Variational inference and image generation

CPSC 440/550: Advanced Machine Learning

cs.ubc.ca/~dsuth/440/23w2

University of British Columbia, on unceded Musqueam land

2023-24 Winter Term 2 (Jan–Apr 2024)

Need for Approximate Inference

- We've seen a bunch of models where inference can be intractable:
  - Bayesian logistic regression
  - Markov chains with non-Gaussian continuous states
  - Non-forest graphical models
  - The models today :)
- Monte Carlo methods can solve these problems, but it's so slow and fiddly
- Most common alternative is variational methods

Monte Carlo vs. Variational Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
   - Approximate p with the empirical distribution of samples

$$p(x) \approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(x^{(i)} = x)$$

   - Turns inference into sampling
2. Variational methods:
   - Approximate p with "closest" distribution q from a tractable family

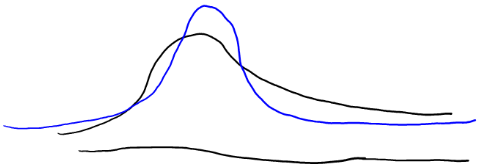
$$p(x) \approx q(x)$$

     - Gaussian, independent Bernoulli, tree-structed UGM, ...

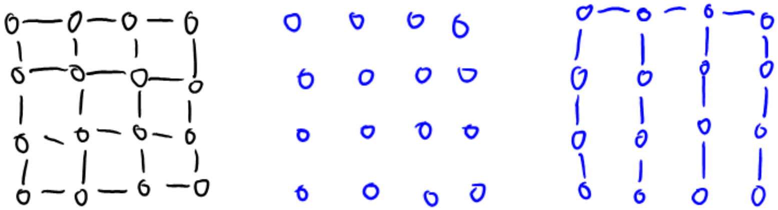
       (or mixtures of these simple distributions)
   - Turns inference into optimization

Variational Inference Illustration

- Approximate non-Gaussian p by a Gaussian q:



- Approximate loopy UGM by independent distribution or tree-structed UGM:



- Variational methods try to find simple distribution q that is closest to target p
  - This isn't consistent like MCMC is, but it can be very fast

Kullback-Leibler (KL) Divergence

- How do we define "closeness" between a distribution p and q?

- A common measure is Kullback-Leibler (KL) divergence between p and q:
$$\mathrm{KL}(p \parallel q) = \int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$$
  - As usual, integral becomes a sum for discrete distributions

- Also called information gain: "information lost when p is approximated by q"
- If $p = q$, we have $\mathrm{KL}(p \parallel q) = 0$ (no information lost)
- Otherwise, $\mathrm{KL}(p \parallel q)$ grows as it becomes hard to predict p from q
- KL is not symmetric: in general, $\mathrm{KL}(p \parallel q) \neq \mathrm{KL}(q \parallel p)$

- Maximumizing likelihood = minimizing $\mathrm{KL}(p_{\text{true}} \parallel p_\theta)$ (bonus slide)

- Unfortunately, this requires summing/integrating over p, or sampling from it
  - ...exactly the problem we're trying to avoid

Minimizing Reverse KL Divergence

- Most variational methods minimize "reverse KL":

$$\mathrm{KL}(q \parallel p) = \int q(x) \log \frac{q(x)}{p(x)} \mathrm{d}x = \int q(x) \log \left(\frac{q(x)}{\tilde{p}(x)} Z \right) \mathrm{d}x$$

  - Not intuitive: "how much information is lost when we approximate q by p"
- "Reverse" KL only needs unnormalized distribution $\tilde{p}$ and expectations over q

$$\begin{aligned} \mathrm{KL}(q \parallel p) &= \int q(x) \log q(x) \mathrm{d}x - \int q(x) \log \tilde{p}(x) \mathrm{d}x + \int q(x) \log(Z) \mathrm{d}x \\ &= \mathop{\mathbb{E}}_{x \sim q} [\log q(x)] - \mathop{\mathbb{E}}_{x \sim q} [\log \tilde{p}(x)] + \underbrace{\log(Z)}_{\text{const. in } q} \end{aligned}$$

- $-\mathbb{E}_{x \sim q} \log q(x) = \mathrm{H}[q]$ is the (differential) entropy of q
  - Value is known for many common choices of q

$$\arg\min_q \mathrm{KL}(q \parallel p) = \arg\max_q \mathop{\mathbb{E}}_{x \sim q} \log \tilde{p}(x) + \mathrm{H}[q]$$

Example: Best Multivariate Gaussian

- We want to find $\max_q \mathbb{E}_{x\sim q}[\log \tilde{p}(x)] + \mathrm{H}[q]$
- For multivariate Gaussians, we have $\mathrm{H}[q] = \frac{1}{2}\log|\mathbf{\Sigma}| + \frac{d}{2}\log(2\pi e)$
- So to find the best multivariate Gaussian approximation, we need to find

$$\underset{\boldsymbol{\mu},\mathbf{\Sigma}}{\arg\max}\, \frac{1}{2}\log|\mathbf{\Sigma}| + \underset{x\sim\mathcal{N}(\boldsymbol{\mu},\mathbf{\Sigma})}{\mathbb{E}} \log \tilde{p}(x) = \underset{\boldsymbol{\mu},\mathbf{L}}{\arg\max}\, \log|\mathbf{L}| + \underset{z\sim\mathcal{N}(\mathbf{0},\mathbf{I})}{\mathbb{E}} \log \tilde{p}(\boldsymbol{\mu} + \mathbf{L}z)$$

- How to optimize this? Can't autodiff through expectation...
- Reparamaterization trick: take variable we're optimizing out of the expectation
- End up with $q = \mathcal{N}(\boldsymbol{\mu}, \mathbf{L}\mathbf{L}^\mathsf{T})$
- If L is lower-triangular with $L_{jj} > 0$ (Cholesky factor), then $|L| = \prod_j L_{jj}$ is easy
  - A3 code for `MultivariateT.mle()` used this trick
- Can take samples for z and run SGD to optimize (but note it's non-convex)

Mean Field / Variational Bayes approximation

bonus!

- Another common scheme is coordinate optimization with an appropriate q
- Consider choosing q as a product of independent q_j

$$q(x) = \prod_{j=1}^{d} q_j(x_j)$$

- If we fix $q_{\neg j}$ and optimize q_j among all distributions, we get (see PML2 10.2)

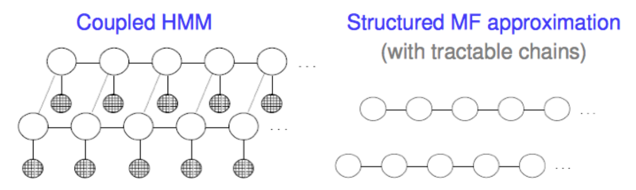
$$q_j(x_j) \propto \exp\left(\mathop{\mathbb{E}}_{q_{\neg j}}\left[\log \tilde{p}(x)\right]\right)$$

- Iterative algorithm: pick j, choose (discrete or conjugate) q_j to match above
  - Each iteration improves the (non-convex) reverse KL

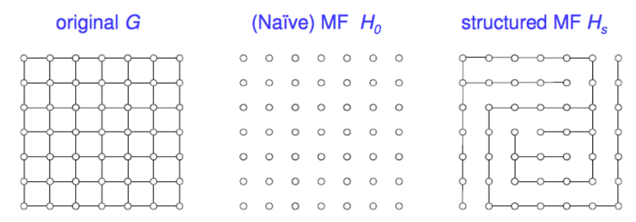
Structured Mean Field

bonus!

