تمرین سری پنجم

امیرحسین محمدی

۹۹۲۰۱۰۸۱

مسئله ۱. کاربرد شبکه‌های بازگشتی (۳+۶ نمره)

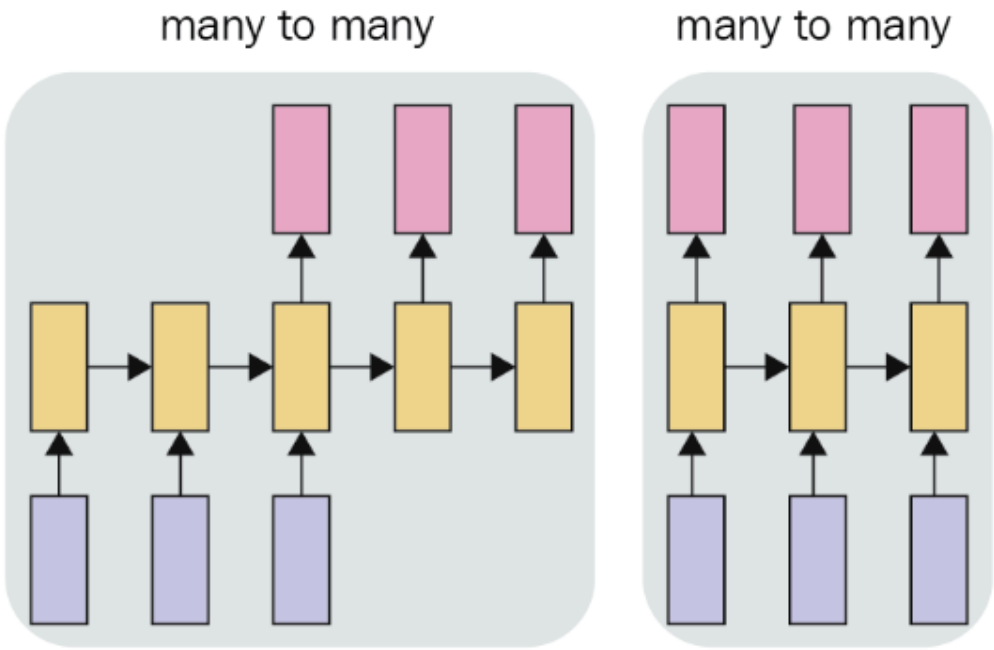
الف) شبکه‌های عصبی بازگشتی در کاربردهای مختلفی در زمینه پردازش رشته‌ها به کار می‌روند. در یک دسته‌بندی کلی این کاربردها را به دسته‌های یک‌ـ‌به‌ـ‌چند (one-to-many) ، چند‌ـ‌به‌ـ‌یک (many-to-one) و چند به چند (many-to-many) تقسیم می‌کنند. همچنین رشته‌های مورد استفاده در آخرین دسته ممکن است از نظر زمانی و ترتیبی، همگام یا غیرهمگام باشند. در مورد هر یک از این دسته‌ها توضیح دهید و از کاربردهای هریک لااقل یک مثال معرفی کنید. (۳ نمره)

ب) مسئله seq2seq در کدام یک از دسته‌های معرفی شده قرار می‌گیرد؟ معماری شبکه عصبی encoder-decoder که برای حل این مسئله به کار می‌رود را توضیح دهید. (۳ نمره)

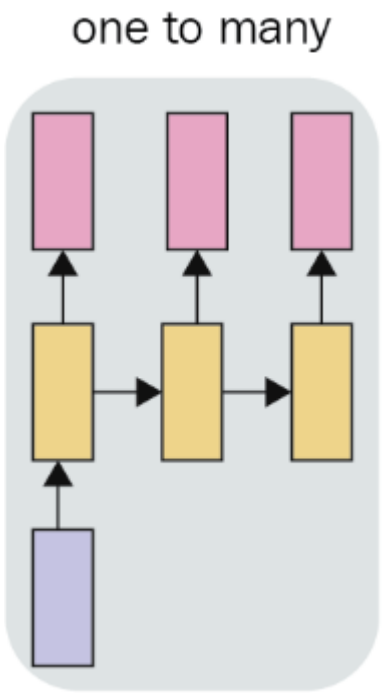
ج) مکانیزم توجه (attention) چیست و چگونه می‌تواند به شبکه‌های encoder-decoder کمک کند. در این مورد توضیح دهید. (۳ نمره)

الف

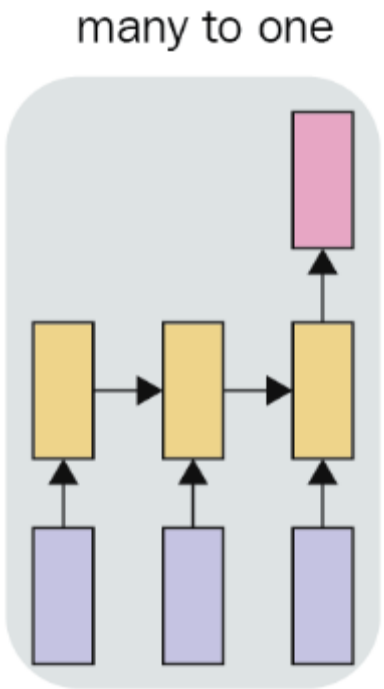
Many to Many: این شبکههای بازگشتی بر این اساس هستند که چند ورودی می گیرند (بنابراین چند Node ورودی دارند) و همچنین چند خروجی دارند، از مثال های مربوط به این شبکه ها می توان به تبدیل تصویر به یک متن یا متن به یک تصویر(زیرا تصویر مجموعه و توالی از پیکس هاست) یا تبدیل و ترجمه یک متن در یک زبان مثل انگلیسی به یک زبان دیگر مثل فارسی استفاده کرد.



One to many: در این مدل از شبکه های عصبی بازگشتی ما یک ورودی داریم (که به شبکه داده می شود) و در ازای آن چندین خروجی داریم مثلا تبدیل یک تصویر به یک عکس یا تبدیل یک کلمه به یک جمله.



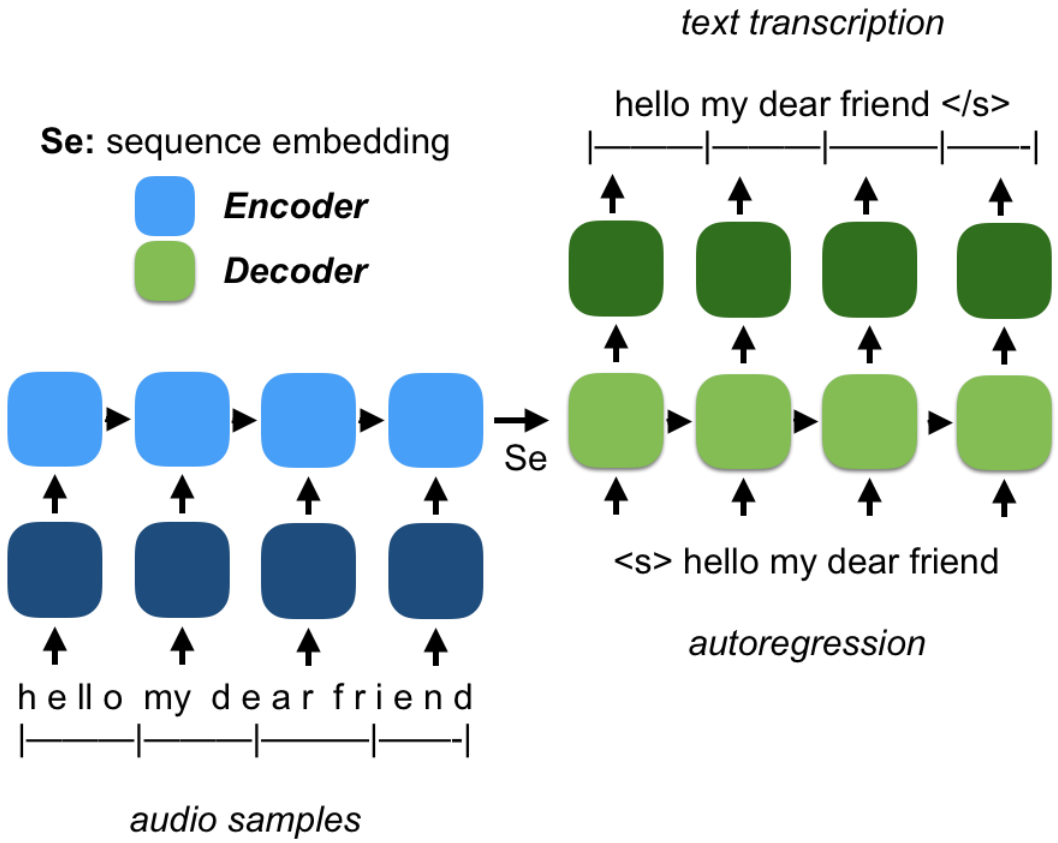
Many to one: در این مدل از شبکه های عصبی بازگشتی ما مجموعه از ورودیها داریم (که به شبکه داده می شود) و در ازای آن یک خروجی داریم مثلا تبدیل یک تصویر به یک کلمه یا تبدیل یک جمله به یک کلمه .



همگام و غیر همگام: در شبکه های عصبی بازگشتی همگام بودن به این معناست که ورودی در زمان های خاص (مثلا کلاک های معین) تبدیل به خروجی می شود مثلا در ترجمه (یک متن) از یک زبان به یک زبان دیگر هر کلمه یا ورودی که بیاید در همان آن به خروجی تبدیل نشود و در یک زبان معین ترجمه انجام شود و تبدیل از یک زبان به زبان دیگر اتفاق بیفتد و خروجی تولید شود. اما در شبکه های عصبی غیر همزمان هر موقع ورودی وارد شبکه شد بلافاصه به خروجی تبدیل می شود مثلا یک عکس بیاید و بلافاصله تبدیل به کلمه شود و یا ترجمه بلافاصله به ازای هر کلمه (هر ورودی) انجام شود.

ب

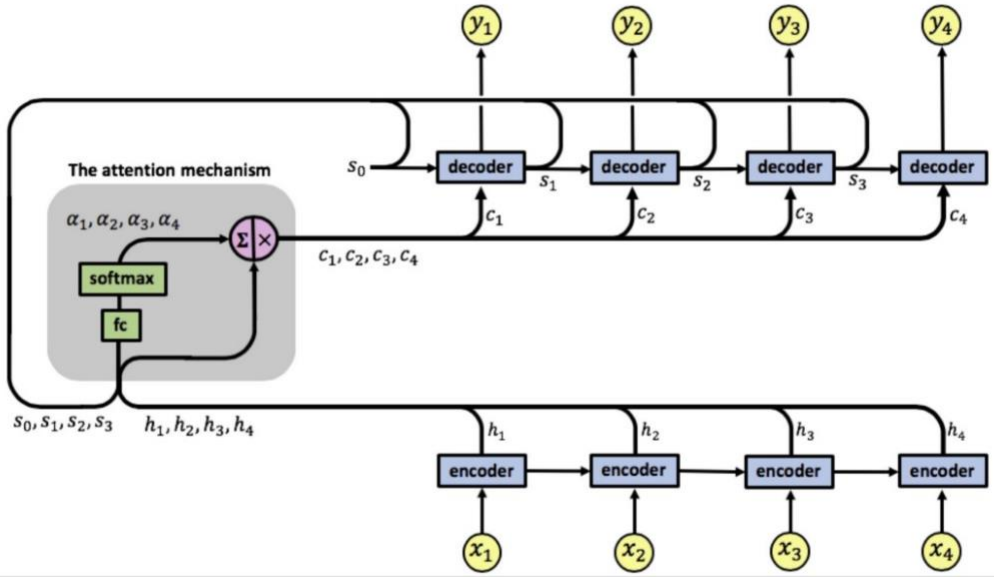
SeqtoSeq از نوع Many to Many است.



