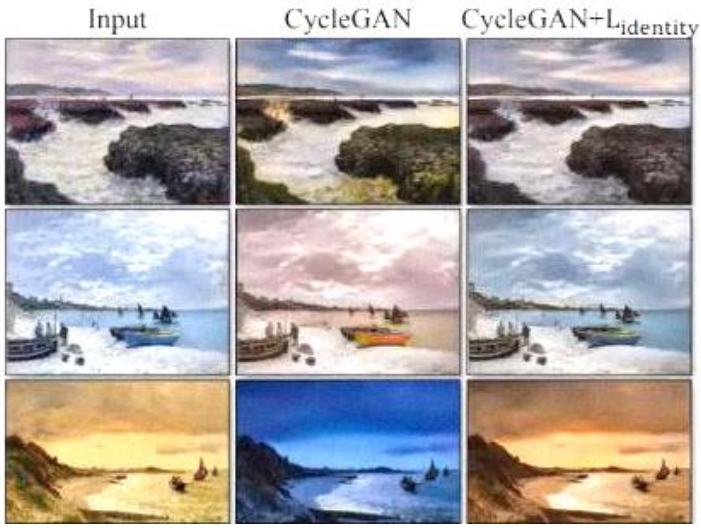
به دلیل وجود این ساختار حلقه مانند که اول از دامنه ی A به دامنه ی B میرویم و بعد از دامنه ی B به دامنه ی A بازمیگردیم، به این ساختار cycle-GAN گفته میشود. به این خاصیت که ورودی اولیه دامنهA طی تبدیل به دامنه ی B و دوباره تبدیل به دامنه ی A تغییر نکند وl2-loss آن کمترین حد ممکن باشد cycle consistency گفته میشود.

حال علاوه بر این l2 loss که بین اولین ورودی و آخرین خروجی است یک مورد دیگر داریم که باید آن را هم در بهینه کنیم.

ما نمیخواهیم رنگ تصویر در طی این تبدیل تغییر زیادی داشته باشد. اگر مدلی که ما طراحی کردیم تا از دامنه ی A به B ببرد این خاصیت را داشته باشد انتظار داریم اگر عکسی از دامنه ی B به آن بدهیم همان عکس را عینا به ما برگرداند چون عملا در دامنه ی مورد انتظار بوده و تغییری روی آن لازم نیست اماموضوعی که هست ممکن است مدل روی رنگ عکس هایی که دیده است بایاس شده باشد و به گونه ای رنگ عکس را تغییر دهد. برای مثال در شکل زیر میبینیم که مدلی که طراحی شده است تا عکس را به منظره تبدیل کند وقتی به آن عکس میدهیم، همان عکس را برگرداند اما میبینیم که روی رنگ عکس هایی که قبلا دیده بایاس شده و هرچند همان عکس را برمیگرداند ولی رنگش را تغییر میدهد.



برای حل این مشکل یک identity loss نیز به تابع هدف اضافه میکنیم به گونه ای رنگ عکس در شرایط گفته شده را حفظ کند.وcolor preservation داشته باشد. پس به ظور خلاصه میخواهیم وقتی ورودی از جنس دامنه ی مورد انتظار خروجی بود هیچ چیز در آن حتی رنگش عوض نشود که با افرودن Lidentity به این هدف میرسیم و به این ویژگی identity mapping گفته میشود.


Input
CycleGAN
CycleGAN+L identity

■

Question SEVEN ……..……...

الف) تابع **objective** برای **random walk** به صورت زیر است :

$$\sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

که در آن **V** تمام رئوس، **u** راس مورد نظر، **NR(u)** مجموعه رئوس انتخاب شده به عنوان همسایه ی **u** و **v** یکی از آن همسایه های **u** است.

