Variational Autoencoder

map input to a distribution (rather than a fixed vector, like in autoencoders for dimensionality reduction)

This distribution p_θ is parameterized by neural network weights θ.

Relationship between input x and latent encoding vector z fully described by:

prior $p_\theta(z)$

likelihood $p_\theta(x|z)$

posterior $p_\theta(z|x)$

If we know θ^* (which we learned), to generate samples:

① sample a $z^{(i)}$ from prior $p_{\theta^*}(z)$

② generate value $x^{(i)}$ from conditional $p_{\theta^*}(x|z=z^{(i)})$

where $\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} p_\theta(x^{(i)})$

For computational tractability, first an approximate function $q_\phi(z|x)$ to output what is likely latent code z given input x.

- $p_\theta(x|z)$ is generative model (probabilistic decoder)
- $q_\phi(z|x)$ is estimated posterior (probabilistic encoder)

We want $q_\phi(z|x)$ to be close to $p_\theta(z|x)$ so

minimize $D_{KL}(q_\phi(z|x) \,||\, p_\theta(z|x))$ → I.A to the ELBO objective.

To encourage less blurring, more use of the latent code, forces disentanglement of the latent space, we can do:

① introduce a mutual information term: into VAE

② allow some Lagrangian weighting parameter in the 2 ELBO terms β-VAE.

ELBO:

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

$$D_{KL}\left(q_\phi(z|x) \| p(z)\right)$$

β-VAE:

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta D_{KL}\left(q_\phi(z|x) \| p(z)\right)$$

when $\beta > 1$ is usually chosen, to force more disentanglement in latent z representation.

The β-VAE objective is closely related to the information bottleneck

$$\max\left[I(z;Y) - \beta I(X;z)\right]$$

where goal was to preserve as much information $I(z;Y)$ about task Y while reducing rate $I(X;z)$ of representation.

Constrain latent space to be discrete

VQ-VAE (vector quantization - VAE)

Intro to Quantization Theory.

Consider an iid sequence of analog random variables

$U_1, U_2, \ldots \sim f_U(u)$

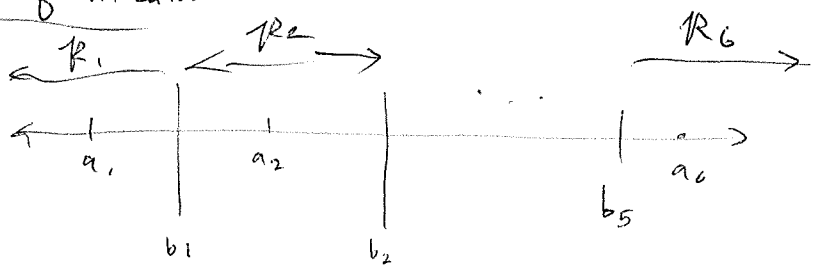
A quantizer maps this sequence into a sequence of discrete r.v.s $V_1, V_2, \ldots$ where V_m should represent U_m, for each m with as little distortion as possible.

If restrict to an alphabet of M possible values quantizer output V_m can't represent losslessly, but more M is generally better

$U_m \rightarrow$ [encoder] $\rightarrow I \in \{1, \ldots, M\} \rightarrow$ [decoder] $\rightarrow V_m$

a common distortion criterion is mean-square error $\mathbb{E}[(U-V)^2]$

Scalar quantization



① given representation points $\{a_j\}$, how should intervals $\{\mathcal{R}_j\}$ be chosen?

nearest neighbor rule: $b_j = \frac{a_j + a_{j+1}}{2}$

② given intervals $\{\mathcal{R}_j\}$, how to choose representation $\{a_j\}$?

centroid: $a_j = \mathbb{E}[U \mid U \in \mathcal{R}_j]$

Lloyd-Max algorithm alternates between them.

Trushkin (1982)

Lloyd-Max: given $M, f_U(u)$

(1) arbitrarily initialize

(2) for each j, set $b_j = \frac{a_j + a_{j+1}}{2}$

(3) for each j, set $a_j = \mathbb{E}[U | U \in [b_{j-1}, b_j]]$

repeat until convergence.