- Common variant is structured mean field: q function includes some of the edges



`http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf`



`http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf`

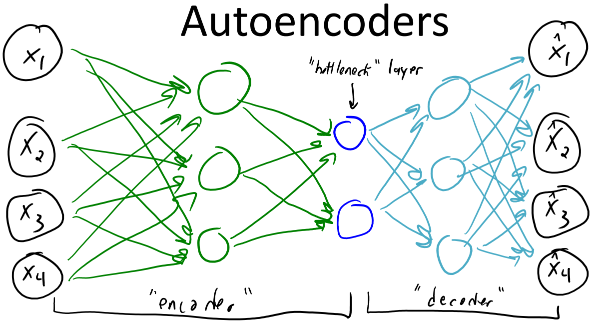
Variational vs. Monte Carlo

- Compared to MCMC, variational methods are typically:
  - more complicated
  - not consistent (q doesn't converge to p if we run the algorithm forever)
  - harder to parallelize
  - better approximations for a given amount of computation
- Variational methods typically have similar cost to MAP
- Combinations of variational inference and stochastic methods:
  - Stochastic variational inference (SVI): use SGD to speed up variational methods
  - Can initialize MCMC parameters based on a variational estimate
  - Variational MCMC: use Metropolis-Hastings with proposals from a variational q

Outline

Autoencoders

- Way back in lecture 6, we talked about auto-encoders:



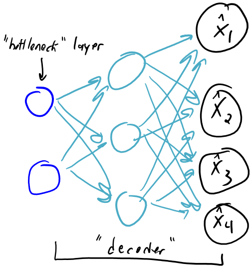
- Autoencoders try to make their output the same as the input
  - Usually have a bottleneck layer with dimension k < input d
  - First layers "encode" the input into bottleneck
  - Last layers "decode" the bottleneck into a (hopefully valid) input

Autoencoders

- Way back in lecture 6, we talked about auto-encoders:

Decoder as Generative Model

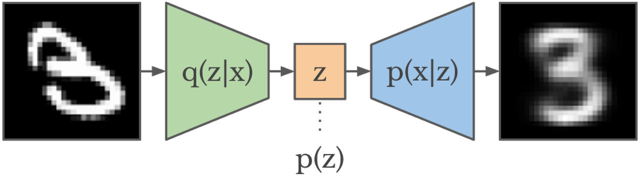
- Consider the decoder part of the network:
  - Takes low-dimensional $z^{(i)}$ and makes features $\hat{x}^{(i)}$
- Can be used for outlier detection:
  - Check distance to original features to detect outliers
- Can be used to generate "new data":
  - If the decoder is good, new values of z that "look like real z" should decode into $\hat{x}$ that "look like real x"
  - To do this "properly," need to estimate the distribution p(z)
    - This is what "Stable Diffusion" does



- There's another option for sampling: make $p(z)$ into something simple
- If $p(z)$ is $\mathcal{N}(\mathbf{0}, \mathbf{I})$, then we can easily sample from it

Variational Auto-encoders

- VAEs choose to make everything probabilistic:



https://danijar.com/building-variational-auto-encoders-in-tensorflow/

  - Encoder network $q_\phi(z \mid x)$ gives a *distribution* over latent codes for x
  - Decoder network $p_\theta(x \mid z)$ gives an x for a given z
  - Prior distribution $p_\theta(z)$ is usually $\mathcal{N}(\mathbf{0}, \mathbf{I})$

- Another view: fitting a deep latent variable model $p_\theta(x) = \int p_\theta(x \mid z) p_\theta(z) \mathrm{d}z$
- We can sample from p_θ ancestrally: $z \sim p_\theta(z)$, $x \sim p_\theta(x \mid z)$
- But if z is high-dimensional, that integral is way too hard; how can we fit θ?
- We use a "recognition" network $q_\phi(z \mid x) \approx p_\theta(z \mid x) = \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(x)}$
  - "Amortized inference" – we amortize the work of conducting (intractable) inference

ELBO

- We'd like to maximize $p_\theta(x) = \int p_\theta(x \mid z) p_\theta(z) \mathrm{d}z$

$$
\begin{aligned}
\log p_\theta(x) &= \mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} [\log p_\theta(x)] \\
&= \mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{p_\theta(z \mid x)} \right] \\
&= \mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)\, q_\phi(z \mid x)}{q_\phi(z \mid x)\, p_\theta(z \mid x)} \right] \\
&= \mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] + \mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} \left[\frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right] \\
&= \mathrm{ELBO}_{\theta,\phi}(x) + \mathrm{KL}(q_\phi(z \mid x) \parallel p_\theta(z \mid x))
\end{aligned}
$$

- Since $\mathrm{KL} \geq 0$, $\mathrm{ELBO}_{\theta,\phi}(x) = \log p_\theta(x) - \mathrm{KL}(q_\phi(z \mid x) \parallel p_\theta(z \mid x)) \leq \log p_\theta(x)$
  - ELBO is the Evidence Lower BOund

Maximizing the ELBO

- Once we know how to evaluate it, we can use as our loss

$$\sum_{i=1}^{n} \mathrm{ELBO}_{\theta,\phi}(x^{(i)}) = \sum_{i=1}^{n} \log p_\theta(x^{(i)}) - \mathrm{KL}(q_\phi(z^{(i)} \mid x^{(i)}) \parallel p_\theta(z^{(i)} \mid x^{(i)}))$$

- Because KL ≥ 0, this is a lower bound on the log-likelihood
- Maximizing over the encoder/recognition parameters ϕ is

$$\arg\max_\phi \sum_{i=1}^{n} \mathrm{ELBO}_{\theta,\phi}(x^{(i)}) = \arg\min_\phi \sum_{i=1}^{n} \mathrm{KL}(q_\phi(z^{(i)} \mid x^{(i)}) \parallel p_\theta(z^{(i)} \mid x^{(i)}))$$

  - Finds a network that gives you a low reverse KL, for any training input $x^{(i)}$
  - Making the inference network better makes the likelihood bound tighter
- If $q_\phi(z \mid x) \approx p_\theta(z \mid x)$ (on the training set),
  maximizing over the probability parameters θ (approximately) maximizes likelihood

Evaluating the ELBO

- We'll actually be able to evaluate the ELBO:

$$
\begin{aligned}
\mathrm{ELBO}_{\theta,\phi}(x) &= \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right] \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)p_\theta(z)}{p_\theta(z)q_\phi(z \mid x)}\right] \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{p_\theta(z)}\right] + \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(z)}{q_\phi(z \mid x)}\right] \\
&= \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}(q_\phi(z \mid x) \parallel p_\theta(z))
\end{aligned}
$$

- First term: $q_\phi(z \mid x)$ should give a latent distribution where decoding to x is likely
- Second term: $q_\phi(z \mid x)$ should be "near" $p_\theta(z)$ (regularization)

Computing the ELBO and its gradient: the reparameterization trick

- We want to maximize the average of

$$\mathrm{ELBO}_{\theta,\phi}(x) = \mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} [\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z))$$

- KL term for a given x is available in closed form if $p(z)$, $q_\phi(z \mid x)$ are Gaussian

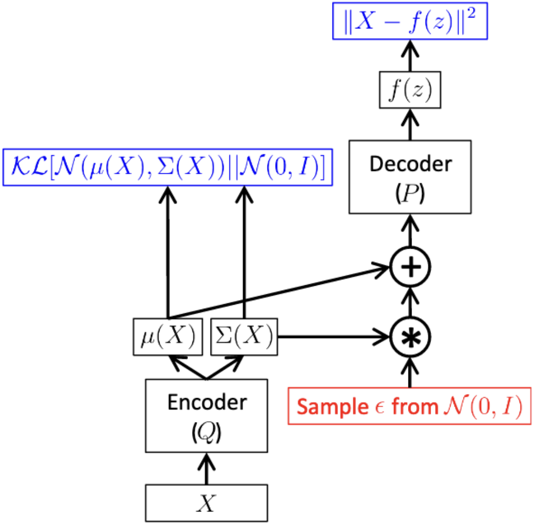
(if $p(z)$ is $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $q_\phi(z \mid x)$ is $\mathcal{N}(\boldsymbol{\mu}_\phi(x), \boldsymbol{\Sigma}_\phi(x))$; regularizes $\|\boldsymbol{\mu}_\phi(x)\|^2$ and $\boldsymbol{\Sigma}_\phi(x)$ to be near $\mathbf{I}$ – bonus)

- For the other term, we need Monte Carlo
- Usually $p_\theta(x \mid z)$ is $\mathcal{N}(f_\theta(z), \sigma^2 \mathbf{I})$, so $\log p_\theta(x \mid z) = -\frac{1}{\sigma^2}\|x - f_\theta(z)\|^2 + \text{const}$
- We need $\mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x \mid z)$
  - Usually estimate with Monte Carlo, with just a single sample for simplicity
- But how do we take ∇_ϕ of this expectation? Use reparameterization trick again:

$$\mathop{\mathbb{E}}_{z \sim q_\phi(z|x)} [\log p_\theta(x \mid z)] = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \log p_\theta(x \mid z = \boldsymbol{\mu}_\phi(x) + \boldsymbol{\Sigma}_\phi(x)^{\frac{1}{2}} \epsilon)$$

