ENGR-E 533 “Deep Learning Systems” Lecture 12: Generative Models

Minje Kim

Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: http://minjekim.com

Research Group: http://saige.sice.indiana.edu

Meeting Request: http://doodle.com/minje

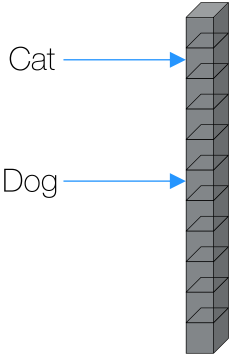
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Latent Representation of Data

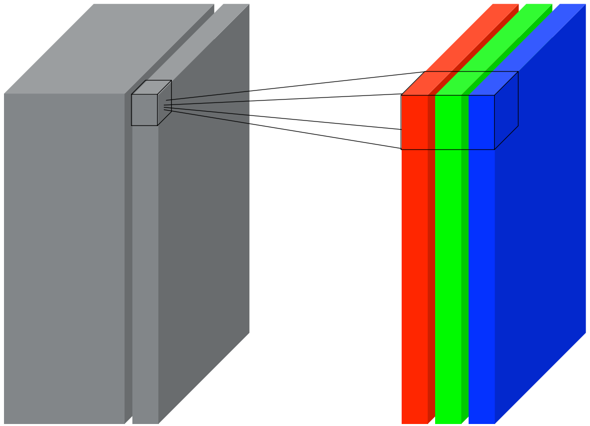
- A decoder made of deconvolution layers

- Suppose I want to create an image of a cat
  - What should the code vector look like?
    - Perhaps a one-hot vector?
- How about a dog?
- Any problem with this?



• • •

Deconvolution Layers



A colorful cat image

  - We can’t really distinguish different cants
  - We can’t really differentiate cats and lions versus cats and dogs

Latent Representation of Data

- Latent embedding vector

- How about this?

0	0.81	0	0	0	0	0	0	0	0	← A cat
0	0.78	0	0	0	0	0	0	0	0	← Another cat

- An even better representation

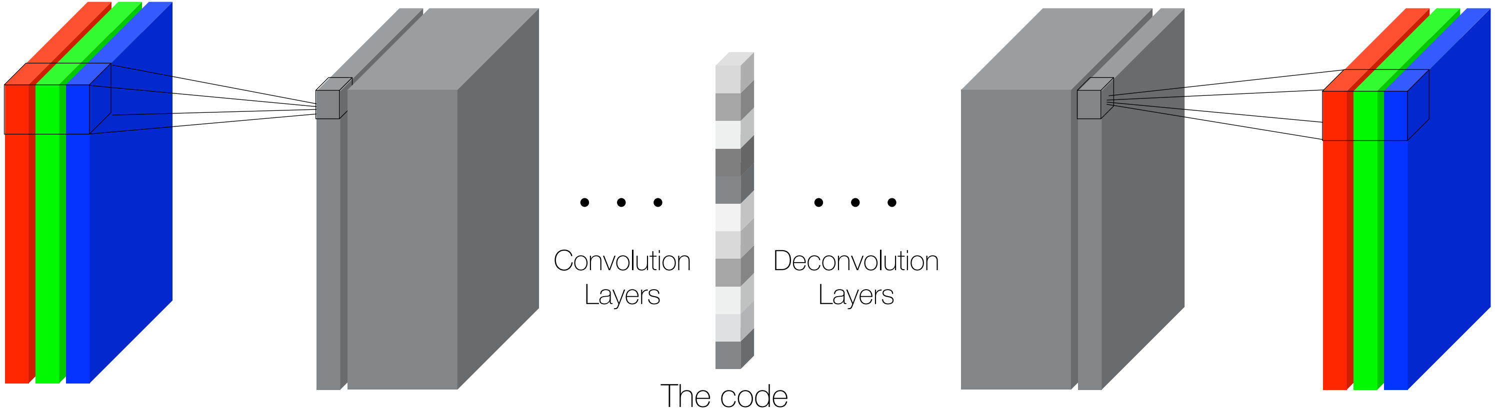
0	0.81	0	0	0	0.1	0	0	0	0	← A cat a little looking like a dog
0	0.78	0	0	0	0.55	0	0	0	0	← Another cat looking a lot like a dog

