

Perspectives on Variational Autoencoders

Marco Henrique de Almeida Inácio

October 24, 2022

Work plan

- Basics of Bayesian inference.
- Points about generalization and generative models.
- Basics of variational inference.
- The structure of the problem we wish to solve.
- Variational autoencoders as the desired solution.
- Some properties of variational autoencoders.
- Two sample comparison with variational autoencoders.
- Code examples with PyStan, NumPy and Pyro: available at <https://pytutorial.marcoincio.com/sections/vae/>.

Bayesian inference

Assume data D come from a distribution.

Example:

$$D = (Y_1, Y_2, \dots, Y_n)$$

$$Y_i \sim \text{Bernoulli}(\alpha) \text{ i.i.d.}$$

Bayesian inference

Assume data D come from a distribution.

Example:

$$D = (Y_1, Y_2, \dots, Y_n)$$

$$Y_i \sim \text{Bernoulli}(\alpha) \text{ i.i.d.}$$

We want to infer about α . We start with a initial distribution for α , called prior distribution.

Bayesian inference

Assume data D come from a distribution.

Example:

$$D = (Y_1, Y_2, \dots, Y_n)$$

$$Y_i \sim \text{Bernoulli}(\alpha) \text{ i.i.d.}$$

We want to infer about α . We start with a initial distribution for α , called prior distribution. For example:

$$\alpha \sim \text{Beta}(2, 2)$$

Given those two and the observed dataset d , we can derive our posterior distribution from Bayes rule.

$$P(\alpha|D = d) = \frac{P(D = d|\alpha)P(\alpha)}{P(D = d)}$$

Given those two and the observed dataset d , we can derive our posterior distribution from Bayes rule.

$$P(\alpha|D = d) = \frac{P(D = d|\alpha)P(\alpha)}{P(D = d)}$$

In this case we have an analytic solution:

$$\alpha|D = d \sim \text{Beta}(2 + \text{sum}(d), 2 + \text{len}(d) - \text{sum}(d))$$

If instead, we have:

$$D = (Y_1, Y_2, \dots, Y_n)$$

$$Y_i \sim \text{Normal}(\mu, \sigma^2 = c) \text{ i.i.d.}$$

$$\text{With prior: } \mu \sim \text{Normal}(\mu_0, \sigma_0^2)$$

We also have an analytic solution:

$$\mu | D = d \sim \text{Normal} \left(\frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{c}} \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^n x_i}{c} \right), \left(\frac{1}{\sigma_0^2} + \frac{n}{c} \right)^{-1} \right)$$

(see Wikipedia's "Conjugate prior" article for more cases with analytic solution)

In more general cases, however, we can generate MCMC samples from $\alpha|D = d$ and using specific algorithms such as Metropolis and HMC.

Sampling replications

Once we have our posterior, we can sample new data (replications) from it:

$$\begin{aligned} P(\tilde{Y}_j | D = d) &= \int P(\tilde{Y}_j | D = d, \alpha = k) \text{pdf}_{\alpha | D=d}(k) dk \\ &= \int P(\tilde{Y}_j | \alpha = k) \text{pdf}_{\alpha | D=d}(k) dk \end{aligned}$$

Or if α was discrete:

$$P(\tilde{Y}_j | D = d) = \sum_k P(\tilde{Y}_j | \alpha = k) P(\alpha = k | D = d)$$

Sample replications

$$P(\tilde{Y}_j | D = d) = \int P(\tilde{Y}_j | \alpha = k) \text{pdf}_{\alpha|D=d}(k) dk$$

If we have MCMC posterior samples from $\alpha | D = d$, then we can sample from the $\tilde{Y}_j | \alpha$ for each sample to obtain replications (so we approximate the pdf to discrete distribution).

This Bayesian setup is naturally generative and handles probabilistic modelling quite well

(and with a frequentist estimator, we can also generate a confidence interval for new predictions, although it'd be a little more tricky when there is no analytic solution)

But, it can be too simplistic when we have more complex, multidimensional data and, it's problematic for big data in particular, as we'll see.

