Statistical Distance Latent Regulation Loss for Latent Vector Recovery

Jeongik Cho[1]

jeongik.jo.01@gmail.com[1]

Abstract

Finding a latent vector that can generate specific data using a generative model is called latent vector recovery. When performing gradient descent based latent recovery, the latent vector being recovered may escape the train latent distribution. To prevent this, latent regulation loss or resampling was used in some papers.

In this paper, assuming that the generative model is trained with IID(Independent and Identically Distributed) random variables, I propose a statistical distance latent regulation loss that considers the train latent distribution as a one-dimensional distribution, the latent vector as a sample distribution, and the distance between the two distributions as a latent regulation loss. The statistical distance latent regulation loss considers the correlation between each element of the latent vector, so better latent vector recovery is possible.

In addition, I compared the performance of latent regulation losses and resampling methods of other papers as well as statistical distance latent regulation losses using several statistical distances.

In conclusion, the performance of Bhattacharyya latent regulation loss was the best when the train latent vector followed the normal distribution, and the Lukaszyk Karmowski regulation loss showed the best performance otherwise.

1. Statistical distance latent regulation loss

The generative model (generator) G is trained to convert the d_z-dimensional multivariate random variable $Z \in R^{d_z}$ following a certain distribution to the d_x-dimensional multivariate random variable $X \in R^{d_x}$.

In the case of GAN, usually latent vector $Z \sim U(a,b)^{d_z}$ or $Z \sim N(\mu,\sigma^2)^{d_z}$, and in the case of VAE, each element of latent vector Z follows a normal distribution with different means and variances. Finding the ideal latent vector z^* that can generate any data x sampled from data distribution X through gradient descent using pre-trained generator G is called latent vector recovery. There are gradient descent-based and encoder-based methods for latent vector recovery. The encoder-based method requires additional encoder training. In this paper, only the gradient descent-based method is covered.

The gradient descent based latent vector recovery receives the error between the data $G(z_p)$ generated through the latent vector z_p and the received data x as loss, and performs gradient descent repeatedly for the latent vector z_p.

The following function is a function that performs latent vector recovery based on gradient descent.

$function\ latent\_recovery(x_p, G, n, opt)$:

$z_p \leftarrow initialize(\,)$

$repeat\ t\ times$:

$L \leftarrow diff\left(x_p, G(z_p)\right)$

$z_p \leftarrow z_p - opt\left(\frac{\Delta L}{\Delta z_p}\right)$

$return\ z_p$

$initialize$ is a function that initializes the values of z_p. t is the number of times to perform gradient descent. opt is an optimizer. $diff$ is a function that measures the difference between two data. Through the above function, z_p can be found that minimizes $diff\left(x_p, G(z_p)\right)$.

However, when z_p exists to make $diff\left(x_p, G(z_p)\right)$ small enough, the z_p may not be the ideal latent vector z^*. For example, suppose in the MNIST handwriting data, x_p is the handwriting data of the number "1", $G(z_p)$ currently produces the number "0" handwriting data, and $z_p[1]$ (the first element of z_p) represents the width of the handwriting data. If the other elements of z_p remain unchanged and $z_p[1]$ becomes extremely low, the width of the letter becomes very narrow, which can look like the number 1. However, for z_p where $z_p[1]$ is extremely low, $P(Z = z_p)$ will be very low or zero. In this case, z_p cannot be regarded as a good latent vector that can generate x_p, and since it has already converged to the local optima, it is possible that additional gradient descent may not generate the number 1. That is, a good latent vector z_p can be regarded as a value that minimizes $diff\left(x_p, G(z_p)\right)$ while maximizing $P(Z = z_p)$.

To maximize $P(Z = z_p)$, latent regulation loss L_{lr} is added to loss L in [3, 4], and some elements of z_p are resampling after gradient descent in [1].

The latent vector recovery using latent regulation loss proceeds as follows.

$function\ latent\_recovery(Z, x_p, G, n, opt)$:

$z_p \leftarrow initialize(\,)$

$repeat\ t\ times$:

$L \leftarrow diff\left(x_p, G(z_p)\right) + \lambda_{lr} L_{lr}$

$z_p \leftarrow z_p - opt\left(\frac{\Delta L}{\Delta z_p}\right)$

$return\ z$

L_{lr} is the latent regulation loss weight, and λ_{lr} is the latent regulation loss weight.

In this paper, to maximize $P(Z = z_p)$, assuming that Z follows a certain IID random variable A^{d_z}, I propose a method that uses the statistical distance between the distribution A and the sample distribution $\{s | s \in z_p\}$ as the latent regulation loss L_{lr}. Since the statistical distance latent regulation loss can consider the relationship between each element of z_p, a better latent vector z_p can be found.

$$L_{lr} = Dist\left(A, \{s|s \in z_p\}\right)$$

$Dist$ is a function that represents the statistical distance between two distributions. Among several statistical distances, this paper used four statistical distances: Bhattacharyya distance, Wasserstein distance, Energy distance, and Lukaszyk Karmowski distance.

The following table shows the required conditions and features by latent regulation loss or resampling method.

Name	Z~ALL	Z~IID	Z~N	Z~U	Remarks
Bhattacharyya distance			O		
Wasserstein distance		O	O	O	
Energy distance		O	O	O	
Lukaszyk Karmowski distance		O	O	O	
Trick discriminator	O	O	O	O	Hard to find hyperparameter, slow speed
Z score square			O		
Z score absolute			O		
Logistic cutoff			O		Information lost
Truncated normal cutoff			O		Information lost
Boundary resampling				O	Information lost, no hyperparameter

Table 1. Features by method

Z~ALL in the above table means that Z can be used regardless of distribution, and Z~IID means that Z can be used when following IID distribution. The yellow items in the table are not suggested in other papers. The "trick discriminator" is the loss proposed in [3]. Z score square is the loss suggested in [4]. The logistic cutoff and truncated normal cutoff are the resampling methods proposed in [5]. Boundary resampling is a resampling method proposed in [1]. Resampling methods slow convergence because information loss occurs when resampling.

2. Experiments

For the experiment, pre-trained GAN using LSGAN [6] adversarial loss was used. Latent vector dimension $d_z = 256$. The model was trained using $optimizer = Adam(learning\ rate = 10^{-5}), epoch = 200, batch\ size = 32$. MNIST handwriting dataset [7] was used for training. FID of GAN trained with $Z \sim N(0,1)^{d_z}$ was 8.30641, and $Z \sim U(-1,1)^{d_z}$ was 5.423009.

For the performance evaluation, I measured how well the generated data were classified into a classifier with an accuracy of 99.34%. Classifier was trained with $optimizer = Adam(learning\ rate = 10^{-5}), epoch = 50, batch\ size = 32$. $initialize()$ was $zeros()$. For $diff$, the l_1 loss with the best result in [2] was used. The number of gradient descent iterations is $t = 100$ and optimizer $opt = Adam$. For evaluation, only 1000 randomly selected from 10000 test data were used. Wasserstein distance, Energy distance, and

Lukaszyk Karmowski distance were measured by sampling enough samples (1000 samples) from the train latent distribution Z. Logistic cutoff and truncated normal cutoff were excluded from the experiment due to too low performance and difficult hyperparameter search.

The following tables show the accuracy of the classifier according to the learning rate and hyperparameter when $Z \sim N(0, 1^2)^{d_z}$.

Bhattacharyya distance			Latent regulation loss weight							
			No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010		0.701	0.699	0.716	0.741	0.725	0.715	0.683	0.707
	0.0032		0.805	0.816	0.796	0.800	0.814	0.826	0.839	0.812
	0.0100		0.798	0.802	0.803	0.802	0.830	0.863	0.907	0.893
	0.0320		0.747	0.757	0.770	0.793	0.862	0.906	0.942	0.898
	0.1000		0.657	0.650	0.665	0.748	0.891	0.918	0.864	0.735
	0.3200		0.540	0.525	0.530	0.535	0.572	0.665	0.707	0.395
	1.0000		0.458	0.428	0.408	0.415	0.425	0.465	0.563	0.347

Table 2. Bhattacharyya latent regulation loss accuracy

Lukaszyk karmowski			Latent regulation loss weight							
			No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010		0.701	0.724	0.733	0.720	0.716	0.724	0.702	0.689
	0.0032		0.805	0.795	0.820	0.811	0.800	0.822	0.828	0.829
	0.0100		0.798	0.809	0.807	0.820	0.834	0.898	0.899	0.852
	0.0320		0.747	0.772	0.773	0.805	0.881	0.933	0.930	0.904
	0.1000		0.657	0.684	0.701	0.853	0.897	0.896	0.838	0.706
	0.3200		0.540	0.583	0.624	0.820	0.872	0.668	0.473	0.374
	1.0000		0.458	0.442	0.475	0.720	0.639	0.345	0.261	0.188

Table 3. Lukaszyk karmowski latent regulation loss accuracy

Z score square			Latent regulation loss weight							
			No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010		0.701	0.727	0.689	0.726	0.723	0.713	0.709	0.656
	0.0032		0.805	0.812	0.806	0.815	0.803	0.808	0.811	0.756
	0.0100		0.798	0.810	0.800	0.830	0.873	0.887	0.890	0.826
	0.0320		0.747	0.738	0.793	0.880	0.914	0.912	0.894	0.826
	0.1000		0.657	0.725	0.890	0.926	0.889	0.870	0.749	0.625
	0.3200		0.540	0.814	0.883	0.865	0.698	0.500	0.383	0.272
	1.0000		0.458	0.868	0.765	0.552	0.355	0.264	0.230	0.222

Table 4. Z score square latent regulation loss accuracy

Z score absolute		Latent regulation loss weight							
		No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010	0.701	0.755	0.724	0.719	0.719	0.631	0.396	0.237
	0.0032	0.805	0.819	0.814	0.828	0.816	0.745	0.535	0.291
	0.0100	0.798	0.819	0.833	0.833	0.868	0.856	0.668	0.378
	0.0320	0.747	0.767	0.772	0.846	0.898	0.890	0.771	0.502
	0.1000	0.657	0.669	0.717	0.882	0.906	0.851	0.750	0.523
	0.3200	0.540	0.566	0.622	0.813	0.844	0.696	0.502	0.426
	1.0000	0.458	0.451	0.466	0.703	0.682	0.463	0.312	0.301

Table 5. Z score absolute latent regulation loss accuracy

Wasserstein distance		Latent regulation loss weight							
		No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010	0.701	0.702	0.713	0.743	0.720	0.676	0.536	0.397
	0.0032	0.805	0.803	0.801	0.793	0.777	0.716	0.579	0.388
	0.0100	0.798	0.797	0.820	0.797	0.747	0.674	0.611	0.337
	0.0320	0.747	0.752	0.731	0.716	0.733	0.705	0.631	0.461
	0.1000	0.657	0.651	0.683	0.817	0.879	0.857	0.808	0.636
	0.3200	0.540	0.546	0.601	0.816	0.895	0.875	0.875	0.762
	1.0000	0.458	0.442	0.471	0.742	0.691	0.517	0.462	0.392

Table 6. Wasserstein distance latent regulation loss accuracy

Energy distance		Latent regulation loss weight							
		No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010	0.701	0.731	0.720	0.747	0.694	0.729	0.667	0.547
	0.0032	0.805	0.825	0.826	0.798	0.796	0.759	0.668	0.533
	0.0100	0.798	0.827	0.822	0.771	0.783	0.711	0.616	0.484
	0.0320	0.747	0.734	0.738	0.734	0.742	0.677	0.648	0.456
	0.1000	0.657	0.660	0.689	0.698	0.856	0.873	0.810	0.614
	0.3200	0.540	0.563	0.566	0.586	0.711	0.865	0.877	0.835
	1.0000	0.458	0.471	0.476	0.442	0.500	0.604	0.660	0.592

Table 7. Energy distance latent regulation loss accuracy

Trick discriminator		Latent regulation loss weight							
		No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010	0.701	0.648	0.484	0.133	0.097	0.111	0.090	0.107
	0.0032	0.805	0.786	0.597	0.214	0.111	0.092	0.091	0.084
	0.0100	0.798	0.791	0.589	0.242	0.130	0.099	0.095	0.099
	0.0320	0.747	0.687	0.496	0.176	0.123	0.115	0.110	0.082
	0.1000	0.657	0.586	0.397	0.156	0.102	0.089	0.118	0.108
	0.3200	0.540	0.497	0.308	0.122	0.100	0.098	0.093	0.104
	1.0000	0.458	0.410	0.275	0.112	0.111	0.093	0.117	0.087

Table 8. Trick discriminator latent regulation loss accuracy

The following tables show the accuracy of the classifier according to the learning rate and hyperparameter when $Z \sim U(0, 1^2)^{d_z}$. Trick discriminator was excluded due to its low performance and slow speed. Bhattacharyya distances, Z score absolute, and Z score square were excluded because they were defined only in $Z \sim N^{d_z}$.

Lukaszyk Karmowski distance			Latent regulation loss weight							
			No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010		0.743	0.756	0.738	0.739	0.736	0.759	0.748	0.743
	0.0032		0.828	0.830	0.815	0.801	0.784	0.829	0.830	0.824
	0.0100		0.748	0.777	0.772	0.759	0.797	0.874	0.899	0.898
	0.0320		0.614	0.657	0.637	0.724	0.849	0.927	0.913	0.883
	0.1000		0.515	0.505	0.539	0.633	0.903	0.890	0.788	0.667
	0.3200		0.417	0.432	0.396	0.538	0.798	0.642	0.458	0.320
	1.0000		0.342	0.340	0.353	0.380	0.629	0.376	0.223	0.202

Table 9. Lukaszyk Karmowski latent regulation loss accuracy

Wasserstein distance			Latent regulation loss weight							
			No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010		0.743	0.743	0.766	0.746	0.753	0.738	0.632	0.435
	0.0032		0.828	0.818	0.800	0.829	0.770	0.716	0.603	0.437
	0.0100		0.748	0.746	0.726	0.719	0.679	0.658	0.570	0.461
	0.0320		0.614	0.633	0.672	0.679	0.694	0.689	0.667	0.523
	0.1000		0.515	0.514	0.503	0.646	0.856	0.880	0.884	0.829
	0.3200		0.417	0.432	0.384	0.533	0.813	0.806	0.772	0.715
	1.0000		0.342	0.316	0.335	0.394	0.643	0.450	0.330	0.312

Table 10. Wasserstein latent regulation loss accuracy

Energy distance			Latent regulation loss weight							
			No loss	0.010	0.032	0.100	0.320	1.000	3.200	10.000
Optimizer learning rate	0.0010		0.743	0.766	0.748	0.751	0.755	0.753	0.686	0.536
	0.0032		0.828	0.803	0.814	0.793	0.763	0.722	0.621	0.508
	0.0100		0.748	0.734	0.741	0.723	0.684	0.637	0.581	0.457
	0.0320		0.614	0.644	0.658	0.686	0.699	0.634	0.576	0.440
	0.1000		0.515	0.525	0.497	0.530	0.713	0.877	0.836	0.766
	0.3200		0.417	0.419	0.448	0.424	0.450	0.611	0.798	0.825
	1.0000		0.342	0.356	0.348	0.343	0.382	0.380	0.460	0.506

Table 11. Energy latent regulation loss accuracy

Boundary resampling						
			Without resmapling		Boundary resampling	
Optimizer learning rate	0.0010		0.743		0.767	
	0.0032		0.828		0.814	
	0.0100		0.748		0.748	
	0.0320		0.614		0.781	
	0.1000		0.515		0.839	
	0.3200		0.417		0.760	
	1.0000		0.342		0.611	

Table 12. Boundary resampling accuracy

3. Conclusion

When $Z \sim N^{d_z}$, Bhattacharyya latent regulation loss showed the best performance. However, when $Z \sim U^{d_z}$, Lukaszyk Karmowski regulation loss showed the best performance. Overall, performance was good when using statistical distance latent regulation loss.

4. References

[1] Zachary C. Lipton, Subarna Tripathi,

"Precise Recovery of Latent Vectors from Generative Adversarial Networks"

https://arxiv.org/abs/1702.04782

[2] Arun Patro ; Vishnu Makkapati ; Jayanta Mukhopadhyay

Evaluation of Loss Functions for Estimation of Latent Vectors from GAN

https://ieeexplore.ieee.org/document/8517097/authors#authors

[3] Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, Minh N. Do

Semantic Image Inpainting with Deep Generative Models

https://arxiv.org/abs/1607.07539

[4] Antonia Creswell, Anil A Bharath

Inverting The Generator Of A Generative Adversarial Network (II)

https://arxiv.org/abs/1802.05701

[5] Nicholas Egan, Jeffrey Zhang, Kevin Shen

Generalized Latent Variable Recovery for Generative Adversarial Networks

https://arxiv.org/abs/1810.03764

[6] Xudong Mao, Qing Li, Haoran Xie, Raymond

Y.K. Lau, Zhen Wang, Stephen Paul Smolley

Least Squares Generative Adversarial Networks

https://arxiv.org/abs/1611.04076

[7] Yann LeCun, Corinna Cortes, Christopher J.C. Burges

THE MNIST DATABASE of handwritten digits

http://yann.lecun.com/exdb/mnist/