این معماری از دو بخش Encoder و Decoder تشکیل شده است، در بخش encoder خروجی هر مرحله (لحظه) به خروجی مرحله قبل به اضافه ورودی آن مرحله بستگی دارد. و این روند تا انتها ادامه دارد و در آخرین مرحله خروجی encoder به عنوان ورودی وارد بخش Decoder می شوند. در بخش Decoder هم در هر مرحله خروجی مرحله قبل به عنوان ورودی مرحله بعد وارد می شود و همچنین خروجی لحظه قبل با استفاده از یک تابع فعال ساز به عنوان ورودی مرحله بعد وارد می شود. و این روند نیز تکرار می شود تا به خروجی نهایی برسیم همانطور که از شکل مشخص است.

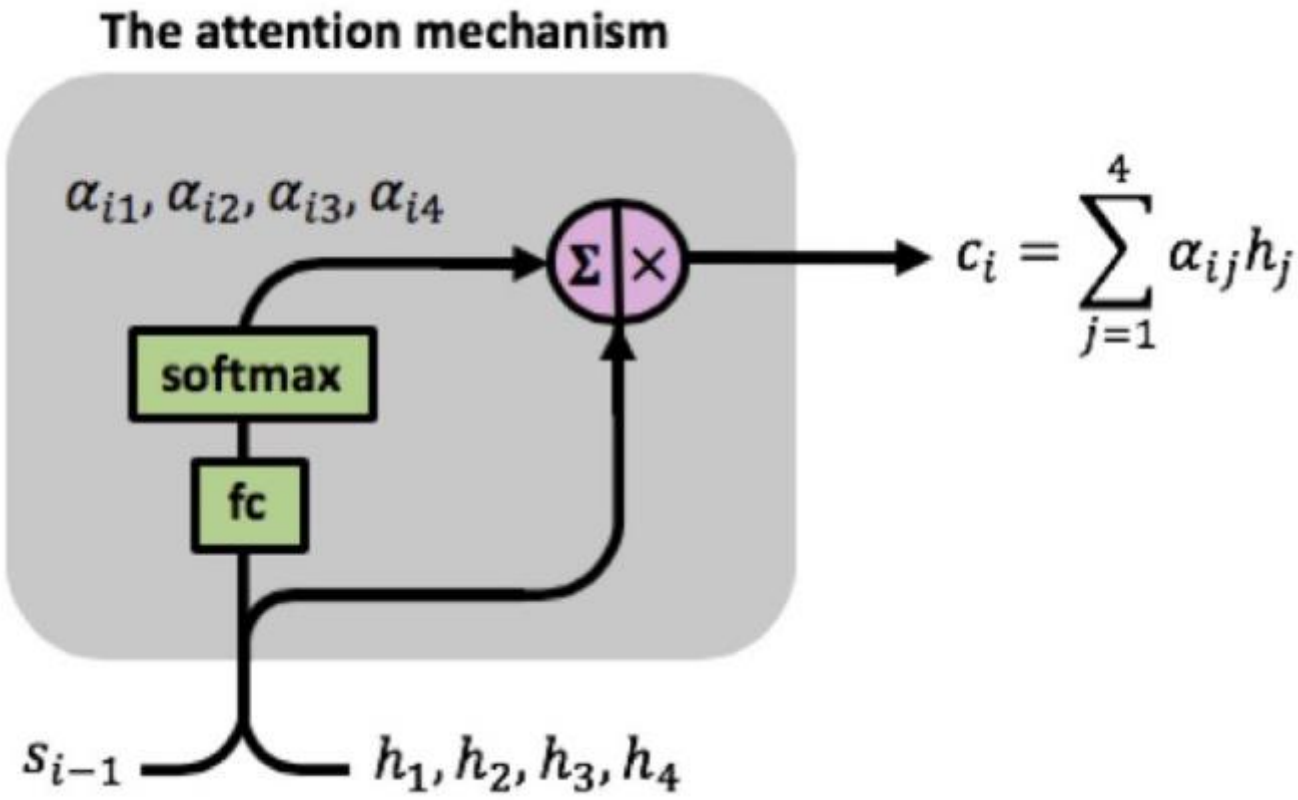
ج

همانطور که در شبکه های Seq to Seq دیدیم این شبکه ها شامل دو بخش Decoder و Encoder هستند. همانطور که دیدیم پس از انجام عملیات Encoder خروجی مرحله آخر باید به عنوان ورودی وارد Decoder شود، مشکل موجود در این مسائل این است که اگر مثلا طول یک متن (خروجی Encoder) بیش از یک اندازه معین شود امکان تمامی اطلاعات وجود ندارد. و این مسئله می تواند باعث افت دقت شود. مثلا اگر مثلا اگر در مسئله ترجمه از یک زبان به زبان دیگر بخواهیم کل متن را یک جا ترجمه کنیم ممکن است این مشکل گفته شده پیش بیاید (و ما بخشی از اطلاعات رو به خاطر مشکل ظرفیت بردار انتقال از دست بدهیم زیرا مجبوریم داده ها Compress کنیم مثلا)، اما اگر مثلا جمله به جمله یا کلمه به کلمه این اتفاق بیفتد این مشکل به وجود نمی آید. برای حل این مشکل می توانیم از روش Attention استفاده کنیم. در مکانیزیم Attention ما بر روی یک بخش از ورودی تمرکز میکنیم و یک بخش از خروجی را به تبع آن تولید می کنیم و این مسئله روند آموزش و یادگیری را آسان تر می کند و کارآیی را افزایش می دهد. در واقع در این روش با استفاده از یک روش دو طرفه ما داده ها میشکنیم به بخش های کوچکتر و با استفاده از آنها خروجی را تولید می کنیم.



همانطور که از شکل مقابل مشخص است، در این شبکه که از Attention استفاده شده است که در آن Encoder از 4 گام تشکیل شده است و خروجی این encoder سپس وارد Attention می شود(h1,h2,..) و همچنین Stateهای Decoder (S1,S2,..)هم وارد Attention می شود که با استفاده از این دو Context Vector ها تولید می شود(c1, c2,...). Context Vector، Decoder را قادر می سازد که روی یک بخش خاص از ورودی تمرکز کَند و خروجی مربوط به آنرا تولید کند. هر Context Vector حاصل جمع وزن داری از خروجی های Encoder است (h1,h2,..) که هر کدام از آنها حاوی اطلاعاتی است.

بنابراین با تمرکز بر روی قسمتهای پیرامون بردار i که بخشی از دنباله ی ورودی است، بردارهای h1، h2، h3، h4با وزن aijاندازه گیری و درجه ارتباط آنها با ورودی xj محاسبه می شود و خروجی لحظه i یعنی yi تولید میشود.



مسئله ۲. آموزش شبکه‌های بازگشتی (۹ نمره)

الف) می‌دانیم که الگوریتم backpropagation به صورت گسترده برای آموزش شبکه‌های عصبی feed-forward استفاده می‌شود. در مورد شبکه‌های بازگشتی از الگوریتم مشابهی به نام backpropgataion through time استفاده می‌شود. در مورد این الگوریتم توضیح دهید و تفاوت آن را با backpropagation عادی بیان کنید.(۳ نمره)

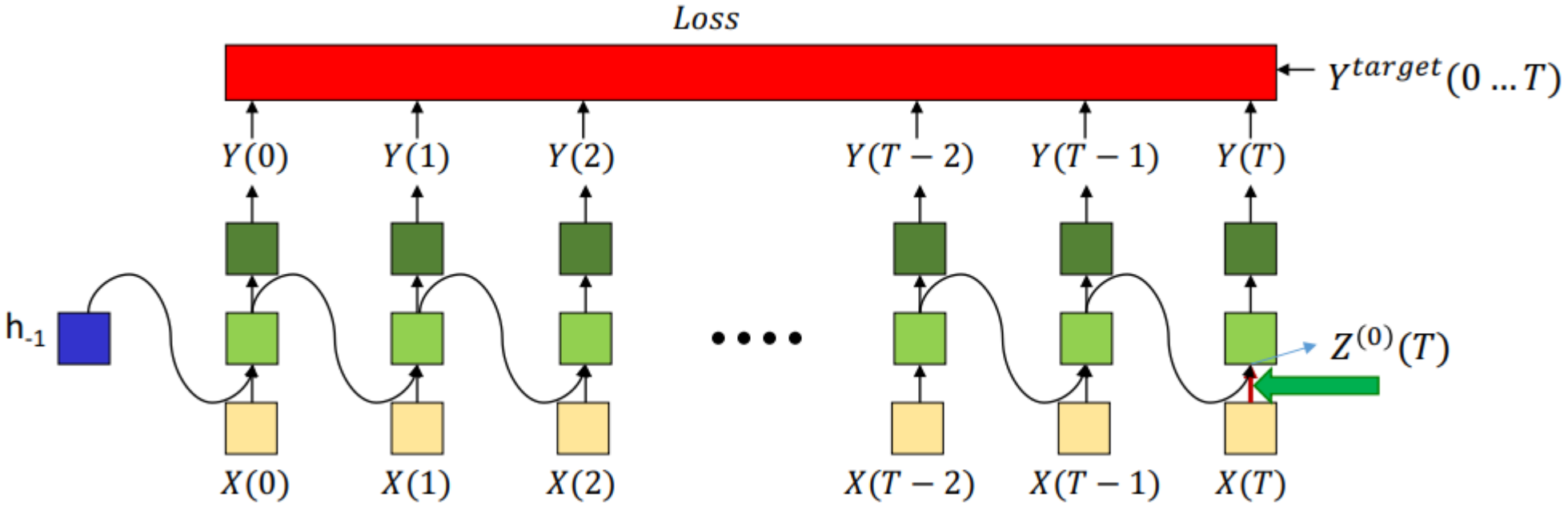
ب) یکی از مشکلات جدی در آموزش شبکه‌های عصبی بازگشتی مشکل vanishing gradient و gradient explosion می‌باشد. این دو مشکل را شرح دهید و توضیح دهید که چرا این مشکلات در شبکه‌های عصبی بازگشتی شدیدتر هستند.(۳ نمره)

پ) توضیح دهید که معماری LSTM چگونه می‌تواند به عنوان راه حلی برای رفع دو مشکل ذکر شده به کار گرفته شود.(۳ نمره)

الف

Back propagation در شبکه های عصبی بازگشتی بسیار شبیه به عملیات Back propagation در شبکه های عصبی معمولی است (feed-forward). اما در شبکه های عصبی feed-forward ما نسبت مشتق تابع loss را نسبت به وزن ها محاسبه می کنیم اما در شبکه های بازگشتی ما ما مشتق تابع loss را نسبت به خروجی در زمان t در نظر می گیریم برای همین مسئله به این نوع Back propagation در شبکه های عصبی بازگشتی Back Propagation Through Time می گویند.

