

Modelos Generativos Profundos para Imágenes

Autoencoders variacionales (parte 2)

Pablo Musé

pmuse@fing.edu.uy

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Agenda

① Repaso

② Aprendizaje variacional

Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

① Repaso

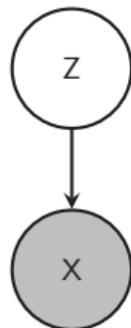
② Aprendizaje variacional

Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

Modelos en variables latentes: definición



Un modelo en variables latentes define una densidad de probabilidad

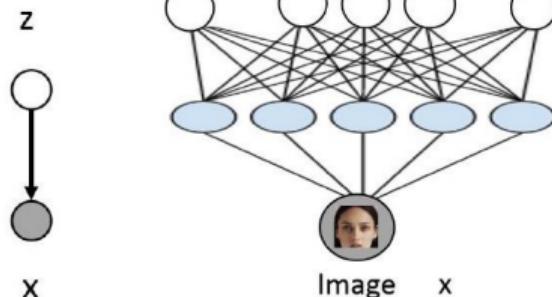
$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- Variables observadas **X**: los objetos de alta dimensión que queremos modelar, y que están en nuestro conjunto de entrenamiento.
- Variables latentes **Z**: no se encuentran en el conjunto de datos, pero están asociadas a **X** según lo especificado por $p(\mathbf{x}, \mathbf{z})$.

Modelos en variables latentes profundos

Se utilizan redes neuronales para modelar las condicionales:

- ① $\mathbf{z} \sim \mathcal{N}(0, I)$
- ② $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$, $\mu_\theta, \Sigma_\theta$ son redes neuronales.



Aunque $p(\mathbf{x}|\mathbf{z})$ es simple, la marginal $p(\mathbf{x})$ es muy compleja/flexible.

Aprendizaje no supervisado de representaciones:

- Una vez entrenado, \mathbf{Z} codifica factores latentes de variación significativos (*features*).
- Como antes, los *features* se pueden calcular a través de $p(\mathbf{z}|\mathbf{x})$.
- La V.A. continua \mathbf{Z} da una parametrización suave de \mathbf{X} . E.g.: caras \mathbf{x} similares (misma edad, género, etc.) tienen \mathbf{z} similares.
⇒ Podemos hacer aritmética en \mathbf{z} : e.g., dados \mathbf{x}_0 y \mathbf{x}_1 , inferimos \mathbf{z}_0 y \mathbf{z}_1 usando $p(\mathbf{z}|\mathbf{x}_i)$, interpolamos $\mathbf{z}_\lambda = \lambda \mathbf{z}_1 + (1 - \lambda) \mathbf{z}_0$, y generamos $\mathbf{x}_\lambda \sim p(\mathbf{x}|\mathbf{z}_\lambda)$.

Recap

- Los modelos de variables latentes nos permiten definir modelos complejos $p(\mathbf{x})$ en términos de bloques simples $p(\mathbf{x}|\mathbf{z})$
- Son naturales para tareas de aprendizaje no supervisado (clustering, aprendizaje de representaciones, etc.)
- Más difícil de aprender comparado con modelos totalmente observados, porque $p(\mathbf{x})$ es difícil de evaluar (y optimizar)

Inferencia variacional

Vimos que para cualquier dato \mathbf{x} , tenemos:

$$\log p(\mathbf{x}; \theta) = \text{ELBO}(\theta, q) + D_{KL}(q(\mathbf{z}) || p(\mathbf{z}|\mathbf{x}; \theta)) \geq \text{ELBO}(\theta, q).$$

La inferencia variacional produce una aproximación de $\log p(\mathbf{x}; \theta)$ encontrando una $q(\mathbf{z})$ que haga que $\text{ELBO}(q)$ sea ajustado:

- ① Elegimos $q(\mathbf{z}; \phi)$ para que sea una distribución de probabilidad (manejable) sobre \mathbf{z} parametrizada por ϕ (parámetros variacionales).
- ② Encontramos un buen ϕ^* maximizando el ELBO y haciendo la cota lo más ajustada:

$$\log p(\mathbf{x}; \theta) \geq \max_{\phi} \text{ELBO}(\theta, \phi)$$

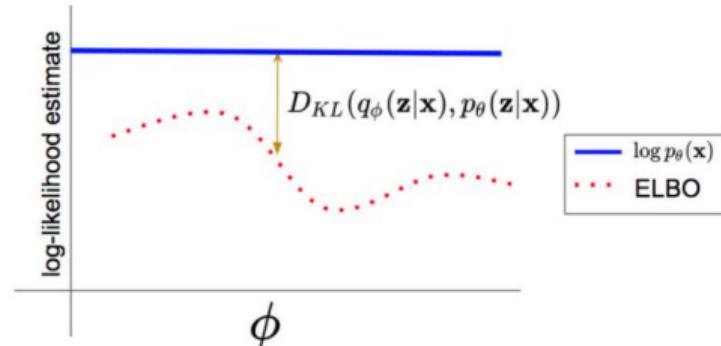
Teniendo un buen $\text{ELBO}(\theta, \phi^*)$ que se ajusta a $\log p(\mathbf{x}; \theta)$, podemos optimizar θ .

La cota inferior de evidencia

$$\underbrace{\mathcal{L}(\mathbf{x}; \theta, \phi)}_{\text{ELBO}} := \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi))$$

$$\log p(\mathbf{x}; \theta) = \mathcal{L}(\mathbf{x}; \theta, \phi) + D_{KL}(q(\mathbf{z}; \phi) \| p(\mathbf{z}|\mathbf{x}; \theta))$$

$$\log p(\mathbf{x}; \theta) \geq \mathcal{L}(\mathbf{x}; \theta, \phi)$$



Como vimos, la q que realiza la igualdad es $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}; \theta)$, pero no podemos calcularlo así porque en general $p(\mathbf{z}|\mathbf{x}; \theta)$ no es calculable.

- Cuanto mejor $q(\mathbf{z}; \phi)$ aproxime el *posterior* $p(\mathbf{z}|\mathbf{x}; \theta)$:
 - Menor será $D_{KL}(q(\mathbf{z}; \phi) \| p(\mathbf{z}|\mathbf{x}; \theta))$,
 - Más cerca estará la ELBO de $\log p(\mathbf{x}; \theta)$.