P(v | Zu) احتمال بودن **u** در همسایگی **v** است و به صورت زیر محاسبه میشود که مقدار درون **exp** ضرب داخلی بردارهای **embed** شده ی **u,v** هستند (این احتمال حالت یک سافت را مکس دارد):

$$P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_n)}$$

پس نهایتا **objective funvtion** به صورت زیر خواهد بود :

$$\sum_{u \in V} \sum_{v \in N_R(u)} -\log \frac{\exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_n)}$$

میتوان دید به دلیل اینکه در این فرمول یک بار روی کل **V** ها سیگما زده میشود و یه بار هم در مخرج روی کل **V** ها سیگما زده میشود از اردر **O(V²)** خواهد بود. و این مشکل اصلی الگوریتم است.

برای بهبود این مشکل از **negative sampling** باید استفاده کنیم و تابع هدف را با دو تغییر زیر تقریب میزنیم:

۱. به جای **exp** از سیگوید استفاده میکنیم. سیگموید نیز مقادیر بین ۰ و ۱ تولید میکند و میتوان از آن تعبیر احتمال داشت.
۲. به جای استفاده از تمام نمونه ها در مخرج برای محاسبه ی احتمال، تنها **k** سمپل رندوم را انتخاب میکنیم. تعداد **k** برای هر نود با درجه ی آن نود متناسب خواهد بود. به طور کلی **negative samling** یک فرم از **Noise Contrastive Estimation** است که برای تخمین زدن کمینه ی **log softmax** استفاده میشود. ایده ی کلی این است که **k** نود باید از نود هایی انتخاب شوند که همسایه ی **u** نیستند (در واقع نمونه های منفی باید سمپل شوند) اما در اپلیکیشن ها اصولا از این قید چشم پوشی میشود چون تعداد سمپل هایی که همسایه نیستند اصولا بسیار بیشتر از سمپل های همسایه است پس حالت دوم نیز قابل قبول خواهد بود.

تابع هدف به شکل زیر تغییر خواهد کرد :

$$\sum_{u \in V} \sum_{v \in N_R(u)} \left(-\log\left(\sigma(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v)\right) - \sum_{i=1}^{k} \log\left(\sigma(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_{n_i})\right) \right)$$

ب)

توضیح **TransE** : در این الگوریتم تلاش برای **embed** کردن یک گراف به یک وکتور است به گونه ای که در گراف اولیه یال ها نشان دهنده ی رابطه ای بین نود ها هستند (مثلا نود ها میتوانند اشخاص باشند و یال ها نشانه ی خویشاوند بودن، دوست داشتن، دشمن بودن و .. باشد یا نود ها اشخاص و غذا و یال ها نشان دهنده ی غذا های مورد علاقه باشند) در روش **TransE** هدف ما این است که دو موجودیت و رابطه ی بین آن ها را **embed** کنیم به گونه ای که اگر **a** با رابطه ی **r** به **b** برود در فضای **embed** شده، **a,b,r** سه وکتور باشند به گونه ای که **a+r=b** شود.

تابع هدف **TransE** به شکل زیر است که در آن بخش اول (در جمله ی دوم ماکسیمم گیری) برای این قید است که برای یک رابطه شرط جمع وکتور ها برقرار باشد و بخش دوم برای این است که ضمن اینکه هر موجودیت و رابطه ی بین آن ها درست **embed** شده باشد، موجودیت هایی که از هم دورند و رابطه هایشان متفاوت است واقعا از هم دور باشند و وروی هم نیوفتند.

$$L = \sum_{(h,r,t)\in S} \sum_{(h',r',t')\in S'} max(0,\ \gamma + d_r(h,t) - d_{r'}(h',\ t'))$$

$$S' = \{(h',r,t)|h' \in E, (h',r,t) \notin S\} \cup \{(h,r,t')|t' \in E, (h,r,t') \notin S\}$$

مثالی برای مشکل روابط یک به چند :

