Week 9

Progress during 24-30th Aug

Hao SUN

August 30, 2017

Contents

Progress in this week

- Classification task
  1. A paper: Learning Hierarchical Features from Generative Models
  2. Some mathematical intuitons and interpretations
  3. Use a hierarchical structure that can deal with larger dataset

Generation vs. classification

Knowing more about the high-level features can not help to generate consistent images. Knowing more about the low-level features can not help to perform classification tasks.

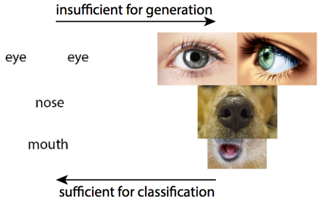


Figure: Fig1. of the paper[1]

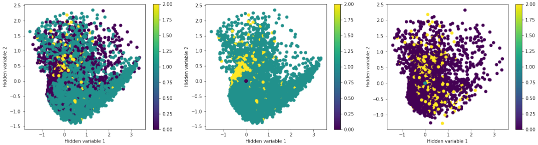
If a hierarchical generative model attempts to reconstruct an image based on these high-level features, it could generate inconsistent images, even when each part can be perfectly generated.

[1] Learning Hierarchical Features from Generative Models, Shengjia Zhao et al. 2017

Can VAE learn high-level features?

The most successful generative models often use only a single layer of latent variables [2], and those that use multiple layers only show modest performance increases in quantitative metrics such as log-likelihood [3].

1. when using more hidden variables (30), the representation ability would be more sufficient to generate consistant images. But the features are more likely to be in lower-level. So that unsupervised clustering methods can not work here.
2. when using less hidden variables (2), the representation ability would be less sufficient to generate consistant images, although the features are more likely to be in higher-level. So that accuracy of unsupervised learning is limited.



[2] Radford et al., 2015; van den Oord et al., 2016

[3] Sønderby et al., 2016; Bachman, 2016

A hierarchical design

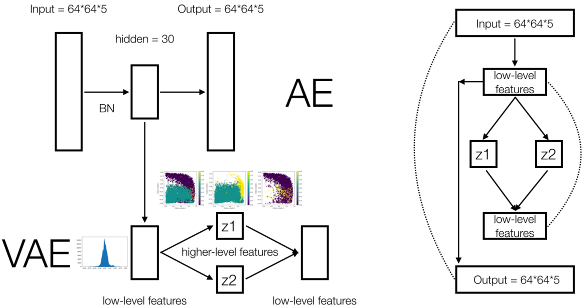
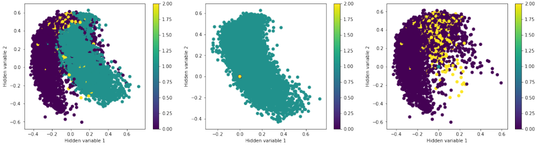


Figure: Left:AE + VAE; Right: VAE with the loss defined by AE

A hierarchical design

1. AE for low-level feature representation
    - Reproduce the original images with 30 hidden variables is easy
    - It's more robust to normalization methods.
    - Few hyperparameters to tune.
    - Can get 93% supervised accuracy
2. Then a VAE for classification
    - Use VAE instead of AE to use the double peak Gaussian prior
    - Lots of hyperparameters to tune (m, s, KL-term...); faster and easier
    - Was sensitive to initialization; use batchnormalization in AE
    - The optimization goal is more accessible: reproduce the 30 hidden variables in AE using 2 hidden units.



Math part[4]

- "Generative modeling" is a broad area of machine learning which deals with models of distributions $P_{gt}(X)$, defined over datapoints X in some potentially high-dimensional space $\mathcal{X}$.
- Our goal is to learn a $P(X)$ that is as similar as possible to $P_{gt}(X)$
- Our SDSS images are $64 \times 64 \times 5$-dimensional images. Sample from such high-dimensional space is computationally expensive.
- An idea is that, if we can use some latent variable z that we can easily sample according to some probability density function (PDF) $P(z)$ and then use a family of deterministic functions $f(z;\theta)$, parameterized by a vector θ in some space Θ. s.t.

$$P(X) = \int P_\theta(X|z)P(z)\,dz$$

[4] Tutorial on Variational Autoencoders, CARL DOERSCH, Carnegie Mellon / UC Berkeley

Math part cont.

We are aiming at maximize the probability of each X in the training set under the entire generative process, according to:

$$P(X) = \int P_\theta(X|z)P(z)dz \tag{1}$$

The intuition here is maximum likelihood
In order to implement gradient descent (or any other optimization technique) to increase $P(X)$ by making $f(z;\theta)$ approach X for some z, the choice of this output distribution is often Gaussian, i.e. $P(X|z;\theta) = N(X|f(z;\theta), \sigma^2 * I)$
Now the problems in optimizing (1) are:

1. how to define the latent variables z
2. how to deal with the integral over z

Math part cont.

1. how to define the latent variables z
   - Any distribution in d dimensions can be generated by taking a set of d variables that are normally distributed and mapping them through a sufficiently complicated function
   - Provided powerful function approximators, we can simply learn a function which maps our independent, normally-distributed z values to whatever latent variables might be needed for the model
   - $P(z) = \mathcal{N}(0, I)$
2. how to deal with the integral over z
   - $P(X) \approx \frac{1}{n} \sum_i P(X|z_i)$
   - Since $P(X|z)$ is an isotropic Gaussian, the negative log probability of X is proportional squared Euclidean distance between $f(z)$ and X