Variational and Information Theoretic Principles in Neural Networks

Karl Stratos

1 Autoencoder

Assume a fixed input $x \in \mathbb{R}^d$. We have an **encoder** Ψ which defines $P_\Psi(z|x)$: a distribution over hidden representations $z \in \mathbb{R}^{d'}$ conditioning on x. We also have a **decoder** Φ which defines $P_\Phi(x|z)$: a distribution over inputs $x \in \mathbb{R}^d$ conditioning on z.

Gaussian encoder. Define $P_\Psi(z|x) := \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$ where $\mu = \tanh(Wx)$ and $\sigma^2 = \tanh(Vx)$. The trainable parameters of Ψ are $W, V \in \mathbb{R}^{d' \times d}$.

Bernoulli decoder. Assume $x \in \{0,1\}^d$. Define $P_\Phi(x|z) := \prod_i p_i^{x_i}(1-p_i)^{1-x_i}$ where $p = (p_1 \dots p_d) = \text{softmax}(\tanh(Wz))$. The trainable parameter of Φ is $W \in \mathbb{R}^{d \times d'}$.

Let R be a regularization penalty on the encoder. The autoencoding objective is to minimize

$$J(\Psi, \Phi) := \mathop{\mathbf{E}}_{z \sim P_\Psi(\cdot|x)} [-\log P_\Phi(x|z)] + R(\Psi)$$

1.1 Basic Autoencoder

The encoder is deterministic and there is no regularization. Let $\Psi : \mathbb{R}^d \to \mathbb{R}^{d'}$ be a feedforward and $P_\Psi(\Psi(x)|x) := 1$. Then

$$J(\Psi, \Phi) = -\log P_\Phi(x|\Psi(x))$$

Because there's no randomness, it's important to make $d' < d$ to avoid the trivial scenario $f(x) = x$ and $P_\Phi(x|x) = 1$.

1.2 Denoising Autoencoder

Assume the same setting in the basic autoencoder, but add a noise distribution $\mathcal{E}$ over $\mathbb{R}^d$. Then

$$J(\Psi, \Phi) = \mathop{\mathbf{E}}_{\epsilon \sim \mathcal{E}} [-\log P_\Phi(x|\Psi(x+\epsilon))]$$

This is equivalent to an autoencoder with dropout (at the input layer) if $\epsilon_i = -x_i$ with probability 0.5 and zero otherwise. But the denoising autoencoder (Vincent et al., 2008) precedes dropout (Srivastava et al., 2014).

1.3 Variational Autoencoder (VAE)

We no longer assume that the encoder deterministic. We also regularize the encoder. The choice of encoder is $P_\Psi(z|x) := \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$ where $\mu, \sigma^2 \in \mathbb{R}^{d'}$ are functions of x (the Gaussian encoder example above). The choice of regularization is its KL divergence from the standard multivariate normal. The VAE objective is then given by

$$J(\Psi, \Phi) = \mathop{\mathbf{E}}_{z \sim P_\Psi(\cdot|x)} \left[-\log P_\Phi(x|z) \right] + D_{\text{KL}} \left(P_\Psi(z|x) || \mathcal{N}(z|0, I_{d'}) \right) \tag{1}$$

1.3.1 Optimization

Let's first just think about how to train Φ and Ψ. By choice the regularization term is given in closed form and differentiable:

$$D_{\text{KL}} \left(P_\Psi(z|x) || \mathcal{N}(z|0, I_{d'}) \right) = \frac{1}{2} \left(\sum_{i=1}^{d} \sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2 \right)$$

Now we need to express the first term as a differentiable function of μ and σ^2 in order to train Ψ. Actually sampling $z \sim \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$ and using z to compute $P_\Phi(x|z)$ will "sever" the connection to μ and σ^2 in the computation graph. But by reparameterizing[1]

$$\mathcal{N}(z|\mu, \text{diag}(\sigma^2)) = \mu + \mathcal{N}(z|0, I_{d'}) \odot \sigma^2$$

the first term is equivalently written as

$$\mathop{\mathbf{E}}_{\epsilon \sim \mathcal{N}(\cdot|0, I_{d'})} \left[-\log P_\Phi(x|\mu + \sigma^2 \odot \epsilon) \right]$$

In summary, training involves drawing $\epsilon \sim \mathcal{N}(\cdot|0, I_{d'})$ and taking a gradient step on

$$-\log P_\Phi(x|\mu + \sigma^2 \odot \epsilon) + \frac{1}{2} \left(\sum_{i=1}^{d} \sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2 \right)$$

1.3.2 Justification

VAE is (almost) just a regularized autoencoder. By making the encoding look like the standard normal, we put a spherical structure over the hidden representation. A catchy side effect is that once the model is trained, we can *generate* random inputs by replacing the encoder step with $z \sim \mathcal{N}(\cdot|0, I_{d'})$. Random MNIST digit images generated this way abound on the internet. But VAE can also be viewed as EM: optimizing the ELBO on a certain generative model (see Appendix A.3.1). The model is

$$P_\Phi(z, x) = \mathcal{N}(z|0, I_{d'}) P_\Phi(x|z)$$

This defines a **marginalized likelihood**

$$P_\Phi(x) = \int_{z \in \mathbb{R}^{d'}} \mathcal{N}(z|0, I_{d'}) P_\Phi(x|z) dz \tag{2}$$

Our goal is to compute the MLE

$$\Phi^{\text{MLE}} := \arg\max_\Phi \ \log P_\Phi(x)$$

Assuming we want to avoid computing (2), we can do EM. Introduce an auxilliary distribution $P_\Psi(z|x)$ (encoder) and maximize ELBO

$$\text{ELBO}(\Phi, \Psi) = \mathop{\mathbf{E}}_{z \sim P_\Psi(z|x)} \left[\log P_\Phi(x|z) \right] - D_{\text{KL}} \left(P_\Psi(z|x) || \mathcal{N}(z|0, I_{d'}) \right)$$

which is the (negative) VAE objective in (1).

[1] This trick, among other things, cannot accommodate discrete z.

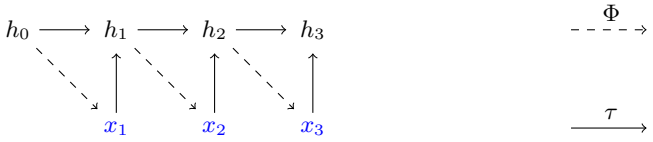
1.4 Variational RNN

It should be emphasized that the principle underlying VAE is EM (optimizing the ELBO lower bound under an auxiliary posterior) where latent variables are required to be continuous to enable the reparameterization trick. That means we can generally use it to introduce continuous latent variables into a model. A neat example is the variational RNN of Chung et al. (2016).

Let h_0 denote a constant initial hidden state. Given an input sequence $x_{\leq T} := x_1 \dots x_T$, a standard RNN *deterministically* computes hidden state transition $h_t = \tau(x_t, h_{t-1})$. Each h_t is a function of $x_{\leq t}$ and is used to define

$$p(x_{\leq T}) = \prod_{t=1}^{T} P_\Phi(x_t|h_{t-1})$$

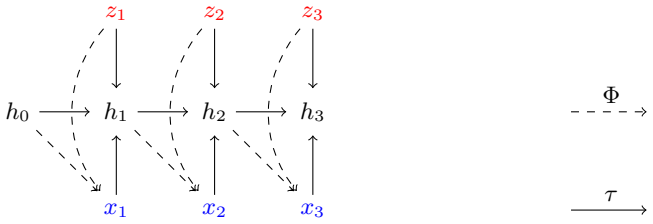
where $P_\Phi(x_t|h_{t-1})$ parametrizes $p(x_t|x_{<t})$. See the picture:



Chung et al. propose to improve the model's expressiveness by introducing variability also in hidden state transition. This is achieved by introducing a continuous latent variable $z_t \sim \mathcal{N}(\cdot|0, I_d)$ at each time step. If inputs *and* latent variables are given, then hidden state transition is again deterministic: $h_t = \tau(z_t, x_t, h_{t-1})$. Each h_t is a function of $z_{\leq t}$ as well as $x_{\leq t}$ and is used to define the joint probability

$$p(z_{\leq T}, x_{\leq T}) = \prod_{t=1}^{T} p(z_t, x_t|z_{<t}, x_{<t}) = \prod_{t=1}^{T} \mathcal{N}(z_t|0, I_d) P_\Phi(x_t|h_{t-1}, z_t)$$

where $P_\Phi(x_t|h_{t-1}, z_t)$ parametrizes $p(x_t|z_{\leq t}, x_{<t})$. See the picture:



Because z_t is unobserved, however, h_t is no longer deterministic. Just like in VAE, our goal is to find parameters τ and Φ that maximize the **marginalized log likelihood**:

$$\log p(x_{\leq T}) = \log \int_{z_{\leq T}} p(x_{\leq T}, z_{\leq T})\, d(z_{\leq T})$$

Instead of dealing with the integral over $z_{\leq T}$, we choose to maximize the ELBO lower bound on $\text{ELBO}_\tau(\Phi, \Psi) \leq \log p(x_{\leq T})$. An auxilliary distribution $P_\Psi(z_t|h_{t-1}, x_t)$ is introduced to parametrize the posterior distribution:

$$p(z_{\leq T}|x_{\leq T}) = \prod_{t=1}^{T} p(z_t|z_{<t}, x_{\leq T}) = \prod_{t=1}^{T} P_\Psi(z_t|h_{t-1}, x_t)$$

The auxilliary distribution again takes a Gaussian form to enable a closed-form expression of the KL term. Specifically, $P_\Psi(z_t|h_{t-1}, x_t) = \mathcal{N}(z_t|\mu_{\Psi,t}, \text{diag}(\sigma^2_{\Psi,t}))$ where $(\mu_{\Psi,t}, \sigma^2_{\Psi,t}) = \Psi(x_t, h_{t-1})$. The lower bound $\text{ELBO}_\tau(\Phi, \Psi)$ is then

$$\begin{aligned}
&\mathop{\mathbf{E}}_{z_{\leq T}\sim P_\Psi(\cdot|x_{\leq T})}\left[\log P_\Phi(x_{\leq T}|z_{\leq T})\right] - D_{\text{KL}}\left(P_\Psi(z_{\leq T}|x_{\leq T})||p(z_{\leq T})\right) \\
&= \mathop{\mathbf{E}}_{z_{\leq T}\sim P_\Psi(\cdot|x_{\leq T})}\left[\sum_{t=1}^{T} \log P_\Phi(x_t|h_{t-1}, z_t) - D_{\text{KL}}\left(\mathcal{N}(z_t|\mu_{\Psi,t}, \text{diag}(\sigma^2_{\Psi,t}))||\mathcal{N}(z_t|0, I_d)\right)\right]
\end{aligned}$$

where the hidden states are computed by τ using the latent variables drawn from Ψ. In summary, the training consists of the following steps. Given $x_{\leq T}$, for $t = 1 \ldots T$,

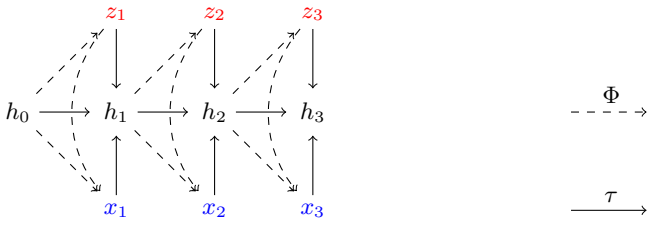
1. (Reparametrization trick) Draw $z_t \sim P_\Psi(\cdot|h_{t-1}, x_t)$ by computing $(\mu_{\Psi,t}, \sigma^2_{\Psi,t}) \leftarrow \Psi(x_t, h_{t-1})$ and then setting $z_t = \mu_{\Psi,t} + \sigma^2_{\Psi,t} \odot \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(\cdot|0, I_d)$.
2. Set $h_t = \tau(z_t, x_t, h_{t-1})$.

and take a gradient step on

$$\sum_{t=1}^{T} \frac{1}{2}\left(\sum_{i=1}^{d}[\sigma^2_{\Psi,t}]_i + [\mu_{\Psi,t}]^2_i - 1 - \log[\sigma^2_{\Psi,t}]_i\right) - \log P_\Phi(x_t|z_t, h_{t-1})$$

1.4.1 Conditional Prior

The actual model of Chung et al. makes z_t depend on h_{t-1}, thereby making it a function of $x_{<t}$ and $z_{<t}$. See the picture:



In functional form, $z_t \sim \mathcal{N}(\cdot|\mu_{\Phi,t}, \text{diag}(\sigma^2_{\Phi,t}))$ where $(\mu_{\Phi,t}, \sigma^2_{\Phi,t}) = \Phi(h_{t-1})$. This puts temporal structure over latent variables, but it's now not clear how to calculate

$$D_{\text{KL}}\left(P_\Psi(z_{\leq T}|x_{\leq T})||p(z_{\leq T})\right)$$

since we must marginalize $p(z_{\leq T}, x_{\leq T})$ over all possible $x_{\leq T}$ to calculate $p(z_{\leq T})$. The paper doesn't resolve this issue: they simply condition the prior on the current input sequence $x_{\leq T}$ and calculate

$$D_{\text{KL}}\left(P_\Psi(z_{\leq T}|x_{\leq T})||p(z_{\leq T}|x_{\leq T})\right)$$

This lets the ELBO lower bound be expressed as

$$\mathop{\mathbf{E}}_{z_{\leq T}\sim P_\Psi(\cdot|x_{\leq T})}\left[\sum_{t=1}^{T} \log P_\Phi(x_t|h_{t-1}, z_t) - D_{\text{KL}}\left(P_\Psi(z_t|h_{t-1}, x_t)||\mathcal{N}(z_t|\mu_{\Phi,t}, \sigma^2_{\Phi,t})\right)\right]$$

where the position-wise KL divergence can again be calculated in closed form. This model is shown to perform significantly better, but the calculation of ELBO is approximate.

2 Generative Adversarial Network (GAN)

2.1 As a Minimax Game

This is the motivation of the original GAN paper (Goodfellow et al., 2014). Let $\mathcal{P}(x)$ denote the **true input distribution** over $\mathbb{R}^d$. GAN learns $\mathcal{P}$ without any explicit parametrization of $\mathcal{P}$ by an adversarial framework.

Some definitions:

- $\pi(z)$ is a distribution over some **noise space** $\mathcal{Z}$, for instance a uniform distribution over a box $[-1,1]^{d'}$ or a spherical Gaussian over a unit ball (recommended).
- $G : \mathcal{Z} \to \mathbb{R}^d$ is a deterministic **generator** that transfroms a noise into an input. This can be a simple feedforward, a convolutional net, etc.
  - We can now sample a random input as follows: draw $z \sim \pi(\cdot)$ and then set $x = G(z)$. This defines $P_G(x, z) := p(z)\,[[x = G(z)]]$. The **fake input distribution** $P_G(x)$ is obtained by marginalizing over z.
- $D : \mathbb{R}^d \to [0,1]$ is a deterministic **discriminiator** that computes the probability of x being from $\mathcal{P}$ (i.e., rather than P_G). We will sometimes write $P_D(1|x)$ to denote $D(x)$ and $P_D(0|x)$ to denote $1 - D(x)$.

GAN defines an objective function so that G and D compete with each other. In particular, the task of fooling D provides the necessary gradient information to train G. The objective is

$$\min_G \max_D V(G, D)$$

where[2]

$$V(G, D) := \mathop{\mathbf{E}}_{x \sim \mathcal{P}(\cdot)} [\log D(x)] + \mathop{\mathbf{E}}_{x \sim P_G(\cdot)} [\log (1 - D(x))] \tag{3}$$

2.1.1 The Minimax Solution

For any fixed G,

$$V(G, D) = \int_{x \in \mathbb{R}^d} \mathcal{P}(x) \log D(x) + P_G(x) \log (1 - D(x))\, dx$$

is maximized by

$$D(x) = \frac{\mathcal{P}(x)}{\mathcal{P}(x) + P_G(x)}$$

So we can write

$$\begin{aligned}
\max_D V(G, D) &= \int_{x \in \mathbb{R}^d} \mathcal{P}(x) \log \frac{\mathcal{P}(x)}{\mathcal{P}(x) + P_G(x)} + P_G(x) \log \frac{P_G(x)}{\mathcal{P}(x) + P_G(x)} dx \\
&= D_{\mathrm{KL}} \left(\mathcal{P}(x) || \mathcal{P}(x) + P_G(x)\right) + D_{\mathrm{KL}} \left(P_G(x) || \mathcal{P}(x) + P_G(x)\right) \\
&= -\log 4 + D_{\mathrm{KL}} \left(\mathcal{P}(x) \middle|\middle| \frac{\mathcal{P}(x) + P_G(x)}{2}\right) + D_{\mathrm{KL}} \left(P_G(x) \middle|\middle| \frac{\mathcal{P}(x) + P_G(x)}{2}\right) \\
&= -\log 4 + 2 \underbrace{\mathrm{JSD} \left(\mathcal{P}(x) || P_G(x)\right)}_{\geq 0}
\end{aligned} \tag{4}$$

[2] This objective, to be differentiable, doesn't accommodate discrete x.

This means

$$\min_G \max_D V(G, D) = -\log 4$$

where the minimizing G and maximizing D satisfy $P_G(x) = \mathcal{P}(x)$ and $D(x) = 1/2$. Thus if the models have enough capacity, when the generator *wins* (i.e., forces the discriminator predict randomly), we have learned a fake distribution P_G that is equal to the true distribution $\mathcal{P}$.

2.2 As Variational Optimization of JSD

Forget the adversarial setup in the previous section. We now just want to model $\mathcal{P}(x)$ with a distribution $P_G(x)$ parametrized by a network G. To this end, we propose to minimize the Jensen-Shannon divergence (JSD) between $\mathcal{P}$ and P_G:

$$J(G) := 2\text{JSD}\left(\mathcal{P}(x)||P_G(x)\right)$$

It's unclear how we can optimize this. But we see in (4) that

$$J(G) = \max_D V(G, D) + \log 4$$

By plugging this in, we see that the solution is given by

$$\arg\min_G J(G) = \arg\min_G \max_D V(G, D)$$

which is the GAN objective.

But there's some subtlety here. The GAN objective is a lower bound on $J(G)$: for all D,

$$J(G) \geq V(G, D) + \log 4 \tag{5}$$

But G is *minimizing* this *lower* bound! To be clear, the derivation is correct "in the limit" (i.e., the minimax objective is always fully optimized, so that we can always swap $J(G)$ with $\max_D V(G, D)$), but it becomes nonsensical when this full optimization is not realized because the error is in the wrong direction. Some suspect this is a source of instability in training GANs.

2.3 As Variational Optimization of Mutual Information

By JSD's relation to mutual information, we can write the JSD objective $J(G)$ as $I_G(Y; X)$ where

- Y is a fair coin: $y \sim \text{Ber}(\cdot|\frac{1}{2})$.
- X_1 is the true input: $x_1 \sim \mathcal{P}(\cdot)$.
- X_2 is the fake input: $x_2 \sim P_G(\cdot)$.
- X is set to X_1 if $Y = 1$; it is set to X_2 if $Y = 0$.

Thus GAN can be viewed as solving

$$\min_G I_G(Y; X)$$

by (incorrectly) minimizing its lower bound $V(G, D)$ as before. We can rederive the lower bound from mutual information by observing that $I_G(X; Y) = H(Y) - H_G(Y|X)$ and ($\Pr(X, Y)$ denotes

the joint probability under the model):

$$\begin{aligned}
-H_G(Y|X) &= \mathop{\mathbf{E}}_{X}\left[\mathop{\mathbf{E}}_{Y|X}\left[\log \Pr(Y=y|X=x)\right]\right] \\
&\geq \mathop{\mathbf{E}}_{X}\left[\mathop{\mathbf{E}}_{Y|X}\left[\log P_D(Y=y|X=x)\right]\right] \\
&= \mathop{\mathbf{E}}_{Y}\left[\mathop{\mathbf{E}}_{X|Y}\left[\log P_D(Y=y|X=x)\right]\right] \\
&= \frac{1}{2}\mathop{\mathbf{E}}_{x\sim\mathcal{P}(\cdot)}\left[\log D(x)\right] + \frac{1}{2}\mathop{\mathbf{E}}_{x\sim P_G(\cdot)}\left[\log\left(1-D(x)\right)\right] \\
&= \frac{1}{2}V(G,D)
\end{aligned}$$

2.4 Training

An idealized training session will look like this: for each true input $x \in \mathbb{R}^d$ in the data,

- Draw $z \sim \pi(\cdot)$.
- Compute $\tilde{x} = G(z)$.
- Update the parameters of D by a gradient step on $-\log D(x) - \log\left(1 - D(\tilde{x})\right)$.
- ~~Update the parameters of G by a gradient step on $\log\left(1 - D(G(z))\right)$.~~
- Update the parameters of G by a gradient step on $-\log D(G(z))$. Fixed points are the same, but it's more numerically stable (esp. early in the training when G is poor and $D(G(z)) \approx 1$).

But in practice, training GAN is notoriously difficult. A few hacks include: normalizing the input, updating D and G with separate batches, label smoothing for D, etc. See here for details.

2.5 Information Maximizing GAN (InfoGAN)

We solve (for some regularization parameter $\lambda > 0$)

$$\min_G I_G(Y;X) - \lambda I_G(Z;G(Z))$$

in hope that we make $z \in \mathcal{Z}$ a "latent code" that's highly correlated with the output of the generator. Maximizing $I_G(Z;G(Z))$ is equivalent to maximizing

$$-H(Z|G(Z)) = \mathop{\mathbf{E}}_{z\sim\pi(\cdot)}\left[\mathop{\mathbf{E}}_{z'\sim P_G(\cdot|G(z))}\left[\log P_G(z'|G(z))\right]\right]$$

whose maximum is zero. But the posterior $P_G(z|x) = P_G(x,z)/P_G(x)$ is difficult to calculate and difficult to sample from. To bypass this difficulty, we introduce a model Q that approximates $P_G(z|x)$ with $P_Q(z|x)$ and observe

$$-H(Z|G(Z)) \geq \mathop{\mathbf{E}}_{z\sim\pi(\cdot)}\left[\mathop{\mathbf{E}}_{z'\sim P_G(\cdot|G(z))}\left[\log P_Q(z'|G(z))\right]\right]$$

To get rid of the need to sample from $P_G(\cdot|G(z))$, define $X := G(Z)$ and write

$$\begin{aligned}
\mathop{\mathbf{E}}_{z\sim\pi(\cdot)}\left[\mathop{\mathbf{E}}_{z'\sim P_G(\cdot|G(z))}\left[\log P_Q(z'|G(z))\right]\right] &= \mathop{\mathbf{E}}_{Z}\left[\mathop{\mathbf{E}}_{X|Z}\left[\mathop{\mathbf{E}}_{Z'|X}\left[\log P_Q(z'|x)\right]\right]\right] \\
&= \mathop{\mathbf{E}}_{X}\left[\mathop{\mathbf{E}}_{Z|X}\left[\mathop{\mathbf{E}}_{Z'|X}\left[\log P_Q(z'|x)\right]\right]\right] \\
&= \mathop{\mathbf{E}}_{X}\left[\mathop{\mathbf{E}}_{Z|X}\left[\log P_Q(z|x)\right]\right] \\
&= \mathop{\mathbf{E}}_{Z}\left[\mathop{\mathbf{E}}_{X|Z}\left[\log P_Q(z|x)\right]\right] \\
&= \mathop{\mathbf{E}}_{z\sim\pi(\cdot)}\left[\log P_Q(z|G(z))\right]
\end{aligned}$$

In summary, InfoGAN uses an upper bound obtained by introducing Q,

$$-I(Z;G(Z)) \le \mathop{\mathbf{E}}_{z\sim\pi(\cdot)}\left[\log P_Q(z|G(z))\right] - H(Z) \tag{6}$$

to optimize $I_G(Y;X) - \lambda I_G(Z;G(Z))$ as

$$\min_G \max_{D,Q} V(G,D) + \lambda \mathop{\mathbf{E}}_{z\sim\pi(\cdot)}\left[\log P_Q(z|G(z))\right]$$

Note that InfoGAN minimizes a lower bound on $I_G(X;Y)$ (wrong direction) and minimizes an upper bound on $-\lambda I_G(Z;G(Z))$ (correct direction).

2.5.1 Training

Here's again an idealized training session: for each true input $x \in \mathbb{R}^d$ in the data,

- Draw $z \sim \pi(\cdot)$.
- Compute $\tilde{x} = G(z)$.
- Update the parameters of D by a gradient step on $-\log D(x) - \log(1 - D(\tilde{x}))$.
- Update the parameters of G by a gradient step on $-\log D(G(z)) - \log P_Q(z|G(z))$.
- Update the parameters of Q by a gradient step on $\log P_Q(z|\tilde{x})$.

Q and D are both mappings from $\mathbb{R}^d$ and their base parameters are usually tied. For example, if $\mathcal{Z}$ is discrete, Q maps x to a vector of length $|\mathcal{Z}|$ and then applies softmax. If $\mathcal{Z}$ is continuous, Q maps x to a vector of Gaussian parameters.

3 The Information Bottleneck Method

Tishby et al. (2000) consider a compression problem: encode a signal $x \in X$ into a "codeword" $z \in Z$. The goal is to learn $p(z|x)$ by optimizing some measure of compression quality. The first measure is "information rate", the number of bits needed to specify z. Thus minimizing $I(Z;X)$ is considered, but this has a trivial solution of making z unrelated to x.

Rather, they propose to introduce another variable $y \in Y$ with joint distribution $p(x,y) = p(y)p(x|y)$. Now the second measure of compression quality is to make z a minimal sufficient statistics of x with respect to y:

$$\max_{p(z|x)} I(Y;Z) - I(X;Z)$$

where the first term makes z informative of y and the second term (the information rate) makes z concise. Note that this is with respect to the joint distribution $p(x, y, z) = p(y)p(x|y)p(z|x)$ and the Markov chain $Y \to X \to Z$. As the authors remark, we squeeze the information that X provides about Y through a "bottleneck" formed by a limited set of codewords Z. They also characterize an optimal coding rule $p(z|x)$ as a function of $p(z)$ and the KL divergence between $p(y|x)$ and $p(y|z)$.

Tishby and Zaslavsky (2015) apply this to deep learning as follows. Y is the the target label, X is the input, and Z is the hidden state. Under this framework, the goal of deep learning is viewed as transforming the input into a concise representation that's maximally predictive of the output. This is further motivated by the observation that the original input is usually inadequate for correct classification under linear models (Appendix C) While the framework gives a nice way to think about the generalization/compression quality of neural networks, a shortcoming of this work is the lack of a practical algorithm for the compression problem. This is addressed by the next section.

4 Information Theoretic Co-Training

This work can be motivated in multiple ways. (1) It's in a similar spirit with the classical co-training where we have two views (X, Y) and try to identify whatever Z they agree on. (2) It can be seen as an effort to measure mutual information between arbitrary random variables (X, Y), which is notoriously difficult. (3) It's also a particular probabilistic formulation to compress the joint information content of (X, Y).

Let's use (2). We have random variables (X, Y) with some unknown joint density $\mathcal{P}(X, Y)$ (with corresponding conditional/marginalized densities $\mathcal{P}(X)$, $\mathcal{P}(Y)$, $\mathcal{P}(Y|Z)$, etc.). Assuming that all we can do is getting iid samples from $\mathcal{P}(X, Y)$, how can we measure the mutual information $I(Y; X) = H(Y) - H(Y|X)$? This is not computable since we can't calculate $\mathcal{P}$. One approach is to introduce a model Ψ that approximates $\mathcal{P}(Y|X)$ as $P_\Psi(Y|X)$, optimize the cross-entropy loss H_Ψ with and without conditioning on X, and take $I(Y; X) \approx H_\Psi(Y) - H_\Psi(Y|X)$. If Ψ is expressive enough to achieve $P_\Psi(Y|X) = \mathcal{P}(Y|X)$, then given infinitely many samples we will have $I(Y; X)$.

But an argument against this is that directly modeling Y this way is numerically unstable. Also, it doesn't result in any compression (3); assume we're interested in this. Thus we consider the following approach: we introduce a latent variable Z and model $P_\Psi(Z|Y)$ instead (the choice of Y is arbitrary). This defines a joint density over (X, Y, Z) as $\mathcal{P}_\Psi(X, Y, Z) = \mathcal{P}(X, Y)P_\Psi(Z|Y)$. We note that by data processing inequality

$$I(Y; X) \geq I_\Psi(Z; X)$$

Thus we now try to maximize $I_\Psi(Z; X) = H_\Psi(Z) - H_\Psi(Z|X)$.[3] In particular, it subsumes the first approach if we set $Z = Y$.

Now we just need to be able to measure $H_\Psi(Z)$ and $H_\Psi(Z|X)$. For simplicity we assume $z \in \mathcal{Z}$ is discrete. Given n iid samples $(x_1, y_1) \dots (x_n, y_n) \sim \mathcal{P}(X, Y)$, an unbiased estimate of $H_\Psi(Z)$

[3]Note here that if we choose to maximize $I_\Psi(Z; Y)$, the objective will "discard" X and reduce to autoencoding on Y.

is given by

$$\widehat{H}_\Psi(Z) := -\sum_{z\in\mathcal{Z}} \widehat{P}_\Psi(Z)\log\widehat{P}_\Psi(Z)$$

where

$$\widehat{P}_\Psi(Z) := \frac{1}{n}\sum_{i=1}^{n} P_\Psi(z|y_i)$$

The other term $H_\Psi(Z|X)$ is more difficult because for a given value of x we need to integrate over y to compute $P_\Psi(z|x) = (\int_y \mathcal{P}_\Psi(x,y,z))/(\int_y \sum_z \mathcal{P}_\Psi(x,y,z))$. So we introduce a **second encoder** Φ that encodes z from x (not y!), and approximate $P_\Phi(z|x) \approx P_\Psi(z|x)$. As usual in variational optimization, since

$$\begin{aligned} H_\Psi(Z|X) &:= \mathop{\mathbf{E}}_{\substack{(x,y)\sim\mathcal{P}(\cdot)\\ z\sim P_\Psi(\cdot|y)}}[-\log P_\Psi(z|x)] \\ &= \underbrace{\mathop{\mathbf{E}}_{\substack{(x,y)\sim\mathcal{P}(\cdot)\\ z\sim P_\Psi(\cdot|y)}}[-\log P_\Phi(z|x)]}_{H_{\Psi,\Phi}(Z|X)} - \underbrace{D_{\mathrm{KL}}\left(P_\Psi(z|x)||P_\Phi(z|x)\right)}_{\geq 0} \end{aligned}$$

we now maximize $H_{\Psi,\Phi}(Z|X)$ instead. This can be estimated from samples as

$$\widehat{H}_{\Psi,\Phi}(Z|X) := -\frac{1}{n}\sum_{i=1}^{n}\sum_{z\in\mathcal{Z}} P_\Psi(z|y_i)\log P_\Phi(z|x_i)$$

In summary, the objective is

$$\max_{\Psi,\Phi} H_\Psi(Z) - H_{\Psi,\Phi}(Z|X)$$

One way to interpret is that Ψ tries to predict Z from Y, Φ tries to predict Z from X, and they collaborate until they agree. In particular, if $\Psi^* = \arg\max_\Psi H_\Psi(Z) - \min_\Phi H_{\Psi,\Phi}(Z|X)$, then an optimal Φ is obtained by just minimizing $H_{\Psi^*,\Phi}(Z|X)$ which is the usual cross-entropy loss where supervision on Z is given by Ψ^*. The training algorithm is as follows.

InfoCotrain
Input: $S_n = \{(x_1, y_1) \dots (x_n, y_n)\}$ drawn iid from $\mathcal{P}$, latent (discrete) space $\mathcal{Z}$, int E
Output: $P_\Psi(z|y)$ and $P_\Phi(z|x)$ that try to maximize a lower bound on $I(Y; X)$

1. Initialize Ψ and Φ.
2. For $e = 1 \dots E$,
   (a) For each $z \in \mathcal{Z}$, estimate

$$\widehat{P}_\Psi(z) \leftarrow \frac{1}{n} \sum_{i=1}^{n} P_\Psi(z|y_i)$$

   (b) Set

$$\widehat{H}_\Psi(Z) \leftarrow -\sum_{z \in \mathcal{Z}} \widehat{P}_\Psi(z) \log \widehat{P}_\Psi(z)$$

   (c) Set

$$\widehat{H}_{\Psi,\Phi}(Z|X) \leftarrow -\frac{1}{n} \sum_{i=1}^{n} \sum_{z \in \mathcal{Z}} P_\Psi(z|y_i) \log P_\Phi(z|x_i)$$

   (d) Take gradient steps

$$\Psi \leftarrow \Psi - \eta_e \nabla_\Psi \left\{ \widehat{H}_{\Psi,\Phi}(Z|X) - \widehat{H}_\Psi(Z) \right\}$$
$$\Phi \leftarrow \Phi - \eta_e \nabla_\Phi \left\{ \widehat{H}_{\Psi,\Phi}(Z|X) \right\}$$

References

- Extracting and Composing Robust Features with Denoising Autoencoders (Vincent et al., 2008)
- Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Srivastava et al., 2014)
- Auto-Encoding Variational Bayes (Kingma and Welling, 2013)
- A Recurrent Latent Variable Model for Sequential Data (Chung et al., 2016)
- Generative Adversarial Nets (Goodfellow et al., 2014)
- The information bottleneck method (Tishby et al., 2000)
- Deep Learning and the Information Bottleneck Principle (Tishby and Zaslavsky, 2015)
- Information Theoretic Co-Training (McAllester, 2017)

A The Latent Variable Paradigm

The data consists of $x \in \mathcal{X}$. We define a model Φ that computes $P_\Phi(x, z)$ where $z \in \mathcal{Z}$ is unobserved. We often assume model structure $P_\Phi(x, z) := P_\Phi(z)P_\Phi(x|z)$. z makes the generative story for x more natural, leading to a model that's both better fit and more interpretable. $\mathcal{X}$ and $\mathcal{Z}$ can be anything: discrete or continuous or a mix of both.

A.1 Example Models

Model	$\mathcal{X}$	$\mathcal{Z}$	Φ	$P_\Phi(z)$	$P_\Phi(x\|z)$
GMM	$\mathbb{R}^d$	$[m]$	$\pi, \{\mu_z, \Sigma_z\}_{z=1}^m$	$\pi(z)$	$\mathcal{N}(x\|\mu_z, \Sigma_z)$
HMM	$[n]^{\mathbb{N}}$	$[m]^{\mathbb{N}}$	t, o	$\prod_i t(z_i\|z_{i-1})$	$\prod_{i=1}^N o(x_i\|z_i)$
PCFG	$[n]^{\mathbb{N}}$	tree	t, o	$\prod_{r\in z} t(r)$	$\prod_{i=1}^N o(x_i\|z_{\pi(x_i)})$
VAE	$\{0,1\}^d$	$\mathbb{R}^{d'}$	$\mathbb{R}^{d'} \mapsto [0,1]^d$	$\mathcal{N}(z\|0, I_{d'})$	$\prod_{i=1}^d \Phi_i(z)^{x_i}(1-\Phi_i(z))^{(1-x_i)}$

A.2 Applications

	unsupervised learning	**generation**	**supervised learning**
data	x	x	(x, y)
goal	understand x	generate new x'	learn $x \mapsto y$
model	$P_\Phi(x, z)$	$p(z)P_\Phi(x\|z)$	$P_\Phi(x, y, z)$
role of z	description of x	seed	refinement on (x, y)
ex. x	document	face image	speech frames
ex. y	—	—	transcription of x
ex. z	topics	seed vector	boundaries in x
inference	$\arg\max_z P_\Phi(z\|x)$	$z \sim p(\cdot)$ $x' \sim P_\Phi(\cdot\|z)$	$\arg\max_y P_\Phi(y\|x)$ or $\arg\max_y \max_z P_\Phi(y, z\|x)$

A.3 Learning

The best estimator is usually given by MLE. If we observe z, we can compute

$$\Phi^{\text{SUP}} := \arg\max_\Phi \log P_\Phi(x, z) \tag{7}$$

Since we don't observe z, we have to compute

$$\Phi^{\text{MLE}} := \arg\max_\Phi \log P_\Phi(x) \tag{8}$$

This problem is (obviously) more difficult. The usual rhetoric is that 1. (7) is concave and has a closed-form solution for a wide class of distributions $P_\Phi(x, z)$ (e.g., exponential families) whereas (8) is non-concave with no such nice solution, 2. we can do gradient ascent, but gradients are complicated and we must suffer local optima. This line of argument is now questionable, because progress in automatic differentiation and local search makes it totally possible for direct gradient ascent on (8) to be effective; we can do this as long as we can estimate $\log P_\Phi(x)$ reasonably well.

But this is the motivation for EM.

A.3.1 Derivation of EM

Φ wants to maximize $\log P_\Phi(x)$, which is nontrivial in many cases. **However**, Φ can often trivially maximize $\log P_\Phi(x, z)$.[4] What if a friend Ψ told him what z should look like by $P_\Psi(z|x)$? Then again Φ can trivially maximize the **expected value** of $\log P_\Phi(x, z)$ under Ψ. If the friend is an oracle and gives him the best z (that is, best under Φ^{MLE}), we are done!

Of course, Φ can't expect an oracle friend. In fact, in the absence of any other information, the best Ψ can do is based on what Φ knows, so he must propose $P_\Psi(z|x) = P_\Phi(z|x)$. That's okay, because

$$\Phi_2 = \arg\max_\Phi \mathop{\mathbf{E}}_{z\sim P_{\Phi_1}(z|x)} [\log P_\Phi(x, z)]$$

is not necessarily the same as Φ_1. If Φ changes, then the friend can make a new proposal based on the changed Φ. In this way, Φ and Ψ continue to cooperate until they agree. This is the heart of EM.

To make this argument formal, we rewrite $\log P_\Phi(x)$ as

$$\begin{aligned}
\log P_\Phi(x) &= \mathop{\mathbf{E}}_{z\sim P_\Psi(\cdot|x)} [\log P_\Phi(x)] \\
&= \mathop{\mathbf{E}}_{z\sim P_\Psi(\cdot|x)} \left[\log \frac{P_\Phi(z|x)P_\Phi(x)}{P_\Phi(z|x)}\right] \\
&= \mathop{\mathbf{E}}_{z\sim P_\Psi(\cdot|x)} [\log P_\Phi(x, z)] - \mathop{\mathbf{E}}_{z\sim P_\Psi(\cdot|x)} [\log P_\Phi(z|x)] \\
&= \underbrace{\mathop{\mathbf{E}}_{z\sim P_\Psi(\cdot|x)} [\log P_\Phi(x, z)] - \mathop{\mathbf{E}}_{z\sim P_\Psi(\cdot|x)} [\log P_\Psi(z|x)]}_{\text{ELBO}(\Phi,\Psi)} + \underbrace{D_{\text{KL}}\left(P_\Psi(z|x)||P_\Phi(z|x)\right)}_{\geq 0}
\end{aligned}$$

where we assume $P_\Phi(z|x) > 0$ for all $z \in \mathcal{Z}$ for convenience in the second equality. Here, among other things, $\text{ELBO}(\Phi, \Psi)$ is a lower bound on $\log P_\Phi(x)$. It's tight iff $P_\Psi(z|x) = P_\Phi(z|x)$. EM is an alternating maximization of ELBO, that is:

$$\begin{aligned}
\Psi^{(t+1)} &= \arg\max_\Psi \text{ ELBO}(\Phi^{(t)}, \Psi) \\
\Phi^{(t+1)} &= \arg\max_\Phi \text{ ELBO}(\Phi, \Psi^{(t+1)})
\end{aligned}$$

This view makes the idea of cooperation between Φ and Ψ precise; the burden on one becomes lighter given the other's feedback. It's easy to see that the solution is given by

$$\Psi^{(t+1)} \in \{\Psi : \ P_\Psi(z|x) = P_{\Phi^{(t)}}(z|x)\} \tag{9}$$

$$\Phi^{(t+1)} = \arg\max_\Phi \mathop{\mathbf{E}}_{z\sim P_{\Psi^{(t+1)}}(\cdot|x)} [\log P_\Phi(x, z)] \tag{10}$$

The upshot is that it's easy to compute $\Phi^{(t+1)}$. It's also easy to see that this iteration always increases the original objective $\log P_\Phi(x)$ unless we are at a stationary point of ELBO, since

- Step (9) not only increases ELBO but makes it **tight**:

$$\text{ELBO}(\Phi^{(t)}, \Psi^{(t+1)}) = \log P_{\Phi^{(t)}}(x)$$

[4] The usual motivation is that if $P_\Phi(x, z)$ is an exponential family, MLE is given in closed-form by its sufficient statistics (in the fully observed setting).

- Step (10), then, can only increase the original objective:

$$\max_{\Phi} \text{ ELBO}(\Phi, \Psi^{(t+1)}) \geq \text{ELBO}(\Phi^{(t)}, \Psi^{(t+1)}) = \log P_{\Phi^{(t)}}(x)$$

In many scenarios, we don't even bother explicitly introducing Ψ. Then EM is just

$$\Phi^{(t+1)} = \arg\max_{\Phi} \mathop{\mathbf{E}}_{z \sim P_{\Phi^{(t)}}(\cdot|x)} [\log P_{\Phi}(x, z)]$$

This justifies the name of the technique: we iteratively **maximize** the **expectation**. This form also most closely mirrors the mechanics of EM in many settings. For instance, to estimate an HMM with EM, we (1) compute expected counts under the current parameter values (with the forward-backward algorithm), and (2) maximize the expected $\log P_{\Phi}(x, z)$ (i.e., MLE on the expected counts).

What makes ELBO so mysterious is that it can be expressed in multiple equivalent forms, and each of these forms justifies a particular step in an algorithm.

- **Lower bound on marginalized log likelihood.**

$$\text{ELBO}(\Phi, \Psi) = \log P_{\Phi}(x) - D_{\text{KL}}\left(P_{\Psi}(z|x) || P_{\Phi}(z|x)\right)$$

  This form was used to justify Step (9): with Φ held constant, it's tightly maximized to $\log P_{\Phi}(x)$ by Ψ satisfying $P_{\Psi}(z|x) = P_{\Phi}(z|x)$.

- **Expected log likelihood with entropy regularization on $P_{\Psi}(z|x)$.**

$$\text{ELBO}(\Phi, \Psi) = \mathop{\mathbf{E}}_{z \sim P_{\Psi}(z|x)} [\log P_{\Phi}(x, z)] - \mathop{\mathbf{E}}_{z \sim P_{\Psi}(z|x)} [\log P_{\Psi}(z|x)]$$

  This form was used to justify Step (10): with Ψ held constant, it's maximized by Φ that maximizes the first term (an easy problem).

- **Autoencoder with KL regularization between $P_{\Psi}(z|x)$ and $P_{\Phi}(z)$.**

$$\text{ELBO}(\Phi, \Psi) = \mathop{\mathbf{E}}_{z \sim P_{\Psi}(z|x)} [\log P_{\Phi}(x|z)] - D_{\text{KL}}\left(P_{\Psi}(z|x) || P_{\Phi}(z)\right)$$

  This form can be easily verified by manipulating the form immediately above. This is the training objective of VAE.

B Fubini's Theorem

Let X and Y be any spaces equipped with probability measures. Fubini's theorem states that for all functions $f : X \times Y \to \mathbb{R}$ satisfying $\int_{X \times Y} |f(x, y)| \, d(x, y) < \infty$,

$$\int_{X \times Y} f(x, y) d(x, y) = \int_X \left(\int_Y f(x, y) dy \right) dx = \int_Y \left(\int_X f(x, y) dx \right) dy$$

An important application of Fubini is changing the order of expectation:

$$\begin{aligned}
\mathop{\mathbf{E}}_{X}\left[\mathop{\mathbf{E}}_{Y|X}\left[f(x,y)\right]\right] &= \int_X \Pr(X=x)\left(\int_Y \Pr(Y=y|X=x)f(x,y)dy\right)dx \\
&= \int_X \left(\int_Y \Pr(X=x,Y=y)f(x,y)dy\right)dx \\
&= \int_{X\times Y} \Pr(X=x,Y=y)f(x,y)d(x,y) \qquad (11) \\
&= \int_{X\times Y} \Pr(Y=y)\Pr(X=x|Y=y)f(x,y)d(x,y) \\
&= \int_Y \Pr(Y=y)\left(\int_X \Pr(X=x|Y=y)f(x,y)dx\right)dy \qquad (12)
\end{aligned}$$

where steps (11) and (12) are by Fubini and give us

$$\mathop{\mathbf{E}}_{X\times Y}\left[f(x,y)\right] = \mathop{\mathbf{E}}_{X}\left[\mathop{\mathbf{E}}_{Y|X}\left[f(x,y)\right]\right] = \mathop{\mathbf{E}}_{Y}\left[\mathop{\mathbf{E}}_{X|Y}\left[f(x,y)\right]\right]$$

C Linear Model for Classification

An input is a feature vector $X \in \mathbb{R}^d$ and an output is $Y \in \{0,1\}$. There is some true distribution $\Pr(X,Y)$. Given $x \in R^d$, a logistic regressor models its **log odd** as

$$\log\frac{\Pr(Y=1|X=x)}{\Pr(Y=0|X=x)} = \underbrace{\log\frac{\Pr(X=x|Y=1)}{\Pr(X=x|Y=0)}}_{w^\top x} + \underbrace{\log\frac{\Pr(Y=1)}{\Pr(Y=0)}}_{b}$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the parameters of the model. Thus the regressor can express the distribution when the first term can be written as $w^\top x$. This is possible iff

1. Features $x_1 \dots x_d \in \mathbb{R}$ are conditionally independent given the label.
2. The log ratio of $p(x_i|1)$ and $p(x_i|0)$ is proportional to $x_i \in \mathbb{R}$.

Then

$$\log\frac{\Pr(X=x|Y=1)}{\Pr(X=x|Y=0)} = \sum_{i=1}^{d}\log\frac{\Pr(X_i=x_i|Y=1)}{\Pr(X_i=x_i|Y=0)} = \sum_{i=1}^{d} w_i x_i$$

For instance, if $\Pr(Y=1)=\pi$ and $\Pr(X=x|Y=y) = \mathcal{N}(\mu^y, \mathrm{diag}(\sigma^2))$ for each $y \in \{0,1\}$, both requirements are satisfied and the parameters are given in closed form. But in general, we can't expect the representation X to be "disentagled" in such an idealized way.