فرض کنیم شبکه بازگشتی زیر را در نظر بگیریم:



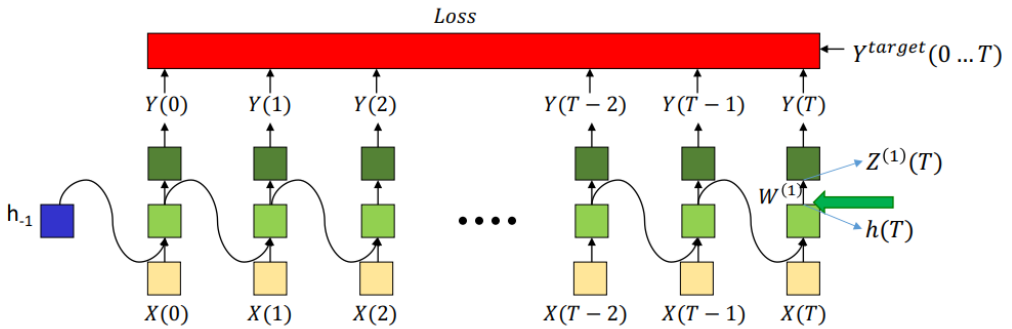
برای محاسبه ی عملیات Back Propagation Through Time در شبکه ی فرض شده داریم:

1

- محاسبه ی مشتق تابع loss در لحظه ی t نسبت به تابع فعالیت Z(T)

$$\nabla_{Z^{(1)}(T)} Loss = \nabla_{Z^{(1)}(T)} Y(T)\, \nabla_{Y(T)} Loss$$

$$\frac{dLoss}{dZ_i^{(1)}(T)} = \sum_j \frac{dLoss}{dY_j(T)} \frac{dY_j(T)}{dZ_i^{(1)}(T)}$$

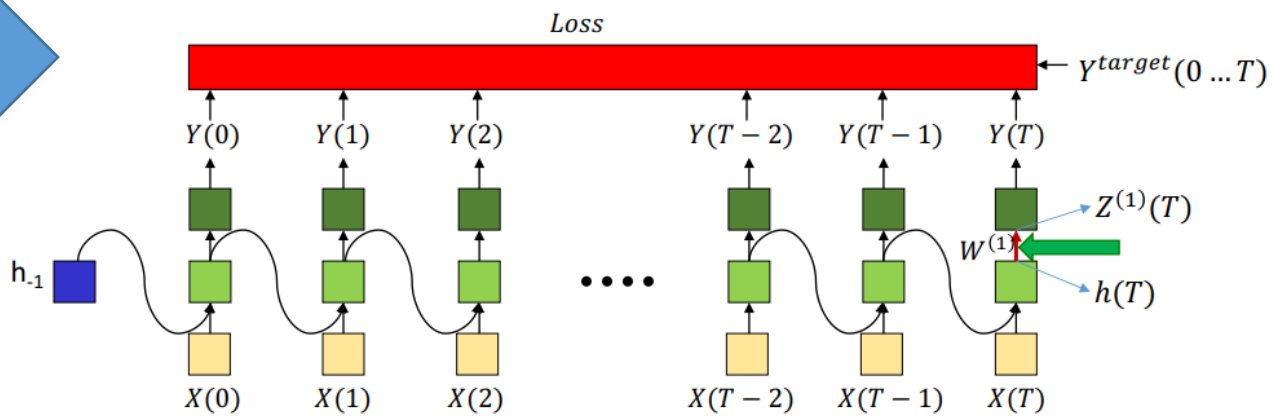


2

- محاسبه ی مشتق تابعloss نسبت به h(T)

$$\nabla_{h(T)} Loss = W^{(1)}\ \nabla_{Z^{(1)}(T)} Loss$$

$$\frac{dLoss}{dh_i(T)} = \sum_j \frac{dLoss}{dZ_j^{(1)}(T)} \frac{dZ_j^{(1)}(T)}{dh_i(T)} = \sum_j w_{ij}^{(1)} \frac{dLoss}{dZ_j^{(1)}(T)}$$

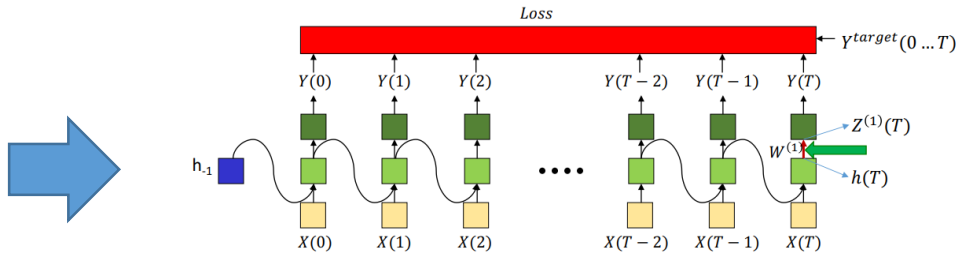


3

$$\nabla_{W^{(1)}} Loss = h(T) \nabla_{Z^{(1)}(T)} Loss$$

$$\frac{dLoss}{dw_{ij}^{(1)}} = \frac{dLoss}{dZ_j^{(1)}(T)} h_i(T)$$

- محاسبه ی مشتق تابعloss نسبت به W^1

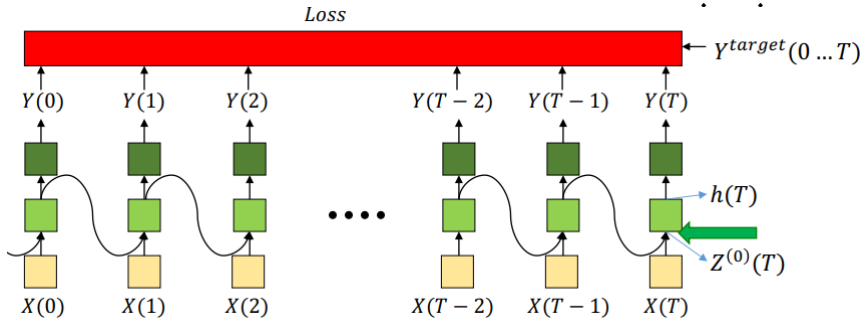


4

$$\nabla_{Z^{(0)}(T)} Loss = \nabla_{Z^{(0)}(T)} h(T) \nabla_{h(T)} Loss$$

$$\frac{dLoss}{dZ_i^{(0)}(T)} = \frac{dLoss}{dh_i(T)} \frac{dh_i(T)}{dZ_i^{(0)}(T)}$$

- محاسبه ی مشتق تابعloss نسبت به Z^0

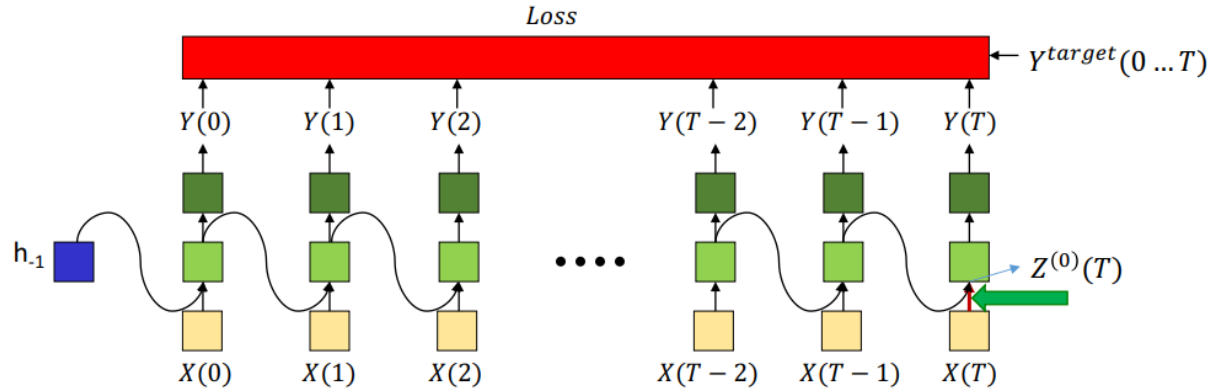


5

$$\nabla_{W^{(0)}} Loss = X(T) \nabla_{Z^{(0)}(T)} Loss$$

$$\frac{dloss}{dw_{ij}^{(0)}} = \frac{dLoss}{dZ_j^{(0)}(T)} X_i(T)$$

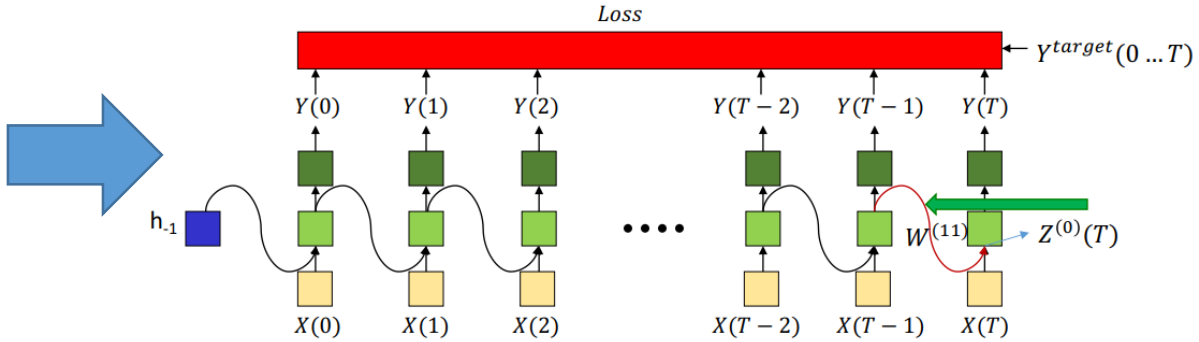
- محاسبه ی مشتق تابعloss نسبت به w^0



6

$$\nabla_{W^{(11)}} Loss = h(T-1)\nabla_{Z^{(0)}(T)} loss$$

$$\frac{dLoss}{dw_{ij}^{(11)}} = \frac{dLoss}{dZ_j^{(0)}(T)} h_i(T-1)$$

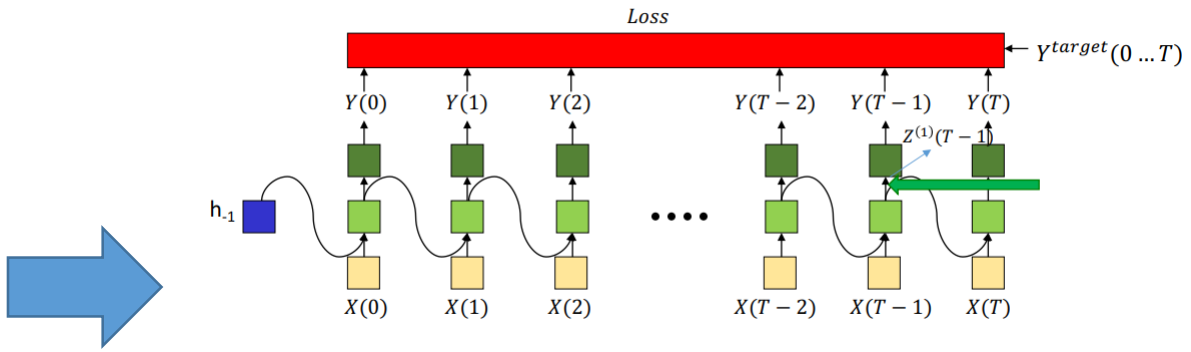


بنابراین به همین ترتیب خواهیم داشت برای مشتق تابع loss نسبت به Z(T-1) خواهیم داشت:

1

$$\nabla_{Z^{(1)}(T-1)} Loss = \nabla_{Z^{(1)}(T)} Y(T-1)\, \nabla_{Y(T-1)} Loss$$

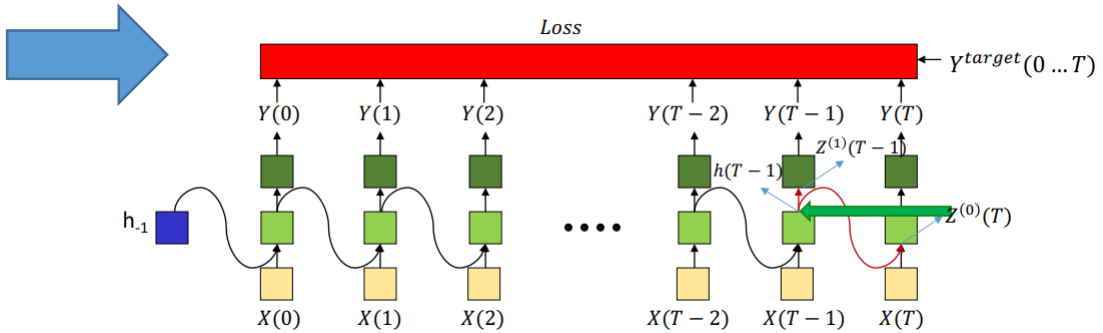
$$\frac{dLoss}{dZ_i^{(1)}(T-1)} = \frac{dLoss}{dY_i(T-1)} \frac{dY_i(T-1)}{dZ_i^{(1)}(T-1)}$$