On the other hand, some powerful models like dense neural networks do not have this easy way to extract generative and probabilistic/uncertainty measurements (not that it's not possible).

So, it would be nice to have generative and probabilistic modeling framework that leverages on the state-of-art machine learning power house.

Why generalize

Points to think about:

Why generalize? What are the consequence?

- Bias variance tradeoff.
- Class complexity, assumptions, generalization and irreducible error.
- Convergence to the oracle of the class, and convergence “speed”.
- Decision theory and impossibility of a dominant estimator for all classes of problems (except for some extremely simple ones).

See https://pytutorial.marcoincio.com/sections/bias_variance_mse_convergence/ for some “theory” /examples regarding this topic.

Why generative?

Why generative modeling and/or density estimation in machine learning? What are the consequences?

- Definition of the objective at hand: distinction between trying to predict a point with low MSE and predicting something similar to possible data points.
- Necessity of generating new similar data (replications): data augmentation, new instances for structured data (e.g.: sound, images, molecules).
- Probabilistic measurement, uncertainty quantification, interpretability of results, discrepancy of data points or datasets.
- Possibility of include a probabilistic priori knowledge and understand its impact.
- Take advantage of power-full new machine learning tools for these classes of problems.

Properties

How does it each methodology compare in terms of probabilistic modeling and generative:

	Probabilistic modeling	Generative
Bayesian inference	Naturally	Easy
Frequentist inference	Yes	Tricky
Machine learning	Generally not	Tricky

MCMC scalability to big data

So, given a desired posterior distribution

$$P(Z|D = d) = \frac{P(D = d|Z)P(Z)}{P(D = d)}$$

We can generate MCMC samples and the show is over. However, this approach has its drawbacks such as big data scalability: MCMC is incompatible with data subsampling (Betancourt, 2015).

So, let's stick with Bayesian inference for now, and consider a more scalable alternative to MCMC.

Very very very complex notation ahead

But first things first.

Notation warning: let's assume that P is a probability measure and, that $P^{(Z)}$ is the distribution of Z relative to this measure ($P^{(Z|D=d)}$ being the posterior distribution of Z).

That is: $P^{(Z)}(c) = P(Z = c)$ and $P^{(Z)}(A) = P(Z \in A)$.

Let's assume we can define other probabilities measures, say Q , and therefore, we would have $Q^{(Z)}$ as the distribution of Z relative to Q (which might be distinct from $P^{(Z)}$!).

Variational inference

So, instead of doing MCMC inference, let's approximate P by some Q_ϕ . That is, find the closest distribution inside a class of distributions indexed by a parameter ϕ .

We can then use the Kullback–Leibler divergence as the criterion of proximity between them:

$$\begin{aligned} \mathbf{D}_{KL}(Q_\phi^{(Z)} | P(Z|D=d)) \\ = E_{Q_\phi} [\log Q_\phi(Z) - \log P(Z|D = d)] \end{aligned}$$

$$\begin{aligned} \mathbf{D}_{KL}(Q_\phi^{(Z)} | P(Z|D=d)) \\ = E_{Q_\phi} [\log Q_\phi(Z) - \log P(Z|D = d)] \end{aligned}$$

Where: $E_{Q_\phi}(f(Z)) = \int_{\Omega} f(Z(k))Q(dk) = \int_{\mathbb{R}} f(k)Q^{(Z)}(dk)$.

If Z is a discrete random variable: $\sum_{\mathbb{Z}} f(k)Q(Z = k)$.

If Z is a continuous random variable: $\int_{\mathbb{R}} f(k) \text{pdf}_{Q^{(Z)}}(k) dk$.