① Repaso

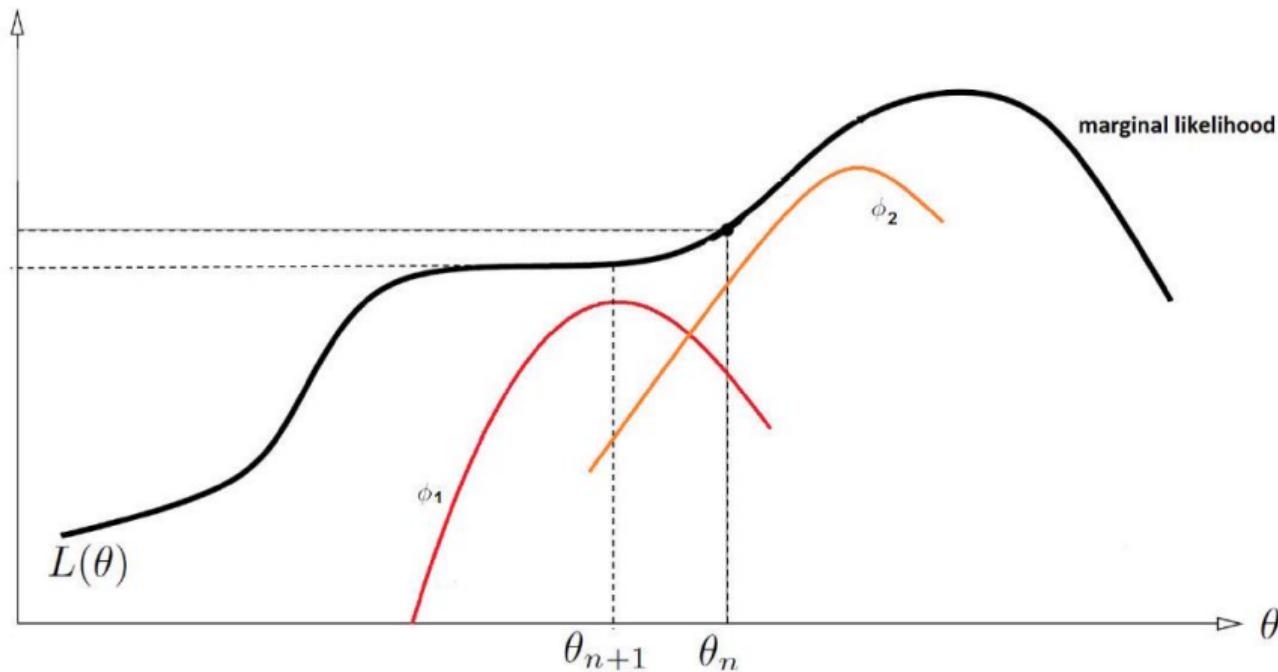
② Aprendizaje variacional

Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

Aprendizaje variacional



$\mathcal{L}(\mathbf{x}; \theta, \phi_1)$ y $\mathcal{L}(\mathbf{x}; \theta, \phi_2)$ son ambas cotas inferiores. Queremos optimizar conjuntamente θ y ϕ

ELBO aplicada a todo el conjunto de datos

- La cota inferior de la evidencia (ELBO) se cumple para cualquier $q(\mathbf{z}; \phi)$:

$$\log p(\mathbf{x}; \theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi)) = \underbrace{\mathcal{L}(\mathbf{x}; \theta, \phi)}_{\text{ELBO}}$$

- Aprendizaje por máxima verosimilitud (sobre todo el conjunto de datos):

$$\begin{aligned}\ell(\theta; \mathcal{D}) &= \sum_{\mathbf{x}_n \in \mathcal{D}} \log p(\mathbf{x}_n; \theta) \geq \sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n) \\ \implies \max_{\theta} \ell(\theta; \mathcal{D}) &\geq \max_{\theta, \phi_1, \dots, \phi_N} \sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n).\end{aligned}$$

- **Obs.:** usamos diferentes *parámetros variacionales* ϕ_n para cada muestra \mathbf{x}_n , porque el verdadero posterior $p(\mathbf{z}|\mathbf{x}_n; \theta)$ es diferente entre las \mathbf{x}_n .

① Repaso

② Aprendizaje variacional

Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

Aprendizaje mediante inferencia variacional estocástica (SVI)

Objetivo: optimizar $\sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n)$ como una función de $\theta, \phi_1, \dots, \phi_N$ usando descenso por gradiente (estocástico) → minimizar $-\sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n)$

$$\begin{aligned}\mathcal{L}(\mathbf{x}_n; \theta, \phi_n) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi_n) \log p(\mathbf{z}, \mathbf{x}_n; \theta) + H(q(\mathbf{z}; \phi_n)) \\ &= \mathbb{E}_{q(\mathbf{z}; \phi_n)} [\log p(\mathbf{z}, \mathbf{x}_n; \theta) - \log q(\mathbf{z}; \phi_n)]\end{aligned}$$

- ① Inicializar $\theta, \phi_1, \dots, \phi_N$
 - ② Muestrear aleatoriamente un \mathbf{x}_n de \mathcal{D}
 - ③ Optimizar $\mathcal{L}(\mathbf{x}_n; \theta, \phi_n)$ como una función de ϕ_n :
 - ① Repetir $\phi_n = \phi_n + \eta \nabla_{\phi_n} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n)$
 - ② hasta la convergencia a $\phi_n^* \approx \operatorname{argmax}_{\phi} \mathcal{L}(\mathbf{x}_n; \theta, \phi)$
 - ④ Calcular $\nabla_{\theta} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n^*)$
 - ⑤ Actualizar θ en la dirección del gradiente. Ir al paso 2.
- ¿Cómo calcular los gradientes?
En general no hay forma cerrada para las esperanzas ⇒ usamos muestreo de Monte Carlo

Aprendizaje de modelos generativos profundos

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi)) \\ &= \mathbb{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)]\end{aligned}$$

- **Nota:** eliminamos el subíndice n de \mathbf{x}_n y ϕ_n por simplicidad.
- Para evaluar la cota, muestreamos $\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}; \phi)$ y estimamos

$$\mathbb{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)] \approx \frac{1}{K} \sum_k (\log p(\mathbf{z}_k, \mathbf{x}; \theta) - \log q(\mathbf{z}_k; \phi)).$$

- **Supuesto clave:** $q(\mathbf{z}; \phi)$ es tratable, i.e., fácil de muestrear y evaluar.
- Ahora queremos calcular $\nabla_{\theta} \mathcal{L}(\mathbf{x}; \theta, \phi)$ y $\nabla_{\phi} \mathcal{L}(\mathbf{x}; \theta, \phi)$

Aprendizaje de modelos generativos profundos

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi)) \\ &= \mathbb{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)]\end{aligned}$$

Queremos calcular $\nabla_{\theta} \mathcal{L}(\mathbf{x}; \theta, \phi)$ y $\nabla_{\phi} \mathcal{L}(\mathbf{x}; \theta, \phi)$

- ① El gradiente con respecto a θ es fácil:

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)] &= \mathbb{E}_{q(\mathbf{z}; \phi)} [\nabla_{\theta} \log p(\mathbf{z}, \mathbf{x}; \theta)] \\ &\approx \frac{1}{K} \sum_k \nabla_{\theta} \log p(\mathbf{z}^k, \mathbf{x}; \theta)\end{aligned}$$

- ② El gradiente con respecto a ϕ es más complicado porque la esperanza depende de ϕ
⇒ Usaremos el truco de la reparametrización.

① Repaso

② Aprendizaje variacional

Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

Truco de la reparametrización

Queremos calcular el gradiente con respecto a ϕ de: $\mathbb{E}_{q(\mathbf{z}; \phi)}[r(\mathbf{z})] = \int q(\mathbf{z}; \phi)r(\mathbf{z})d\mathbf{z}$.

- Sea $q(\mathbf{z}; \phi) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \sigma^2 I)$, parámetros $\phi = (\boldsymbol{\mu}, \sigma)$. Muestreamos $\mathbf{z} \sim q(\mathbf{z}; \phi)$ como:

- ① Muestrear $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$
- ② $\mathbf{z} = \boldsymbol{\mu} + \sigma \boldsymbol{\varepsilon} = g(\boldsymbol{\varepsilon}; \phi)$. g es determinista!

- Luego,

$$\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}; \phi)}[r(\mathbf{z})] = \int q(\mathbf{z}; \phi)r(\mathbf{z})d\mathbf{z} = \mathbb{E}_{\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)}[r(g(\boldsymbol{\varepsilon}; \phi))] = \int \mathcal{N}(\boldsymbol{\varepsilon})r(\boldsymbol{\mu} + \sigma \boldsymbol{\varepsilon})d\boldsymbol{\varepsilon}$$

$$\nabla_{\phi} \mathbb{E}_{q(\mathbf{z}; \phi)}[r(\mathbf{z})] = \nabla_{\phi} \mathbb{E}_{\boldsymbol{\varepsilon}}[r(g(\boldsymbol{\varepsilon}; \phi))] = \mathbb{E}_{\boldsymbol{\varepsilon}} [\nabla_{\phi} r(g(\boldsymbol{\varepsilon}; \phi))] \quad (\text{intercambiamos } \nabla_{\phi} \text{ y } \mathbb{E}_{\boldsymbol{\varepsilon}})$$

- Estimación MC fácil si r y g son diferenciables con respecto a ϕ , y $\boldsymbol{\varepsilon}$ es fácil de muestrear:

$$\mathbb{E}_{\boldsymbol{\varepsilon}} [\nabla_{\phi} r(g(\boldsymbol{\varepsilon}; \phi))] \approx \frac{1}{K} \sum_k \nabla_{\phi} r(g(\boldsymbol{\varepsilon}_k; \phi)), \text{ donde } \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_K \sim \mathcal{N}(0, I).$$

Aprendizaje de Modelos Generativos Profundos

Volviendo a nuestro problema de maximizar

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi)) \\ &= \mathbb{E}_{q(\mathbf{z}; \phi)} \underbrace{[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)]}_{r(\mathbf{z}, \phi)}\end{aligned}$$

⇒ Caso ligeramente más complicado: tenemos $\mathbb{E}_{q(\mathbf{z}; \phi)}[r(\mathbf{z}, \phi)]$ en lugar de $\mathbb{E}_{q(\mathbf{z}; \phi)}[r(\mathbf{z})]$

- Aún se puede usar reparametrización, con $\mathbf{z} = \boldsymbol{\mu} + \sigma \boldsymbol{\varepsilon} = g(\boldsymbol{\varepsilon}; \phi)$ como antes:

$$\mathbb{E}_{q(\mathbf{z}; \phi)}[r(\mathbf{z}, \phi)] = \mathbb{E}_{\boldsymbol{\varepsilon}}[r(g(\boldsymbol{\varepsilon}; \phi), \phi)] \approx \frac{1}{K} \sum_k r(g(\boldsymbol{\varepsilon}_k; \phi), \phi)$$

$$\mathbb{E}_{\boldsymbol{\varepsilon}} [\nabla_{\phi} r(g(\boldsymbol{\varepsilon}; \phi), \phi)] \approx \frac{1}{K} \sum_k \nabla_{\phi} r(g(\boldsymbol{\varepsilon}_k; \phi), \phi).$$

- El gradiente se calcula usando la regla de la cadena (autograd).

① Repaso

② Aprendizaje variacional

Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

Inferencia amortizada

$$\max_{\theta} \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi_1, \dots, \phi_N} \sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi_n)$$

- Hasta ahora usamos **un conjunto de parámetros variacionales ϕ_n para cada muestra \mathbf{x}_n**
⇒ No escala a grandes volúmenes datos.
- **Idea: Amortización**
 - Se aprende **una única función paramétrica f_λ** que mapea cada \mathbf{x} a un conjunto de (buenos) parámetros variacionales.
 - Como hacer regresión sobre $\mathbf{x}_n \mapsto \phi_n^*$
 - Por ejemplo, si para $n = 1, \dots, N$ tenemos $q(\mathbf{z}|\mathbf{x}_n) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_n, I)$, i.e., $\mathbf{x}_n \mapsto \boldsymbol{\mu}_n$, entrenamos una **única** red neuronal f_λ que mapee cada \mathbf{x}_n a $\boldsymbol{\mu}_n$.
 - En la literatura, para $q(\mathbf{z}; f_\lambda(\mathbf{x}))$ se usa en general la notación $q_\phi(\mathbf{z}|\mathbf{x})$ y ϕ son los parámetros de la red f_λ .

Aprendizaje con inferencia amortizada

- **Objetivo:** minimizar $-\sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi)$ como una función de θ, ϕ usando SGD.

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q_{\phi}(\mathbf{z}|\mathbf{x})) \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_{\phi}(\mathbf{z}|\mathbf{x})]\end{aligned}$$

- ① Inicializar $\theta^{(0)}, \phi^{(0)}$
- ② Muestrear aleatoriamente \mathbf{x}_n de \mathcal{D}
- ③ Calcular $\nabla_{\theta} \mathcal{L}(\mathbf{x}_n; \theta, \phi)$ y $\nabla_{\phi} \mathcal{L}(\mathbf{x}_n; \theta, \phi)$ // usar reparametrización, como antes
- ④ Actualizar θ, ϕ en la dirección del gradiente:

$$\begin{aligned}\theta^{(k+1)} &= \theta^{(k)} + \nabla_{\theta} \mathcal{L}(\mathbf{x}_n; \theta^{(k)}, \phi^{(k)}) \\ \phi^{(k+1)} &= \phi^{(k)} + \nabla_{\phi} \mathcal{L}(\mathbf{x}_n; \theta^{(k)}, \phi^{(k)}).\end{aligned}$$

① Repaso

② Aprendizaje variacional

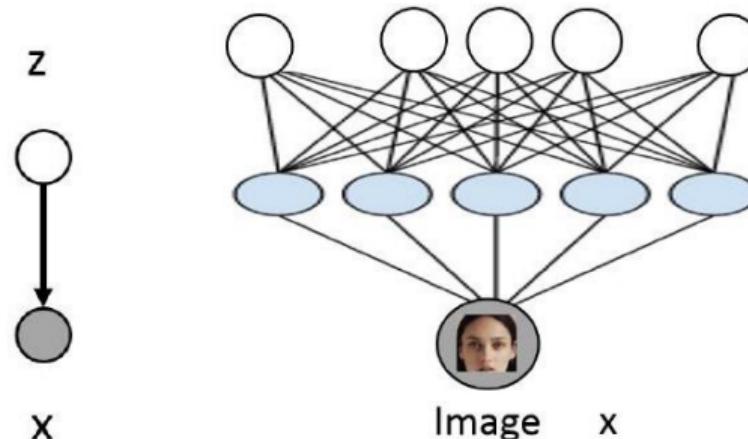
Optimización estocástica
Reparametrización

③ Inferencia amortizada

④ Autoencoder variacional

Recordemos los modelos profundos en variables latentes gaussianas

Extensión de los GMM a mezclas de un número infinito de gaussianas:



- ① Prior $\mathbf{z} \sim \mathcal{N}(0, I)$
- ② $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\Sigma}_\theta(\mathbf{z}))$, $\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta$ son redes neuronales.

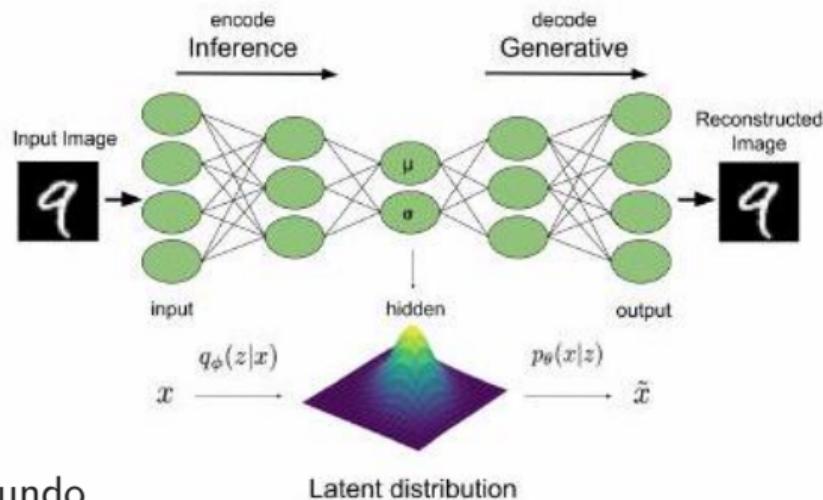
⇒ Si bien $p(\mathbf{z}|\mathbf{x})$, la verosimilitud marginal $p(\mathbf{x})$ es muy compleja / flexible.

Autoencoder variacional

Algoritmo de modelado generativo profundo definido por las siguientes características:

① Modelo probabilista:

- $p(\mathbf{x}, \mathbf{z}; \theta)$ modelo en variables latentes profundo
 - La condicional $p(\mathbf{x}|\mathbf{z}; \theta)$ puede tomar una forma no gaussiana.
 - La variable latente \mathbf{Z} es gaussiana.
- La red de inferencia aproximada $q_\phi(\mathbf{z}|\mathbf{x})$ es una gaussiana condicional.
- $p(\mathbf{x}|\mathbf{z}; \theta)$, $q_\phi(\mathbf{z}|\mathbf{x})$ están parametrizadas por NNs (e.g. $(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$, $(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$).



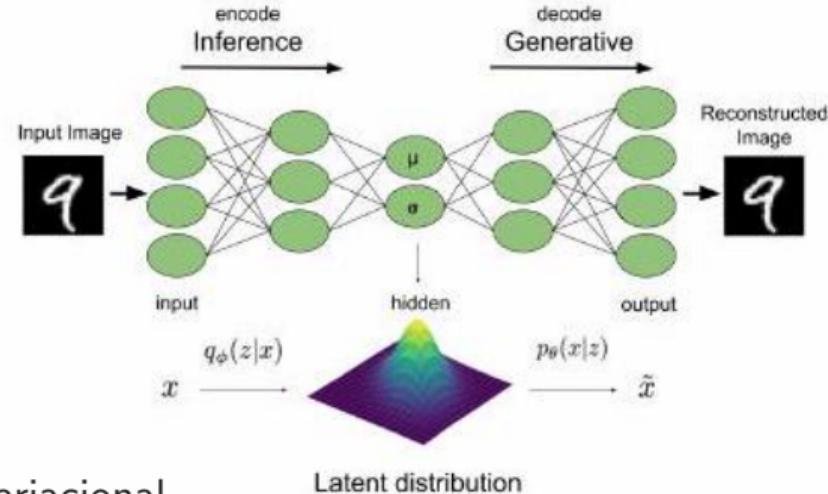
Autoencoder variacional

Algoritmo de modelado generativo profundo definido por las siguientes características:

② Entrenamiento:

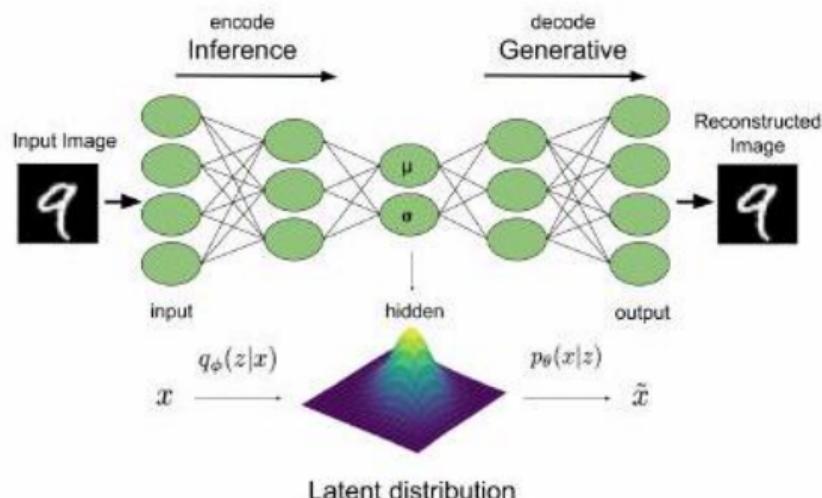
- El modelo se optimiza usando inferencia variacional para maximizar el ELBO:

$$\max_{\theta} \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi} \sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{L}(\mathbf{x}_n; \theta, \phi).$$



- Los gradientes se estiman por Monte Carlo usando el truco de la parametrización.

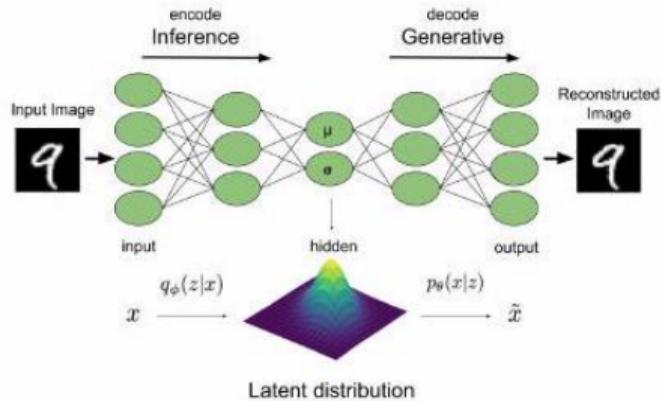
Perspectiva de autoencoder



- ① Se toma una muestra x_n
- ② Se mapea x_n en \hat{z} según $q_\phi(z|x_n)$ (encoder).
- ③ Se reconstruye \hat{x} muestreando de $p(x|\hat{z}; \theta)$ (decoder).

Perspectiva de autoencoder

¿Qué efecto tiene la función objetivo que se entrena (maximización del ELBO)?



$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))\end{aligned}$$

Perspectiva de autoencoder

¿Qué efecto tiene la función objetivo que se entrena (maximización del ELBO)?

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

- **1^{er} término:** fomenta que $\hat{\mathbf{x}} \approx \mathbf{x}_n$
(que \mathbf{x}_n sea probable según $p(\mathbf{x}|\hat{\mathbf{z}}; \theta)$)
⇒ Término de reconstrucción
- **2^o término:** fomenta que $\hat{\mathbf{z}}$ se distribuya
como el prior $p(\mathbf{z})$
⇒ Término de regularización

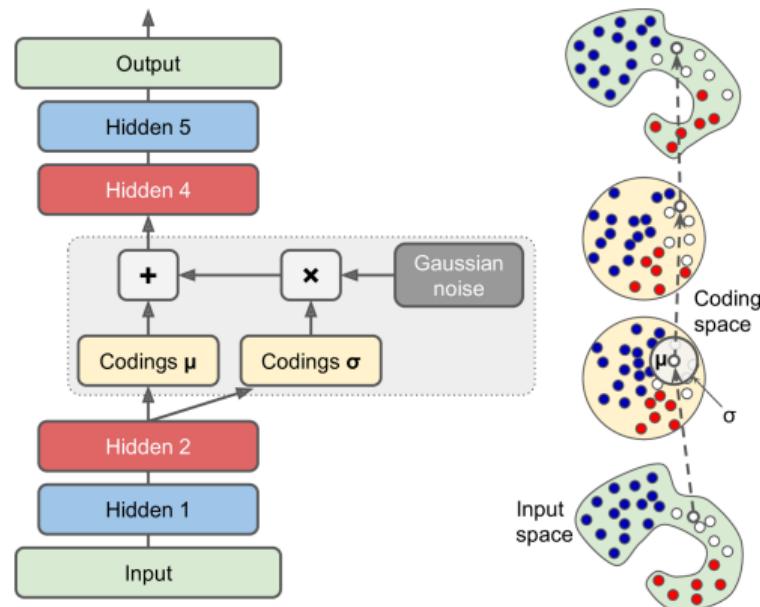
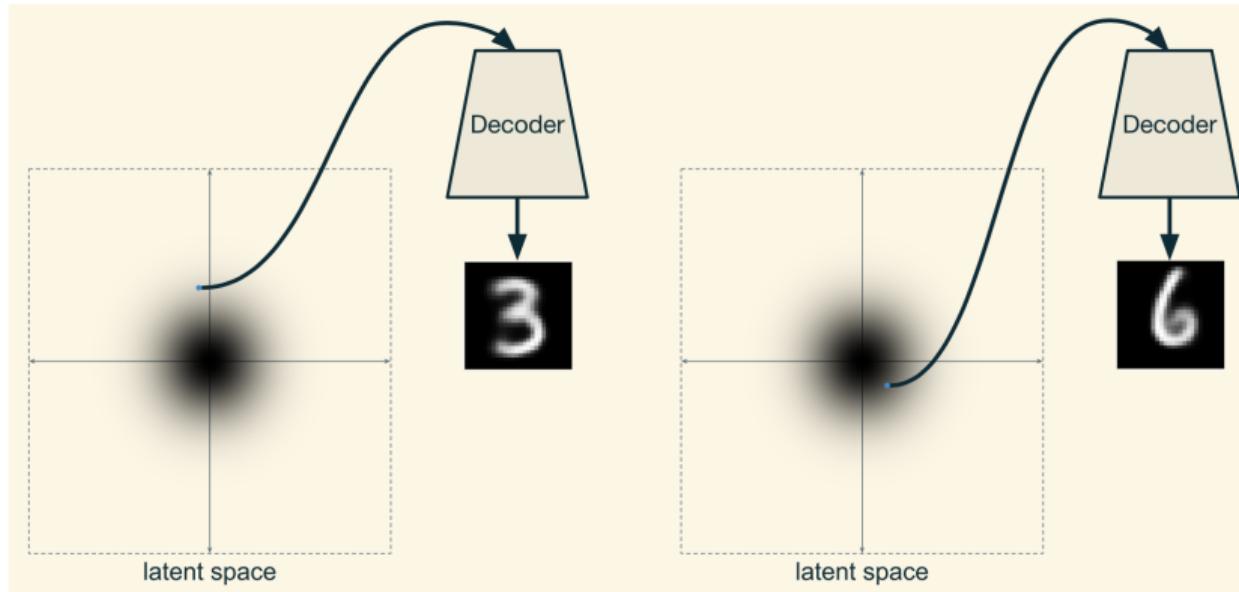


Figura tomada de A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow*. O'Reilly, 2023.