2

$$\nabla_{h(T-1)} Loss = W^{(1)} \nabla_{Z^{(1)}(T-1)} Loss + W^{(11)} \nabla_{Z^{(0)}(T)} Loss$$

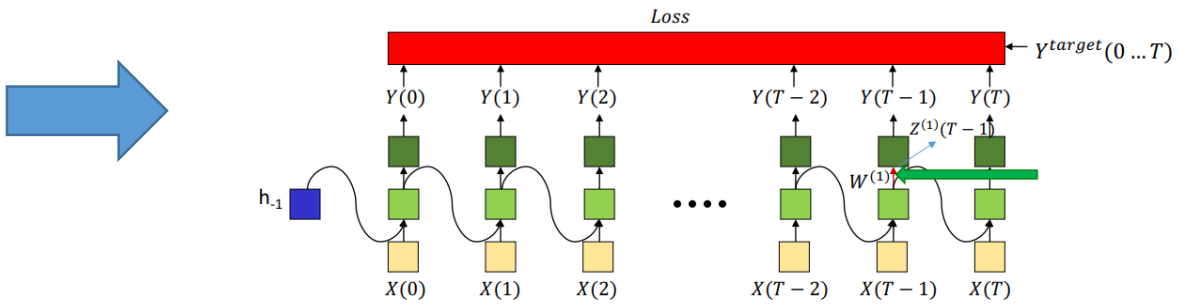
$$\frac{dLoss}{dh_i(T-1)} = \sum_j w_{ij}^{(1)} \frac{dLoss}{dZ_j^{(1)}(T-1)} + \sum_j w_{ij}^{(11)} \frac{dLoss}{dZ_j^{(0)}(T)}$$



3

$$\nabla_{W^{(1)}} Loss \mathrel{+}= h(T-1) \nabla_{Z^{(1)}(T-1)} Loss$$

$$\frac{dLoss}{dw_{ij}^{(1)}} \mathrel{+}= \frac{dLoss}{dZ_j^{(1)}(T-1)} h_i(T-1)$$



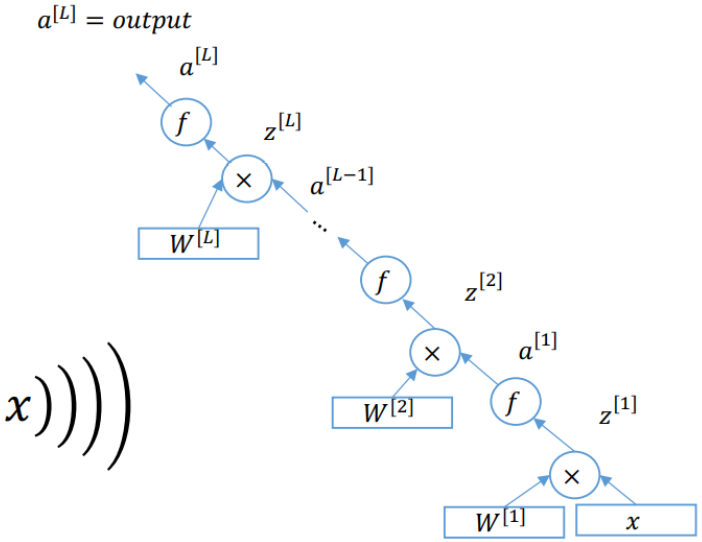
و به همین ترتیب ...

ب

در شبکه های عصبی می دانیم که برای به روز رسانی وزن ها باید مشتق تابع loss را نسبت به ورودی اولیه محاسبه کنیم، بنابراین طبق طبق الگوریتم Back Propagation ما به یک زنجیره متوالی از ضرب مشتقات در یک دیگر رو به رو خواهیم بود. اط طرفی می دانیم که در شبکه های عمیق این مسئله به صورت شدید تری وجود دارد (چون تعداد لایه ها بیشتر است بنابراین برای محاسبه ی مشتق تابع loss نسبت به ورودی با یک زنجیره بزرگتری مواجه هستیم):

ساختار شبکه عصبی

$$Output = a^{[L]}$$
$$= f(z^{[L]})$$
$$= f(W^{[L]}a^{[L-1]})$$
$$= f(W^{[L]}f(W^{[L-1]}a^{[L-2]})$$
$$= f\left(W^{[L]}f\left(W^{[L-1]}\dots f\left(W^{[2]}f(W^{[1]}x)\right)\right)\right)$$



$$Loss(x) = E\left(f^{[L]}\left(W^{[L]}f^{[L-1]}\left(W^{[L-1]}f^{[L-2]}(\dots W^{[1]}x)\right)\right)\right)$$

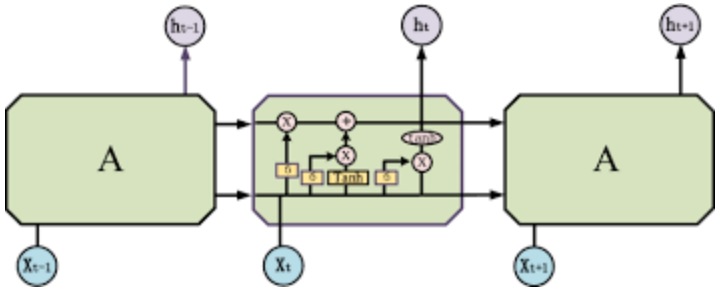
$$Loss(x) = E\left(f^{[L]}\left(W^{[L]}f^{[L-1]}\left(W^{[L-1]}f^{[L-2]}(\dots W^{[1]}x)\right)\right)\right)$$

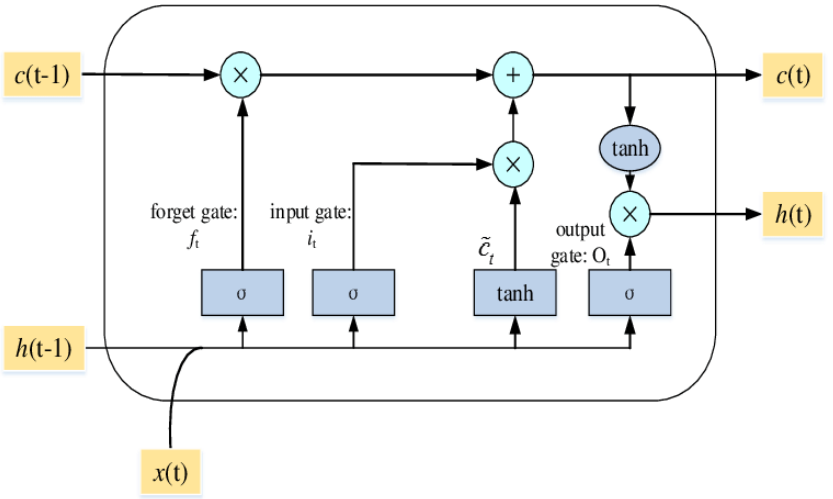
$$\nabla_{f^{[l]}} Loss = \nabla_{f^{[L]}} Loss . \nabla f^{[L]} . W^{[L]} . \nabla f^{[L-1]} . W^{[L-1]} \dots \nabla f^{[l+1]} . W^{[l+1]}$$

همانطور که ما از رابطه ی بالا مشخص است ما ما حاصل ضرب یک سری گرادیان مواجه هستیم، بنابراین در اثر ضرب متوالی بردارهای وزن در بردارهای ویژه ای که مقدار بزرگ تر از یک دارند (در اثر لایه های زیاد و شبکه های عمیق) مقادیر وزن ها به صورت خیلی زیاد تغییر کند explosion gradient رخ می دهد.
به همین ترتیب اثر ضرب متوالی بردارهای وزن در بردارهای ویژه ای که مقدار کوچکتر از یک دارند مقادیر وزن ها به صورت خیلی ناچیز تغییر تغییر می کند و Vanishing gradient رخ می دهد. همچنین می دانیم که حاصل توابع فعالیت Tanh و sigmoid و ReLU در شبکه های عصبی معمولی و بازگشتی مقداری کمتر از یک دارند.
این مسئله نیز در شبکه های عصبی بازگشتی نیز وجود دارند، زیرا در شبکه های عصبی ما با ساختاری مواجه هستیم که یک ماتریس وزن داریم و در طول فرآیند آموزش پیوسته این مقادیر وزن دستخوش تغییرات می شود (به دلیل ضرب های متوالی). همچنین ما در شبکه های بازگشتی با مسئله ی ظرفیت حافظه هم مواجه هستیم، زیرا ما پیوساع مقادیر و اطلاعات لحظات مختلف را ذخیره می کنیم بنابراین ظرفیت حافظه محدود است و از یک حدی بیشتر هم نمی توانیم حافظه تخصیص دهیم بنابراین ممکن است بنابریان ممکن گرادیان به لحظات خیلی قبل تر به نرسد و ما باید بر اساس نتایج فعلی تصمیم گیری کنیم که این مسئله می تواند باعث شود یا خیلی ناچیز وزن ها را به روز رسانی کنیم (vanishing) و یا به صورت خیلی زیاد تغییر کند(explosion).

پ

همانطور که اسم شبکه ی LSTM مشخص است Long Short-Term Memory این شبکه طراحی شده است که مشکل مطرح شده در شبکه های بازگشتی که به دلیل کمبود ضرفیت حافظه به وجود می آمد را بر طرف کند. چون همانطور که اشاره شد ممکن از ما اطلاعات مربوط به مراحل قبل را نتوانیم ذخیره کنیم (فراموش کنیم) و این مثلاه می تواند به طور چشم گیر در بروز رسانی وزن ها (Vanishing یا explosion) تاثیر بگذارد.





این شبکه بر این اساس طراحی شد است که با تقسیم به ضرب ها به ضرب های کوچکتر (ضرب ها در زمان های مختلف رخ می دهد) مصرف حافظه و عملیات را کاهش دهد. این معماری از چند گیت تشکیل شده است.Forget gateInput gate مشخص می کند چه مقداری از حافظه وارد حافظه شود (i_1). (f_t) برای این است که بتوانیم تعیین کنیم تا چه زمانی میخواهیم مقادیر موجود در حافظه حفظ شود. $C^{\sim}$ اثر ورودی بر روی خروجی را تعیین می کند و میزان آن در حافظه را مشخص می کند. $C_t = f_t * C_{t-1} + i_t * C_t^{\sim}$ این رابطه از دو جز تشکیل شده است. $f_t * C_{t-1}$ به این صورت است که مشخص می کند چه مقدار از حافظه و تا چه مدت مورد استفاده قرار بگیرد. $i_t * C_t^{\sim}$ مشخص می کند چه مقدار از ورودی در حافظه قرار داشته باشد.

