CPSC 440/540: Advanced Machine Learning

Variational Inference and VAEs

Danica Sutherland (building on materials from Mark Schmidt)

University of British Columbia

Winter 2023

Outline

Need for Approximate Inference

- We have seen a variety of models where inference can be intractable:
  - Bayesian logistic regression.
  - Markov chains with non-Gaussians continuous states.
  - Non-forest graphical models.
  - LDA topic modeling.
- 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,

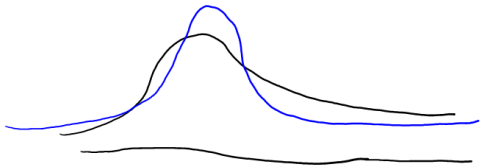
$$p(x) \approx q(x).$$

     - E.g., Gaussian, independent Bernoulli, or tree UGM.

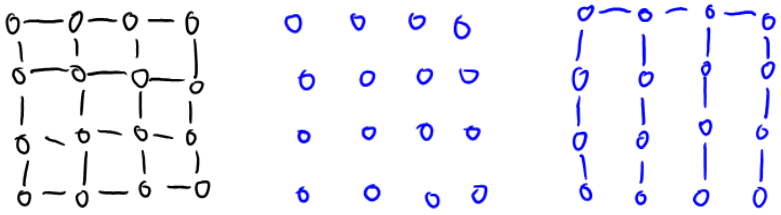
       (or mixtures of these simple distributions)
   - Turns inference into optimization.

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 and q are the same, 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.
    - Note that KL is not symmetric: in general, $\mathrm{KL}(p \parallel q) \neq \mathrm{KL}(q \parallel p)$.
- Maximum likelihood is the same as minimizing $\mathrm{KL}(p_{\text{true}} \parallel p_\theta)$ (see bonus slide).
- Unfortunately, this requires summing/integrating over p.
  - ... and that's exactly the problem we're trying to solve.

Minimizing Reverse KL Divergence

- Most variational methods minimize "reverse KL,"

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

  - 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.

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

Variational Approximation with a Multivariate Gaussian

- We want to find $\min_q \mathbb{E}_{x\sim q}[\log q(x)] - \mathbb{E}_{x\sim q}[\log \tilde{p}(x)]$.
- First term is minus the differential entropy, $\mathrm{H}[q] = -\int q(x)\log q(x)\mathrm{d}x$.
- For multivariate Gaussians, we have $H[q] = \frac{1}{2}\log\det\Sigma + \frac{d}{2} + \frac{d}{2}\log(2\pi)$.
- So to find the best multivariate Gaussian approximation, we need to find

$$\arg\max_{\mu,\Sigma} \tfrac{1}{2}\log\det\Sigma + \mathop{\mathbb{E}}_{x\sim\mathcal{N}(\mu,\Sigma)} \log\tilde{p}(x) = \arg\max_{\mu,L} \log\det L + \mathop{\mathbb{E}}_{z\sim\mathcal{N}(0,I)} \log\tilde{p}(\mu+Lz).$$

- End up with $q = \mathcal{N}(\mu, LL^\mathsf{T})$.
  - If L is lower-triangular with $L_j j > 0$ (Cholesky factor), then $\det L = \prod_j L_{jj}$ is easy.
- One instance of the "reparamaterization trick" to optimize an expectation.
- Can take samples for z and run SGD to optimize (but note it's non-convex).

Coordinate Optimization: Mean Field Approximation

- Another common scheme is coordinate optimization with an appropriate q.
- Consider minimizing reverse KL when q is a product of independent q_j,

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

where we choose q to be discrete or conjugate (e.g. Gaussian).

- If we fix q_{-j} and optimize the functional q_j we obtain (see PML2 10.2)

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

which we can use to update q_j for a particular j.

Coordinate Optimization: Mean Field Approximation

- Each iteration we choose a j and set q based on mean (of neighbours),

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

- This improves the (non-convex) reverse KL on each iteration.

- Applying this update is called:
  - Mean field method (graphical models).
  - Variational Bayes (Bayesian inference).

Three Coordinate-Wise Algorithms

- 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 the lecture 20 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

- Consider a pairwise discrete UGM:

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

- 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 $M_j(x_j)$.