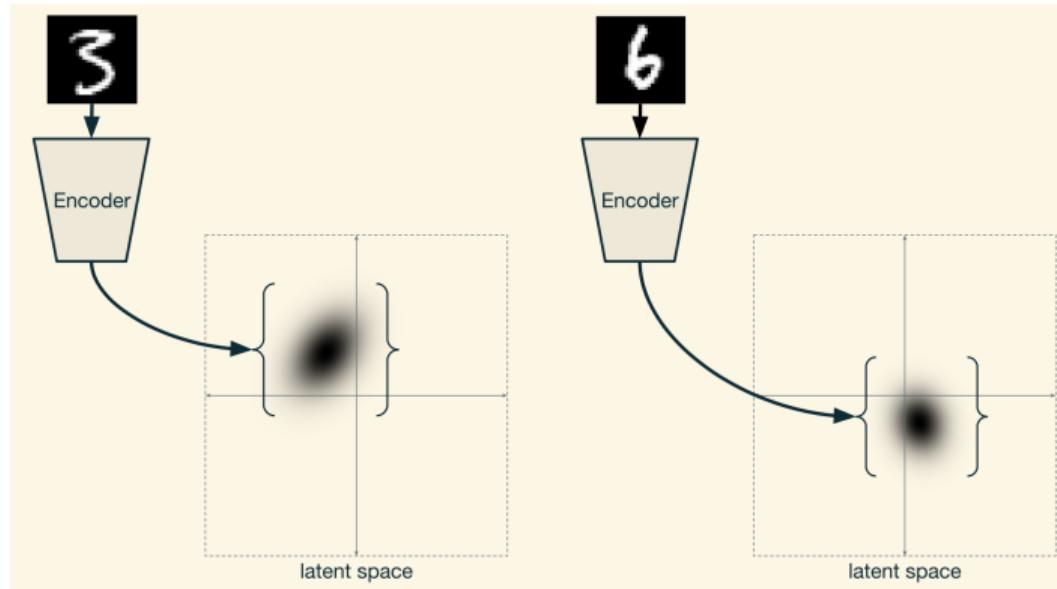
Autoencoders variacionales: estructura del espacio latente

Ejemplo: MNIST



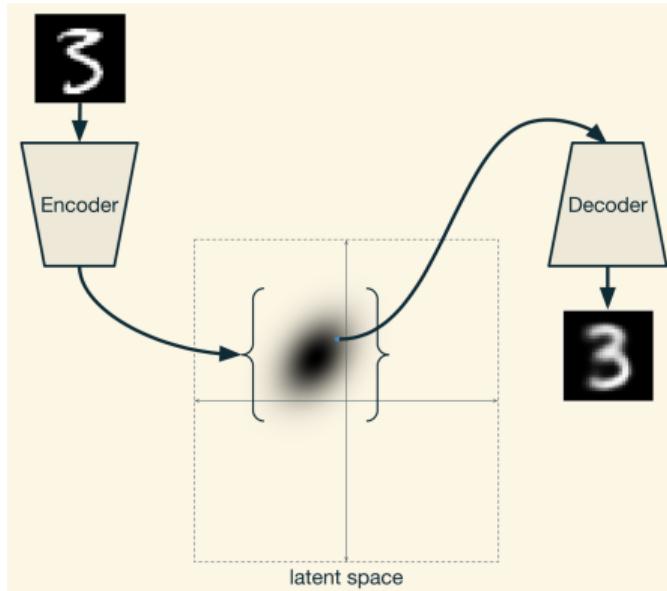
Autoencoders variacionales: estructura del espacio latente

Ejemplo: MNIST



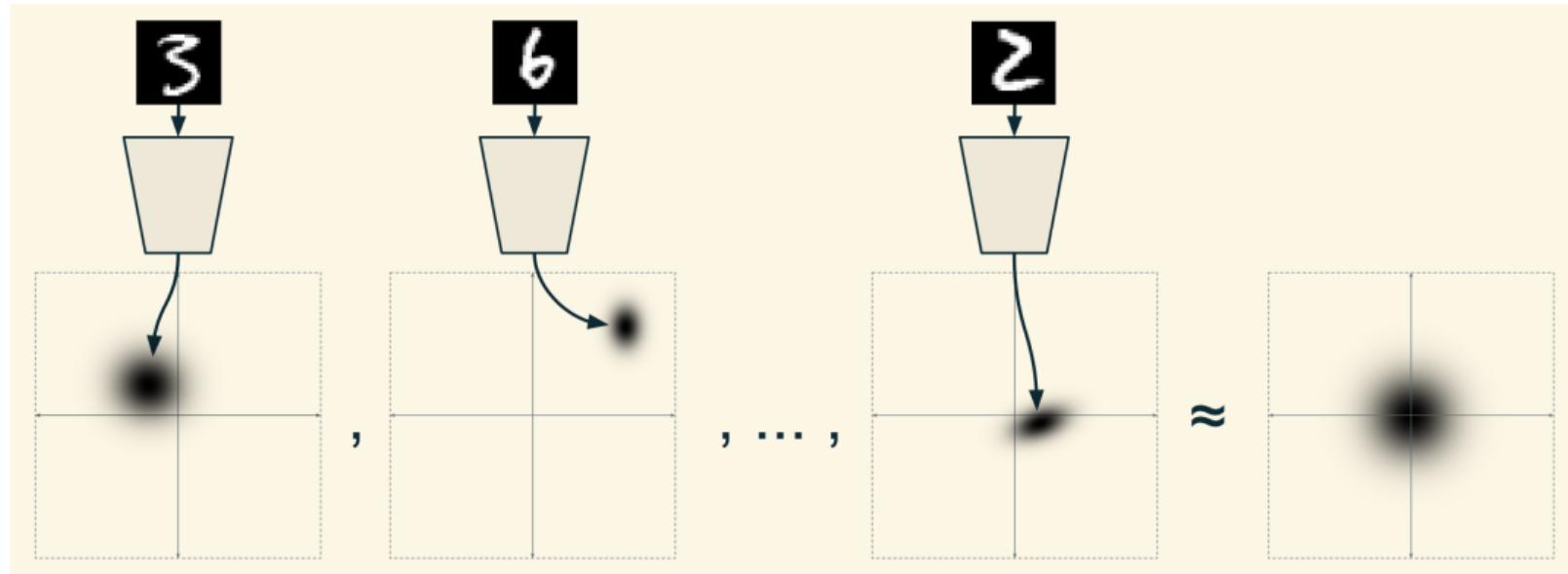
Autoencoders variacionales: estructura del espacio latente

