Variational inference and VAEs

CPSC 440/550: Advanced Machine Learning

`cs.ubc.ca/~dsuth/440/24w2`

University of British Columbia, on unceded Musqueam land

2024-25 Winter Term 2 (Jan–Apr 2025)

Last time: approximate inference

- Bayesian inference requires computing expectations with respect to posterior,

$$\mathbb{E}[f(\theta)] = \int_\theta f(\theta)\, p(\theta \mid x) \mathrm{d}\theta$$

  - If $f(\theta) = \theta$, we get posterior mean of θ
  - If $f(\theta) = p(\tilde{x} \mid \theta)$, we get posterior predictive
  - If $f(\theta) = \mathbb{1}(\theta \in S)$ we get probability of S (e.g., marginals)

- But posterior often doesn't have a closed-form expression
  - Bayesian linear regression – $w \sim \mathcal{N}(m, V)$; $y \mid x, w \sim \mathcal{N}(w^\mathsf{T} x, \sigma^2)$ – does
  - Bayesian logistic regression – $p(y \mid x, w) = 1/(1 + \exp(-y\, w^\mathsf{T} x))$ – doesn't
  - More complex models almost never do

- Our two main tools for approximate inference:
  1. Monte Carlo methods
  2. Variational methods

Approximate Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
   - Approximate expectations based on samples,

$$\mathop{\mathbb{E}}_{X \sim p} f(X) \approx \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)})$$

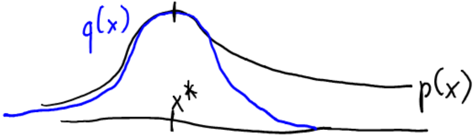
   - Turns inference into sampling
   - Simple Monte Carlo: exactly as above, if we can take iid samples from p
   - Rejection sampling: get p samples from q samples and $M \geq \max_x \tilde{p}(x)/q(x)$
   - Importance sampling: estimate p expectations based on reweighting q samples
   - Markov chain Monte Carlo: a little later in the course
2. Variational methods:
   - Approximate p with "closest" distribution q from a tractable family,

$$\mathop{\mathbb{E}}_{X \sim p} f(X) \approx \mathop{\mathbb{E}}_{X \sim q} f(X)$$

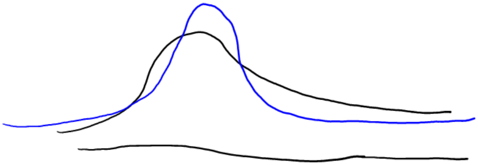
   - q could be Gaussian, product of Bernoulli, any other model with easy inference...
   - Turns inference into optimization

Variational Inference Illustration

- Example: approximate a non-Gaussian p by a Gaussian q
  - Theoretical justification that most posteriors are "eventually" Gaussian
- Laplace approximation: find the mode x^*, then
  match first two derivatives of log-likelihood at the mode: $\mathcal{N}\left(x^*, [\nabla^2 \log p(x^*)]^{-1}\right)$
  - Still works with an unnormalized $\tilde{p}(x)/Z = p(x)$:
    $\log p(x) = \log \tilde{p}(x) - \log Z$ has same mode and derivatives as $\log p$



- Is this the "best" Gaussian approximation? What if we want non-Gaussian approx?



Kullback-Leibler (KL) Divergence

- We'd like to find the "closest" q to our target p
- How do we define "closeness" between a distribution p and q?

- A common measure is Kullback-Leibler (KL) divergence between p and q:

$$\mathrm{KL}(p \parallel q) = \mathop{\mathbb{E}}_{X \sim p} \log \frac{p(X)}{q(X)}$$

- 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, $\mathrm{KL}(p \parallel q) \neq \mathrm{KL}(q \parallel p)$

- Maximizing likelihood = minimizing $\mathrm{KL}(p_{\text{true}} \parallel p_\theta)$ (bonus slide)
- Unfortunately, computing this requires integrating over or sampling from p
    - …exactly the problem we're trying to avoid

Minimizing Reverse KL Divergence

- Most variational methods minimize "reverse KL", as showed up with the ELBO:

$$\mathrm{KL}(q \,\|\, p) = \mathop{\mathbb{E}}_{X \sim q} \log \frac{q(X)}{p(X)} = \mathop{\mathbb{E}}_{X \sim q} \log \left(\frac{q(x)}{\tilde{p}(x)} Z \right)$$