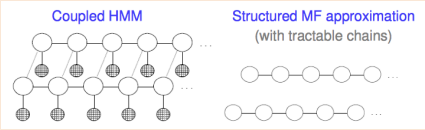
- Gibbs for updating a node j with 2 neighbours (i and k).
  1. Compute $M_j(x_j) = \phi_j(x_j)\phi_{ij}(x_i, x_j)\phi_{jk}(x_j, x_k)$ for all x_j.
  2. Sample x_j proportional to $M_j(x_j)$.

- 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 $M_j(x_j)$.

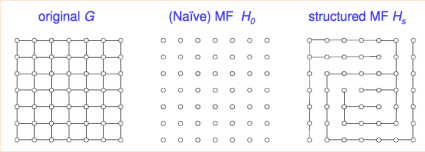
Structure Mean Field

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

- Original LDA paper proposed a structured mean field approximation.

Variational vs. Monte Carlo

bonus!

- 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.

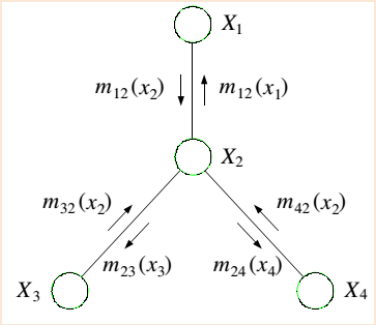
- Combinations of variational inference and stochastic methods:
  - Stochastic variational inference (SVI): use SGD to speed up variational methods.
  - Can initialize MCMC parameters based on a variational estimate.
  - Variational MCMC: use Metropolis-Hastings with proposals from a variational q.

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.

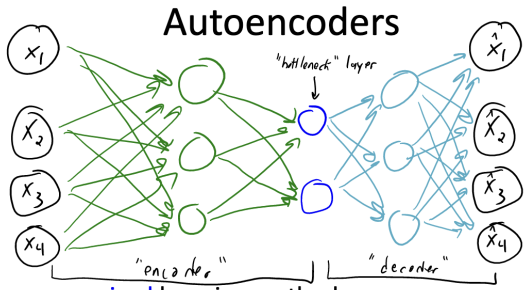
- Alternate view: write exact inference as constrained convex optimization.
  - Writing inference as maximizing entropy with constraints on marginals.
    - See bonus slides from the exponential family lecture.
  - Different methods correspond to different entropy/constraint approximations.
    - Mean 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`

Outline

Autoencoders

- Way back in lecture 5, we talked about auto-encoders:



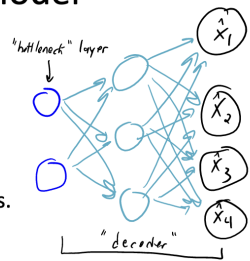
- This is an unsupervised learning method.
  - There are no labels y.
- Relationship to principal component analysis (PCA):
  - With squared error and linear network, equivalent to PCA.
    - Size of bottleneck layer gives number of latent factors k in PCA.
  - With non-linear transforms: a non-linear/deep generalization of PCA.

Autoencoders

- Way back in lecture 5, we talked about auto-encoders:

Decoder as Generative Model

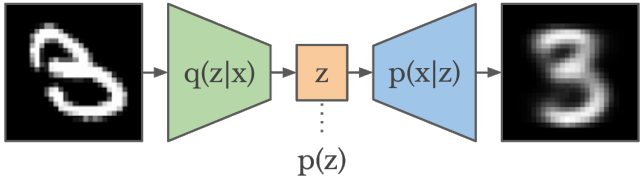
- Consider the decoder part of the network:
  - Takes low-dimensional z^i and makes features $\hat{x}^i$.
- Can be used for outlier detection:
  - Check distance to original features to detect outliers.
- Can be used to generate new data:
  - The z close to training examples should generate new valid "samples."
  - But this is not actually sampling, since we aren't modeling p(z) yet.



- Let's fix that "not actually sampling" part, to get a real generative model.

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 $\mathcal{N}(0, I)$
- Another view: fitting the distribution $p_\theta(x) = \int p_\theta(x \mid z) p_\theta(z) \mathrm{d}z$ to data
  - Plus 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
  - We can sample from p_θ ancestrally: $z \sim p_\theta(z)$, $x \sim p_\theta(x \mid z)$.