Ejemplo: MNIST



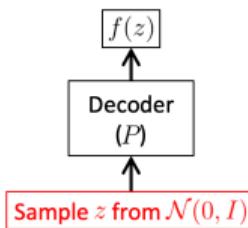
Autoencoders variacionales: estructura del espacio latente

Ejemplo: MNIST



Autoencoders variacionales: resultados

Generación de muestras aleatorias
sorteando $\mathbf{z} \sim N(0, I)$



8 5 1 7 8 1 4 8 2 8	3 1 6 5 1 0 7 6 7 2	2 8 9 1 3 8 5 9 3 8	2 2 0 8 9 2 3 9 0 0
9 6 8 3 9 6 0 3 1 9	8 5 9 4 6 8 2 1 6 2	8 3 8 2 7 9 3 5 3 8	7 5 9 9 1 1 7 1 4 4
3 3 7 1 3 6 5 1 7 9	6 1 5 3 2 8 8 1 3 3	8 5 5 9 4 7 9 5 1 3	8 9 6 2 0 8 2 8 2 9
8 9 0 8 6 9 1 4 6 3	2 1 6 8 4 1 0 0 4 1	1 9 2 8 5 8 3 1 9 2	2 4 8 4 3 8 7 0 6 1
9 2 3 3 3 1 3 8 6	5 1 9 1 0 1 5 3 5 9	2 7 3 6 4 3 0 2 0 3	5 4 7 9 1 9 9 9 1 0
6 9 9 8 6 1 6 6 6 5	6 5 6 1 4 9 1 7 5 8	5 9 7 0 5 9 3 8 4 5	6 8 8 4 9 4 8 2 8 1
9 5 2 6 6 5 1 8 9 9	1 3 4 3 9 8 3 4 7 0	6 9 4 3 6 2 8 5 5 2	7 5 8 2 9 6 1 3 5 3
1 9 7 1 3 1 2 8 2 3	4 5 8 2 9 7 0 1 5 9	8 4 9 0 5 0 7 0 6 6	7 9 8 7 2 7 9 3 9 0
0 4 6 1 2 3 2 0 8 8	6 8 4 4 8 7 2 3 4 3	7 4 3 6 2 0 3 6 0 1	4 5 2 4 3 9 0 1 8 4
9 7 5 4 9 3 4 8 5 1	2 6 4 5 6 0 9 7 7 8	2 1 2 0 9 7 1 9 5 0	2 8 7 3 3 1 6 2 3 6

Espacio latente 2-D

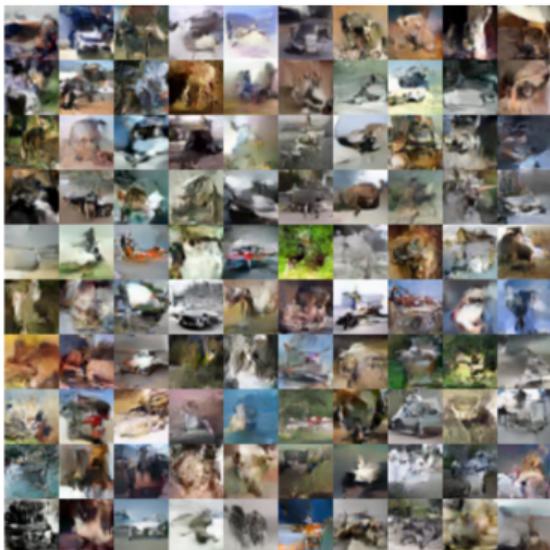
Espacio latente 5-D

Espacio latente 10-D

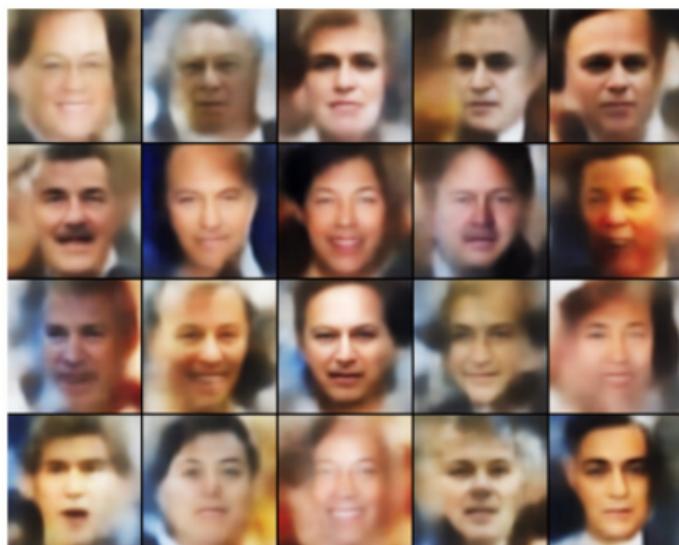
Espacio latente 20-D

Autoencoders variacionales: resultados

32x32 CIFAR-10



Labeled Faces in the Wild



Figuras de (I) Dirk Kingma et al. 2016; (D) Anders Larsen et al. 2017.

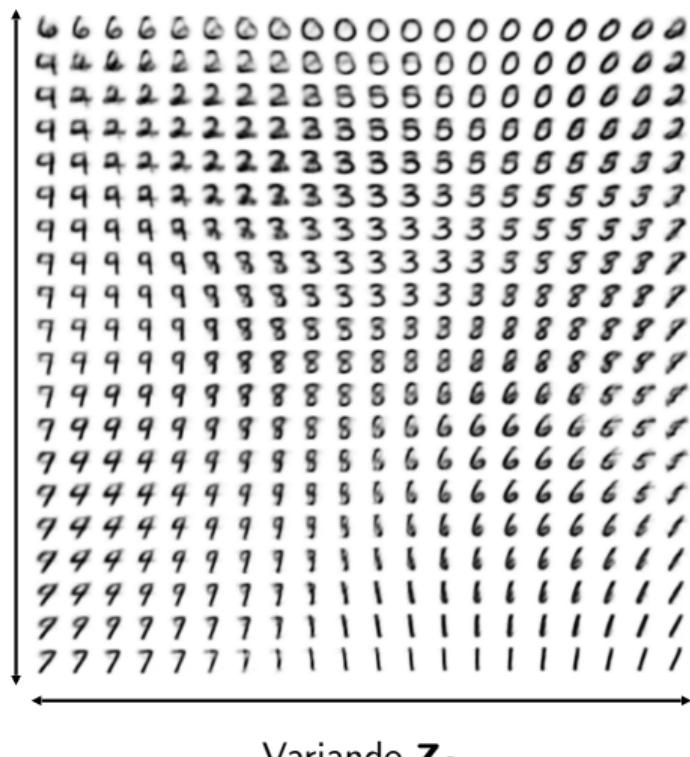
Autoencoders variacionales: resultados

El prior diagonal sobre $p(\mathbf{z})$ hace que las dimensiones de \mathbf{z} sean independientes