  - Not very intuitive: "how much information is lost when we approximate q by p"
  - Does give some guarantee on approximating bounded functions (bonus)
- "Reverse" KL only needs unnormalized distribution $\tilde{p}/Z = p$ and expectations in q

$$\mathrm{KL}(q \,\|\, p) = \mathop{\mathbb{E}}_{X \sim q}[\log q(X)] - \mathop{\mathbb{E}}_{X \sim q}[\log \tilde{p}(X)] + \underbrace{\mathop{\mathbb{E}}_{X \sim q}[\log(Z)]}_{\text{const. in } q}$$

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

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

Example: Best Multivariate Gaussian

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

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

- How to optimize this? Can't autodiff through expectation...
- Reparamaterization trick: take relevant variable out of the expectation
  - Use Leibniz rule $\frac{\partial}{\partial a}\mathbb{E}_{x\sim p} f(a,x) = \mathbb{E}_{x\sim p}\frac{\partial}{\partial a} f(a,x)$ when p doesn't depend on a
  - Change variables to $q = \mathcal{N}(\boldsymbol{\mu}, \mathbf{L}\mathbf{L}^\mathsf{T})$; use $\left|\mathbf{L}\mathbf{L}^\mathsf{T}\right| = |\mathbf{L}||\mathbf{L}^\mathsf{T}| = |\mathbf{L}|^2$
  - If L is lower-triangular with $L_{jj} > 0$ (Cholesky factor), then $|L| = \prod_j L_{jj}$ is easy
- Can take samples for z and run SGD to optimize (but note it's non-convex)

Reparameterization trick

- Another view on why we can't autodiff through the expectation:

$$\nabla_\theta \mathop{\mathbb{E}}_{x \sim p_\theta} f(x) = \nabla_\theta \int f(x) p_\theta(x) \mathrm{d}x = \int f(x) \nabla_\theta p_\theta(x) \mathrm{d}x$$

  and what do we do with that? (well, see bonus slide)
- But if we write $x = g(\varepsilon, \theta)$ for $\varepsilon \sim r$ (standard normal, uniform, ...),

$$\begin{aligned}\nabla_\theta \mathop{\mathbb{E}}_{x \sim p_\theta} f(x) &= \nabla_\theta \mathop{\mathbb{E}}_{\varepsilon} f(g(\varepsilon, \theta)) = \nabla_\theta \int f(g(\varepsilon, \theta)) r(\varepsilon) \mathrm{d}\varepsilon \\ &= \int \nabla_\theta \left[f(g(\varepsilon, \theta)) \right] r(\varepsilon) \mathrm{d}\varepsilon = \mathop{\mathbb{E}}_{\varepsilon} \left[\nabla_\theta f(g(\varepsilon, \theta)) \right]\end{aligned}$$

  which is autodiff-friendly if we take Monte Carlo samples for ε
- Need g to be differentiable (i.e. x should be continuous)
  - Tricks to avoid this; "Gumbel-Softmax" = "Concrete" distribution

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

Outline

Deep latent variable model

- So far, we've built generative models out of relatively simple parts
  - Gaussian mixture is $Z \sim \mathrm{Cat}(\pi)$, $X \mid (Z = z) \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$
  - Can maximize $p(x) = \sum_{z=1}^{k} \pi_z \mathcal{N}(x; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ with EM/GD/...
- Discriminative models allow using arbitrary functions (e.g. deep nets) inside

$$Y \mid (X = x) \sim \mathcal{N}(g_\theta(z), \sigma^2)$$

  - Can maximize $p(y \mid x) = \mathcal{N}(x; f_\theta(x), \sigma^2)$ with GD/...

- Can we get a generative model with an arbitrary (deep) function in it?
- An example deep latent variable model (X is d-dimensional, Z is k-dimensional):

$$Z \sim \mathcal{N}(\mathbf{0}_k, \mathbf{I}_k) \qquad X \mid (Z = z) \sim \mathcal{N}(g_\theta(z), \sigma^2 \mathbf{I}_d)$$

  - Want to maximize $p(x) = \displaystyle\int \mathcal{N}(z; \mathbf{0}_k, \mathbf{I}_k) \mathcal{N}(x; g_\theta(z), \sigma^2 \mathbf{I}) \mathrm{d}z$
  - How?

