[Paper review 37]

Adversarial Autoencoders

(Hensman, et al., 2013)

[Contents]

1. Abstract

propose AAE (Adversarial Auto Encoder)

- probabilistic autoencoder
- uses GAN to perform VI

2. Introduction

building scalable generative models to capture rich distributions

- ex) RBM, DBN, DBM : trained by MCMC

  (compute gradient of log-likelihood, which becomes more imprecise as training oes on)

  (∵ Markov Chains are unable to mix between modes fast enough)

- recently : trained via **back-propagation**

1) VAE (2014), importance weighted autoencoders (2015)

- use **recognition network** to predict posterior over latent variable

2) GAN (2014)

- use **adversarial training procedure**

3) GMMN (2015)

- Generative Moment Matching Networks
- use **momennt matching cost function** to learn the data distribution

This paper : propose **AAE** that can turn **autoencoder into generative model!**

Autoencoder is trained with dual objectives

- 1) traditional reconstruction error criterion
- 2) adversarial training criterion

  (matches the aggregated **posterior distn** of latent representation to an arbitrary **prior distn**)

2-1. GAN

- min-max adversarial game with generative model (G) and discriminative model (D)
- objective function

  $\min_G \max_D \mathrm{E}_{\mathrm{x} \sim p_{\text{data}}}[\log D(\mathrm{x})]+\mathrm{E}_{\mathrm{z} \sim p(\mathbf{z})}[\log (1-D(G(\mathbf{z}))].$
- using alternating SGD in 2 stages
  - step 1) train D
  - step 2) train G

3. Adversarial Autoencoders

notation

- $\mathbf{x}$: input
- $\mathbf{z}$: latent code vector
- $p(\mathbf{z})$: prior distn
- $q(\mathbf{z} \mid \mathbf{x})$: encoding distn
- $p(\mathbf{x} \mid \mathbf{z})$: decoding distn
- $p(\mathbf{x})$: model distribution

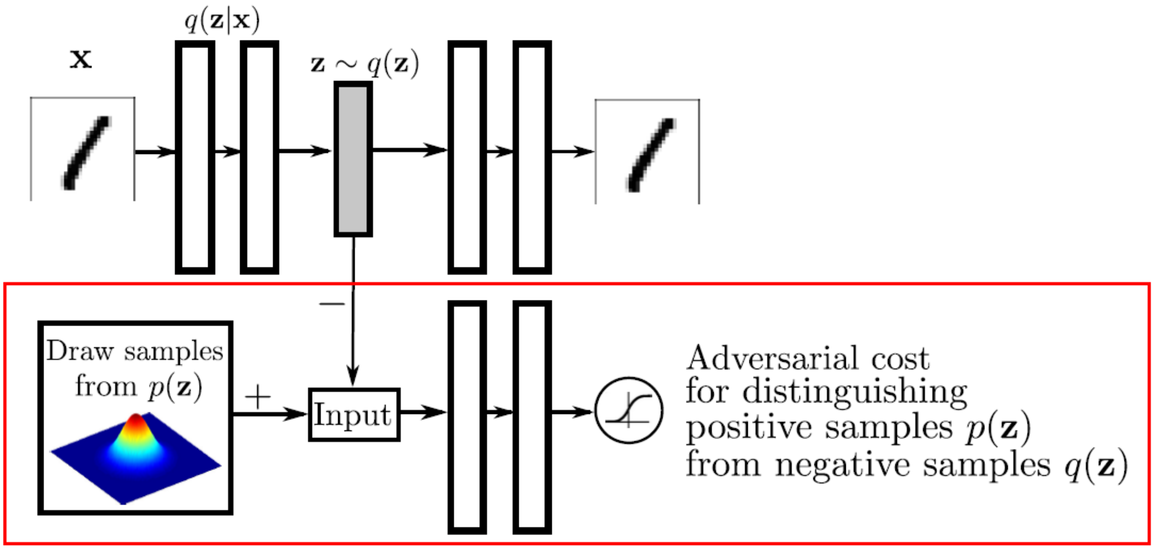
encoding function defines an aggregated posterior of $q(\mathbf{z})$ as below :