  - And so on

- Eventually we need a latent representation with real numbers

Autoencoders with A Priori Knowledge

- Regularized autoencoders



- What do we want from the codes?
  - Minimal loss of information
- How do you define “information”?
  - The decoder should generate a similar output to the input
  - Originally similar input examples share similar codes
  - **As a generative model**
    - **Should be able to create examples from unseen codes**

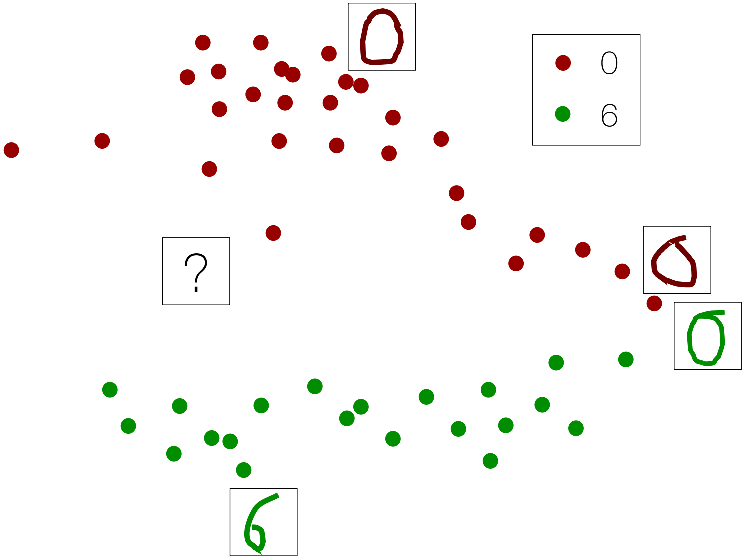
Autoencoders with A Priori Knowledge

- Regularized autoencoders

- Problems with vanilla AEs
  - No control over the latent variables
  - Why do we need the control in the first place?
- A sample in the latent space
  - e.g. Could be something in between the two classes
  - Can we be more specific?
- The p.d.f. of the latent variables given the data set