Deep latent variable model

- We'd like to do MLE/similar for

$$Z \sim \mathcal{N}(\mathbf{0}_k, \mathbf{I}_k) \qquad X \mid (Z = z) \sim \mathcal{N}(g_\theta(z), \sigma^2 \mathbf{I}_d)$$

  which is

$$\max_\theta \sum_i \log \mathop{\mathbb{E}}_{z^{(i)} \sim \mathcal{N}(\mathbf{0}_k, \mathbf{I}_k)} \left[\mathcal{N}(x^{(i)}; g_\theta(z^{(i)}), \sigma^2 \mathbf{I}) \right]$$

- Could potentially approximate this integral with Monte Carlo + reparam trick:

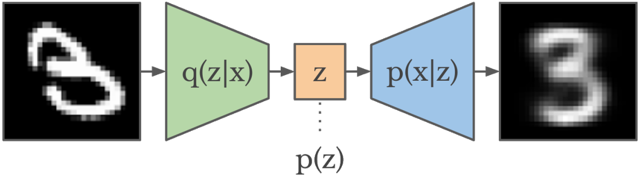
$$\max_\theta \sum_i \log \left(\frac{1}{M} \sum_{j=1}^{M} \mathcal{N}(x^{(i)}; g_\theta(z^{(i,j)}), \sigma^2 \mathbf{I}) \right) \qquad \text{for } z^{(i,j)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

  - But this converges really slowly when k is large / g_θ is complicated: need a huge M

Amortized inference

- EM would alternate between
  - Expectation: $q(Z \mid X, \Theta) = p(Z \mid X, \Theta)$, compute $\mathbb{E}_{Z\sim q} \log p(X, Z \mid \Theta)$
  - Maximization of $\mathbb{E}_{Z\sim q} \log p(X, Z \mid \Theta)$ in Θ
- The E step (inferring Z given X) is hard here!
  - (So is the M step, in the normal deep network way)

- Have some complicated function from X to Z
- Idea: instead of exactly solving the inference problem, let's approximate it
- ...with a neural network $q_\phi(z \mid x)$



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

  - Named variational autoencoder (VAE):
    encode image into latent code z, decode back to approximation of original image

ELBO

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

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

- Since $\mathrm{KL} \geq 0$, $\mathrm{ELBO}_{\theta,\phi}(x) = \log p_\theta(x) - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)) \leq \log p_\theta(x)$
  - ELBO is the Evidence Lower BOund
  - Same as we used in EM, except that we've separated it per-sample here

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

- To efficiently evaluate the ELBO here:

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

- 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

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

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

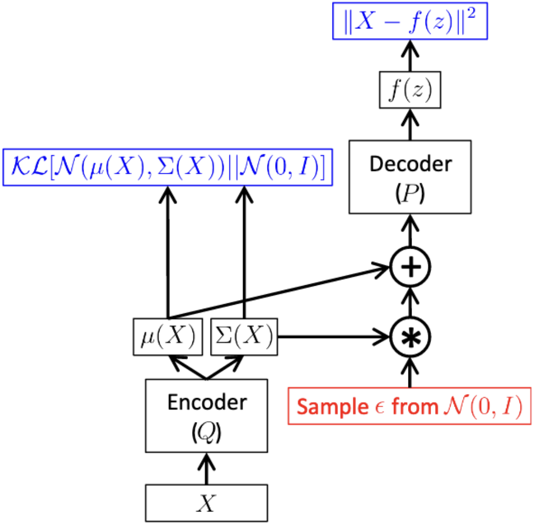
  (if $p(z)$ is $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $q_\phi(z \mid x)$ is $\mathcal{N}(\boldsymbol{\mu}_\phi(x), \boldsymbol{\Sigma}_\phi(x))$; regularizes $\|\boldsymbol{\mu}_\phi(x)\|^2$ and $\boldsymbol{\Sigma}_\phi(x)$ to be near $\mathbf{I}$ – bonus)

- 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 $\log p_\theta(x \mid z) = -\frac{1}{2\sigma^2}\|x - f_\theta(z)\|^2 + \text{const}$
- We need $\mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x \mid z)$
  - Estimate with Monte Carlo
  - Can usually use just a single step for simplicity: if q_ϕ is "good," should be okay
- But how do we take ∇_ϕ of this expectation? Use reparameterization trick again:

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

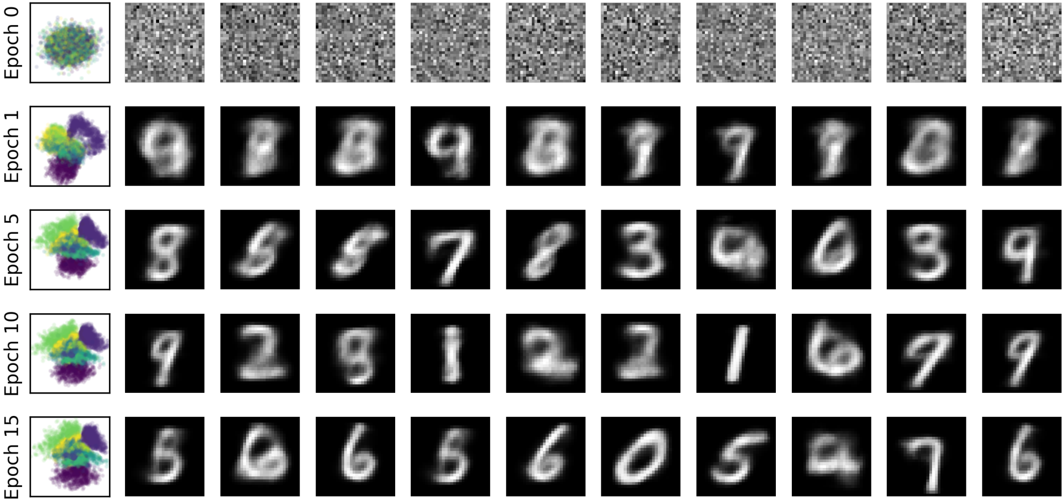
- Take a Monte Carlo sample for ϵ; now have something we can autodiff
- Now just do SGD to maximize $\frac{1}{n}\sum_{i=1}^n \widehat{\mathrm{ELBO}}_{\theta,\phi}(x^{(i)})$

A VAE



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

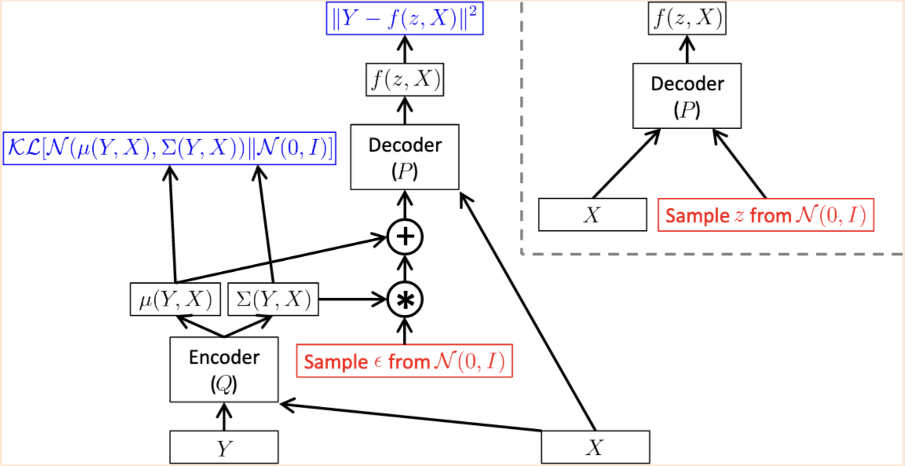
A VAE on MNIST



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

Conditional VAE

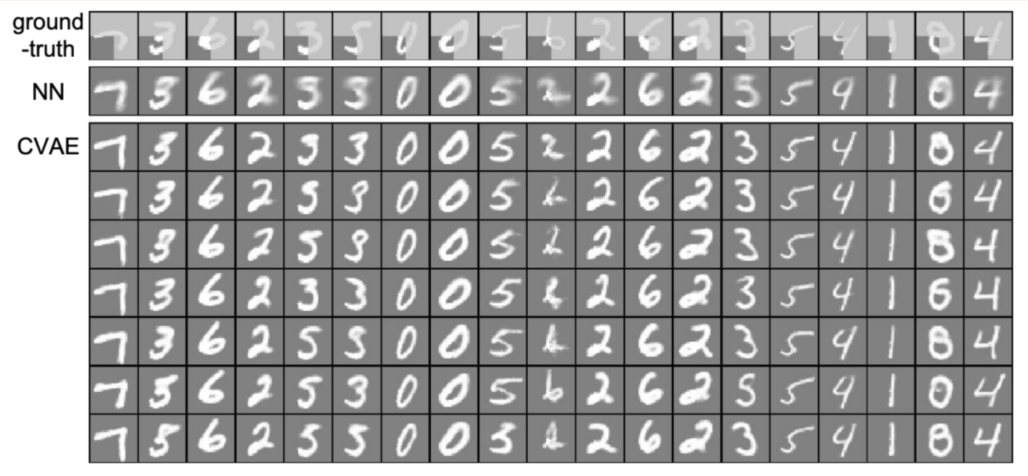
bonus!



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

Summary

- Variational inference: choose $q \approx p$ and estimate $\mathbb{E}_{X \sim p} f(X) \approx \mathbb{E}_{X \sim q} f(X)$
  - Reparameterization trick often helps find best q with SGD
  - Not consistent (q doesn't converge to p if we run the algorithm forever)
  - Can be complicated and hard to parallelize
  - Only needs unnormalized density
  - Often better approximation for a given amount of computation than MCMC
- Variational auto-encoders (VAEs) are a deep latent variable model
  - $p(x) = \int p(z)p(x \mid z)\mathrm{d}z$
  - Learn an "inference" / "recognition" network $q(z \mid x)$ to approximate inference
  - Minimizing the ELBO maximizes a lower bound on the likelihood
- Next up: how to design a VAE and how to use it

Maximum likelihood minimizes KL

bonus!

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

REINFORCE estimator

bonus!

- Alternative gradient estimator to the reparameterization trick:

$$\nabla_\theta \mathop{\mathbb{E}}_{x \sim p_\theta} f(x) = \nabla_\theta \int f(x) p_\theta(x) \mathrm{d}x = \int f(x) \nabla_\theta p_\theta(x) \mathrm{d}x$$

- Notice that $\nabla_\theta \log p_\theta(x) = \frac{1}{p_\theta(x)} \nabla_\theta p_\theta(x)$, so

$$\nabla_\theta \mathop{\mathbb{E}}_{x \sim p_\theta} f(x) = \int f(x) \nabla_\theta \log p_\theta(x) p_\theta(x) \mathrm{d}x = \mathop{\mathbb{E}}_{x \sim p_\theta} [f(x) \nabla_\theta \log p_\theta(x)]$$

- So if we can evaluate $\nabla_\theta \log p_\theta(x)$ for any x, we can estimate the gradient!
- Called REINFORCE estimator:
  - Doesn't require x to be continuous, e.g. it can be categorical with parameters a differentiable function of θ
  - Unbiased estimator
  - Tends to have large variance

Reverse KL guarantees

bonus!

- Does $\mathrm{KL}(q \parallel p)$ being small tell us anything about $|\mathbb{E}_{X\sim p} f(X) - \mathbb{E}_{X\sim q} f(X)|$?
- For a bounded f with $|f(x)| \le F$ for all x, we have for any p and q that

$$\begin{aligned}\left| \mathop{\mathbb{E}}_{X\sim p} f(X) - \mathop{\mathbb{E}}_{X\sim q} f(X) \right| &= 2F\,\frac{1}{2} \left| \mathop{\mathbb{E}}_{X\sim p} \frac{f(X)}{F} - \mathop{\mathbb{E}}_{X\sim q} \frac{f(X)}{F} \right| \\ &\le 2F\,\mathrm{TV}(p,q) \\ &\le F\sqrt{2\,\mathrm{KL}(q \parallel p)}\end{aligned}$$

  - Here we used the total variation distance, which is (supremum is a fancy version of max)

$$\mathrm{TV}(p,q) = \frac{1}{2} \sup_{f:\forall x, |f(x)|\le 1} \left| \mathop{\mathbb{E}}_{X\sim p} f(X) - \mathop{\mathbb{E}}_{Y\sim q} f(Y) \right|$$

  - Pinsker's inequality says that $\mathrm{TV}(p,q) \le \sqrt{\frac{1}{2}\,\mathrm{KL}(p \parallel q)}$; note TV is symmetric
    - `arxiv.org/abs/2202.07198` gives a nice overview of when this is tight
- Not a super tight bound, but does give some reassurance
- Can min. another "integral probability metric" for better bound, but usually harder

Computing the ELBO and its gradient: KL term

bonus!

- We want to maximize the average of

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

- 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)

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

- 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