$q(\mathbf{z})=\int_{\mathbf{x}} q(\mathbf{z} \mid \mathbf{x}) p_d(\mathbf{x}) d \mathbf{x}.$

AAE = AE+ "Regularization"

(by matching the aggregated posterior $q(\mathbf{z})$ to an arbitrary prior $p(z)$)

To do so, **adversarial network is attached on top of hidden code vector of the autoencoder**



Both **(1) adversarial network** and **(2) autoencoder** are trained JOINTLY with 2 phases.

- phase 1) **reconstruction phase**
  - autoencoder updates encoder & decoder to minimize reconstruction error
- phase 2) **regularization phase**
  - adversarial network
    - first updates D
    - next updates G

Several possible choices of encoder $q(\mathbf{z} \mid \mathbf{x})$:

- **1) Deterministic**
  - just same as standard AE
  - stochasticity is only from $q(\mathbf{z})$ (data distribution)
- **2) Gaussian posterior**
  - $z_i \sim \mathcal{N}\left(\mu_i(\mathrm{x}), \sigma_i(\mathrm{x})\right)$.
  - use reparam-trick
- **3) Universal approximator posterior**
  - assume $q(\mathbf{z} \mid \mathbf{x}, \eta)=\delta(\mathbf{z}-f(\mathbf{x}, \eta))$
  - then,
    - posterior : $q(\mathbf{z} \mid \mathbf{x})=\int_\eta q(\mathbf{z} \mid \mathbf{x}, \eta) p_\eta(\eta) d \eta$
    - aggregated posterior : $q(\mathbf{z})=\int_{\mathbf{x}} \int_\eta q(\mathbf{z} \mid \mathbf{x}, \eta) p_d(\mathbf{x}) p_\eta(\eta) d \eta d \mathbf{x}$
  - stochasticity is from
    - 1) $q(\mathbf{z})$ (data distribution)
    - 2) random noise input η at input of the encoder

Summary

- **1)** : may produce $q(\mathbf{z})$ that is not very smooth
- **2) and 3)** : additional sources of stochasticity, which could help regularization by smoothing out $q(\mathbf{z})$

In this rest of paper, will use **1) Deterministic** version

3-1. Relationship to VAE

VAE summary

minimize the below upperbound on the NLL

$$\begin{aligned} E_{\mathbf{x}\sim p_d(\mathbf{x})}[-\log p(\mathbf{x})] &< E_{\mathbf{x}}\left[\mathrm{E}_{q(\mathbf{z}|\mathbf{x})}[-\log(p(\mathbf{x}\mid\mathbf{z}))]\right]+E_{\mathbf{x}}[\mathrm{KL}(q(\mathbf{z}\mid\mathbf{x})\|p(\mathbf{z}))] \\ &= E_{\mathbf{x}}\left[\mathbf{E}_{q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{x}\mid\mathbf{z})]\right]-E_{\mathbf{x}}[H(q(\mathbf{z}\mid\mathbf{x}))]+E_{q(\mathbf{z})}[-\log p(\mathbf{z})] \\ &= E_{\mathbf{x}}\left[\mathbf{E}_{q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{x}\mid\mathbf{z})]\right]-E_{\mathbf{x}}\left[\sum_i \log\sigma_i(\mathbf{x})\right)\Bigg]+E_{q(\mathbf{z})}[-\log p(\mathbf{z})]+\text{const} \\ &= \text{ Reconstruction } - \text{ Entropy } + \text{CrossEntropy}(q(\mathbf{z}),p(\mathbf{z})) \end{aligned}$$

.

