

STAT 238 - Bayesian Statistics

Lecture Thirty Eight

Spring 2026, UC Berkeley

Aditya Guntuboyina

01 May 2026

As an example of variational inference, we shall look at an example of Variational Auto-Encoders (VAE) which are used for generative modeling.

1 Variational AutoEncoders

Consider data points y_1, \dots, y_n , where each y_i is a binary vector in $\{0, 1\}^{d_y}$. One can think of each y_i as an image, flattened from its 2D pixel grid into a single vector of length d_y , where each entry corresponds to one pixel. In general, pixel values can take many forms—for instance, grayscale images use intensities in $\{0, 1, \dots, 255\}$, and color images use three such values per pixel (one per RGB channel). Here we focus on the simplest case: *binary* images (also called black-and-white or bilevel images), in which each pixel is either off (0) or on (1). For example, a 28×28 binary image yields $d_y = 28^2 = 784$.

We build up a model for the data in stages. Since each coordinate of y_i is binary, the most basic assumption is

$$y_i \mid p_i \stackrel{\text{independent}}{\sim} \text{Bernoulli}(p_i),$$

where $p_i \in [0, 1]^{d_y}$ specifies the probability that each pixel is on (Bernoulli is interpreted componentwise, so pixels are conditionally independent given p_i). In other words, we are assuming that each $y_i \in \{0, 1\}^{d_y}$ is Bernoulli with its own parameter $p_i \in [0, 1]^{d_y}$. We need to further model p_1, \dots, p_n (otherwise, there will be too many parameters to yield anything useful). Here it is more convenient to work on the logit scale: let

$$\eta_i = \log \frac{p_i}{1 - p_i} \in \mathbb{R}^{d_y},$$

applied componentwise.

We shall assume η_1, \dots, η_n are i.i.d with some common distribution on \mathbb{R}^{d_y} . Further, we assume that this common distribution, despite living in \mathbb{R}^{d_y} , is in fact concentrated on (or near) a much lower-dimensional set. A natural way to encode this is to write

$$\eta_i = f_\theta(z_i) \quad \text{where } z_i \stackrel{\text{i.i.d.}}{\sim} N(0, I_d),$$

where $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^{d_y}$ is a neural network with parameters θ and $d \ll d_y$. The pushforward of $N(0, I_d)$ under f_θ defines a d -dimensional family of distributions on \mathbb{R}^{d_y} , and the flexibility

of f_θ ensures this family is rich enough to model the structure we care about. The Gaussian prior on z_i is essentially a convention: any continuous distribution on \mathbb{R}^d can be written as a deterministic transformation of a standard Gaussian, so fixing $z_i \sim N(0, I_d)$ and letting f_θ be flexible loses no generality (it is also analytically convenient, as we will see when designing an inference procedure). Putting the pieces together, and writing σ for the componentwise sigmoid, the overall model can be written in the following alternative way:

$$z_i \stackrel{\text{i.i.d.}}{\sim} N(0, I_d) \quad \text{and} \quad y_i | z_i \sim \text{Bernoulli}(\sigma(f_\theta(z_i)))$$

with, for example,

$$f_\theta(z_i) = B \text{ReLU}(Az_i + a) + b \quad \text{with } \theta = (A, a, B, b) \quad (1)$$

The unknown parameters in this model are the neural network parameters θ . The latent variables z_1, \dots, z_n are also unobserved. The z_i 's can be integrated out and the model becomes

$$y_i \stackrel{\text{i.i.d.}}{\sim} p_\theta$$

where

$$p_\theta(y) = \int p_\theta(y | z) \varphi(z) dz$$

with $\varphi(z)$ being the density of $N(0, I_d)$, and

$$p_\theta(y | z) = \prod_{j=1}^{d_y} (\sigma(f_{\theta,j}(z)))^{y_j} (1 - \sigma(f_{\theta,j}(z)))^{1-y_j} \quad \text{for } y = (y_1, \dots, y_{d_y}) \in \{0, 1\}^{d_y}.$$

The likelihood then is given by:

$$f_{\text{data}|\theta}(y_1, \dots, y_n) = \prod_{i=1}^n \left[\int p_\theta(y_i | z_i) \varphi(z_i) dz_i \right]$$

and the corresponding log-likelihood is

$$\log f_{\text{data}|\theta}(y_1, \dots, y_n) = \sum_{i=1}^n \log \left[\int p_\theta(y_i | z_i) \varphi(z_i) dz_i \right]. \quad (2)$$