مسئله ۳. Exposure Bias (۵ نمره)

شبکه‌های تکرار شونده به طور کلی با هدف مدل کردن یک دنباله آموزش می‌یابند. این شبکه‌ها معمولاً از دو روند متفاوت برای فاز آموزش و تست استفاده می‌کنند. به این صورت که در فاز آموزش، خروجی/برچسب درست در مرحله $t-1$ از مجموعه دادگان گرفته می‌شود و به عنوان ورودی گام tام استفاده می‌شود. به عبارت دیگر شبکه در هر گام، خروجی درست مرحله قبل را از مجموعه دادگان دریافت می‌کند. از طرفی در فاز تست از آن جایی که طبیعتاً مجموعه دادگانی در دسترس نیست، خروجی گرفته شده از خود مدل در مرحله $t-1$ به عنوان ورودی گام t در نظر گرفته می‌شود. این تفاوت بین دو فاز باعث به وجود آمدن مشکلی به نام exposure bias می‌شود. این مشکل را توضیح دهید و یک مثال از حالتی که این مشکل می‌تواند کیفیت خروجی مدل را به شدت کاهش دهد بیاورید. به نظر شما چرا نمی‌شود از روندی که در فاز تست وجود دارد در فاز آموزش هم استفاده کنیم؟

همانطور که در صورت سوال نیز ذکر شده است، شبکه های بازگشتی شامل دو روند متفوات برای آموزش و تست هستند به طوری که ما در مرحله ی آموزش، ما از لیبل های واقعی داده ها استفاده می کنیم و این در حالی است که در مرحله ی تست این اتفاق می افتد چون طبیعتا ما لیبل مربوطه را نداریم. این مسئله سبب به وجود آمدن مشکلی به نامExposure bias می شود به این معنِ است که در مسائلی همچون ترجمه ماشینی با استفاده از شبکه های عصبی (NMT) ممکن است دامنه داده های آموزش متفاوت از داده های تست باشد و این مسئله می تواند خروجی های داده های تست را به صورت چشمگیری تحت تاثیر قرار دهد، مثلا فرض کنیم که ورودی های داده های آموزشی علاوه برب متن های عادی مجموعه ای اشعار شعرای معروف نیز هست، همانطور که می دانیم قواعد نگارشی اشعار متفاوت از متون معمولی است بنابریان ممکن است فضای آموزش دیده شده در مرحله ی آموزش به نوعی باشد که قابل استفاده در فضای داده های تست نباشد. بنابراین اگر خروجی های پیش بینی شده در فاز آموزش به هر دلیلی غیر دقیق باشد(مانند مثال گفته شده) می تواند نتایج داده های تست را تحت تاثیر قرار دهد زیرا همانطور که گفته شد بخش تست از داده های واقعی نمی تواد استفاده کند. در مورد اینکه چرا نمی شود روند بخش آموزش شبکه های بازگشتی را نیز شبیه روند بخش تست طراحی کرد، می توان به این مسئله اشاره کرد که ما در فضای آموزش کاملا ما از یک حالت رندم و تصادفی شروع می کنیم و به دنبال آن هسیتم با استفاده از ارزیابی عملکرد شبکه طی فرآیند آموزش (با استفاده از تابع هزینه و مقادیر واقعی) شبکه را آموزش دهیم به طوریکه به فضای دادگان ما نزدیک شود و آموزش ببیند، بنابراین اگر شبیه فرآیند آموزش ما لیبل های مورد نظر داده های آموزش را به شبکه ندهیم این مسئله سبب می شود تا نتوانیم از تابع هزینه به نحو مطلوب استفاده کینم و با استفاده از آن وزن های آموزش دیده را به نحو مطلوبی به روز رسانی کنیم. بنابراین اگر فرآنید آموزش نیز به درستی انجام نشود (بر اساس لیبل داده های واقعی) نمی توانیم از آن در مرحله ی تست و پیش بینی داده های دیده نشده به نحوه مطلوبی استفاده کنیم.

مسئله ۴. شبکه مولد متخاصمی (۱۵ نمره)

تابع هدف شبکه مولد تخاصمی به صورت زیر است

$$\mathcal{V}(G,D) = \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{x\sim p_g}[log(۱ - D(x))] \quad (۱)$$

که در آن G شبکه مولد و D شبکه تمییزدهنده است و می‌خواهیم آن را نسبت به D بیشینه و نسبت به G کمینه کنیم. حال با توجه به توضیحات داده شده به سوالات زیر پاسخ دهید.

الف) با فرض نامحدود بودن ظرفیت تمییزدهنده، تمییزدهنده بهینه (D^*) را بدست آورید.(۵ نمره)

ب) حال با فرض بهینه بودن تمییز دهنده، نشان دهید کمینه کردن تابع هزینه GAN معادل کمینه کردن فاصله Jensen Shannon بین توزیع دادگان آموزش و توزیع شبکه مولد است.(۵ نمره)

ج)حال با توجه به قسمت قبل نشان دهید که $p_g \approx p_{data}$ یک نقطه بهینه برای تابع هدف می‌باشد.(۵ نمره)

الف

طبق تابع هدف GAN داریم:

$$\min_{\phi} \max_{\theta} \quad [\mathbb{E}_{x\sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$

- با ساده سازی رابطه ی بالا با استفاده از رابطه ی امید ریاضی داریم:

$$\min_{\phi} \max_{\theta} \int_x p_{data}(x) \log D_\theta(x) + \int_z p(z) \log(1 - D_\theta(G_\phi(z)))$$

- با توجه به این می دانیم توزیع $p_G(x)$ ، X هایی را که توسط تابع مولد تولید شده است مشخص می کند، می توانیم رابطه بالا را به صورت زیر بازنویسی کنیم:

$$\min_{\phi} \max_{\theta} \int_x p_{data}(x) \log D_\theta(x) + \int_x p_G(x) \log(1 - D_\theta(x))$$

- با ترکیب انتگرال ها خواهیم داشت:

$$\min_{\phi} \max_{\theta} \int_x (p_{data}(x) \log D_\theta(x) + p_G(x) \log(1 - D_\theta(x)))\, dx$$

$$\min_{\phi} \max_{\theta} \int_x \left(p_{data}(x) \log D_\theta(x) + p_G(x) \log(1 - D_\theta(x))\right) dx$$

- ما به دنبال پیدا کردن تمیز دهنده ی بهینه هستیم، بنابراین از رابطه بالا نسبت به D_θ مشتق بگیریم و برابر صفر قرار دهیم:

$$\frac{d}{d(D_\theta(x))} \left(p_{data}(x) \log D_\theta(x) + p_G(x) \log(1 - D_\theta(x))\right) = 0$$

- ساده سازی رابطه ی بالا:

$$p_{data}(x)\frac{1}{D_\theta(x)} + p_G(x)\frac{1}{1 - D_\theta(x)}(-1) = 0 \quad \Rightarrow \quad \frac{p_{data}(x)}{D_\theta(x)} = \frac{p_G(x)}{1 - D_\theta(x)}$$

$$\frac{p_{data}(x)}{D_\theta(x)} = \frac{p_G(x)}{1 - D_\theta(x)} \quad \Rightarrow \quad D_\theta(x) = \frac{p_{data}(x)}{p_G(x) + p_{data}(x)}$$

$$D_G^*(G(x)) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

تمیز دهنده بهینه

ب

- طبق رابطه تابع هدف GAN داریم:

$$\mathcal{V}(G,D) = \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{x\sim p_g}[log(1 - D(x))]$$

- همچنین به ازای تمیز دهنده بهینه داریم:

$$D_G^*(G(x)) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

- با جاگذاری تمیز دهنده بهنیه در رابطه تابع هدف GAN داریم:

$$E_{x\sim P_{data}}\left[log\frac{P_{data}(x)}{P_{data}(x) + PG(x)}\right] + E_{x\sim PG}\left[log1 - \frac{P_{data}(x)}{P_{data}(x) + PG(x)}\right] =$$

$$E_{x\sim P_{data}}\left[log\frac{P_{data}(x)}{P_{data}(x) + PG(x)}\right] + E_{x\sim PG}\left[log\frac{P_{PG}(x)}{P_{data}(x) + PG(x)}\right]$$

$$E_{x\sim P_{data}}\left[log\frac{P_{data}(x)}{P_{data}(x)+PG(x)}\right]+E_{x\sim PG}\left[log\frac{P_{PG}(x)}{P_{data}(x)+PG(x)}\right]$$

- یک $-log4$ به رابطه ی بالا اضافه می کنیم و برای خنثی سازی تاثیر آن هر دو لگاریتم موجود در رابطه را بر دو تقسیم می کنیم:

$$E_{x\sim P_{data}}\left[log\frac{\frac{P_{data}(x)}{P_{data}(x)+PG(x)}}{2}\right]+E_{x\sim PG}\left[log\frac{\frac{P_{PG}(x)}{P_{data}(x)+PG(x)}}{2}\right]-log4$$

- رابطه ی بالا را می توانیم به صورت زیر بازنویسی کنیم:

$$-log4+kl(pg||\frac{p_{data+pg}}{2}+kl(p_{data}||\frac{p_{data+pg}}{2})$$

Jensen shannon

$$-log4+2JS(p_{data},pG)$$

❖ می دانیم که JS بزرگتر مساوی صفر است بنابراین حد پایین رابطه ی بالا log4– است. بنابراین کمینه کردن تابع هزینه GAN برابر است با معادله ی کمینه فاصله JS است.

ج

- داریم:

$$D_G^*(G(x)) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

طبق فرضیات مسئله

$$p_G = p_{data}$$

بنابراین

$$D_G^* = \frac{p_{data}}{p_{data} + p_G} = \frac{1}{2}$$

- طبق رابطه ی تابع هدف GAN داریم:

$$V(G, D_G^*) = \int_x p_{data}(x) \log D(x) + p_G(x) \log (1 - D(x))\, dx$$

ثابت شد

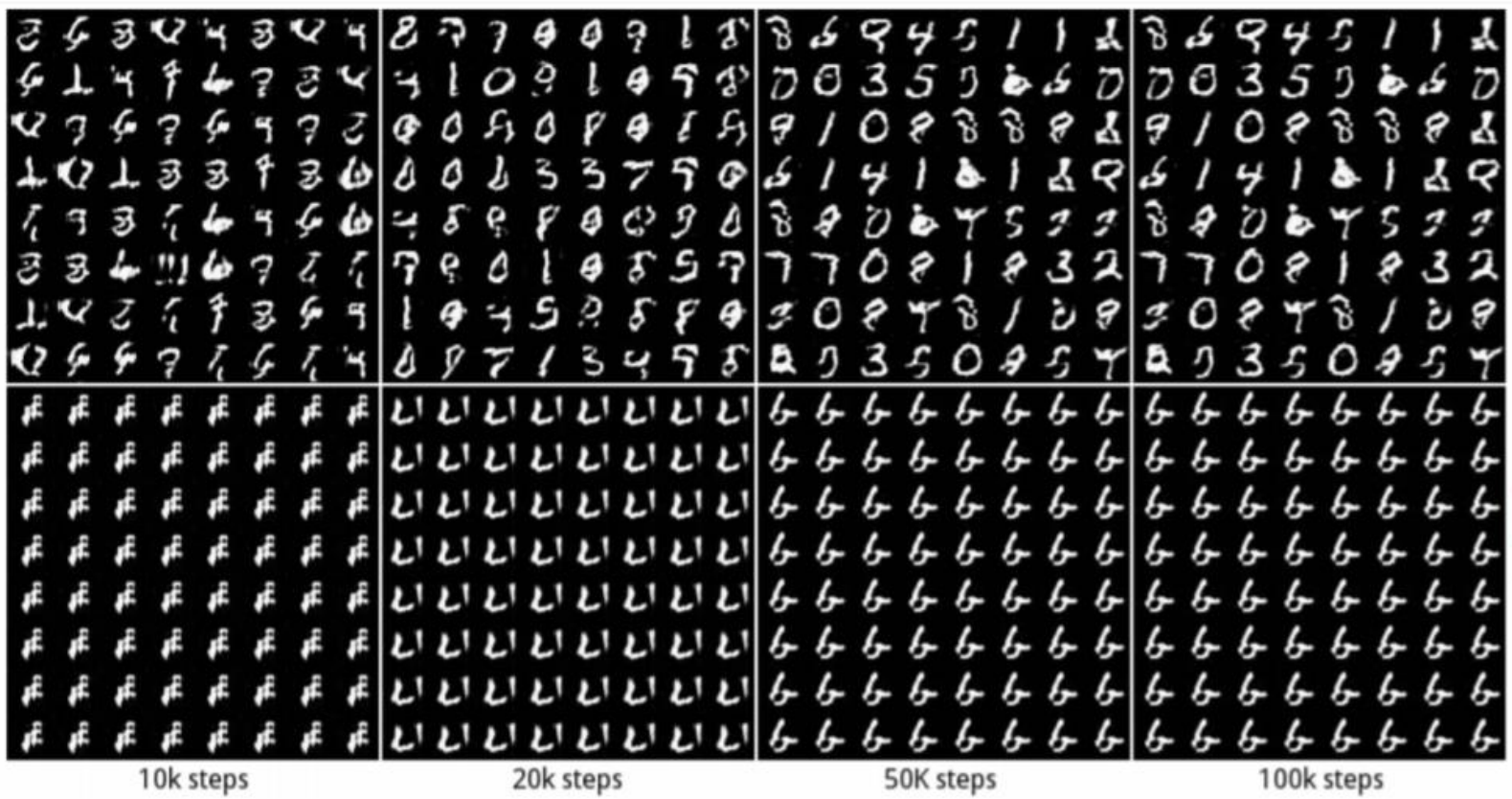
جایگذاری ½ در رابطه بالا

$$\int_x p_{data}(x) \log \frac{1}{2} + p_G(x) \log \left(1 - \frac{1}{2}\right) dx \rightarrow \log 2 \int_x p_G(x)dx - \log 2 \int_x p_{data}(x)dx \rightarrow -2 \log 2 = -\log 4$$

مسئله ۵. چسبندگی قله (۵ نمره)

آموزش شبکه‌های مولد تخاصمی با چالش هایی همراه است. یکی از این چالش ها مشکل Mode Collapse است.
این مشکل را شرح دهید و توضیح دهید چرا با این مشکل در آموزش این شبکه‌ها مواجه هستیم.