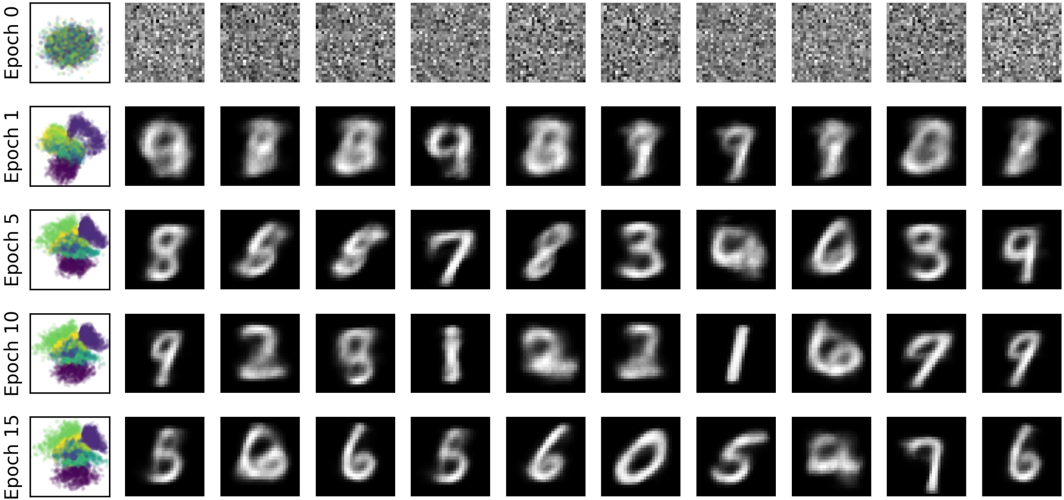
- Take a Monte Carlo sample for ϵ; now have something we can autodiff
- Now just do SGD to maximize $\frac{1}{n} \sum_{i=1}^n \widehat{\mathrm{ELBO}}_{\theta,\phi}(x^{(i)})$

A VAE



https://arxiv.org/pdf/1606.05908.pdf

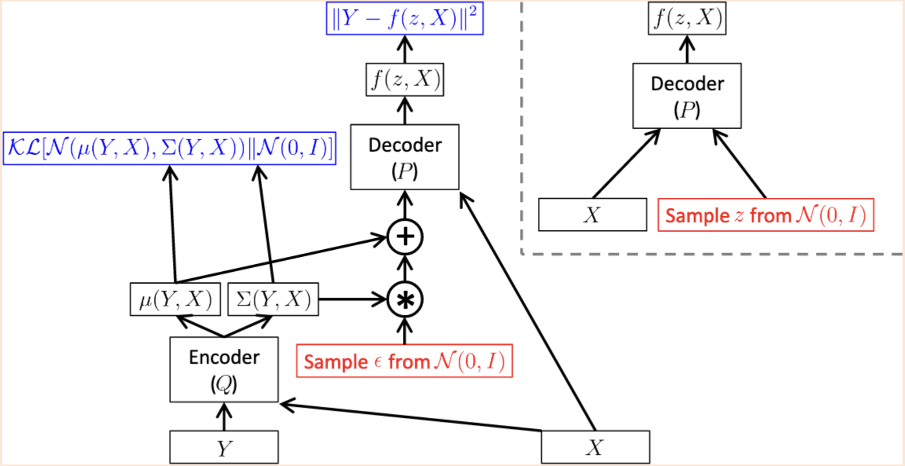
A VAE on MNIST



https://danijar.com/building-variational-auto-encoders-in-tensorflow/

Conditional VAE

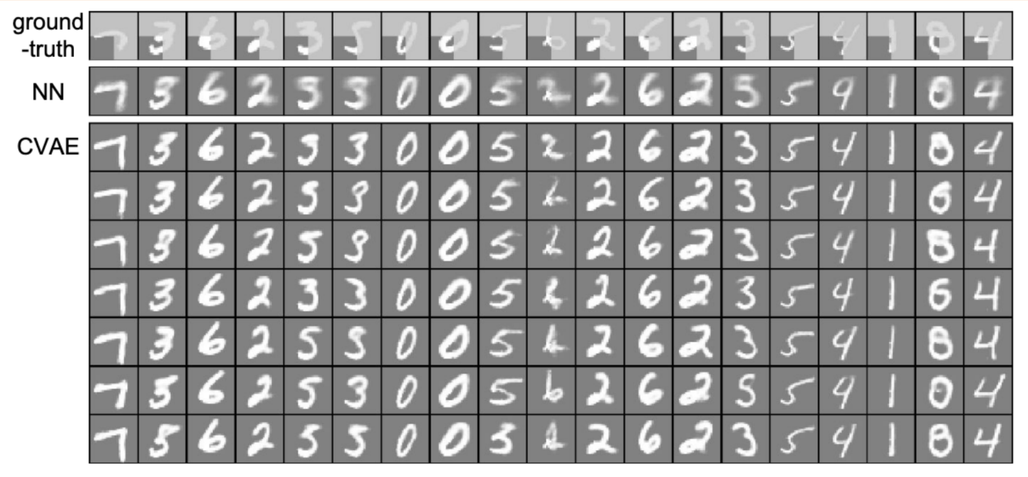
bonus!



https://arxiv.org/pdf/1606.05908.pdf

Conditional VAE to “in-paint” on MNIST

bonus!



https://papers.nips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf

Posterior collapse

bonus!

- What if we use a *really powerful* decoder $p_\theta(x \mid z)$?
- For example, an autoregressive model based on

$$p_\theta(x \mid z) = p_\theta(x_1 \mid z) p_\theta(x_2 \mid x_1, z) \cdots p_\theta(x_d \mid x_1, \ldots, x_{d-1}, z)$$

- If you try this, get great samples. . . that tend to ignore z entirely
- Remember $\mathrm{ELBO}_{\theta,\phi}(x) = \mathbb{E}_{z \sim q_\phi(z \mid x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z))$
    - If $p_\theta(x \mid z)$ ignores z, $q_\phi(z \mid x)$ can be just $p_\theta(z)$ and KL becomes 0

VQ-VAE

bonus!

- One way to avoid this: vector quantized VAE uses a discrete latent space
- Encoder maps to a single discrete value of the latent; learn a prior on them
- Autoregressive decoder is encouraged to “commit” to a latent

- VQ-VAE-2 uses two-layer hierarchical latents
  - Autoregressive prior on the latents, but a fast feed-forward decoder

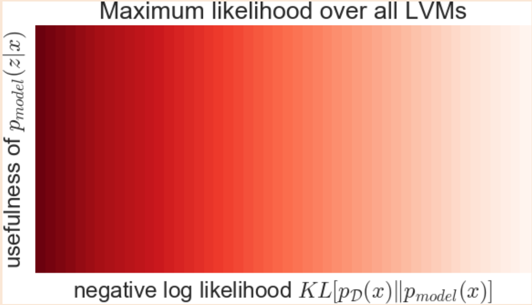
Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

https://arxiv.org/pdf/1906.00446.pdf

Representation Learning with Latent Variable Models

bonus!

- We'd often like a "useful" $p_\theta(z \mid x)$
- Maximum likelihood minimizes KL between target and $p_\theta(x) = \int p_\theta(x, z) \mathrm{d}z$
- Objective wants a good fit for $p_\theta(x)$; doesn't care about usefulness at all
  - True for *any* objective that only cares about $p_\theta(x)$, not just MLE

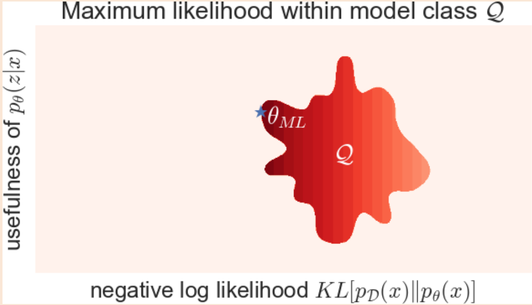


https://www.inference.vc/maximum-likelihood-for-representation-learning-2/

Representation Learning with Latent Variable Models

bonus!

- We'd often like a "useful" $p_\theta(z \mid x)$
- Maximum likelihood minimizes KL between target and $p_\theta(x) = \int p_\theta(x, z) \mathrm{d}z$
- Objective wants a good fit for $p_\theta(x)$; doesn't care about usefulness at all
  - True for *any* objective that only cares about $p_\theta(x)$, not just MLE
- But we don't actually maximize over all latent variable models

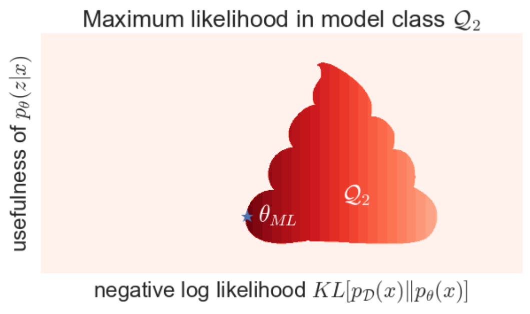


https://www.inference.vc/maximum-likelihood-for-representation-learning-2/

Representation Learning with Latent Variable Models

bonus!