- where $q(\mathbf{z})=\int_{\mathbf{X}}\int_{\eta} q(\mathbf{z}\mid\mathbf{x},\eta)p_d(\mathbf{x})p_\eta(\eta)d\eta d\mathbf{x}$.
- $q(\mathbf{z}\mid\mathbf{x})$: Gaussian
- $p(\mathbf{z})$: arbitrary distributon

Three error term

- 1) reconstruction
- 2) entropy : encourgages large variances for the posterior
- 3) cross entropy : cross entropy between $q(\mathbf{z})$ and $p(\mathbf{z})$

AAE vs VAE

- in AAE, replace second two terms with **an adversarial training procedure** that encourages $q(\mathbf{z})$ to match the whole distribution of $p(\mathbf{z})$
- in order to backpropagate
  - (VAE) need to have access to "exact functional form" of prior
  - (AAE) only need to be able to sample from prior

  $\rightarrow$ AAE can impose complex distn without accessing explicit functional form of distn

3-2. Relation to GANs and GMMNs

GAN vs AAE

- (GAN) impose data distribution at pixel level on the output layer of NN
- (AAE) training to capture the data distribution & much simpler distn is imposed in much lower dimension

  $\rightarrow$ better test-likelihood

GMMN vs AAE

- (GMMN) use maximum mean discrepency (minimizing the distance between all moments of model & data distn) can be combined with pre-trained droupout auto encoders

GMMN+AE first trains standard dropout AE & then fits distribution

- (AAE) adversarial training procedure acts as regularizer

  $\rightarrow$ better test-likelihood

3-3. Incorporating Label Information in the Adversarial Regularization

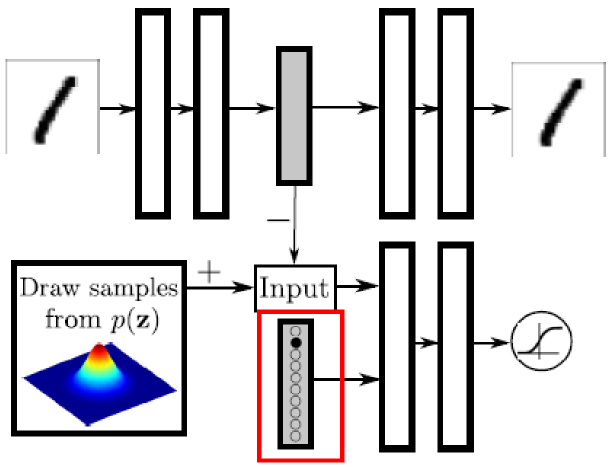
When the data is labeled, use the information!

- leverage partial (or complete) label info to REGULARIZE the latent representation

Semi-supervised approach

- add one-hot vector, which acts as switch that selects correspoding decision boundary of discriminative network, given the class label

  (extra class : for UNKNOWN class .... use full mixture of Gaussian)



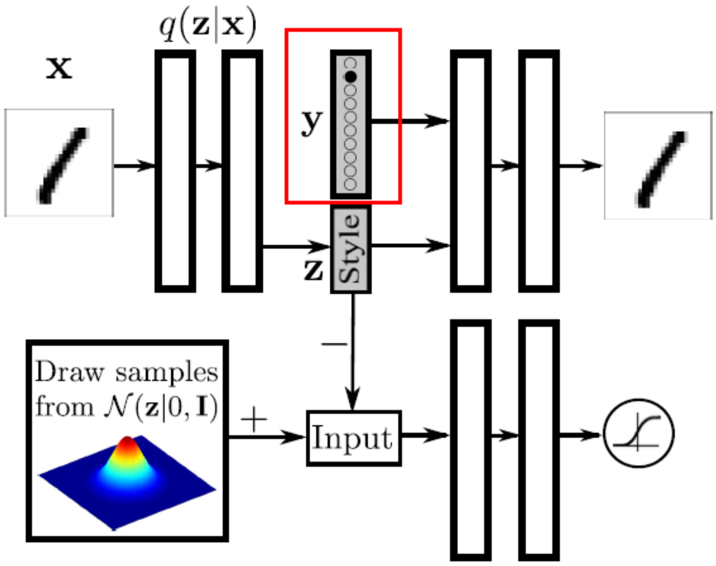
4. Supervised AAE

Generative models have become most popular approaches for **semi-supervised learning**!

In this section, deal with **"fully supervised"** case

Alter network architecture, to provide **one-hot vector encoding** of the label **to the decoder**

(force the network to retain all information independent of the label in $\mathbf{z}$)



5. Semi-Supervised AAE

In this section, deal with **"semi-supervised"** case

Assumption : data is generated by ....

- "latent class variable $\mathbf{y}$ " (which comes from Categorical distn)
- "continuous latent variable $\mathbf{z}$ " (which comes from Gaussian distn)

$$p(\mathbf{y}) = \mathrm{Cat}(\mathbf{y}) \quad p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid 0, \mathbf{I}).$$

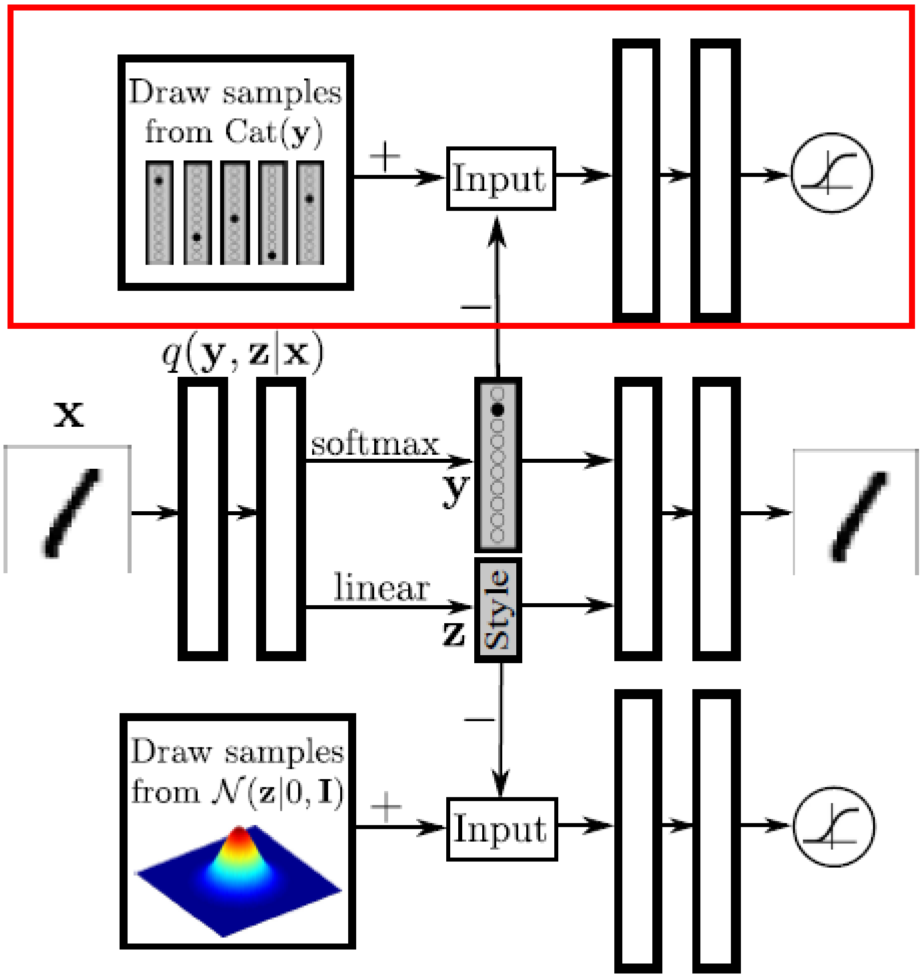
AAE predicts both $\mathbf{y}$ and $\mathbf{z}$, using encoder $q(\mathbf{z}, \mathbf{y} \mid \mathbf{x})$