در حوزه ی یادگیری ماشین گاهی لیبل های مجموعه دادگان ما بیش از یک و دو لیبل است، مثلا در مجموعه ی دادگان MNIST مثلا در دیتاست ارقام ۱۰ دسته (۱۰ لیبل) وجود دارد. این مسئله زمانی بیشتر جلوه می کند که به طور مسال ما بر روی تصاویر انسان ها کار میکنیم از نظر پراکندگی و گوناکونی زیادی داریم که هر دسته و مجموعه از یک توزیع پیروی می کند. بنابراین ما در یک دیتا ست چند دسته ای مثل مجموعه ی ارقام، با فضای ابعادی (مسئله ای) مواجه هستیم که از چند توزیع با قله های مختلف تشکیل شده است و از تونع زیادی نسبتا برخوردار هستند. تصور ما این است که وقتی از GAN استفاده می کنیم شبکه ی GAN تمامی این توزیع ها را آموزش ببیند و در خروجی به ازای هر دسته ما نمونه ای را آموزش دیده باشد و آن را Generate کند. بنابراین ما توقع داریم که در خروجی مجموعه ی متنوعی از مجموعه دادگان مسئله تولید شود. اما در عمل این اتفاق نمی افتد و در خروجی ما با یک از زیر مجموعه ی کوچکی از فضای مسئله مواجه هستیم. همانطور که در شکل زیر می بینیم:

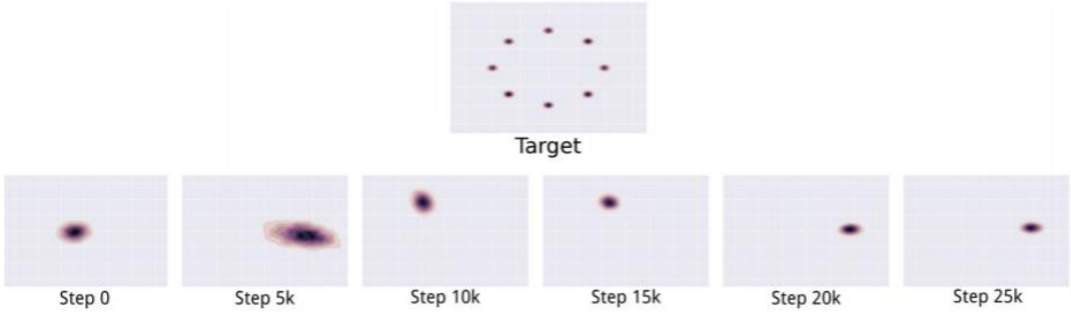


توقع ما از شبکه GAN

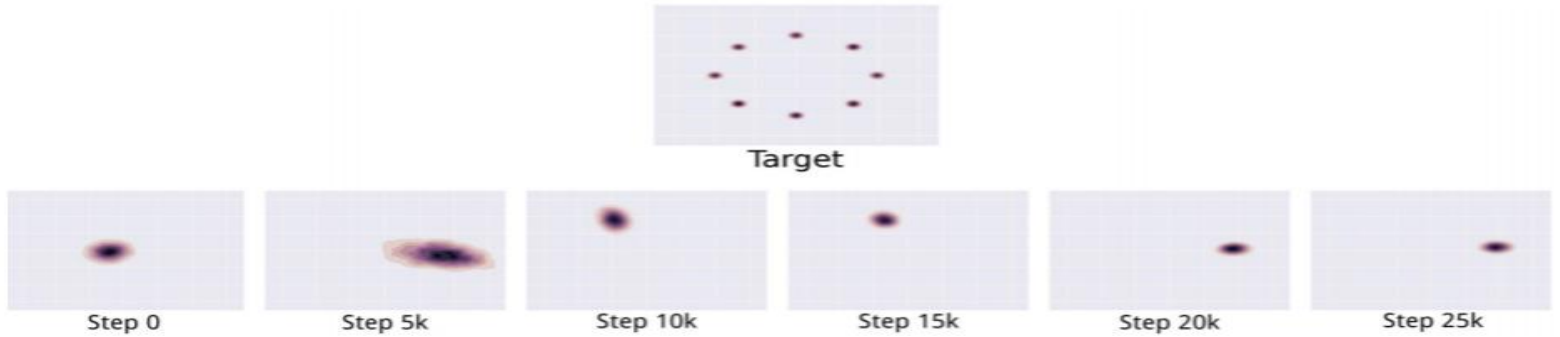
چیزی که در عمل اتفاق افتاده است. و فقط در گام های مختلف (100K) عدد ۶ تولید شده است

البته باید اشاره داشت که این مشکل همیشه مانند مثال قبل نیست که فقط یک نمونه به طور کلی در گام مختلف آموزش یاد گرفته شود و تولید شود (complete collapse)و این مسئله کمتر متداول است. این مسئله معمولا به صورت نسبی اتفاق می افتد و مثلا یک بخش از فضای مسئله آموزش می بیند و آن بخش تولید می شود (partial collapse). همانطور که در شکل زیر می بینید تصاویر که رنگ زیرین آنها یکسان تقریبا از یک اوزیع هستند

به این مشکل به وجود آمده در GAN اصطلاحا mode collapse می گویند. در واقع همانطور که گفته شد mode collapse زمانی رخ می دهد که که توزیع های متنوعی در فضتی مسئله داشته باشیم(که این مسئله مثلا به خاطر افزایش تعداد لیبل های زیاد مسئله ناشی می شود). همانطور که در شکل زیر می بینید.

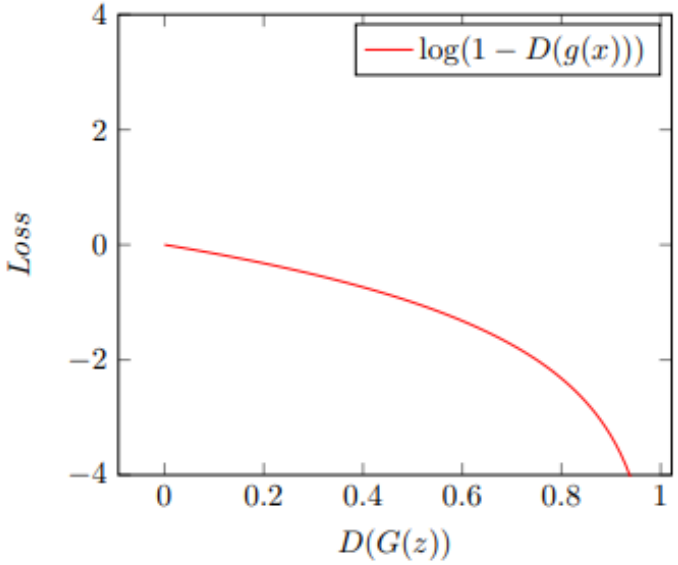


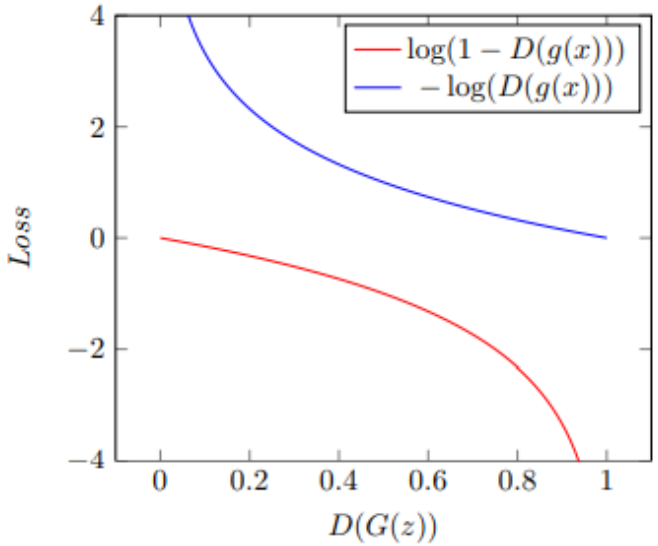
دلیل به وجود آمدن پدیده ی Mode collapse به این دلیل است که اگر Generator قدرتمند باشد خیلی سریع آموزش می بیند در یک MODE (همانطور که از شکل زیر مشخص است) و داده هایی تولید می کند که به راحتی می تواند Discriminator را فریب دهد (زیرا مولد در یک گام آموزش می بیند و تمیز دهند ده یک گام دیگر). بنابراین پس از آن بر به صورت تدریجی Discriminator آموزش می بیند که این MODE را به عنوان داده های fake دسته بندی کند. بنابراین زمانی که Discriminator در تشخیص داده های fake به اندازه ی کافی قدرتمند می شود، Generator یک ضعف دیگر در Discriminator پیدا می کند و به یک MODE جدید Collapse می کند و تا زمانی که مجددا Discriminator تنظیم شود و بتواند این MODE را به عنوان fake پیش بینی کند. بنابراین مجددا دوباره Generator به حالت اولیه بر می گردد و همین پروسه مجددا تکرار می شود. همان طور که گفته شد وقع complete collapse خیلی نادر است اما partial collapse که در بخش از توزیع رخ می دهد متداول است.



در این قسمت می‌خواهیم تاثیر استفاده از توابع هدف متفاوت را برای اموزش ببینیم.

۱) مشکل استفاده از $\mathbb{E}_{x\sim p_g}[log(۱ - D(x))]$ به عنوان تابع هدف شبکه مولد که می‌خواهیم آن را کمینه کنیم را توضیح دهید و توضیح دهید چگونه استفاده از تابع هدف $\mathbb{E}_{x\sim p_g}[log(D(x))]$ که می‌خواهیم آن را بیشینه کنیم مشکل ذکر شده را برطرف می‌کند. (۵ نمره)





- با توجه به تابع هزینه مطرح شده برای Generator زمانی که در ابتدای آموزش داده ها تقریبا بسیار شبیه به Fake هستند ما به دنبال آن هستیم تا با استفاده از گرادیان بخش Generator را قوی تر کنیم اما همانطور که از شکل مقابل مشخص است D(G(z)) نزیک به صفر است و تابع هزینه در ناحیه تقریبا مسطح و flat است بنابریان این مسئله باعث می شود که گرادیان تقریبا نزدیک به صفر باشد. این در صورتی است که Discriminator به سرعت به دلیل گرادیان بالا قدرتمند تر می شود اما مولد همینطور ضعف می ماند.

- بنابراین به جای تابع هدف $\mathbb{E}_{x\sim p_g}[log(1 - D(x))]$ می توانیم از تابع هدف $\mathbb{E}_{x\sim p_g}[log(D(x))]$

استفاده کنیم و آنرا بیشنیه کنیم در این صورت همانطور که از شکل مقابل مشخص است مسئله کوچک بودن مقدار گردایان که در تابع هدف قبلی وجود داشت در این تابع هدف جدید بهبود می یابد.