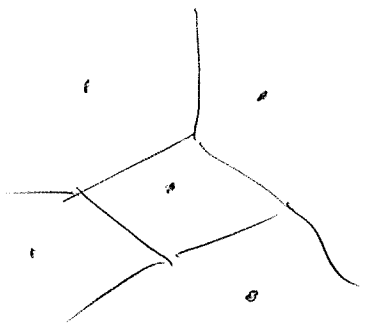
Under some convexity conditions (Trushkin, 1982), finds global optimum.

[Shroma, 1978 has dynamic programming formulation].

Vector quantization: consider longer vectors. (i.e. $n=2$)

(1) region R_j must be set of points (u, u') that are closer to (u_j, u_j') than any other representation point.

(2) R_j are minimum distance region (i.e. Voronoi regions for given representation points)



Convex polygonal regions where boundaries are perpendicular bisectors

(2) centroids still $\mathbb{E}[\vec{U} | \vec{U} \in R_j]$

Vector Quantization Advantages (Gray & Linde, 1980)

(1) memory advantage

(2) space-filling advantage

(3) shape advantage.

VQ-VAE

using a discrete latent space should prevent "posterior collapse", ie. blurring.

- encoder $q_\phi(z|x)$
- decoder $p_\theta(x|z)$
- prior $p(z)$.

Define latent embedding space as subset of $\mathbb{R}^D$ with K representative points:

$e_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, K$ which are possible embedding vectors.

The encoder output $E(x) = z_e$ goes through nearest-neighbor lookup to match one of the K embedding vectors, and then this is input to decoder.

$$z_q(x) = Q(E(x)) = e_k \quad \text{where } k = \operatorname{argmin}_i \|E(x) - e_i\|_2.$$

Another view (posterior is categorical distribution that is one-hot):

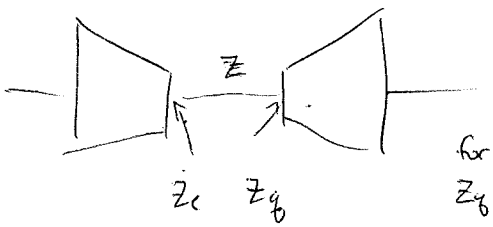
$$q(z = k|x) = \begin{cases} 1 & \text{for } k = \operatorname{argmin}_i \|E(x) - e_i\|_2 \\ 0 & \text{else} \end{cases}$$

$$= \begin{cases} 1 & \text{for } k = \operatorname{argmin}_i \|z_e(x) - e_i\|_2 \\ 0 & \text{else.} \end{cases}$$

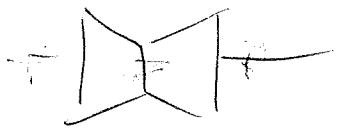
For training, since argmin is non-differentiable on discrete space, gradient of loss function $\mathcal{L}$ (eg ELBO or something like that)

$\nabla_z \mathcal{L}$ from decoder input z_q copied to encoder output z_e.

VQ-VAE



for optimization
$z_q = z_e$



Consider 3 terms in L for VQ-VAE.

1. reconstruction loss which optimizes decoder, encoder.
2. due to copying of gradient from $z_e(x)$ to $z_q(x)$, need some way to learn embedding space, i.e. vector quantizer under MSE.
3. to ensure encoder commits to an embedding and its output doesn't grow, have "commitment loss".

Let $sg(\cdot)$ be stop-gradient operator that is defined as identity at forward computation and has zero partial derivatives, effectively constraining its operand to be a non-updated constant.

$$L = \|x - D(e_k)\|_2^2 + \|sg[E(x)] - e_k\|_2^2 + \beta \|E(x) - sg(e_k)\|_2^2$$

reconstruction loss | VQ loss | commitment loss

encourages encoder output to stay close to embedding space and prevents it from fluctuating too frequently from one code vector to another

$$L = \log p(x \mid z_q(x)) + \| sg[z_e(x)] - e \|_2^2 + \beta \| z_e(x) - sg[e] \|_2^2$$

Also learn a prior over the ~~latent~~ latent discrete codebook so we can sample from it

For generation: decoder can receive input vectors sampled from similar distribution as in training

[fancy in VQ-VAE 2]

---

output-interpretable VAE (oi-VAE)

introduce structured sparsity into the generator.

- factors generator across observation dimensions with separate generator per group of variable.
  - → coupled through shared latent space and joint training