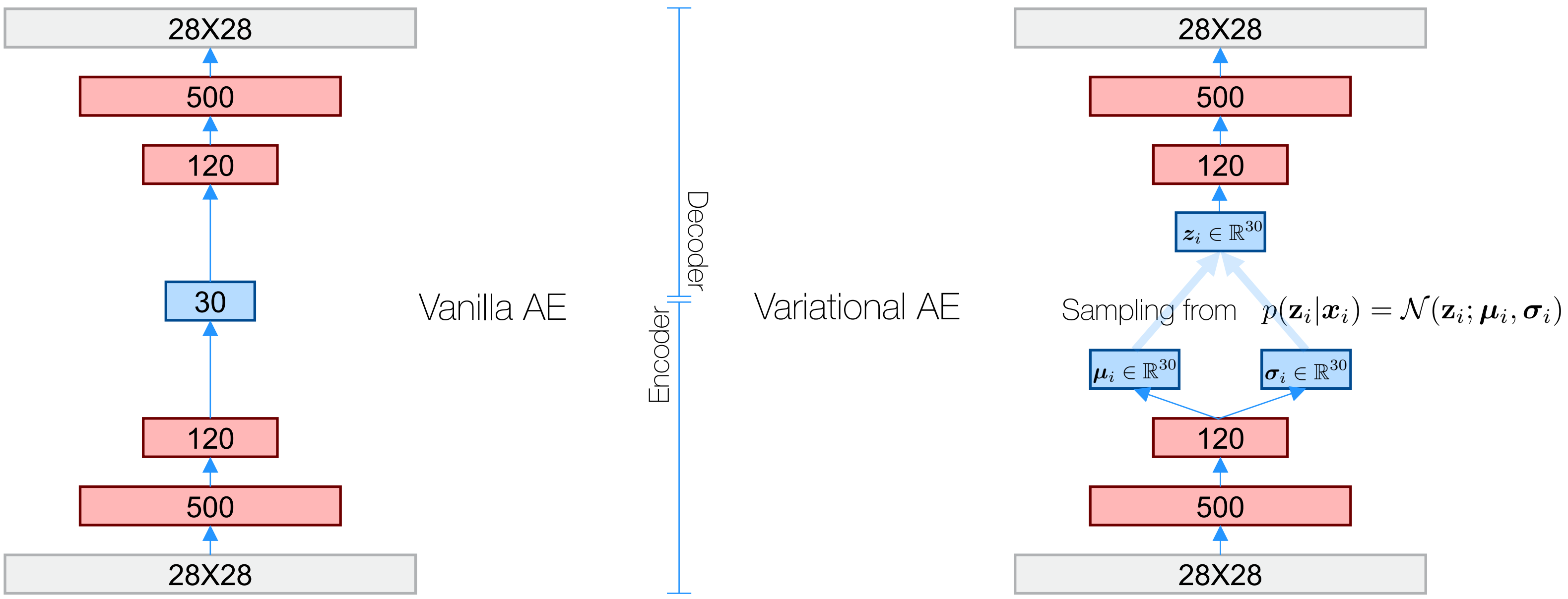
  $$p(\mathbf{z}|\boldsymbol{X})$$

  - We want to know it better
    - This could be a nasty one
  - Furthermore, we want it to be more intuitively distributed
    - e.g. z_1 is for thickness; z_2 is for rotation; etc.
    - Why?
  - How could you "generate" a new example with a choice of thickness if the p.d.f. is too complicated?



Variational AutoEncoders

- Latent variables are from multivariate Normal distributions

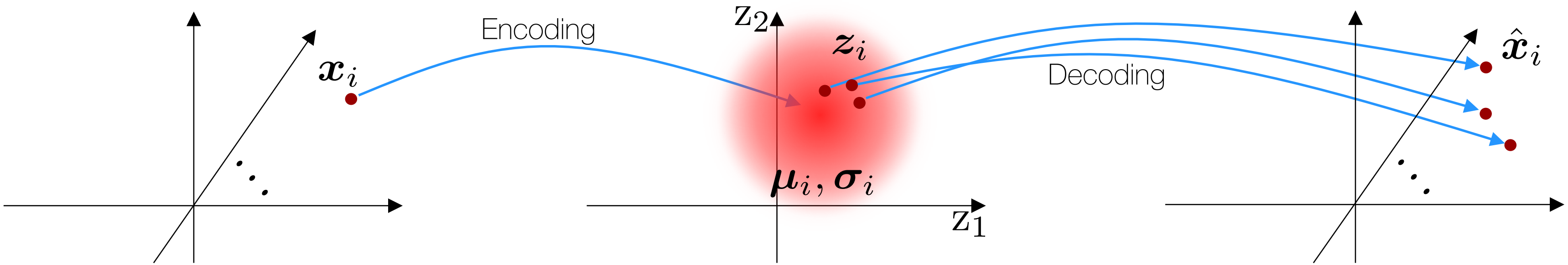


- Why this weird structure?
  - You want an easy distribution for $p(\mathbf{z}_i|\boldsymbol{x}_i)$
  - But it makes training difficult—needs a special structure

Variational AutoEncoders

- Latent variables are from multivariate Normal distributions

Sampling turns decoding into a stochastic process,
enabling it to learn from multiple samples of the LV



Parameterization of the Normal distribution
lets encoding result in a distribution over LV

- During training
  - Every epoch the reconstructed input (output of the decoder) is from a sampled version of the latent variable
  - The training algorithm is exposed to the continuous distribution of the LV, which is not possible with vanilla AE
- During testing
  - The decoder itself is deterministic, but produces stochastic predictions due to the sampling process