ELBO

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

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

- 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 $\mathbb{E}_x \, \mathrm{ELBO}_{\theta,\phi}(x)$ over ϕ, the bound gets tighter for these x.
- Maximizing $\mathbb{E}_x \, \mathrm{ELBO}_{\theta,\phi}(x)$ over θ moves towards maximizing likelihood for p_θ.

ELBO as regularized approximate likelihood

- We can rewrite the ELBO as

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

- If $q_\phi \approx p_\theta$, the first term is approximately $p_\theta(x)$.
- The second term regularizes $q_\phi(z \mid x)$ to stay "near" $p_\theta(z)$.

Computing the ELBO and its gradient: KL term

- We want to maximize the average of

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

- KL term for a given x is often available in closed form.
  - Typically we choose $p_\theta(z)$ to be just $\mathcal{N}(0, I)$.
  - Typically we choose $q_\phi(z \mid x)$ to be $\mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$.
  - In this case, we get that the KL term is just (see PML2 eq 5.80)

$$\mathrm{KL}(\mathcal{N}(\mu_\phi(x), \Sigma_\phi(x)) \parallel \mathcal{N}(0, I)) = \tfrac{1}{2}\left(\|\mu_\phi(x)\|^2 + \mathrm{Tr}\,\Sigma_\phi(x) - \log\det\Sigma_\phi(x) - d\right).$$

  - Most of the time we also choose $\Sigma_\phi(x)$ to be diagonal; determinant is easy.
  - This is just an expression in terms of ϕ that we can autodiff.

Optimizing the ELBO and its gradient: the reparameterization trick

- We want to maximize the average of

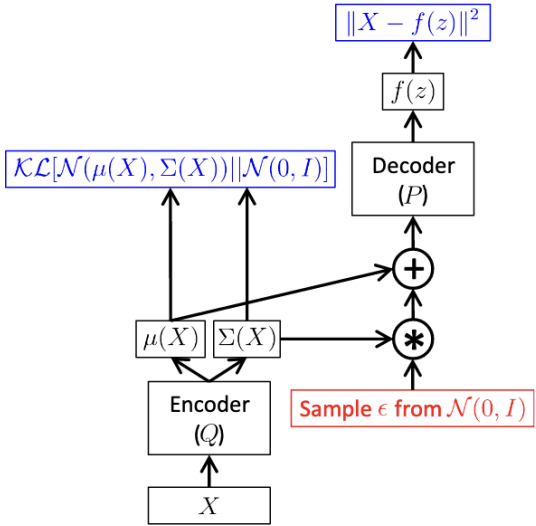
$$\mathrm{ELBO}_{\theta,\phi}(x) = \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} [\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \parallel p(z)).$$

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

$$\mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} [\log p_\theta(x \mid z)] = \mathop{\mathbb{E}}_{\xi\sim\mathcal{N}(0,I)} \log p_\theta(x \mid z = \mu_\phi(x) + \Sigma_\phi(x)^{\frac{1}{2}}\xi).$$

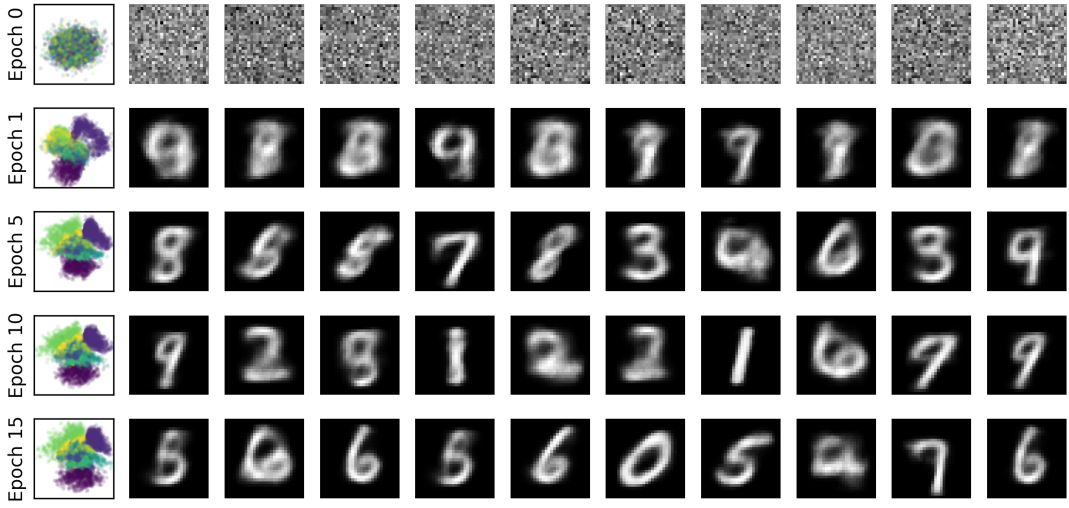
  - Take a Monte Carlo sample for ξ; now have something we can autodiff.
- Now just do SGD to maximize $\frac{1}{n}\sum_{i=1}^n \widehat{\mathrm{ELBO}}_{\theta,\phi}(x^i)$.