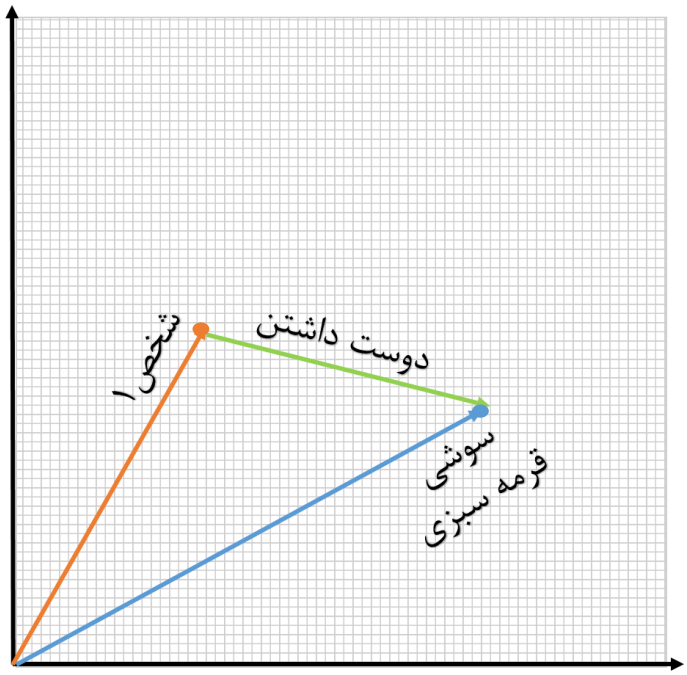
حال فرض کنید روابط زیر را داریم :

شخص۱ غذای۱ را دوست دارد.

شخص ۱ غذای۲ را دوست دارد.

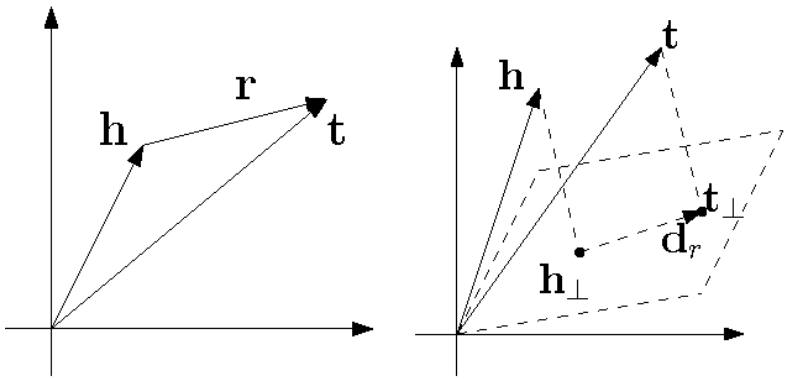
نوع رابطه و طرف اول رابطه در هردوی روابط بالا یکی است و فقط طرف دوم آن است که متفاوت است.

حال **TrensE** هر دو رابطه را مستقلا به طور جدا درست **embed** میکند ولی چون طرف اول روابط یکسان است پس طرف اول هر دو رابطه یک وکتور دارد هم چنین نوع رابطه نیز برای هر دوی آن ها یک وکتور دارد پس طرف دوم رابطه که غذای۱ و غذای۲ است روی هم میافتد در حالی که ممکن است بسیار از هم منفاوت باشند (مثلا قرمه سبزی و سوشی ☺)



واضح است که به همین شکل برای روابط چند به یک و چند به چند نیز دچار مشکل خواهیم بود.

ج) روشی به نام **TransH** وجود دارد که برای بهبود این مشکل وجود امده است. در این روش برای هر رابطه **a,r,b** ابتدا **a** و **b** در فضای مورد نظر **embed** میشود و به ازای رابطه یک بردار نرمال معادل یک صفحه نیز تولید میشود و **a,b** ابتدا روی آن صفحه تصویر میشوند و سپس بین تصویر آن ها روی صفحه، برداری به عنوان رابطه ی میان آن ها پیدا میشود. به این نحو دیگر مشکل ذکر شده وجود نخواهد داشت چون موجودیت ها ابتدا روی صفحه ی نظیر رابطه تصویر میشوند، روابط تداخل ذکر شده را نخواهند داشت.



(a) TransE (b) TransH

■