۲) فرض کنید که دامنه توزیع اصلی و توزیع شبکه مولد با هم همپوشانی نداشته باشند و تمییزدهنده نیز نزدیک به تمییزدهنده بهینه باشد. با فرض‌های مطرح شده به سوال‌های زیر پاسخ دهید.

الف) تمییزدهنده را به صورت $\sigma(a)$ در نظر بگیرید که در آن a ، logits شبکه است. حال گرادیان $log(۱ - D(x))$ را نسبت به logits های شبکه D بدست آورید. (a را به صورت تابعی از ورودی یعنی $f(x)$ در نظر بگیرید) (۳ نمره)

ب) در این حالت چه گرادیانی به شبکه مولد می‌رسد؟ چه مشکلی ایجاد می‌شود؟ (۳ نمره)

ج) حال اگر از تابع $-logD(x)$ به عنوان هدف شبکه مولد استفاده شود آیا باز هم این مشکل وجود دارد؟ (۳ نمره)

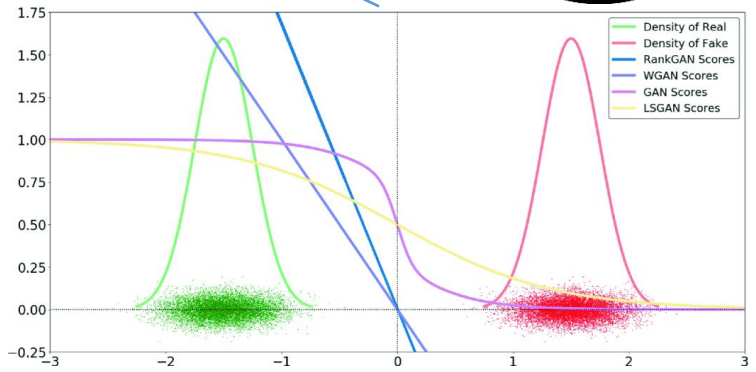
الف

$$\mathcal{V}(G,D) = \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{x\sim p_g}[log(1 - D(x))]$$

فرضیات مسئله

$$\log\big(1 - D(x)\big) = \log\left(1 - \frac{1}{\exp(-a)}\right) = \log\left(\frac{1}{1+\exp(-a)}\right) = -\log(1+\exp(a))$$

$$\frac{\partial}{\partial a}\big(\log\big(1 - D(x)\big)\big) = -\frac{\exp(a)}{1+\exp(a)} = -\frac{1}{1+\exp(-a)} = -\sigma(a)$$



ب

در حالتی که تمییزدهنده بهینه باشد و دامنه توزیع شبکه مولد و داده آموزشی همپوشانی نداشته باشند آنگاه $\sigma(a) = D(x)$ به سمت صفر میل می کند و با توجه به روابط بالا تابع هزینه شبکه Generator نسبت به logitهای شبکه تمیزدهنده برابر است با:

$$\frac{\partial}{\partial a}\big(\log\big(1 - D(x)\big)\big) = \sigma(a)$$

بنابراین تابع بالا به سمت صفر میل می کند. چون در مرحله Backpropagation گردایان لایه ها در هم ضرب می شود، همانطور که دیده شد گرادیان تابع هزینه به سمت صفر میل می کند و گرادیان صفر به Generator میرسد و مسئله ی Vanishing پیش می آید.

اگر تابع هزینه مربوطه با تابع هزینه $-\log D(x)$ جایگزین کنیم خواهیم داشت:

$$\mathcal{V}(G,D) = \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{x\sim p_g} -logD(x)$$

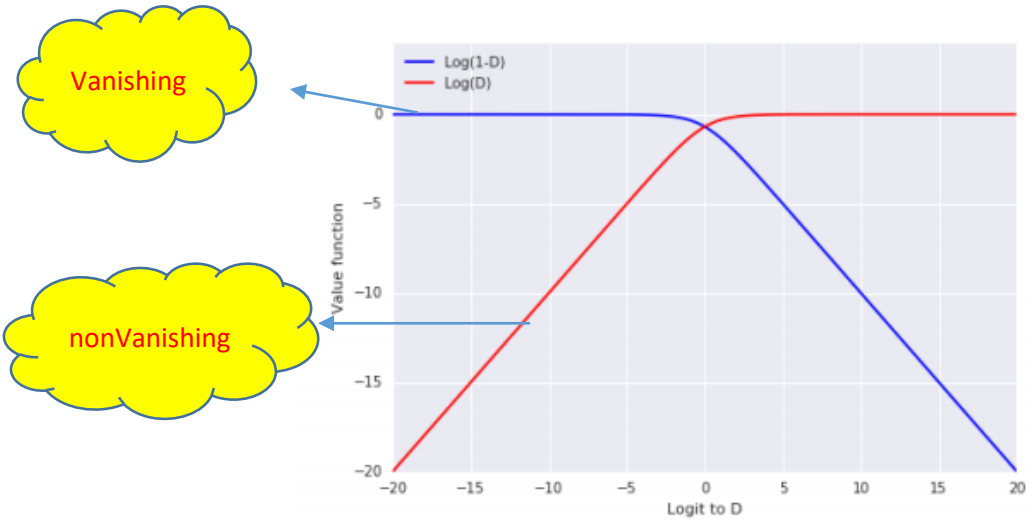
$$-\log\big(D(x)\big) = -\log\left(\frac{1}{1+\exp(-a)}\right) = \log(1+\exp(-a))$$

$$\frac{\partial}{\partial a}(-\log D(x)) = -\frac{\exp(-a)}{1+\exp(-a)} = -(1-\sigma(a))$$

در این حالت اگر نگاه کنیم $\sigma(a) = D(x)$ به سمت صفر می رود آنگاه:

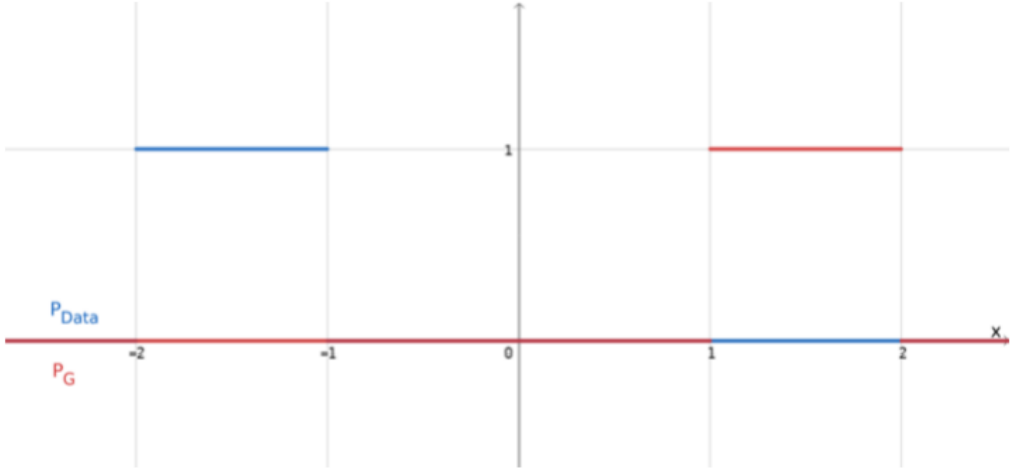
$$\frac{\partial}{\partial a}\log\big(-\log\big(D(x)\big)\big) = -\big(1-\sigma(a)\big) = -1$$

بنابراین همانطور که می بینیم مشکل برطرف می شود و مسئله ما nonvanishing می شود.



مسئله ۷. مشکل تابع فاصله JS (۷ نمره)

(امتیازی) همانطور که می‌دانیم، در شبکه های مولد تخاصمی می‌خواهیم فاصله Jensen Shannon را کمینه کنیم و برای این کار از روش Gradient Descent استفاده می‌کنیم. حال توضیح دهید اگر توزیع دادگان آموزش و توزیع شبکه مولد چه ویژگی نسبت به همدیگر داشته باشند نمی‌توانیم از رویکرد بالا برای آموزش شبکه استفاده کنیم. برای رفع این مشکل چه کاری می‌تواند صورت گیرد؟



همانطور که از دو مثال (دو شکل مقابل مشخص است) ما، دو توزیع یعنی توزیع PG و توزیع Pdata هم پوشانی ندارند. همچنین طبق فرض مسئله رابطه ی Jensen shannon داریم:

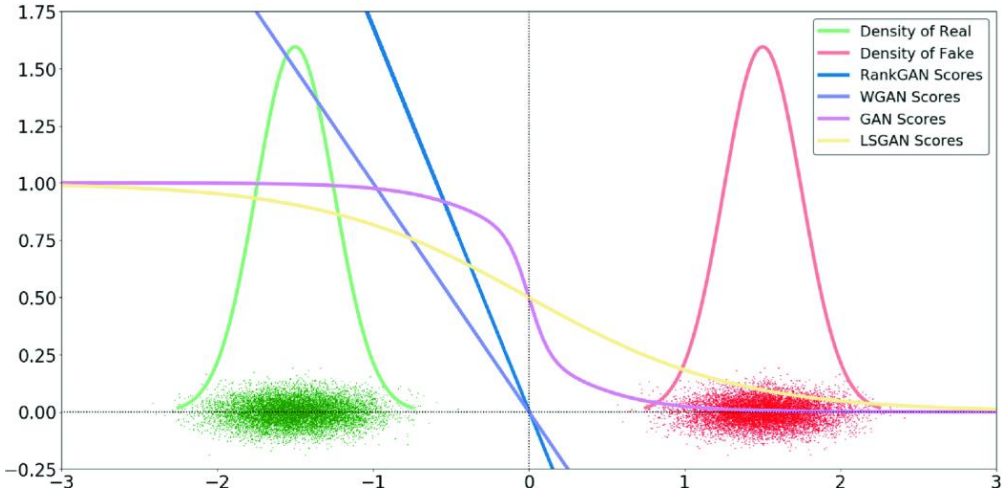
$$D_G^* = \frac{p_{data}}{p_{data} + p_G} = \frac{1}{2}$$

$$V(G, D_G^*) = \int_x p_{data}(x) \log D(x) + p_G(x) \log (1 - D(x))\, dx$$

$$\int_x p_{data}(x) \log \frac{1}{2} + p_G(x) \log \left(1 - \frac{1}{2}\right) dx$$

$$\log 2 \int_x p_G(x) dx - \log 2 \int_x p_{data}(x) dx$$

$$-2 \log 2 \quad = -\log 4$$



همانطور که می بینیم مقدار کمینه ی تابع JS برابر یک مقدار ثابت یعنی log4– شده است. بنابراین زمانی که دو توزیع از یک دیگر فاصله دشاته باشند گرادیانی که که به شبکه مولد می رسد همانطور که درشکل صفحه قبل وجود دارد صفر می شود(زیرا دو توزیع از هم فاصله دارند و با جابه جایی اندک آنها همچنان این مقدار ثابت باقی می ماند و گرادیان صفر به شبکه Generator می شود). برای حل این مشکل می توانیم مقداری نویز به دو توزیع مربوط به داده های Generator و داده های Real اضافه کنیم. این مسئله سبب می شود تا این مسئله که دو توزیع از هم فاصله دارند و هم پوشانی ندارند به نوعی حل شود و دو توزیع باهم هم پوشانی پیدا کنند که این مسئله باعث می شود از وقوع گرادیان صفر جلوگیری می کنیم(اما این مسئله اجتمالی است و ممکن است لزوما این اتفاق رخ ندهد). برای حل مشکل صفر شدن گرادیان روش دیگری نیز وجود دارد به نام wasserstein distance که در واقع به کمترین هزینه جابه جایی از یک توزیع به یک توزیع دیگر است.

$$-2\log 2 \qquad = -\log 4$$

$$\int \gamma(x,y)\,\mathrm{d}y = \mu(x)$$
$$\int \gamma(x,y)\,\mathrm{d}x = \nu(y)$$