Then, decoder uses both **(1) class label** $\mathbf{y}$ (as a one-hot vector) and **(2) continuous hidden code** $\mathbf{z}$ to reconstruct image

There are 2 separate adversarial networks

- first network)
  - impose Categorical distn on the label representation
  - ensures that $\mathbf{y}$ does not carry any style info
- second network)
  - impose Gaussian distn on the style representation
  - ensures that $\mathbf{z}$ is a continuous Gaussian variable
- both networks are trained jointly with SGD, in 3 phases
  - phase 1) reconstruction
    - AE updates encoder and decoder to minimize reconstruction error
  - phase 2) regularization
    - each of adversarial networks updates (1) D and (2) G
  - phase 3) semi-supervised classification phase
    - (if phase 3 does not exist, it is UNSUPERVISED task)

- AE updates $q(\mathbf{y} \mid \mathbf{x})$ to minimize cross-entropy



6. Unsupervised Clustering with AAE

In this section, deal with **"unsupervised"** case

difference with "semi-supervised" case

- (1) no phase 3) in above
- (2) inference network $q(\mathbf{y} \mid \mathbf{x})$ predicts one-hot vector, whose dimension is the number of categories which we specify

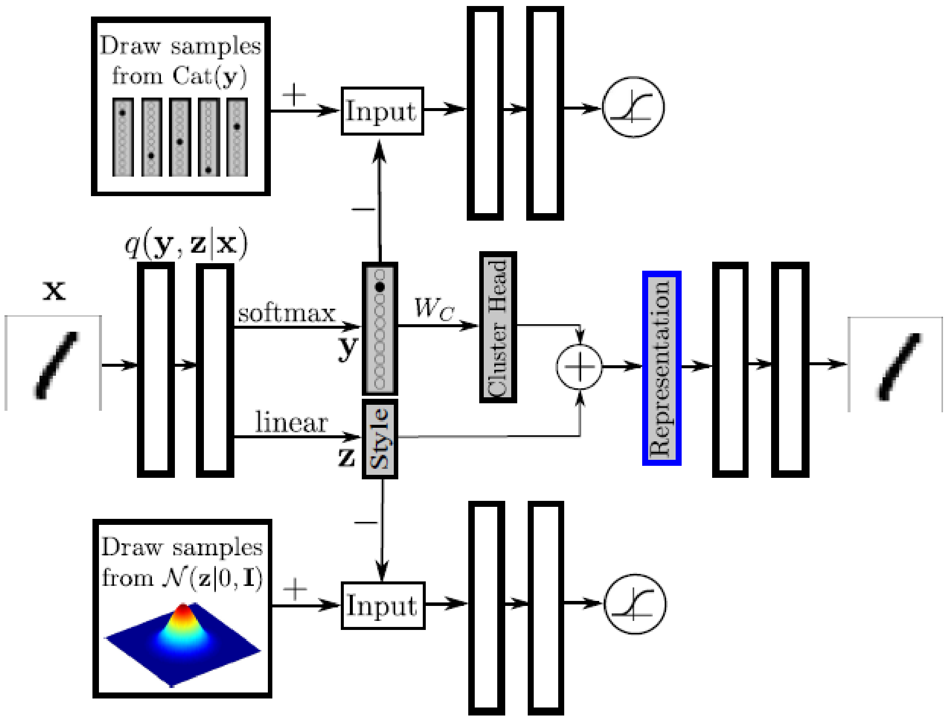
7. Dimensionality Reduction with AAE

visualization of high-dimensional data

- ex) t-SNE

  (drawback : do not have a parametric encoder to encode new data point)

By adding n dim distributed representation of the cluster head, with n dim style representation



8. Conclusion

- propose to use GAN framework as VI algorithm

  (for both discrete & continuous latent variables, in probabilstic AE)
- AAE = generative autoencoder