Now, note that this

$$\begin{aligned} & \mathbf{D}_{KL}(Q_\phi^{(Z)} | P^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log Q_\phi(Z) - \log P(Z|D=d)] \\ &= E_{Q_\phi} [\log Q_\phi(Z) - \log P(D=d|Z) - \log P(Z)] + \log P(D=d) \end{aligned}$$

can be rewritten as:

$$\begin{aligned} & \log P(D=d) - \mathbf{D}_{KL}(Q_\phi^{(Z)} | P^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log P(D=d|Z) + \log P(Z) - \log Q_\phi(Z)] \end{aligned}$$

Since $\log P(D = d)$ is constant in relation to Q , it suffices to maximize

$$\begin{aligned} E_{Q_\phi} [\log P(D = d|Z) + \log P(Z) - \log Q_\phi(Z)] \\ = E_{Q_\phi} [\log P(D = d, Z) - \log Q_\phi(Z)] \end{aligned}$$

This equation is called ELBO: the evidence lower bound.

$$\begin{aligned} & E_{Q_\phi} [\log P(D = d|Z) + \log P(Z) - \log Q_\phi(Z)] \\ &= E_{Q_\phi} [\log P(D = d, Z) - \log Q_\phi(Z)] \end{aligned}$$

Note that this maximization is basically a tradeoff: choose ϕ such that regions where $P(D = d, Z)$ is larger (in average) are favored or such that Q_ϕ get its mass better spread across the parameter space of Z (greater entropy).

$$\begin{aligned} & E_{Q_\phi} [\log P(D = d|Z) + \log P(Z) - \log Q_\phi(Z)] \\ &= E_{Q_\phi} [\log P(D = d, Z) - \log Q_\phi(Z)] \end{aligned}$$

Note that this maximization is basically a tradeoff: choose ϕ such that regions where $P(D = d, Z)$ is larger (in average) are favored or such that Q_ϕ get its mass better spread across the parameter space of Z (greater entropy).

Also note that we switched from a problem of MCMC sampling (parameter space exploration) to a problem of optimization, for which we can use data subsampling, SGD (see, P. Contributors, n.d. and E. Contributors, n.d. for caveats).

Bonus: you can use variational inference as warmup for MCMC sampling, see Hoffman et al., 2019.

Generation of similar instances

Now let's put variational inference on hold for a while.

Given random variable $D = (X_1, X_2, \dots, X_n)$ (e.g.: MNIST handwritten digit), let us generate replications, new instances, for each similar data points, for which. We could do this by encoding information in latent random variables $Z = (Z_1, Z_2, \dots, Z_n)$.

In case of MNIST handwritten digits, for instance, each Z_i could encode information about digit, stroke, angle, calligraphy, etc (Doersch, 2016).

Generation of similar instances

We could work with an hierarchical model Bayesian inference in a classical sense:

$$\begin{aligned} P(\theta, Z|D = d) &= \frac{P(D = d|Z, \theta)P(Z|\theta)P(\theta)}{P(D = d)} \\ &= \frac{P(\theta) \prod_{i=1}^n P(X_i = x_i|Z_i, \theta)P(Z_i|\theta)}{P(D = d)} \end{aligned}$$

Generation of similar instances

We could work with an hierarchical model Bayesian inference in a classical sense:

$$\begin{aligned} P(\theta, Z|D = d) &= \frac{P(D = d|Z, \theta)P(Z|\theta)P(\theta)}{P(D = d)} \\ &= \frac{P(\theta) \prod_{i=1}^n P(X_i = x_i|Z_i, \theta)P(Z_i|\theta)}{P(D = d)} \end{aligned}$$

But this is a little complicated for such a complex problem... defining $P(D = d|Z, \theta)$ is specially tricky for images (high dimension).

So let us instead, we work with a evidence optimization of θ .

Then instead of

$$P(\theta|D = d) = \frac{P(D = d|\theta)P(\theta)}{P(D = d)}$$

We work with maximizing $P(D = d; \theta) = P_{\theta}(D = d)$ (akin to a MLE or even a MAP assuming we had uniform prior on θ for this parameterization).

Let's also consider the following structure:

$$P_{\theta}(D = d|Z) = \prod_{i=1}^n \text{Normal}(X_i = x_i; (\mu_i, \sigma_i) = g_{\theta}(Z_i))$$

$$Z \sim \text{Normal}(0, 1)$$

Here g_{θ} is a complex function with parameter θ . Note that, if $\{g_{\theta} : \theta \in \Theta\}$ is complex enough, we can actually model any distribution of $X_i = x_i|Z_i$! So, we use neural networks due to its flexibility, scalability and the universal approximation theorem.

