Lecture 19: Deep generative models

CS 182/282A (“Deep Learning”)

2022/04/06

Today's lecture

- Today's lecture will be a crash course on **deep generative models**
- Next lecture will be a crash course on **self-supervised learning**
  - Want a full course on these? Check out the Deep Unsupervised Learning course (much of these slides' content are borrowed from there)
- Deep generative models attempt to understand data (using deep networks) in a **label-free way**, specifically through density/distribution modeling
- We will blaze through a number of different applications of generative models, along with different types of models that are suitable for these applications

Density/distribution modeling

Some potential motivations

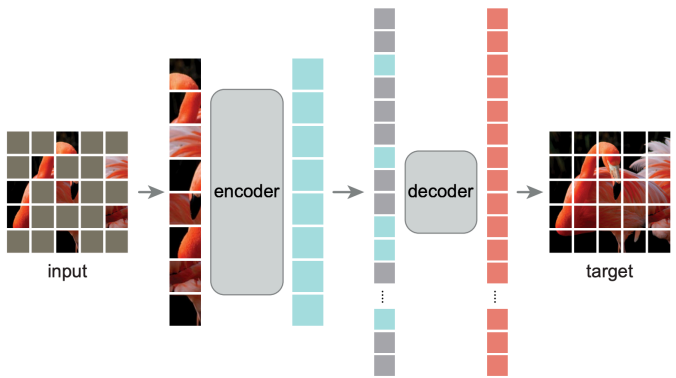
- Why might we be interested in trying to model the data distribution?
- **Generation**: synthesize new data points that are useful/interesting/pretty
  - **Conditional generation**: synthesize specific data points of interest
- **Density modeling**: understand/analyze the likelihood of various data points
  - Doesn't work that well OOD, at least not yet, but still could be useful…
- **Representation learning** (or *compression*, or *dimensionality reduction*, etc.): convert high dimensional data into lower dimensional representations

Have we already seen some generative models?

- Yes we have!

Prompt: Recycling is good for the world, no, you could not be more wrong.

GPT-2: Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources.



Generating new data

- In NLP, **autoregressive models** like GPT reign when it comes to generation
- Masked autoencoders are capable of generating images, but they are far from state-of-the-art and that is not their primary function
- For computer vision, the leading approaches for generation are **generative adversarial networks (GANs)** and, recently, **diffusion/score-based models**
- For audio, the leading approach is also **autoregressive modeling**
- We will briefly cover all of these types of models and more

Generative adversarial networks

- The GAN framework defines optimization as a game between two competitors: a **generator** G that is synthesizing data from noise inputs, and a **discriminator** D which is trying to distinguish real data from synthesized data

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

- We saw discriminators in domain adaptation, though that probably came after
- Theory: optimal $G^\star$ under Bayes optimal $D^\star$ recreates the data distribution
- After training: throw away D, use G to generate new samples

Generating images with GANs

https://twitter.com/goodfellow_ian/status/1084973596236144640

- Many works have contributed to making GANs work better; roughly in order: DCGAN, Improved Training of GANs, WGAN(-GP), Progressive GAN, SN-GAN, SAGAN, BigGAN
  - Better architectures, modified objectives, bags of tricks, scaling up (of course)

Conditional generation with GANs

- There are a number of ways to condition the generator in order to “guide” the data that it produces
- If class information is available, we can directly pass in the label to both G and D
  - Even without class information, including a “label-like” (uniform categorical) input can allow G to do unsupervised discovery of different categories
  - Here, additional objectives may further help, such as maximizing the mutual information between the generator output and the “label-like” input (InfoGAN)
- What else can we condition on?

Unsupervised image-to-image translation

- Another success story of GANs is translating images from one domain (e.g., pictures of horses) into corresponding images of another domain (e.g., zebras)
- Most well known is CycleGAN, which trains two image-to-image generators that turns images from one domain into images of the other domain
  - And, correspondingly, two discriminators (one for each domain)

GANs summary

- GANs are the go-to model for image generation
  - Other types of models are catching up in terms of generated quality, however, they currently have other downsides such as being much slower to generate
- However, GANs are not density models, and there is not really a way to obtain probability estimates of data points from a GAN
  - Consequently, this makes GANs harder to evaluate as well — how can we say whether one GAN is better than another GAN (or some other model)?
  - Metrics for evaluating GANs usually rely on inputting generated images into a pretrained classifier (e.g., *Inception* and *FID* scores), but this is contentious