- We'd often like a "useful" $p_\theta(z \mid x)$
- Maximum likelihood minimizes KL between target and $p_\theta(x) = \int p_\theta(x, z)\mathrm{d}z$
- Objective wants a good fit for $p_\theta(x)$; doesn't care about usefulness at all
    - True for *any* objective that only cares about $p_\theta(x)$, not just MLE
- But we don't actually maximize over all latent variable models
- This relies on our model class (or really, learning process. . .) aligning well

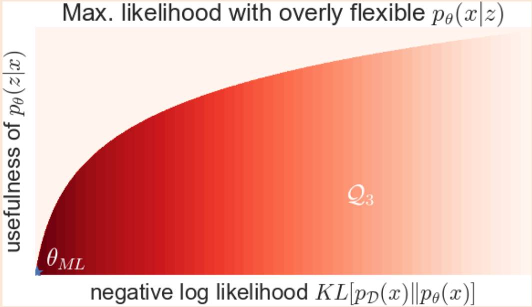


https://www.inference.vc/maximum-likelihood-for-representation-learning-2/

Representation Learning with Latent Variable Models

bonus!

- We'd often like a "useful" $p_\theta(z \mid x)$
- Maximum likelihood minimizes KL between target and $p_\theta(x) = \int p_\theta(x, z)\mathrm{d}z$
- Objective wants a good fit for $p_\theta(x)$; doesn't care about usefulness at all
  - True for *any* objective that only cares about $p_\theta(x)$, not just MLE
- But we don't actually maximize over all latent variable models
- This relies on our model class (or really, learning process...) aligning well
- Real(ish) case: if $p_\theta(x \mid z)$ is too powerful, can ignore z, i.e. useless representation



Representation Learning with VAEs

bonus!

- Maximizing the ELBO isn't *just* MLE...

$$\max_{\phi} \sum_i \mathrm{ELBO}_{\theta,\phi}(x^{(i)}) = \log p_\theta(\mathbf{X}) - \min_{\phi} \sum_i \mathrm{KL}(q_\phi(z^{(i)} \mid x^{(i)}) \parallel p_\theta(z^{(i)} \mid x^{(i)}))$$

  - If ϕ is perfect, it's just the MLE
  - Otherwise, we prefer the kinds of distributions that q_ϕ can successfully reconstruct
- And, to emphasize again, training a VAE isn't just minimizing the ELBO
  - Implicit bias of SGD training procedure likely plays a very important role
  - Likely even more true for complex models, e.g. transformer-based

Outline

Pause

admin

- That's it for "course content" today
- There's a bunch of fun bonus stuff I'd like to go through

- But...the Student Experience of Instruction response rate is
  - Currently 9% for 440, "supposed to be" at least 25%
  - Currently 18% for 550, "supposed to be" at least 65%
- These get used:
  - For me (and administrators) to see anonymously to improve in the future
    - Really is anonymous: I don't see your name, only numeric summaries + each text response to each question (separately, not linked to each other)
    - I only see this well after final grades are submitted
  - For my tenure case
- `seoi.ubc.ca/surveys` or from Canvas
- Teaching evaluations: the good, the bad, and the ugly by Mike Gelbart on `r/UBC`
  - "Think about your biases"; "be specific"; "be kind"

Outline

Normalizing Flows

bonus!

- Based on change-of-variables formula: if $x = f(z)$ for bijective, differentiable f,

$$p(x) = p(z) \left|\det(\nabla_z f^{-1}(z))\right|$$

- Limit layers to be invertible (and square) with easy det; get exact likelihoods
- Some variants: original, Real NVP, MAF, GLOW, FFJORD, Residual Flows

Real Data | Residual Flow

Figure 14: Random samples from 5bit CelebA-HQ 256×256. Most visually appealing batch out of five was chosen.

https://arxiv.org/abs/1906.02735

Autoregressive Models

bonus!

- Use $p(x) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_d \mid x_{1:d-1})$
  - Just a fully-connected DAG model
- Model each $p(x_j \mid x_{1:j-1})$ using some kind of neural net
- Some variants: RNADE, PixelRNN, PixelCNN, WaveNet, MADE
- First models with really good likelihoods and samples for complex datasets
- Very slow: go through an image pixel-by-pixel

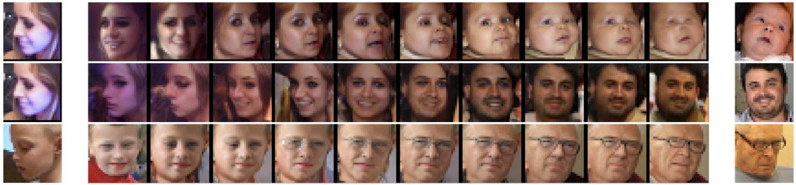


Figure 5: Linear interpolations in the embedding space decoded by the PixelCNN. Embeddings from leftmost and rightmost images are used for endpoints of the interpolation.

https://arxiv.org/abs/1606.05328

- Note: can have interesting behaviour with zero-probability prompts

Energy-Based Models

bonus!

- General term for models like $p_\theta(x) = \frac{1}{Z_\theta} \exp(-\mathcal{E}_\theta(x))$; $\mathcal{E}_\theta$ is "energy"
  - Important example: product of experts $p_1(x) p_2(x)$ has energy $\mathcal{E}_1(x) + \mathcal{E}_2(x)$
- Super-broad category (…essentially any distribution)
- Maximum likelihood: like exponential families, $\nabla_\theta \log \frac{1}{Z_\theta} = \mathbb{E}_{x \sim p_\theta} \nabla_\theta \mathcal{E}_\theta(x)$
  - Can estimate with MCMC sample, e.g. contrastive divergence / Younes algorithm
- Can also fit without estimating Z_θ using score matching, noise-contrastive estimation, Stein discrepancy, adversarial training, …

Score Matching

bonus!

- A way to fit unnormalized generative models
- Hyvärinen score is $s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \tilde{p}_\theta(x) - \underbrace{\nabla_x \log Z_\theta}_{0}$
  - Or we can just learn a function s_θ directly
- Score matching tries to match s_θ to target's Hyvärinen score:

$$\arg\min_\theta \mathop{\mathbb{E}}_{x \sim p_{\text{target}}} \|s_\theta(x) - \nabla_x \log p_{\text{target}}(x)\|^2$$

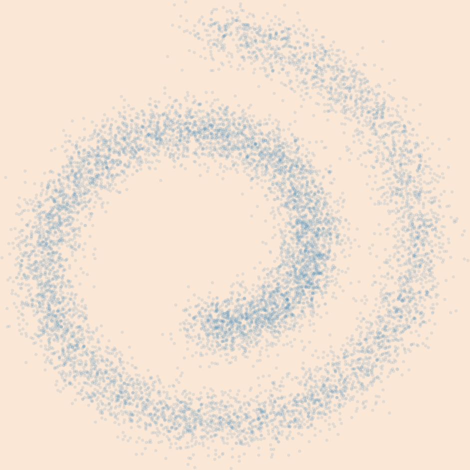
- Under some conditions (using integration by parts), this is equivalent to

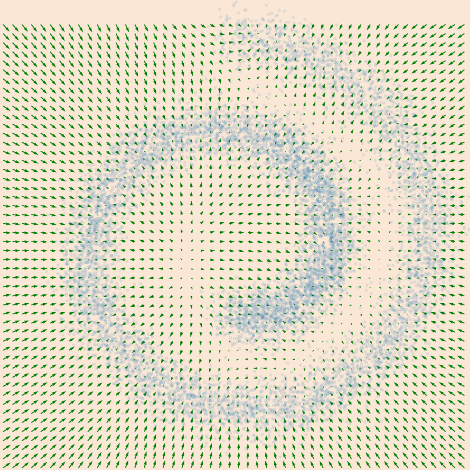
$$\arg\min_\theta \mathop{\mathbb{E}}_{x \sim p_{\text{target}}} \tfrac{1}{2}\|s_\theta(x)\|^2 + \operatorname{Tr}(\nabla_x s_\theta(x))$$

- Denoising score matching, sliced score matching to help with second derivative
- Close connection to contrastive divergence (see PML2 24.3.4)

Score matching a Swiss roll

bonus!





PML2's `score_matching_swiss_roll.ipynb`

Generative Adversarial Networks (GANs)

bonus!