Some might be sad that we are not measuring the uncertainty of θ here, but just imagine the attempting to this here!

Curse of dimensionality

So, now that we gave up on a full Bayesian approach, let's then maximize $P_{\theta}(D = d) = \int P_{\theta}(D = d|Z = z)P^{(Z)}(dz)$:

- We sample from $P^{(Z)}$.
- Input those values on the neural network.
- Apply the output on the likelihood.
- Backpropagate, step and voila... well, not quite so.

We could stop here, if it weren't for something: the curse of dimensionality (see Doersch, 2016).

Variational inference revisited

We can solve the curse using variational inference, but first of all, let's update our previous equation:

$$\begin{aligned} & \log P(D = d) - \mathbf{D}_{KL}(Q_\phi^{(Z)} | P^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log P(D = d|Z) + \log P(Z) - \log Q_\phi(Z)] \end{aligned}$$

with the dependency we now have on θ :

$$\begin{aligned} & \log P_\theta(D = d) - \mathbf{D}_{KL}(Q_\phi^{(Z)} | P_\theta^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log P_\theta(D = d|Z) + \log P(Z) - \log Q_\phi(Z)] \end{aligned}$$

Problem solved?

$$\begin{aligned} & \log P_{\theta}(D = d) - \mathbf{D}_{KL}(Q_{\phi}^{(Z)} | P_{\theta}^{(Z|D=d)}) \\ &= E_{Q_{\phi}} [\log P_{\theta}(D|Z) + \log P(Z) - \log Q_{\phi}(Z) | D = d] \\ &= E_{Q_{\phi}} [\log P_{\theta}(D|Z) | D = d] - \mathbf{D}_{KL}(Q_{\phi}^{(Z)} | P^{(Z)}) \end{aligned}$$

Now can optimize the right hand side of the equation as before given the right choice a Q :

$$Q_{\phi}(Z_i | X_i = x_i) = \text{Distribution}(Z_i; \phi_i)$$

Now can optimize the right hand side of the equation as before given the right choice a Q :

$$Q_{\phi}(Z_i | X_i = x_i) = \text{Distribution}(Z_i; \phi_i)$$

This framework has a problem though...

Now can optimize the right hand side of the equation as before given the right choice a Q :

$$Q_{\phi}(Z_i|X_i = x_i) = \text{Distribution}(Z_i; \phi_i)$$

This framework has a problem though...

We have to optimize a ϕ_i for every instance in the dataset, this is computationally expensive.

Variational inference revisited again

So, we introduce the idea of amortization, by having Q_ϕ dependent on our X_i of interest:

$$\begin{aligned} & \log P_\theta(D = d) - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)} | P_\theta^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log P_\theta(D|Z) + \log P(Z) - \log Q_\phi(Z|D) | D = d] \end{aligned}$$

Problem solved!

$$\begin{aligned} & \log P_\theta(D = d) - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)} | P_\theta^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log P_\theta(D|Z) + \log P(Z) - \log Q_\phi(Z|D) | D = d] \\ &= E_{Q_\phi} [\log P_\theta(D|Z) | D = d] - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)} | P(Z)) \end{aligned}$$

Now can optimize the right hand side of the equation as before given the right choice of Q :

$$Q_\phi(Z_i | X_i = x_i) = \text{Normal}(Z_i; (\mu_i, \sigma_i) = h_\phi(x_i))$$

This approach is called amortized variational inference (see Kim and Pavlovic, 2021, for caveats) and this framework is called variational autoencoder (Kingma & Welling, 2013).