Density models on images

- If we care about more than just image generation, we may be better off trying to learn a density model $p_\theta(\mathbf{x})$
  - It may sometimes (but, as we have discussed, certainly not always) be useful for detecting anomalous $\mathbf{x}$s
  - At the very least, this type of model prescribes a clear approach for both training (MLE) and evaluation (better held out likelihood means better model)
- There are at least three main classes of deep generative models that allow for estimating $p_\theta(\mathbf{x})$ — **autoregressive**, **latent variable**, and **flow-based models**

Autoregressive models

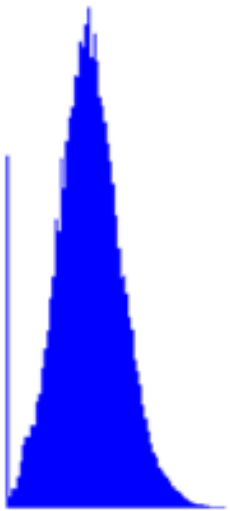
- Conceptually, autoregressive models are the simplest type: we just factorize the density according to the chain rule $p_\theta(\mathbf{x}) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{<i})$
  - $\mathbf{x}$ may naturally be sequential (e.g., audio), or we can define such an ordering
- The key question is then how to define the model such that we get $p_\theta(x_i \mid x_{<i})$
- Think back to how we did this with the transformer decoder — *masking*
- We can do the same for images, e.g., masked convolutions (*PixelCNN(++)*) and masked self-attention (*PixelSNAIL*, *Image Transformer*, *Sparse Transformer*)

Aside: defining a distribution over pixels

- What distribution is $p_\theta(x_i \mid x_{<i})$ for image pixels and how do we parameterize it?
- One simple option is to make it a 256-dim Categorical (softmax) distribution
  - But this is rather expensive and doesn't capture useful inductive biases, e.g., nearby values are closer than far away values
- One effective approach is to use a mixture of (truncated) logistic distributions
- Other simpler options can also sometimes work, e.g., if we treat pixels as continuous values in [0, 1], we can use Beta, Kumaraswamy, or (my favorite) Logit-Normal distributions

Generating from autoregressive models

Figures originally by Aaron van den Oord



Autoregressive models summary

- Autoregressive models offer “best in class” modeling performance, oftentimes both qualitatively in terms of generation and quantitatively in terms of likelihood metrics
- This is true beyond images — there are very good autoregressive models for language (we already knew this) and audio (e.g., a somewhat dated example is WaveNet)
- Similar to GANs, we can modify autoregressive models to do conditional generation
- GANs are still superior when it comes to image generation
- Generating from autoregressive models is also very slow, comparatively speaking
- Lastly, autoregressive models do not naturally provide a notion of a latent space, so they are not used for representation learning

Representation learning

- What deep generative models might be useful for representation learning?
- The model should have a notion of a **latent representation $\mathbf{z}$** that is perhaps lower dimensional or otherwise "simpler" than $\mathbf{x}$
  - Autoregressive models did not have this; GANs did
- The model ideally would have an **encoder** that maps $\mathbf{x}$ to $\mathbf{z}$
  - GANs did not have this, though a variant called BiGAN does
- Bonus: modeling $\mathbf{z}$ as probabilistic could be useful for some applications

Latent variable models, the formalism

- Let's start by treating both $\mathbf{x}$ and $\mathbf{z}$ as probabilistic and see what happens
- We directly model $p(\mathbf{z})$, which is the **prior** (e.g., unit Gaussian) and $p_\theta(\mathbf{x} \mid \mathbf{z})$, which is the **observation model** (or *generator*, *generative model*, *decoder*, …)
- The likelihood of a data point is $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$ — totally intractable
- We are interested in the distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$, which is also intractable…
- Idea: what if we model $p_\theta(\mathbf{z} \mid \mathbf{x})$ with another distribution $q_\phi(\mathbf{z}; \mathbf{x})$?

The evidence lower bound (ELBO)

$$D_{KL}(q_\phi(z;x) \,\|\, p_\theta(z|x)) = \mathbb{E}_{q_\phi}[\log q_\phi(z;x) - \log p_\theta(z|x)]$$

$$= \mathbb{E}_{q_\phi}[\log q_\phi(z;x) - \log p_\theta(x,z)] + \log p_\theta(x)$$

$$= \mathbb{E}_{q_\phi}[-\log p_\theta(x|z) + \log q_\phi(z;x) - \log p(z)] + \log p_\theta(x)$$

$$= -\underbrace{(\mathbb{E}_{q_\phi}[\log p_\theta(x|z)] - D_{KL}(q_\phi \,\|\, p_z))}_{\text{evidence lower bound}} + \log p_\theta(x)$$

$$\geq 0$$

Variational autoencoders (VAEs)

- In **VAEs**, both the observation model $p_\theta(\mathbf{x} \mid \mathbf{z})$ and **recognition model** (also called *encoder*) $q_\phi(\mathbf{z}; \mathbf{x})$ are trained to maximize the evidence lower bound — this is an example of the *variational inference (VI)* framework
- Intuitively, the ELBO contains a reconstruction term and a regularization term
- After training, we may elect to keep around both models
- q_ϕ provides a natural approach for generating representations of new data points
- p_θ, combined with the prior, allows us to synthesize new data points

An aside: other types of autoencoders

- There are other types of deep *autoencoders* (paired encoders and decoders) that we won't cover: denoising autoencoders, bottleneck autoencoders, …
  - These are not used much anymore compared to VAEs
- We have seen the masked autoencoder previously and studied its strengths
  - So far, masked autoencoders have primarily been used for representation learning, and their reconstructions are quite poor
  - Generation from masked autoencoders (and other types of autoencoders) is also non obvious, as there is no prior distribution on the latent variable

VAEs summary

- VAEs provide natural mechanisms for both representation learning and generation
  - Though estimating $p_\theta(\mathbf{x})$ is difficult, a lower bound can easily be obtained
- However, there is typically a tradeoff involved between representation learning and generation quality
- The VAEs which synthesize the best data points and result in the best (lower bounds of) likelihoods utilize complex priors and modeling choices, e.g., quantization and multiple levels of latent variables
  - This can make extracting useful representations more difficult

Briefly: flow models

- Another important class of deep generative models is **flow models**, which parameterize an *invertible* transformation between $\mathbf{x}$ and $\mathbf{z}$
  - Consequently, $\mathbf{z}$ has to be the same dimensionality as $\mathbf{x}$, but it can be defined to follow a much simpler distribution, e.g., independent unit Gaussians
- Exact likelihood computations can be done by transforming $\mathbf{x}$ to $\mathbf{z}$ and following the *change of variables formula* for continuous random variables
- Sampling is also straightforward: sample $\mathbf{z}$ and transform it into $\mathbf{x}$
- The problem of how to define effective invertible transformations is an area of much active research

Briefly: energy-based models (EBMs)

- **EBMs** parameterize an *energy function* $e_\theta(\mathbf{x})$ and define the likelihood as $p_\theta(\mathbf{x}) = \frac{\exp\{-e_\theta(\mathbf{x})\}}{Z_\theta}$, where $Z_\theta = \int \exp\{-e_\theta(\mathbf{x}')\} d\mathbf{x}'$
- Math exercise (if you want): $\nabla_\theta \log p_\theta(\mathbf{x}) = -\nabla_\theta e_\theta(\mathbf{x}) + \mathbb{E}_{p_\theta}[\nabla_\theta e_\theta(X)]$
- Sampling from p_θ (both for estimating the above gradient and generation after training) are typically done by *Markov chain Monte Carlo (MCMC)* techniques — MCMC and VI together comprise the two main "workhorses" of approximate probabilistic inference
- EBMs are primarily appealing due to their similarities to the discriminative model "interface"
- However, it is difficult to obtain density estimates from EBMs, and they are rather hard to train

Briefly: diffusion models

- **Diffusion models** define a process by which $\mathbf{x}$ is transformed, little by little, via additive Gaussian noise, into $\mathbf{z}$, which is pure noise
  - This is similar in spirit to flow models, but we do not require invertibility
- Diffusion models parameterize the *reverse process* ($\mathbf{z}$ to $\mathbf{x}$), and the *forward process* ($\mathbf{x}$ to $\mathbf{z}$) is fixed as incrementally adding small amounts of noise
  - Similar to VAEs, we then train by maximizing the evidence lower bound
- Diffusion models generate impressive samples, however, sampling is expensive

Briefly: score-based models

- **Score-based models** parameterize the score function $s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$
- Training is done via *score matching:* roughly, making $s_\theta(\mathbf{x})$ close to $\nabla_{\mathbf{x}} \log p(\mathbf{x})$
- Recently, a powerful "multi-step" score-based model was proposed which models a process by which images are gradually corrupted via noise — sound familiar?
  - There are deep connections between this model and diffusion models
- Different views of the same model type can lead to new strengths, e.g., this model allows for straightforward exact likelihood computation rather than just a lower bound
- However, some weaknesses are shared, such as the inefficiency of sample generation