This log-likelihood cannot be directly used as the objective function in optimization software such as PyTorch because of the presence of the integral. A natural idea is to discretize the integral by sampling $z_i^{(l)}, l = 1, \dots, M$ from $N(0, I_d)$ and forming the approximate log-likelihood:

$$\log f_{\text{data}|\theta}(y_1, \dots, y_n) \approx \sum_{i=1}^n \log \left[\frac{1}{M} \sum_{l=1}^M p_\theta(y_i | z_i^{(l)}) \right]. \quad (3)$$

The above term (3) is unlikely to be a good approximation to the actual log-likelihood (2). This is because the integral $\int p_\theta(y_i | z_i) \varphi(z_i) dz_i$ in (2) is dominated by the region where $p_\theta(y_i | z_i)$ is large i.e., by the posterior $p_\theta(z_i | y_i)$. This posterior typically is very concentrated in the sense that only a tiny region of $z_i \in \mathbb{R}^d$ yields non-negligible values of $p_\theta(y_i | z_i)$. The prior $z_i \sim N(0, I_d)$, by contrast, spreads mass diffusely over \mathbb{R}^d . The consequence will be that almost every sample $z_i^{(l)}$ lands in a region where $p_\theta(y_i | z_i^{(l)})$ is small, so that the average in (3) is dominated by one or two (or even zero) lucky samples. This makes the average in (3) a highly variance estimate of the integral in (2), and hence unreliable as an optimization objective.

A better approach is to use ideas from variational inference and to maximize the ELBO instead.

2 Use of the ELBO for maximization of $\log f_{\text{data}|\theta}(y_1, \dots, y_n)$

We shall use the following formula whose proof we saw in Lecture 36 (see also Lecture 37):

$$\log f_{\text{data}|\theta}(y_1, \dots, y_n) = \max_q \text{ELBO}(q, \theta) \quad (4)$$

where

$$\begin{aligned} \text{ELBO}(q, \theta) &= \int \cdots \int q(z_1, \dots, z_n) \log \frac{\prod_{i=1}^n p_\theta(y_i, z_i)}{q(z_1, \dots, z_n)} dz_1 \dots dz_n \\ &= \int \cdots \int q(z_1, \dots, z_n) \log \prod_{i=1}^n p_\theta(y_i | z_i) dz_1 \dots dz_n - \int \cdots \int q(z_1, \dots, z_n) \log \frac{q(z_1, \dots, z_n)}{\varphi(z_1) \cdots \varphi(z_n)} dz_1 \dots dz_n \\ &= \sum_{i=1}^n \int \log p_\theta(y_i | z_i) q_i(z_i) dz_i - KL(q \| \varphi \times \cdots \times \varphi). \end{aligned}$$

where q_i is the i -th marginal corresponding to q , and $\varphi \times \cdots \times \varphi$ denotes the distribution of (z_1, \dots, z_n) corresponding to $z_i \stackrel{\text{i.i.d}}{\sim} N(0, I_d)$.

The maximization in (4) is over all probability densities q of z_1, \dots, z_n . We have also seen (again in Lecture 36) that the maximum in (4) is achieved when q is the conditional density of z_1, \dots, z_n given y_1, \dots, y_n and θ :

$$q^*(z_1, \dots, z_n) \propto \prod_{i=1}^n [p_\theta(y_i | z_i) \varphi(z_i)].$$

It is clear that this q^* is a product measure $\prod_{i=1}^n q_i^*(z_i | y_i)$ where

$$q_i^*(z_i | y_i) \propto p_\theta(y_i | z_i) \varphi(z_i) \implies q_i^*(z_i | y_i) = \frac{p_\theta(y_i | z_i) \varphi(z_i)}{\int p_\theta(y_i | z_i) \varphi(z_i) dz_i}.$$

However because of the somewhat complicated form of $p_\theta(y_i | z_i)$, the density q_i^* above is not of a simple form (for example, it is not a Gaussian). We therefore take a simpler class of densities \mathcal{Q} and then maximize the ELBO over \mathcal{Q} . Specifically, we shall take \mathcal{Q} to be the class of all product densities $\prod q_{i,\phi}(z_i | y_i)$ for which each marginal $q_\phi(z_i | y_i)$ is Gaussian:

$$q_\phi(z_i | y_i) = \text{density of } N(\mu_\phi(y_i), \Sigma_\phi(y_i))$$

with

$$\mu_\phi(y_i) = (\mu_{\phi,j}(y_i), j = 1, \dots, d) \quad \text{and} \quad \Sigma_\phi(y_i) = \text{diag}(\sigma_{\phi,j}^2(y_i), j = 1, \dots, d).$$

With this choice \mathcal{Q} , the ELBO becomes

$$\text{ELBO}(q, \theta) = \sum_{i=1}^n \int \log p_\theta(y_i | z_i) q_\phi(z_i | y_i) dz_i - \sum_{i=1}^n KL(q(\cdot | y_i) \| \phi),$$

where we use the fact that the KL between two product distributions becomes the sum of the KL between the marginals. We can further write this as:

$$\text{ELBO}(q, \theta) = \sum_{i=1}^n \mathbb{E}_{z_i \sim q_\phi(\cdot | y_i)} \log p_\theta(y_i | z_i) - \sum_{i=1}^n KL(N(\mu_\phi(y_i), \Sigma_\phi(y_i)) \| N(0, I_d)).$$

The second term above can be written in closed form as (see e.g., https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence#Examples)

$$KL(N(\mu_\phi(y_i), \Sigma_\phi(y_i)) \| N(0, I_d)) = \frac{1}{2} \sum_{j=1}^d \{ \sigma_{\phi,j}^2(y_i) + \mu_{\phi,j}^2(y_i) - \log \sigma_{\phi,j}^2(y_i) - 1 \}.$$

For the first term in the ELBO expression above, we need to compute

$$\mathbb{E}_{z_i \sim q_\phi(\cdot | y_i)} \log p_\theta(y_i | z_i)$$

For this, we can use Monte Carlo averaging. First write $z_i \sim N(\mu_\phi(y_i), \Sigma_\phi(y_i))$ as

$$z_i = \mu_\phi(y_i) + \sigma_\phi(y_i) \odot u_i \quad \text{where } u_i \stackrel{\text{i.i.d.}}{\sim} N(0, I_d).$$

Here \odot stands for the Hadamard product (elementwise multiplication). Note that the components of $\Sigma_\phi(y_i)^{1/2} u_i$ are simply $\sigma_{\phi,j}(y_i) u_{i,j}$. Thus

$$\begin{aligned} \mathbb{E}_{z_i \sim q_\phi(\cdot | y_i)} \log p_\theta(y_i | z_i) &= \mathbb{E}_{u_i \sim N(0, I_d)} \log p_\theta(y_i | \mu_\phi(y_i) + \sigma_\phi(y_i) \odot u_i) \\ &\approx \frac{1}{L} \sum_{l=1}^L \log p_\theta(y_i | \mu_\phi(y_i) + \sigma_\phi(y_i) \odot u_i^{(l)}) \end{aligned}$$

where $u_i^{(l)}, 1 \leq l \leq L, 1 \leq i \leq n$ are i.i.d samples generated from $N(0, I_d)$. Thus the ELBO becomes:

$$\begin{aligned} \text{ELBO}(\phi, \theta) &= \frac{1}{L} \sum_{i=1}^n \sum_{l=1}^L \sum_{j=1}^{d_y} \left\{ y_{i,j} \log \sigma \left(f_{\theta,j}(\mu_\phi(y_i) + \sigma_\phi(y_i) \odot u_i^{(l)}) \right) \right. \\ &\quad \left. + (1 - y_{i,j}) \log \left[1 - \sigma \left(f_{\theta,j}(\mu_\phi(y_i) + \sigma_\phi(y_i) \odot u_i^{(l)}) \right) \right] \right\} \\ &\quad - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^d \{ \sigma_{\phi,j}^2(y_i) + \mu_{\phi,j}^2(y_i) - \log \sigma_{\phi,j}^2(y_i) - 1 \}. \end{aligned}$$

Here σ denotes the sigmoid function while σ_ϕ denotes the variance parameter of the variational distribution q_ϕ .

We simply maximize $\text{ELBO}(\phi, \theta)$ jointly over ϕ and θ to estimate them. We will parametrize μ_ϕ and σ_ϕ using neural networks. For example,

$$\begin{aligned} \mu_\phi(y) &= W_\mu \text{ReLU}(Wy + w) + b_\mu \\ \log \sigma_\phi^2(y) &= W_\sigma \text{ReLU}(Wy + w) + b_\sigma, \end{aligned}$$

with $\phi = (W_\mu, W, w, b_\mu, W_\sigma, b_\sigma)$. Recall also that f_θ will also be parametrized by a neural network as in (1).

With these parametrizations, $\text{ELBO}(\phi, \theta)$ can be maximized using optimization software such as PyTorch.