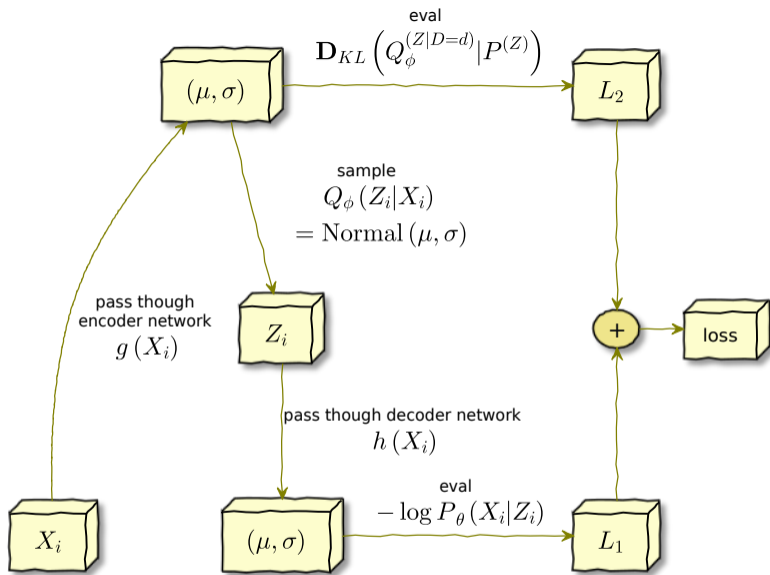


Diagram of the algorithm for optimizing $E_{Q_\phi} [\log P_\theta(D|Z) | D = d] - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)} | P(Z))$

Remarks

$$\begin{aligned} & \log P_\theta(D = d) - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)} | P_\theta^{(Z|D=d)}) \\ &= E_{Q_\phi} [\log P_\theta(D|Z) | D = d] - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)} | P(Z)) \end{aligned}$$

Note however that now we have something slightly different on the left hand side as $P_\theta(D = d)$ is not constant (with regards to what we are optimizing) anymore!

So we are actually minimizing the distance between Q and P while maximizing the evidence.

We have two objectives here: first find a Q_ϕ that is close to P for the parameters that are fully Bayesian (Z), and second, maximize the evidence as was the original goal.

Generating new instances

Now how do we generate new instances \tilde{X}_j 's? If we were working with a fully Bayesian model we could do:

$$P(\tilde{X}_j | D = d) = \int \int P(\tilde{X}_j | \tilde{Z}_j = z, \theta = o) P^{(\tilde{Z}_j)}(dz) P^{(\theta | D=d)}(do)$$

Since we aren't, we work with:

$$P_{\theta}(\tilde{X}_j | D = d) = \int P_{\theta}(\tilde{X}_j | \tilde{Z}_j = z) P^{(\tilde{Z}_j)}(dz)$$

i.e.: sample from $P(Z)$, apply the result on the neural network g_{θ} and the sample from a $Normal((\mu, \sigma) = g_{\theta})$.

Properties 2.0

So, from this we can conclude that:

	Probabilistic modeling	Generative
Bayesian inference	Naturally	Easy
Frequentist inference	Yes	Tricky
Machine learning	Generally not	Tricky
VAE	Yes	Easy

How can VAE have this properties?

But VAEs can be understood as a frequentist method like any other, so why is it easy to generate sample from it unlike many frequentist estimation methods?

Well, first let's see why it's not so easy to learn easy to sample new predictions from a frequentist model:

$$D = (Y_1, Y_2, \dots, Y_n)$$

$$Y_i \sim \text{Normal}(\mu, \sigma^2) \text{ i.i.d.}$$

Once we have the estimator $(\hat{\mu}, \hat{\sigma})$ can we generate new predictions from $\tilde{X}_j \sim \text{Normal}(\hat{\mu}, \hat{\sigma})$?

Once we have the estimator $(\hat{\mu}, \hat{\sigma})$ can we generate new predictions from $\tilde{X}_j \sim \text{Normal}(\hat{\mu}, \hat{\sigma})$?

Once we have the estimator $(\hat{\mu}, \hat{\sigma})$ can we generate new predictions from $\tilde{X}_j \sim \text{Normal}(\hat{\mu}, \hat{\sigma})$?

Not a good idea: it's a low quality prediction: we won't generate prediction intervals with correct frequentist coverage.

We can see this from the angle of Bayesian inference: using the Jeffreys prior, you get an equivalent inference to frequentist Statistics, but the prediction intervals will be different than that because we propagate the uncertainty over $(\hat{\mu}, \hat{\sigma})$ to \tilde{X}_j .