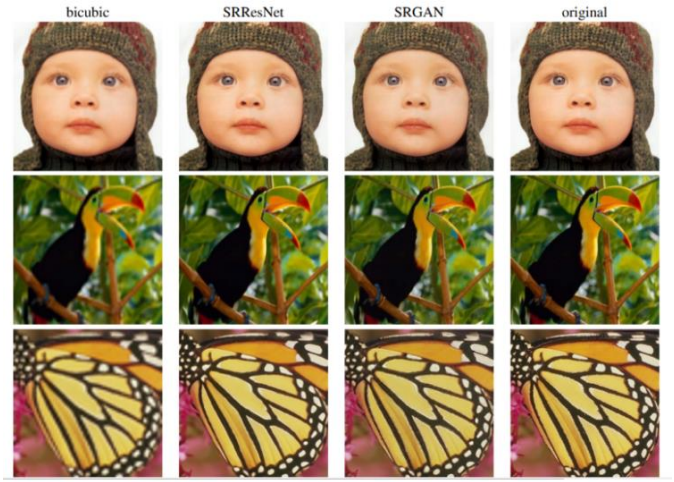
$$\iint c(x,y)\gamma(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int c(x,y)\,\mathrm{d}\gamma(x,y)$$

$$C = \inf_{\gamma\in\Gamma(\mu,\nu)} \int c(x,y)\,\mathrm{d}\gamma(x,y)$$

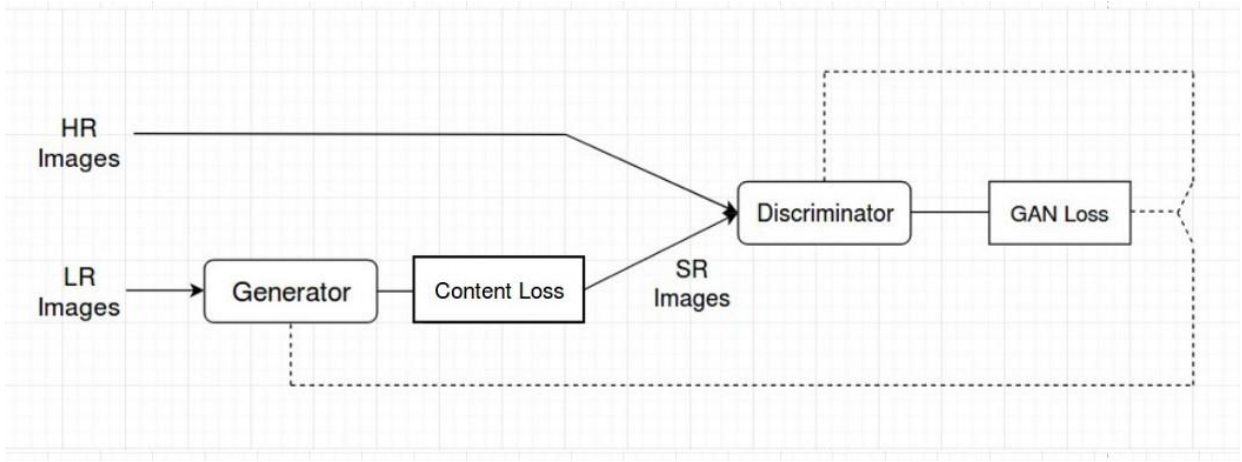
که در این رابطه μ و V دو توزیع مد نظر ما هستند و C تابع هزینه ماست. C(x,y) هزینه ی جابه جایی بین دو نقطه x وy است و $\gamma\in\Gamma(\mu,\nu)$ تمام روش های جابه حایی بین این دو توزیع μ و V است. مقدار $\gamma(x,y)$ نشان دهنده مقداری است که می خواهید از نقطه x به y در دو توزیع μ و V منتقل کنیم. بنابراین $T(\gamma,c)$ هزینه جابه جایی بین دو توزیع μ و V تحت روش جابه جایی $\gamma(x,y)$ خواهد داشت.

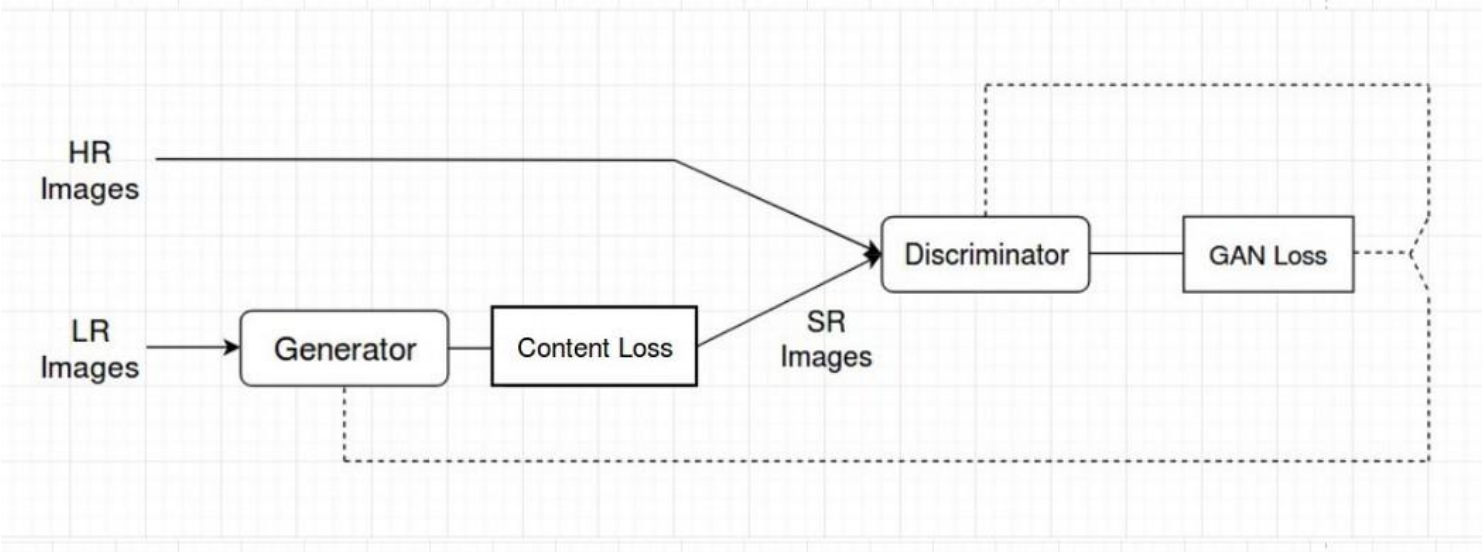
مسئله ۸. کاربرد شبکه‌های مولد متخاصمی (۶ نمره)

یکی از کاربردهای شبکه‌های مولد تخاصمی کاربرد Super Resolution است. در این کاربرد ما می‌خواهیم تصاویر با رزولوشن پایین را تبدیل به تصاویر با رزولوشن بالا کنیم. برای این کار می‌توان از یک شبکه عصبی به این صورت استفاده کرد که به عنوان ورودی، تصاویر با رزولوشن پایین را بگیرد و در خروجی تصاویر با رزولوشن بالا را به ما برگرداند. برای این روش می‌توان از تابع هدف Mean Square Error (MSE) استفاده کرد. مشکلی که در این راهکار به وجود می‌آید آن است که تصاویر تولید شده با رزولوشن بالا، تار می‌شوند. یکی از روش ها برای رفع این موضوع، استفاده از تابع هزینه GAN در کنار تابع هزینه MSE است. فرض کنید دیتاست ما به صورت جفت داده‌های (I^{LR}, I^{HR}) است که شامل تصویر با رزولوشن پایین و تصویر با رزولوشن بالای متناظر با آن است. در مورد ساختار شبکه عصبی مورد استفاده برای این کار و تابع هزینه هر یک از شبکه‌های مولد و تمییزدهنده توضیح دهید.

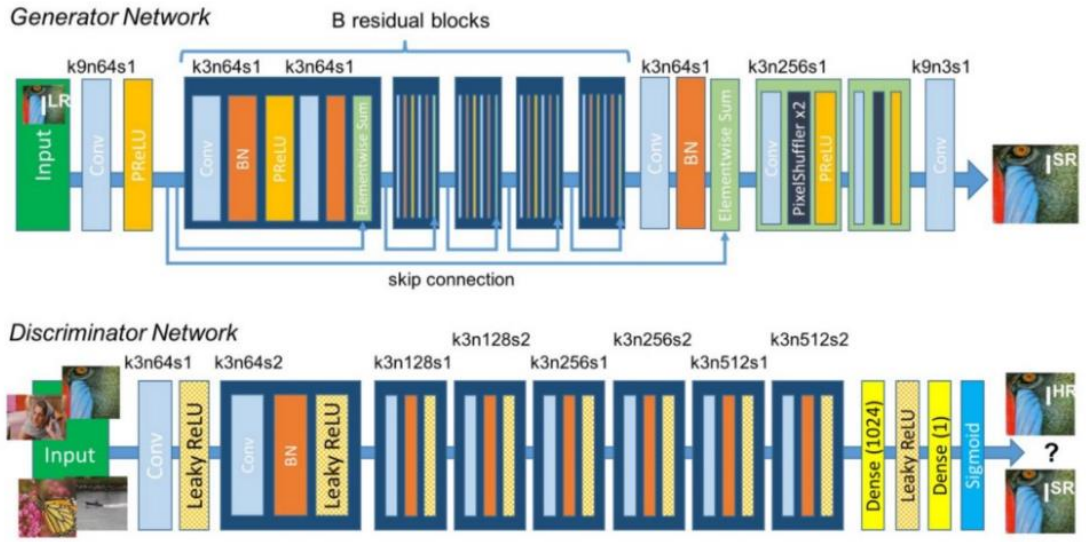


همانطور که گفته شد یکی از کاربرد های شبکه های مولد تخاصمی GAN، ایجاد Super Resolution در تصاویر است. در واقع ما به دنبال این هستیم که تصاویر با رزولوشن پایین را به تصاویر با رزولوشن بالا تبدیل کنیم. برای این منظور از روش هایی در ابتدا استفاده می شد که که از تابع MSE استفاده می کردند و این مسئله باعث می شد که تصاویر با رزولوشن بالا تار شوند. برای این منظور از تابع MSE در کنار GAN استفاده شد. یکی از شبکه های معروف برای افزایش رزولوشن تصاویر شبکه ی SRGAN است(Super-resolution GAN). معماری شبکه به شکل زیر است:





همانطور که از معماری شبکه SRGAN مشخص است ورودی های این شبکه HR (تصاویر با رزولوشن بالا) و LR (تصاویر با رزولوشن پایین) هستند. تصاویر LR وارد شبکه مولد (Generator) می شوند. همچنین ما از یک تمییز دهنده برای تشخیص تصاویر HR استفاده می کنیم و با استفاده از عملیات Backpropagation دو شبکه ی مولد و تمییز دهنده را آموزش می دهیم.



تابع Lossمورد استفاده در شبکه SRGAN، Perceptual Loss نام دارد که از دو بخش تشکیل شده است: Content(Reconstruction) loss و Adversarial loss.

$$l^{SR} = \underbrace{l_{\mathrm{X}}^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}$$

perceptual loss (for VGG based content losses)

Content(Reconstruction) loss

$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

Content Loss

Content loss که در این معماری استفاده شده است تا بتوانیم به جای شباهت پیکسلی (perceptual similarity)، شباهت ادراکی (pixel wise similarity) را حفظ کنیم. این به ما امکان می دهد بافت واقعی عکس را از تصاویر بسیار پایین نمونه گرفته شده بازیابی کنیم

Adversarial loss

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

Adversarial Loss

از این تابع lossبه عنوان بررسی تصویر تولید شده توسط مولد استفاده می‌شود که باعث ایجاد تصاویر با ظاهری طبیعی می‌شود

تابع loss مربوط به تمییز دهنده نیز در این شبکه شبیه همان تمییز دهنده موجود در شبکه GAN است.

$$\min_{\phi} \max_{\theta} \quad [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$