A VAE



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

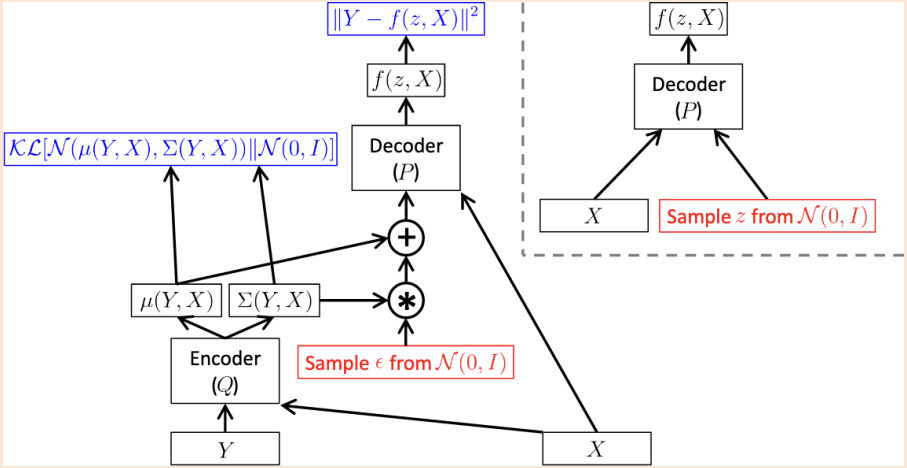
A VAE on MNIST



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

Conditional VAE

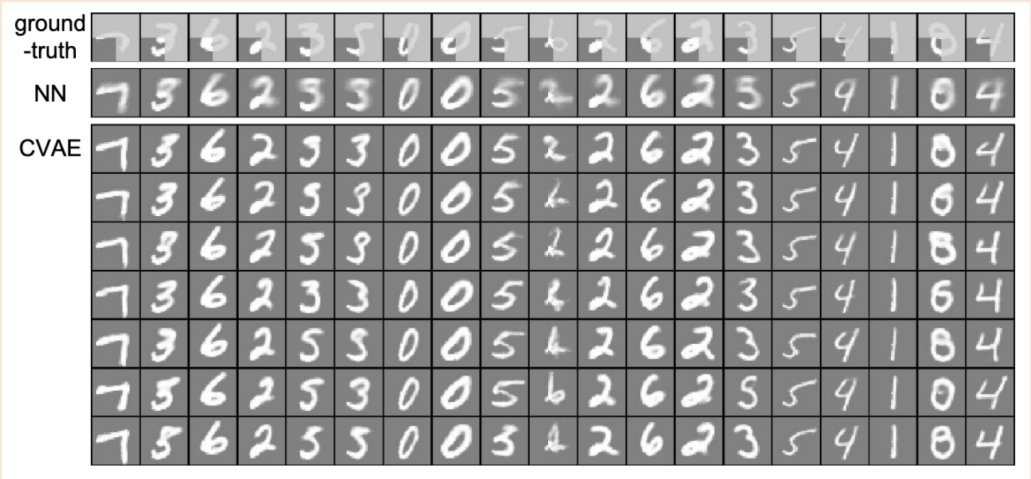
bonus!



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

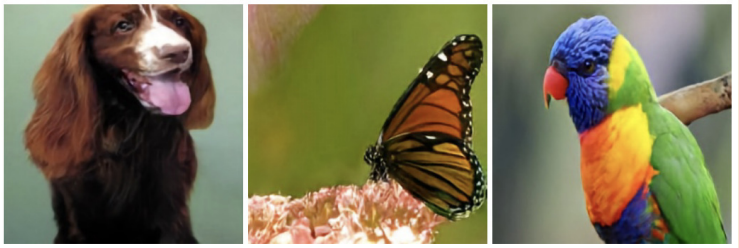


Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

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

β-VAE

bonus!

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

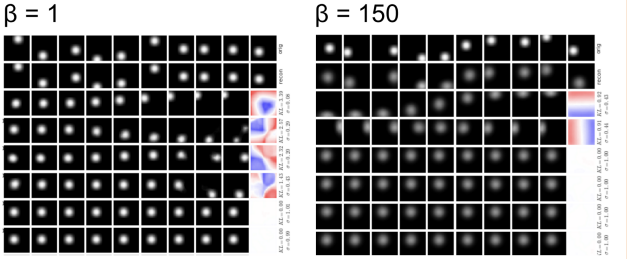


Figure 2: **Entangled versus disentangled representations of positional factors of variation learnt by a standard VAE** ($\beta = 1$) **and β-VAE** ($\beta = 150$) **respectively.** The dataset consists of Gaussian blobs presented in various locations on a black canvas. Top row: original images. Second 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

Summary

- Variational methods approximate p with a simpler distribution q.
  - Mean field approximation minimizes reverse KL divergence with independent q.
  - Loopy belief propagation is a heuristic that often works well.
- Variational auto-encoders (VAEs) do this for a "deep latent variable model."
- Next lecture: how DALLE-2 / Midjourney / Stable Diffusion work.

Maximum likelihood minimizes KL

bonus!

$$
\begin{aligned}
\arg\min_\theta \mathrm{KL}(p_{\mathsf{true}} \,\|\, p_\theta) &= \arg\min_\theta \int p_{\mathsf{true}}(x) \log \frac{p_{\mathsf{true}}(x)}{p_\theta(x)} \mathrm{d}x \\
&= \arg\min_\theta \int p_{\mathsf{true}}(x) \log p_{\mathsf{true}}(x) \mathrm{d}x - \int p_{\mathsf{true}}(x) \log p_\theta(x) \mathrm{d}x \\
&= \arg\min_\theta - \int p_{\mathsf{true}}(x) \log p_\theta(x) \mathrm{d}x \\
&= \arg\max_\theta \mathop{\mathbb{E}}_{x \sim p_{\mathsf{true}}} \log p_\theta(x) \\
&\approx \arg\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x^i)
\end{aligned}
$$

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:

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

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:

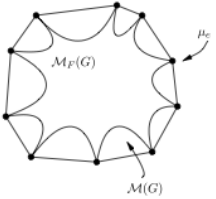


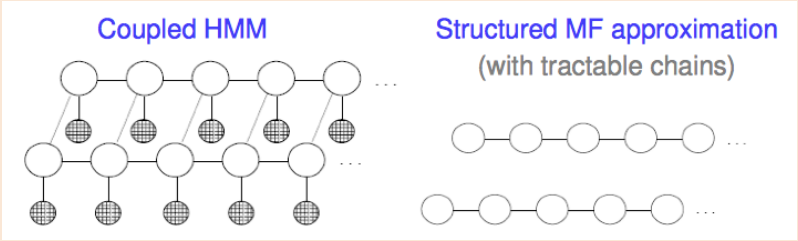
Fig. 5.3 Cartoon illustration of the set $\mathcal{M}_F(G)$ of mean parameters that arise from tractable distributions is a nonconvex inner bound on $\mathcal{M}(G)$. Illustrated here is the case of discrete random variables where $\mathcal{M}(G)$ is a polytope. The circles correspond to mean parameters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

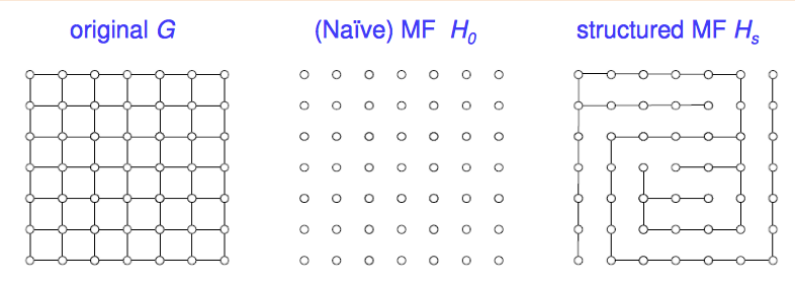


http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf

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.