For this simple case we have an analytic solution (it's a T-student distribution), but for many cases we don't

Once we have the estimator $(\hat{\mu}, \hat{\sigma})$ can we generate new predictions from $\tilde{X}_j \sim \text{Normal}(\hat{\mu}, \hat{\sigma})$? Not a good idea.

So why we do that with variational autoencoders?

Once we have the estimator $(\hat{\mu}, \hat{\sigma})$ can we generate new predictions from $\tilde{X}_j \sim \text{Normal}(\hat{\mu}, \hat{\sigma})$? Not a good idea.

So why we do that with variational autoencoders?

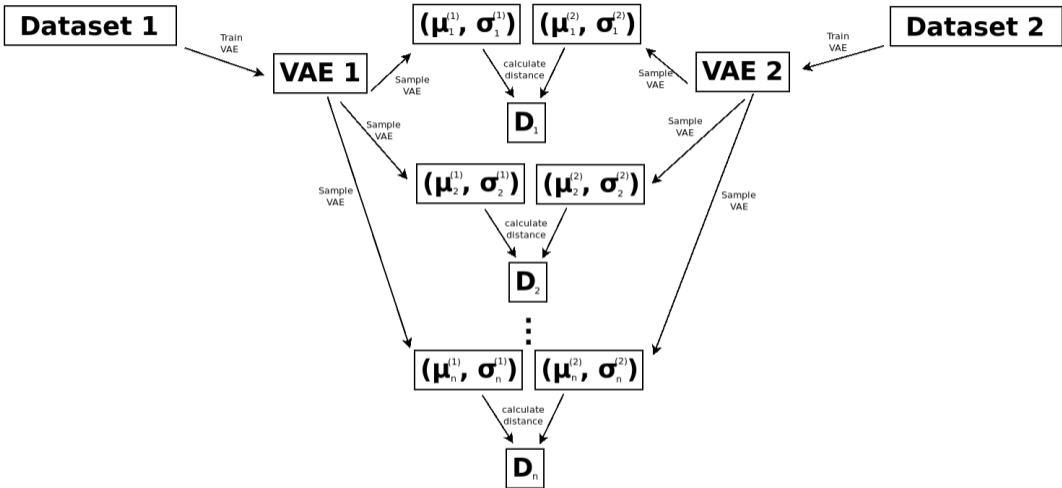
Because variational autoencoders live in a very large/powerful class (e.g.: an infinite mixture of normal distributions) to the point that we ignore propagating the uncertainty over each $(\hat{\mu}_j, \hat{\sigma}_j)$ pair that we sample.

Can VAE also be used as autoencoders, for representation learning and clusterization?






Yes, see:

- <http://thoth.inrialpes.fr/workshop/paiss2019/lecun-20191003-paiss.pdf>
- <https://stats.stackexchange.com/questions/324340>
- <http://pyro.ai/examples/vae.html>




Can VAE be used as density estimators? Absolutely: in Inácio et al., 2021, we use VAE as density estimation method to measure the distance between two sample (and with the help of permutation tests, test the hypothesis of whether the two samples arise from the same distribution).



References I

-  Betancourt, M. J. (2015). The fundamental incompatibility of hamiltonian monte carlo and data subsampling.
-  Contributors, E. (n.d.). $KL(q \parallel p)$ minimization [Accessed: 2018-11-30].
<http://edwardlib.org/tutorials/klqp>
-  Contributors, P. (n.d.). Svi part iii: Elbo gradient estimators [Accessed: 2018-11-30].
http://pyro.ai/examples/svi_part_iii.html
-  Doersch, C. (2016). Tutorial on variational autoencoders.
-  Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., & Vasudevan, S. (2019). Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport.

References II

-  Inácio, M., Izbicki, R., & Gyires-Tóth, B. (2021). Distance assessment and analysis of high-dimensional samples using variational autoencoders. *Inf. Sci. (Ny)*, 557, 407–420.
-  Kim, M., & Pavlovic, V. (2021). Reducing the amortization gap in variational autoencoders: A bayesian random function approach.
-  Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes.