Advanced Deep Learning: Assignment 1

Seung-Hoon Na

May 31, 2020

1 Disentangled representation learning

Simply saying, disentangled representation learning is to find latent factors which are *independent* to other factors (uniquely identified) and *interpretable* to the real world. The 'independence' implies that the change of one disentangled factor affects the generation process by changing only the property corresponding to the factor, while properties from other latent factors are fixed (such as person identify, object style, or color). The 'interpretability' means that such a disentangled factor has a proper meaning in the real world, such that the change of such a latent factor in the real world can be captured and recognized by human in a conscious and grounded manner.

The early studies of the disentangled representation learning in deep learning are broadly categorized into *supervised* or *unsupervised* approaches. One of the classical models in the supervised method is *DC-IGN* [1] that uses graphics codes in 3D rendering engine as disentangled factors and proposes the use of a deep convolutional inverse graphics network as an encoder. In DC-IGN, graphics codes are explicitly provided in training data, as they are revealed in a 3D rendering engine during the generation process. Using the supervised information of target graphics codes, the training process of DC-IGN learns to project images into such "pre-designed" supervised factors and to generate those images from them.

Unsupervised methods of disentangled representation learning discover disentangled factors from training data directly, without making such prior knowledge.

In this project, we will study and implement two classical methods of unsupervised disentangled representation learning – **InfoGAN** [2] and **β-VAE** [3]. The goal of this project is summarized as follows:

1. Completely review InfoGAN and β-VAE, with the detailed derivations of them.
2. Implement InfoGAN and β-VAE using pytorch
3. Empirical comparison of InfoGAN and β-VAE on various datasets.

2 InfoGAN

Read the paper of InfoGAN [2] and write a report that addresses the following problems.

2.1 Model

1. Summarize the main idea of InfoGAN and the difference between InfoGAN and the original Generative adversarial network (GAN).

2. Specify the underlying principle of the following information-regularized minimax game (i.e., Eq. (3) in [2]).

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c)) \tag{1}$$

3. Why a variational method is necessary to estimate $I(c; G(z,c))$?

4. Clearly derive the following lower bound of $I(c; G(z,c))$ (i.e., Eq. (4) of [2]).

$$I(c; G(z,c)) \geq \mathbb{E}_{x\sim G(z,c)} \left[\mathbb{E}_{c'\sim P(c|x)} \, logQ(c'|x) \right] + H(c) \tag{2}$$

   Note: If possible, you don't need to use the original notations. Rather it is better to more clearly describe the notations of random variables. For example, in the paper,$G(z,c)$ is used to indicate either a random variable or a distribution. But, in your derivation, use clear notations to distinguish them.

5. Derive the following lemma (i.e., Lemma 5.1 of [2]).

$$\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x',y)] \tag{3}$$

6. Clearly derive the following lower bound of $I(c; G(z,c))$ (i.e., Eq. (5) of [2]).

$$I(c; G(z,c) \geq \mathbb{E}_{c\sim P(c), x\sim G(z,c)} \left[logQ(c|x) \right] + H(c) = L_I(G,Q) \tag{4}$$

7. Finally derive the following minimax game with a variation regularization of mutual information and a hyperparameter λ (i.e., Eq.(6) of [2]).

$$\min_{G,Q} \max_D V_I(D,G) = V(D,G) - \lambda L_I(G,Q) \tag{5}$$

2.2 Implementation using pytorch

Implement InfoGAN using pytorch in your own codes.

Note: If you use open source codes in some of your part (basic common modules), clearly mention the URL of the open code's github and the part used in your code.

2.3 Experiments

Write additional codes and scripts that perform some of the experiments of the paper (i.e., Section 7.1 and 7.2 of [2]), as required below. For each experiment, prepare a README file that describe how to execute your codes or scripts with clear details.

Perform experiments as follows:

2.3.1 Train InfoGAN on MNIST dataset

1. Train InfoGAN on MNIST dataset using three latent codes c_1, c_2, c_3 where $c_1 \sim Cat(K == 10, p = 0.1)$ and $c_2, c_3 \sim Unif(-1, 1)$, with 62 noise variables, as in Section 7.2 of [2].
2. Refer to Appendix C.1 for detailed neural architectures for the discriminator and generator CNNs. To train InfoGAN, you should also write the corresponding CNN codes for implementing the architectures shown in Appendix C.1.
3. Draw a figure obtained by manipulating latent codes as in the paper (i.e,. Fig. 2 of [2]).

2.3.2 Train InfoGAN on Faces

For this experiment, use the 'faces' dataset of [4].

1. Train InfoGAN on faces dataset using five continous codes, $c_i \sim Unif(-1, 1)$ with $1 \leq i \leq 5$, with 128 noise variables, as in Section 7.2 of [2].
2. Refer to Appendix C.4 for detailed neural architectures for the discriminator and generator CNNs. To train InfoGAN, you should also write the CNN codes for implementing the architectures shown in Appendix C.4.
3. Draw a figure obtained by manipulating latent codes as in the paper (i.e,. Fig. 3 of [2]).

2.3.3 Train InfoGAN on CelebA

For this experiment, use the CelebA dataset of [5].

1. Train InfoGAN on CelebA dataset using 10 ten-dimensional categorical code and 128 noise variables, resulting in a concatenated dimention of 228, as in Section 7.2 and Appendix C.3 of [2].
2. Refer to Appendix C.3 for detailed neural architectures for the discriminator and generator CNNs. To train InfoGAN, you should also write the CNN codes for implementing the architectures shown in Appendix C.3.
3. Draw a figure obtained by manipulating latent codes as in the paper (i.e,. Fig. 6 of [2]).

2.3.4 Train InfoGAN on CASIA-Webface

For this experiment, use the CASIA-Webface of [6] (you can download in the web).

1. To Train InfoGAN on the CASIA-Webface dataset, design neural architectures for the discriminator and generator CNNs, the number of latent codes c with their distribution types, with a proper number of noise variables z.
2. Write your designed codes and train InfoGAN on the CASIA-Webface dataset.
3. Draw a figure obtained by manipulating latent codes, as in the Fig 6 of [2].

3 β-VAE

Read the paper of β-VAE [3] and write a report that addresses the following problems.

3.1 Model

1. Summarize the main idea of β-VAE and the difference between β-VAE and the original variational auto-encoder (VAE) of [7, 8].

2. Describe the underlying principle of the following constraint optimization problem for β-VAE (i.e., Eq. (2) in [3]).

$$\max_{\phi,\theta} \mathbb{E}_{x\sim\mathbf{D}}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]\right] \text{ subject to } D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}) < \epsilon\right) \tag{6}$$

3. Derive the following formula of β-VAE (i.e., Eq. (4) in [3]).

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \beta D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\right) \tag{7}$$

4. Why disentanglement metric is necessary, based on the argument in Section 3 of [3]?

5. Describe the intuitive idea of the disentanglement metric of [3] and present the method clearly.

3.2 Implementation using pytorch

Implement β-VAE using pytorch in your own codes.

Note: If you use open source codes in some of your part (basic common modules), clearly mention the URL of the open code's github and the part used in your code.

3.3 Experiments

In this experiment, you should carry of some of the experiments of [3], comparing to InfoGAN. For each experiment, you should write additional codes and scripts, and need to prepare a README file that describe how to execute your codes or scripts with clear details.

Perform experiments as follows:

3.3.1 Train β-VAE on faces dataset

For this experiment, use the faces dataset of [4], as used in InfoGAN.

1. Refer to Table 1 of Appendix A.1 for encoder and decoder architectures for training VAE and β-VAE (i.e. the row named 3DFaces). Write codes for encoder and decoder used in VAE and β-VAE.

2. Train VAE and β-VAE ($\beta = 20$) on faces dataset, as shown in the paper,

3. Compare InfoGAN, VAE, and β-VAE, and draw a figure obtained by manipulating latent codes as in the paper (i.e,. Fig. 3 of [3]).

3.3.2 Train β-VAE on CelebA dataset

For this experiment, use the CelebA dataset of [5]. as used in InfoGAN in the previous section.

1. Refer to Table 1 of Appendix A.1 for encoder and decoder architectures for training VAE and β-VAE (i.e. the row named CelebA).
2. Train VAE and β-VAE ($\beta = 250$) on CelebA dataset.
3. Compare InfoGAN, VAE, and β-VAE, and draw a figure obtained by manipulating latent codes as in the paper (i.e,. Fig. 1 of [3]).

3.3.3 Train β-VAE on CASIA-Webface

For this experiment, use the CASIA-Webface of [6] (you can download in the web).

1. Design properly for encoder and decoder architectures for training VAE and β-VAE.
2. Train VAE and β-VAE, using a proper value of β, on CASIA-Webface.
3. Compare InfoGAN, VAE, and β-VAE, and draw a figure obtained by manipulating latent codes as in the paper.

3.3.4 Train β-VAE on synthetic 2D shares

Read the Section 4.2 of [3] and address the following questions.

1. Create a synthetic dataset of binary 2D shapes, similar to the way of Section 4.2 in [3]. Of course, you need to write codes for creating the synthetic dataset.
2. Refer to Table 1 of Appendix A.1 for encoder and decoder architectures for training InfoGAN, VAE and β-VAE (i.e. the row named 2D shapes) .Write codes for encoder and decoder used in InfoGAN, VAE and β-VAE.
3. Implement the disentanglement metric by referring to Section 3 of [3].
4. Read Appendix A.6 of [3] and derive the following form of β_{norm}:

$$\beta_{norm} = \frac{\beta M}{\beta N} \tag{8}$$

   where M is the size of latent vector $\mathbf{z}$ and N is the size of input vector $\mathbf{x}$.

5. Train β-VAE on 2D shapes dataset, as shown in the paper. Compute disentanglement metrics of β-VAE varying the size of latent vectors and β. Draw the results of disentanglement metric scores, as shown in Fig. 6 of [3].
6. Train InfoGAN on 2D shapes dataset, compare InfoGAN, VAE, β-VAE ($\beta = 4$).
7. Compare InfoGAN, VAE, and β-VAE, and draw a figure obtained by manipulating latent codes.

4 Required submissions

For this project, you should submit the followings:

1. **Full reports** of addressing all the problems (derivation and summary)
2. **Complete source codes** (the main parts should mostly be written by yourself) of InfoGAN and β-GAN, with proper comments.
3. **Complete additional codes and script**s (i.e, required for carrying out all the experiments).
4. **README files** for performing experiments (i.e., describe how to train and test your models). Prepare scripts and README files such that any user can easily execute training and testing on the dataset in most environments within 10 mins after reading the README file

References

[1] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS '15*, pages 2539–2547. 2015.

[2] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS '16*, page 2180–2188, 2016.

[3] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[4] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *Proceedings of the 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, page 296–301. IEEE Computer Society, 2009.

[5] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, page 3730–3738. IEEE Computer Society, 2015.

[6] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning face representation from scratch. *CoRR*, abs/1411.7923, 2014.

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 year = 2014, url = http://arxiv.org/abs/1312.6114*.

[8] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.