- Generator network $G_\theta(z)$ produces samples based on $p_\theta(z)$
  - Train G_θ to trick a discriminator $D_\phi(x)$ that tries to classify real vs. fake
  - Adversarial game, $\min_\theta \max_\phi$; tricky to optimize
  - Sort of minimizes Jensen-Shannon, $\frac{1}{2}\operatorname{KL}(p_\theta \parallel \frac{p_\theta + p_{\text{target}}}{2}) + \frac{1}{2}\operatorname{KL}(p_{\text{target}} \parallel \frac{p_\theta + p_{\text{target}}}{2})$
    - Variants sort of minimize Wasserstein-1 or other distributional losses
- Not probabilistic – no attempt at computing $\int G_\theta(z) p_\theta(z) \mathrm{d}z$, only sampling

https://arxiv.org/abs/2202.00273

What's the best way to train?

bonus!

- It's not necessarily clear that MLE $= \arg\min_\theta \mathrm{KL}(p_{\text{target}} \parallel p_\theta)$ is best
  - MLE has some nice asymptotic properties, given some (strong!) assumptions
    - Classical results assume there is some θ^* where $p_{\text{target}} = p_{\theta^*}$

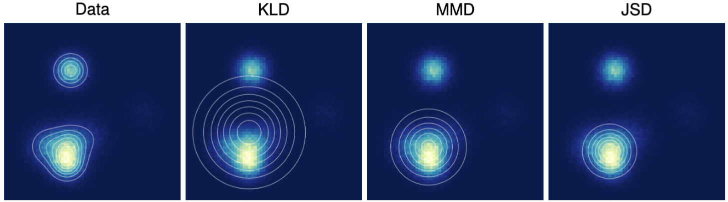


Figure 1: An isotropic Gaussian distribution was fit to data drawn from a mixture of Gaussians by either minimizing Kullback-Leibler divergence (KLD), maximum mean discrepancy (MMD), or Jensen-Shannon divergence (JSD). The different fits demonstrate different tradeoffs made by the three measures of distance between distributions.

https://arxiv.org/abs/1511.01844

- Which one you want depends a lot on what you're using it for

Outline

How do we tell if a generative model is any good anyway?

bonus!

- Held-out log-likelihood would be the usual thing to do for generative models
  - GANs can't do; VAEs under-estimate; energy-based models typically over-estimate
    - (Happens by Jensen's inequality; see this paper, section 3.2, to estimate by how much)
  - Images are usually in $\{0, 1, \dots, 255\}^d$: continuous models can get infinite likelihoods
    - Usually de-quantize by adding uniform noise from $[0, 1)^d$
    - Under-estimates log-likelihood of discrete model with $p_{\text{discrete}}(x) = \int_{[0,1)^d} p_\theta(x + u) \mathrm{d}u$ (Jensen's again; see this paper, section 3.1)
- Connection to sample quality is tenuous in high dimensions
  - Break samples, barely change log-likelihood: $p(x) = 0.001 p_\theta(x) + 0.999$ 💩(x)
    - $\log p(x) \ge \log(0.001 p_\theta) > \underbrace{\log p_\theta(x)}_{\text{scales with } d} - \underbrace{7}_{\text{doesn't}}$
    - On 64×64 ImageNet, PixelCNN beats PixelRNN by 511 nats/img, Conv Draw by 4,514
  - Break log-likelihood, barely change samples: $p = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(\tilde{x}^i, \varepsilon^2 I)$ for $\tilde{x}^i \overset{\text{iid}}{\sim} p_\theta$
    - If N is big and ε tiny, unlikely to see duplicates, but it's a way-overfit KDE

How do we tell if a generative model is any good anyway?

bonus!

How do we tell if a generative model is any good anyway?

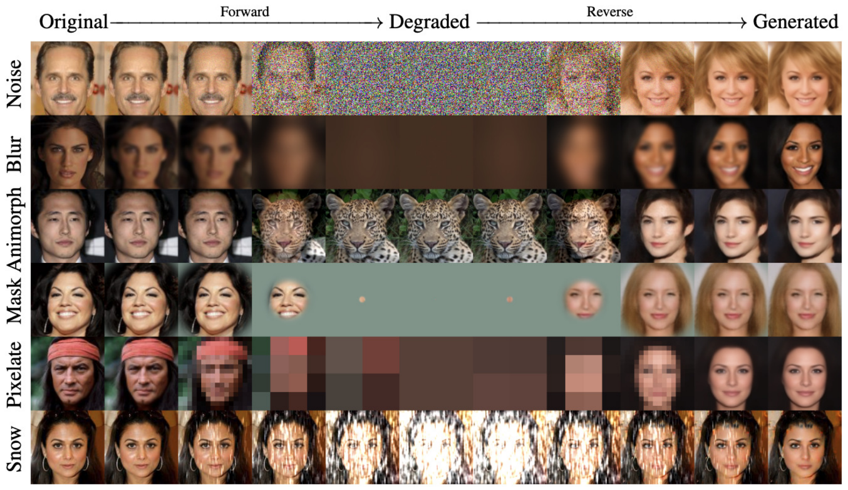
bonus!

- Most common sample evaluation method: Fréchet Inception Distance (FID)
  - Estimate mean, covariance of featurizer pretrained on ImageNet
  - Squared FID: $\|\hat{\mu}_{\text{model}} - \hat{\mu}_{\text{target}}\|^2 + \operatorname{Tr}(\hat{\Sigma}_{\text{model}}) + \operatorname{Tr}(\hat{\Sigma}_{\text{target}}) - 2 \operatorname{Tr}\left((\hat{\Sigma}_{\text{model}} \hat{\Sigma}_{\text{target}})^{\frac{1}{2}} \right)$
  - Motivated as Wasserstein-2 (Fréchet) distance between Gaussians
  - Estimator has low variance but high bias (this paper, section 4 / appendix D)
- Precision/Recall, Density/Coverage metrics
  - Try to disambiguate "all samples look reasonable" versus "covering all the data"
- Classification Accuracy Score
  - Train a classifier on (class-conditional) model samples; see how it does on real data
- All of these have issues with "overfitting" by just reproducing training set

Outline

Diffusion Processes

bonus!



https://arxiv.org/abs/2208.09392

- Non-random ("cold diffusion"): maybe ≈ conditional flow matching

Diffusion Models as Hierarchical VAEs

bonus!

- Start with data point x_0, add noise to get x_1, add noise to get x_2, ...
- Forward process is ($\approx$)fixed; should choose so $q(x_T \mid x_0) \approx p(x_T)$
- Reverse process $p_\theta(x_{t-1} \mid x_t)$ to remove the noise
- Normal ELBO would give us (see (34) to (45) in this note)

$$\log p_\theta(x_0) \geq \overbrace{\mathop{\mathbb{E}}_{q(x_1|x_0)} \log p_\theta(x_0 \mid x_1)}^{\text{reconstruction}} - \overbrace{\mathop{\mathbb{E}}_{q(x_{T-1}|x_0)} \mathrm{KL}(q(x_T \mid x_{T-1}) \,\|\, p(x_T))}^{\text{prior matching; doesn't depend on } \theta}$$
$$- \sum_{t=1}^{T-1} \underbrace{\mathop{\mathbb{E}}_{q(x_{t-1},x_{t+1}|x_0)} \mathrm{KL}(q(x_t \mid x_{t-1}) \,\|\, p_\theta(x_t \mid x_{t+1}))}_{\text{consistency}}$$

Diffusion Models as Hierarchical VAEs

bonus!

- Start with data point x_0, add noise to get x_1, add noise to get x_2, ...
- Forward process is ($\approx$)fixed; should choose so $q(x_T \mid x_0) \approx p(x_T)$
- Reverse process $p_\theta(x_{t-1} \mid x_t)$ to remove the noise
- Nicer ELBO (see (46) to (58) in this note) cancels tons of stuff:

$$\log p_\theta(x_0) \geq \overbrace{\mathop{\mathbb{E}}_{q(x_1|x_0)} \log p_\theta(x_0 \mid x_1)}^{\text{reconstruction}} - \overbrace{\mathrm{KL}(q(x_T \mid x_0) \,\|\, p(x_T))}^{\text{prior matching; no } \theta}$$
$$- \sum_{t=1}^{T-1} \underbrace{\mathop{\mathbb{E}}_{q(x_t|x_0)} \mathrm{KL}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))}_{p_\theta \text{ should match true denoising process}}$$

- Recovers standard VAE ELBO if $T = 1$

Diffusion Models as Hierarchical VAEs

bonus!

$$\arg\max_\theta \mathop{\mathbb{E}}_{q(x_1|x_0)} \log p_\theta(x_0 \mid x_1) - \mathrm{KL}(q(x_T \mid x_0) \,\|\, p(x_T)) - \sum_{t=1}^{T-1} \mathop{\mathbb{E}}_{q(x_t|x_0)} \mathrm{KL}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))$$

- Usual case is fixed normal noise: $q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I)$
  - Implies $q(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{\bar\alpha_t} x_0, (1-\bar\alpha_t) I\right)$ for $\bar\alpha_t = \prod_{\tau=1}^{t}(1-\beta_\tau)$
  - Choose T, β_t such that $\bar\alpha_T \approx 0$, so $q(x_T \mid x_0) \approx \mathcal{N}(0, I)$
  - Get that $q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(x_{t-1}; \gamma_t x_t + \delta_t x_0, \sigma_t^2 I\right)$; $\gamma_t, \delta_t, \sigma_t$ depend only on β_ts
  - We can just choose $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \gamma_t x_t + \delta_t \hat{x}_\theta(x_t, t), \sigma_t^2 I)$!
  - KL, reconstruction terms simplify a lot: get

$$\arg\min_\theta \mathop{\mathbb{E}}_{\substack{x_0 \sim p_{\text{target}} \\ t \sim \mathrm{Unif}\{1,\dots,T\}}} \left[\mathop{\mathbb{E}}_{x_t \sim \mathcal{N}\left(\sqrt{\bar\alpha_t} x_0, (1-\bar\alpha_t) I\right)} \left[\frac{\delta_t^2}{2\sigma_t^2} \begin{cases} \|\hat{x}_\theta(x_1, 1) - x_0 - \gamma_1 x_1\|^2 & \text{if } t = 1 \\ \|\hat{x}_\theta(x_t, t) - x_0\|^2 & \text{otherwise} \end{cases} \right] \right]$$

  - Empirically can choose to ignore weighting δ_t^2 / σ_t^2 and the $t = 1$ special case:

$$\arg\min_\theta \mathop{\mathbb{E}}_{\substack{x_0 \sim p_{\text{target}} \\ t \sim \mathrm{Unif}\{1,\dots,T\}}} \left[\mathop{\mathbb{E}}_{x_t \sim \mathcal{N}\left(\sqrt{\bar\alpha_t} x_0, (1-\bar\alpha_t) I\right)} \left[\|\hat{x}_\theta(x_t, t) - x_0\|^2 \right] \right]$$

Other views of Diffusion Models

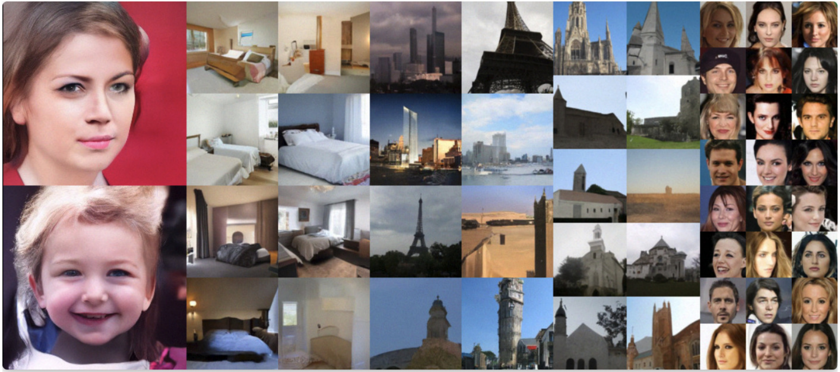
bonus!

- Can view essentially same objective as denoising score matching
- Or as stacked denoising auto-encoders

- Helpful descriptions by: Yang Song, Lilian Weng, Calvin Luo, and PML2 25

"Plain" Diffusion Samples

bonus!



Samples from the NCSNv2 [18] model. From left to right: FFHQ 256x256, LSUN bedroom 128x128, LSUN tower 128x128, LSUN church_outdoor 96x96, and CelebA 64x64.

https://yang-song.net/blog/2021/score/

Infinitely many noise levels

bonus!

- Can take the $T = \infty$ limit based on stochastic differential equations
  - See Yang Song's blog post
- Gives exact log-likelihoods and better ability to condition

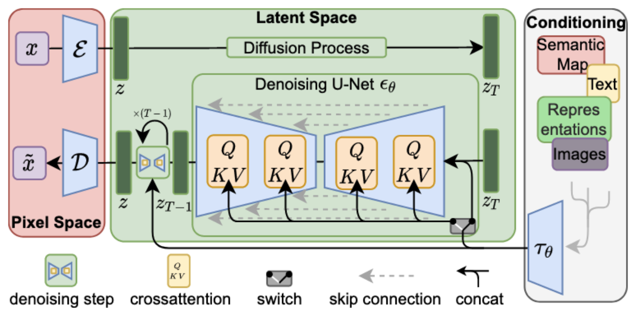
Image inpainting with a time-dependent score-based model trained on LSUN bedroom. The leftmost column is ground-truth. The second column shows masked images (y in our framework). The rest columns show different inpainted images, generated by solving the conditional reverse-time SDE.

https://yang-song.net/blog/2021/score/

Stable Diffusion

bonus!

- Train a fancy, high-quality auto-encoder
- Run diffusion model on the code distribution
- Condition the decoder on text embeddings



https://arxiv.org/abs/2112.10752

ControlNet

bonus!

- Allows "post-processing" to add new kinds of conditioning to pretrained model

bonus!

ARTIFICIAL INTELLIGENCE / TECH / LAW

Getty Images is suing the creators of AI art tool Stable Diffusion for scraping its content

An image created by Stable Diffusion showing a recreation of Getty Images' watermark. Image: The Verge / Stable Diffusion

/ Getty Images claims Stability AI 'unlawfully' scraped millions of images from its site. It's a significant escalation in the developing legal battles between generative AI firms and content creators.

By JAMES VINCENT

Jan 17, 2023, 2:30 AM PST | 18 Comments / 18 New bonus!

Figure 1: Diffusion models memorize individual training examples and generate them at test time. **Left:** an image from Stable Diffusion's training set (licensed CC BY-SA 3.0, see [49]). **Right:** a Stable Diffusion generation when prompted with "Ann Graham Lotz". The reconstruction is nearly identical (ℓ_2 distance = 0.031).

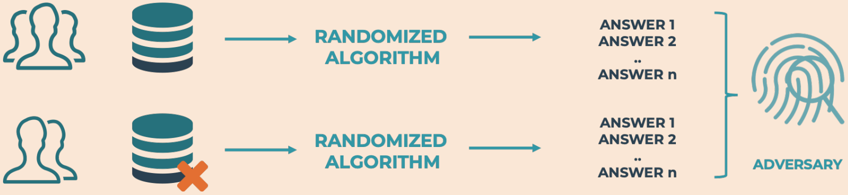
https://arxiv.org/abs/2301.13188

Outline

Privacy

bonus!

- How can we prevent models from memorizing individual data points?
- Leading framework is differential privacy



https://2021.ai/machine-learning-differential-privacy-overview/

- CPSC grad courses: 532P by Mijung Park, sometimes 538L by Mathias Lecuyer

Fairness, Accountability, Transparency

bonus!

- Tons of issues around ML models / applications
- Some have technical (partial) solutions
- Some can only be handled socially
- "Sociotechnical systems" (STS)

- FAccT and AIES conferences
- DSCI 430, focuses mostly on fairness

Causality

bonus!

- 532Y: Causal ML by Mathias Lecuyer
- Math 605D by Elina Robeva (sometimes)

- Closely related to fairness
- More related things to be aware of:
    - Disentanglement
    - Independent components analysis
    - Out-of-distribution generalization, domain adaptation

More Deep Learning: NLP

bonus!

- Big, super-fast thing is large language models
  - We May be Surprised Again: Why I take LLMs seriously

- CPSC 436N: NLP
- CPSC 532V: Commonsense Reasoning in NLP by Vered Shwartz
- 532G (dialogue models) by Giuseppe Carenini
- courses by Muhammad Abdul-Mageed

- 532S: Multimodal Learning with Vision, Language and Sound

More Deep Learning: Vision/Graphics

bonus!

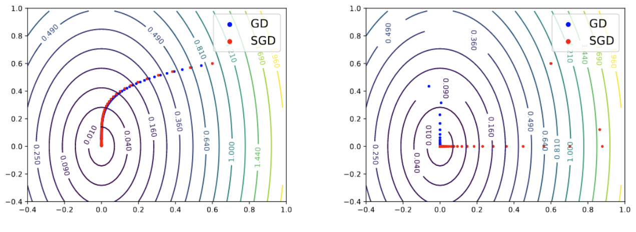
- Lots of vision to do beyond what was in this course!

- CPSC 425: Computer Vision
- 533Y: Visual Geometry with Deep Learning by Kwang Moo Yi
- 533V: Learning to Move by Michiel van de Panne
- Probably a course by Evan Shelhamer

Theory

bonus!

- Why/when do ML models / optimizers work, mathematically?



(a) Small learning rate regime (b) Moderate learning rate regime

https://arxiv.org/abs/2011.02538

- 532D: Statistical Learning Theory by me
- Optimization: 406 and 536M by Michael Friedlander
- Optimization in ML: 5XX by Mark Schmidt (not this year)
- EECE 571Z Convex Optimization by Christos Thrampoulidis
- Various stat courses

Probabilistic/Bayesian/... ML

bonus!

- Probabilistic programming: 532W by Frank Wood
- Stat 520A: Bayesian analysis by Alexandre Bouchard-Côté
- Stat 520B: Variational Bayes by Trevor Campbell
- Stat 547S: Topics on Symmetry by Benjamin Bloem-Reddy
- Stat 520P: Bayesian Optimization by Geoff Pleiss
- ECE 571F: Deep Learning with Structures by Renjie Liao
- Various more stat courses

- Some more things to be aware of:
  - Mutual information/dependence estimation
  - Graph neural networks, deep sets, other structured data
  - Particle filters
  - Bayesian neural networks

Reinforcement learning

bonus!

- 322, 422 – logic, more graphical models, search, planning, some RL
- 522 by David Poole (PGMs, some RL)

- 532J: Never Ending Reinforcement Learning by Jeff Clune
- 533V: Learning to Move by Michiel van de Panne (planned W2)

- Some more things to be aware of:
  - Meta-learning
  - Online learning
  - Active learning
  - Multi-armed bandits
  - Auto-ML

Other stuff

bonus!

- 532C: Human-Centred AI by Cristina Conati (planned W2)
- Somewhat relevant: 539L: Automated Testing by Caroline Lemieux
- 532L: Modes of Strategic Behaviour by Kevin Leyton-Brown
- 545: Algorithms for Bioinformatics by Jiarui Ding
- Math 605D: Tensor decompositions by Elina Robeva (sometimes)
- Math 555: Compressed Sensing by Yaniv Plan

- Possible courses by
  - Kelsey Allen (new in CS+Psych; cognitive science / robotics / ML)
  - Xiaxio Li (ECE; federated learning)
  - Lele Wang (ECE; coding theory)

- Reading groups: `https://ml.ubc.ca/reading-groups/`
- Talks: CAIDA (AI broadly), MILD ("mathematical" ML)

Summary

- Variational methods approximate p with a simpler distribution q
  - Usually minimize reverse KL divergence
    - Because it's (often) easy to evaluate for simple q, not for any fundamental reason
- Variational auto-encoders (VAEs) do this for a "deep latent variable model"
  - $p(x) = \int p(z)p(x \mid z)\mathrm{d}z$
  - Learn a "recognition network" $q(z \mid x)$ to reconstruct z for any given x
  - Minimize the ELBO: lower bound on the likelihood
- Bunch of stuff on other image generative models, but all bonus content

- Next lecture: nothing! there is no next lecture! Bye :)

Maximum likelihood minimizes KL

bonus!

$$
\begin{aligned}
\arg\min_\theta \mathrm{KL}(p_{\mathsf{true}} \parallel p_\theta) &= \arg\min_\theta \int p_{\mathsf{true}}(x) \log \frac{p_{\mathsf{true}}(x)}{p_\theta(x)} \mathrm{d}x \\
&= \arg\min_\theta \underbrace{\int p_{\mathsf{true}}(x) \log p_{\mathsf{true}}(x) \mathrm{d}x}_{\text{doesn't depend on } \theta} - \int p_{\mathsf{true}}(x) \log p_\theta(x) \mathrm{d}x \\
&= \arg\min_\theta - \int p_{\mathsf{true}}(x) \log p_\theta(x) \mathrm{d}x \\
&= \arg\max_\theta \mathop{\mathbb{E}}_{x \sim p_{\mathsf{true}}} \log p_\theta(x) \\
&\approx \arg\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x^{(i)})
\end{aligned}
$$

Three Coordinate-Wise Algorithms

bonus!

- Gibbs sampling is a coordinate-wise method for approximate sampling:
  - Choose a coordinate j to update
  - Sample x_j keeping other variables fixed

- ICM is a coordinate-wise method for approximate decoding:
  - Iterated Conditional Mode; it's in last lecture's bonus slides
  - Choose a coordinate j to update
  - Maximize x_j keeping other variables fixed

- Mean field is a coordinate-wise method for approximate marginalization:
  - Choose a coordinate j to update
  - Update marginal $\underbrace{q_j(x_j)}_{\text{for all } x_j}$ keeping other variables fixed ($q_j(x_j)$ approximates $p_j(x_j)$)

Three Coordinate-Wise Algorithms

bonus!

- Consider a pairwise discrete UGM:

$$p(x_1, x_2, \dots, x_d) \propto \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right),$$

- ICM for updating a node j with 2 neighbours (i and k)
  1. Compute $M_j(x_j) = \phi_j(x_j)\phi_{ij}(x_i, x_j)\phi_{jk}(x_j, x_k)$ for all x_j
  2. Set x_j to the largest value of $M_j(x_j)$

- Gibbs for updating a node j with 2 neighbours (i and k)
  1. Compute $M_j(x_j) = \phi_j(x_j)\phi_{ij}(x_i, x_j)\phi_{jk}(x_j, x_k)$ for all x_j
  2. Sample x_j proportional to $M_j(x_j)$

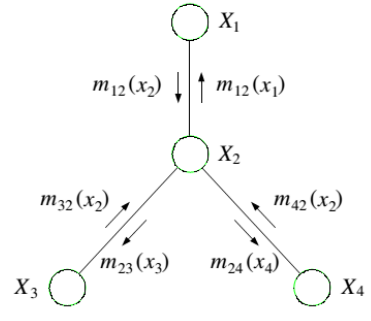
- Mean field for updating a node j with 2 neighbours (i and k)
  1. Compute $M_j(x_j) = \phi_j(x_j)\exp\left(\sum_{x_i} q_j(x_i)\log\phi_{ij}(x_i, x_j) + \sum_{x_k} q_k(x_k)\log\phi_{jk}(x_j, x_k)\right)$
  2. Set $q_j(x_j)$ proportional to $M_j(x_j)$

Previously: Belief Propagation

bonus!

- Generalization of forward-backward to forests is belief propagation.

(undirected graphs with no loops, which must be pairwise)



https://www.quora.com/

Probabilistic-graphical-models-what-are-the-relationships-between-sum-product-algorithm-belief-propagation-and-junction-tree-

- Defines "messages" that can be sent along each edge.

Loopy Belief Propagation

bonus!

- In pairwise UGM, belief propagation "message" from parent p to child c is gven by

$$M_{pc}(x_c) \propto \sum_{x_p} \phi_i(x_p)\phi_{pc}(x_p, x_c)M_{jp}(x_p)M_{kp}(x_p),$$

  assuming that parent p has parents j and k.
  - We get marginals by multiplying all incoming messages with local potentials.

- Loopy belief propagation: a "hacker" approach to approximate marginals:
  - Choose an edge ic to update.
  - Update messages $M_{ic}(x_c)$ keeping all other messages fixed.
  - Repeat until "convergence".
    - We approximate marginals by multiplying all incoming messages with local potentials.

- Empirically much better than mean field; we've spent 20+ years figuring out why.

Discussion of Loopy Belief Propagation

bonus!

- Loopy BP decoding is used for "error correction" in 3G/4G, NASA missions. . . .
  - Called "turbo codes" in information theory.
- Loopy BP is not optimizing an objective function.
  - Convergence of loopy BP is hard to characterize: does not converge in general.
- If it converges, loopy BP finds fixed point of "Bethe free energy":
  - Instead of "Gibbs mean-field free-energy" for mean field, which lower bounds Z.
  - Bethe typically gives better approximation than mean field, but not a bound.
- There are convex variants that upper bound Z.
  - Tree-reweighted belief propagation.
  - Variations that are guaranteed to converge.
    - Convex variants are more consistent but often give worse approximations.
- Messages only have closed-form update for conjugate models.
  - Can approximate non-conjugate models using expectation propagation.

Convex Relaxations

bonus!

- We've overviewed a view of variational methods as minimizing non-convex reverse KL.
- Alternate view: write exact inference as constrained convex optimization.
  - Writing inference as maximizing entropy with constraints on marginals.
    - See bonus slides from the exponential family lecture.
  - Different methods correspond to different entropy/constraint approximations.
    - Mean field and loopy belief propagation relax entropy and marginals in different ways.
    - Weirdly, these approximations are non-convex even though original problem is convex.
  - There are also convex relaxations that approximate with linear programs (or SDPs).
- For an overview of these ideas, see:
  `https://people.eecs.berkeley.edu/~wainwrig/Papers/WaiJor08_FTML.pdf`

Difficulty of Variational Formulation

bonus!

- In exponential family bonus slides, we write inference as a convex optimization:

$$\log(Z) = \sup_{\mu \in \mathcal{M}} \{w^T \mu + H(p_\mu)\},$$

- Did this make anything easier?
  - Computing entropy $H(p_\mu)$ seems as hard as inference.
  - Characterizing marginal polytope $\mathcal{M}$ becomes hard with loops.
- Practical variational methods:
  - Work with approximation/bound on entropy H.
  - Work with approximation to marginal polytope $\mathcal{M}$.

Mean Field Approximation

bonus!

- Mean field approximation assumes

$$\mu_{ij,st} = \mu_{i,s}\mu_{j,t},$$

  for all edges, which means

$$p(x_i = s, x_j = t) = p(x_i = s)p(x_j = t),$$

  and that variables are independent.
- Entropy is simple under mean field approximation:

$$\sum_X p(X) \log p(X) = \sum_i \sum_{x_i} p(x_i) \log p(x_i).$$

- Marginal polytope is also simple:

$$\mathcal{M}_F = \{\mu \mid \mu_{i,s} \geq 0,\ \sum_s \mu_{i,s} = 1,\ \mu_{ij,st} = \mu_{i,s}\mu_{j,t}\}.$$

Entropy of Mean Field Approximation

bonus!

- Entropy form is from distributive law and probabilities sum to 1:

$$
\begin{aligned}
\sum_X p(X)\log p(X) &= \sum_X p(X)\log(\prod_i p(x_i)) \\
&= \sum_X p(X)\sum_i \log(p(x_i)) \\
&= \sum_i \sum_X p(X)\log p(x_i) \\
&= \sum_i \sum_X \prod_j p(x_j)\log p(x_i) \\
&= \sum_i \sum_X p(x_i)\log p(x_i)\prod_{j\neq i} p(x_j) \\
&= \sum_i \sum_{x_i} p(x_i)\log p(x_i)\sum_{x_j \mid j\neq i}\prod_{j\neq i} p(x_j) \\
&= \sum_i \sum_{x_i} p(x_i)\log p(x_i).
\end{aligned}
$$

Mean Field as Non-Convex Lower Bound

bonus!

- Since $\mathcal{M}_F \subseteq \mathcal{M}$, yields a lower bound on $\log(Z)$:

$$\sup_{\mu \in \mathcal{M}_F} \{w^T\mu + H(p_\mu)\} \leq \sup_{\mu \in \mathcal{M}} \{w^T\mu + H(p_\mu)\} = \log(Z).$$

- Since $\mathcal{M}_F \subseteq \mathcal{M}$, it is an inner approximation:

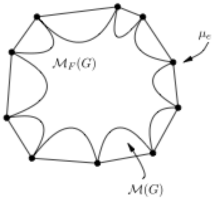


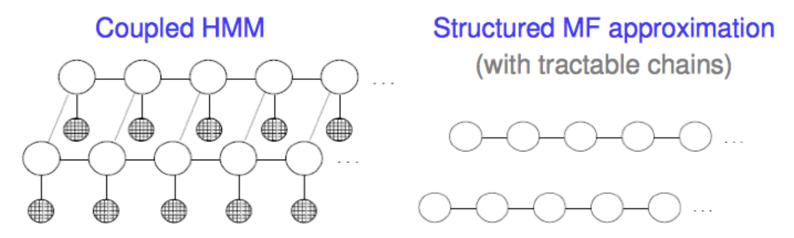
Fig. 5.3 Cartoon illustration of the set $\mathcal{M}_F(G)$ of mean parameters that arise from tractable distributions is a nonconvex inner bound on $\mathcal{M}(G)$. Illustrated here is the case of discrete random variables where $\mathcal{M}(G)$ is a polytope. The circles correspond to mean parameters that arise from delta distributions, and belong to both $\mathcal{M}(G)$ and $\mathcal{M}_F(G)$.

- Constraints $\mu_{ij,st} = \mu_{i,s}\mu_{j,t}$ make it non-convex.
- Mean field algorithm is coordinate descent on $w^T\mu + H(p_\mu)$ over $\mathcal{M}_F$.

Discussion of Mean Field and Structured MF

bonus!

- Mean field is weird:
  - Non-convex approximation to a convex problem.
  - For learning, we want upper bounds on $\log(Z)$.
- Structured mean field:
  - Cost of computing entropy is similar to cost of inference.
  - Use a subgraph where we can perform exact inference.

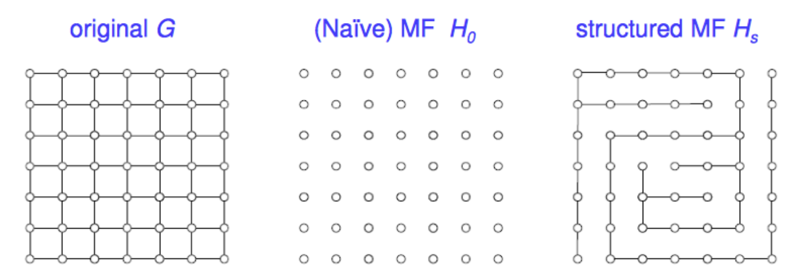


http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf

Structured Mean Field with Tree

bonus!

- More edges means better approximation of $\mathcal{M}$ and $H(p_\mu)$:



http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf

- Fixed points of loopy correspond to using "Bethe" approximation of entropy and "local polytope" approximation of "marginal polytope".

- You can design better variational methods by constructing better approximations.

Computing the ELBO and its gradient: KL term

bonus!

- We want to maximize the average of

$$\mathrm{ELBO}_{\theta,\phi}(x) = \mathop{\mathbb{E}}_{z \sim q_\phi(z \mid x)} [\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \parallel p(z))$$

- KL term for a given x is often available in closed form

- Typically we choose $p_\theta(z)$ to be $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $q_\phi(z \mid x)$ to be $\mathcal{N}(\boldsymbol{\mu}_\phi(x), \boldsymbol{\Sigma}_\phi(x))$
- Then the KL is just (see PML2 eq 5.80)

$$\frac{1}{2}\left(\|\boldsymbol{\mu}_\phi(x)\|^2 + \mathrm{Tr}\, \boldsymbol{\Sigma}_\phi(x) - \log|\boldsymbol{\Sigma}_\phi(x)| - d\right)$$

- Most of the time we also choose $\boldsymbol{\Sigma}_\phi(x)$ to be diagonal; determinant is easy
- This is just an expression in terms of ϕ; we can use autodiff

β-VAE

bonus!

- Put a weight $\beta > 1$ in front of the KL term in the ELBO

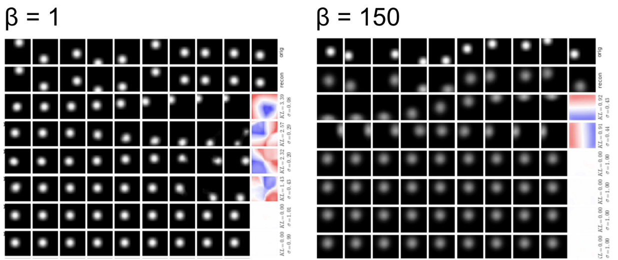


Figure 2: **Entangled versus disentangled representations of positional factors of variation learnt by a standard VAE ($\beta = 1$) and β-VAE ($\beta = 150$) respectively.** The dataset consists of Gaussian blobs presented in various locations on a black canvas. Top row: original images. Second row: the corresponding reconstructions. Remaining rows: latent traversals ordered by their average KL divergence with the prior (high to low). To generate the traversals, we initialise the latent representation by inferring it from a seed image (left data sample), then traverse a single latent dimension (in $[-3, 3]$), whilst holding the remaining latent dimensions fixed, and plot the resulting reconstruction. Heatmaps show the 2D position tuning of each latent unit, corresponding to the inferred mean values for each latent for given each possible 2D location of the blob (with peak blue, -3; white, 0; peak red, 3).

https://arxiv.org/pdf/1804.03599.pdf

- Refined version: see TC-VAE

Wasserstein Auto-Encoder

bonus!

- Different framing for an auto-encoder-based generative model
- Avoids "motivation" for posterior collapse
- Simple version with deterministic encoder/decoder:

$$\min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} \|x^i - \mathrm{dec}_\theta(\mathrm{enc}_\phi(x^i))\|^2 + \lambda D\left(\mathsf{prior}(z), \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left(z = \mathrm{enc}_\phi(x^i)\right)\right)$$

  where D is some distance between probability distributions (kernel MMD, GAN)
- Only makes marginal distribution of zs match the prior, not each one like VAEs
- Can show approximately minimizes Wasserstein distance between model and data