"Disentangling factors of variation"

Variando

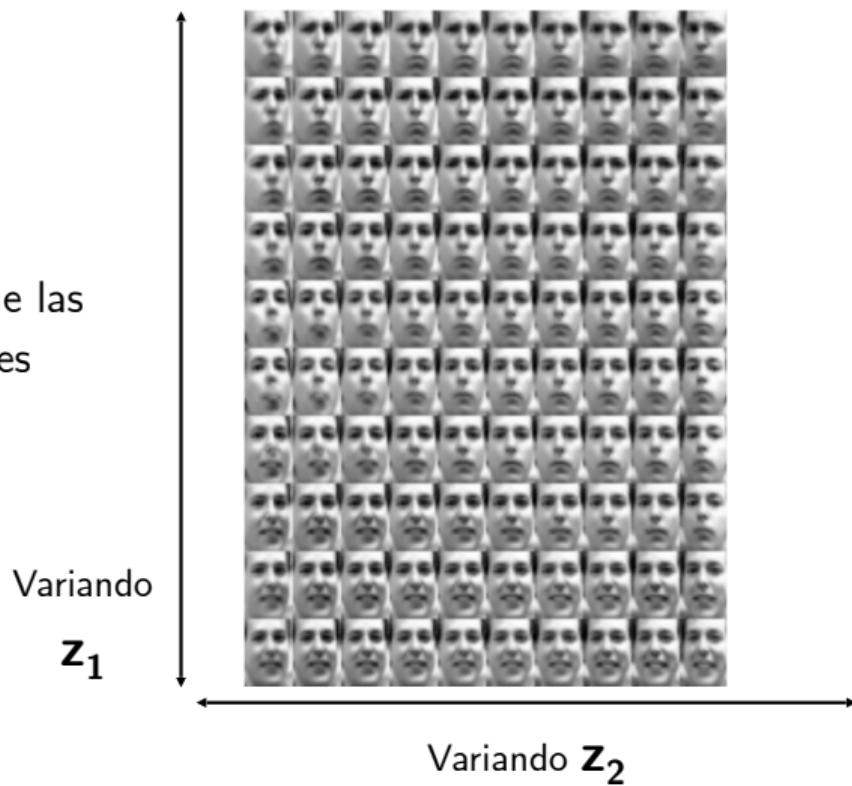
\mathbf{z}_1



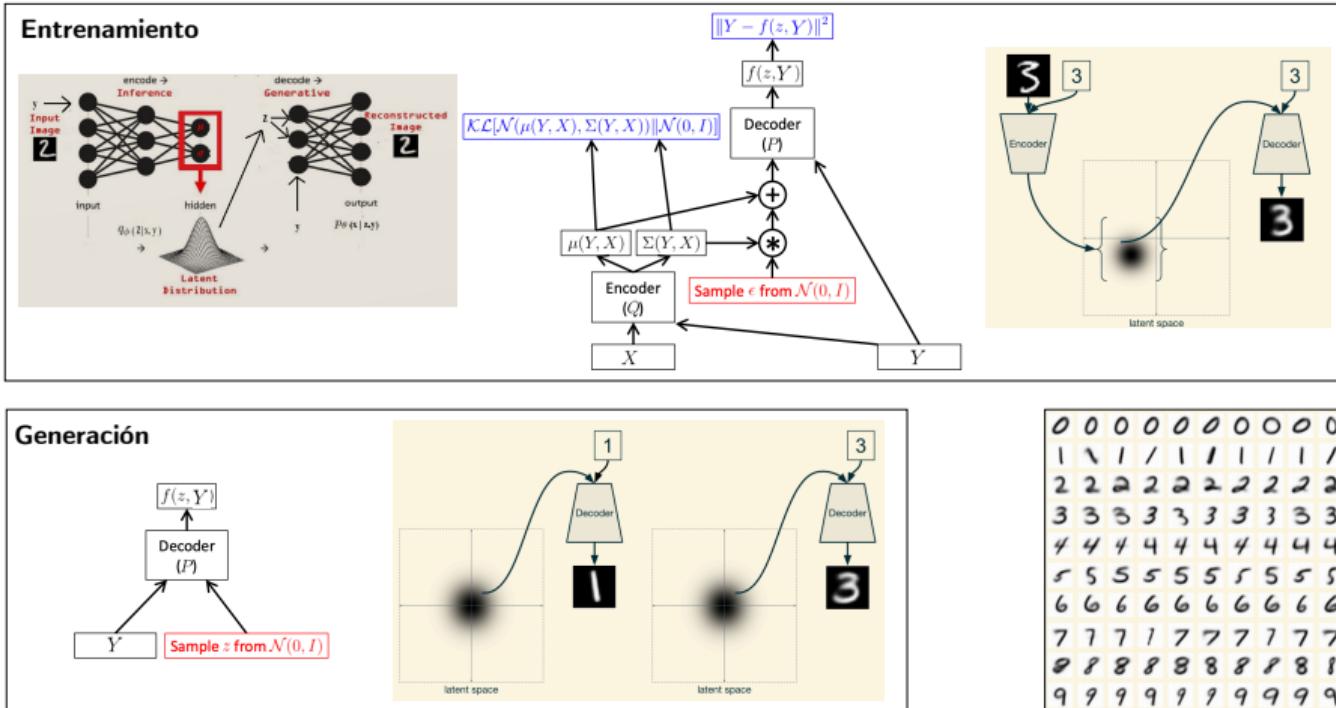
Autoencoders variacionales: resultados

El prior diagonal sobre $p(\mathbf{z})$ hace que las dimensiones de \mathbf{z} sean independientes

"Disentangling factors of variation"



Variational autoencoders condicionados



Resumen de Modelos de Variables Latentes

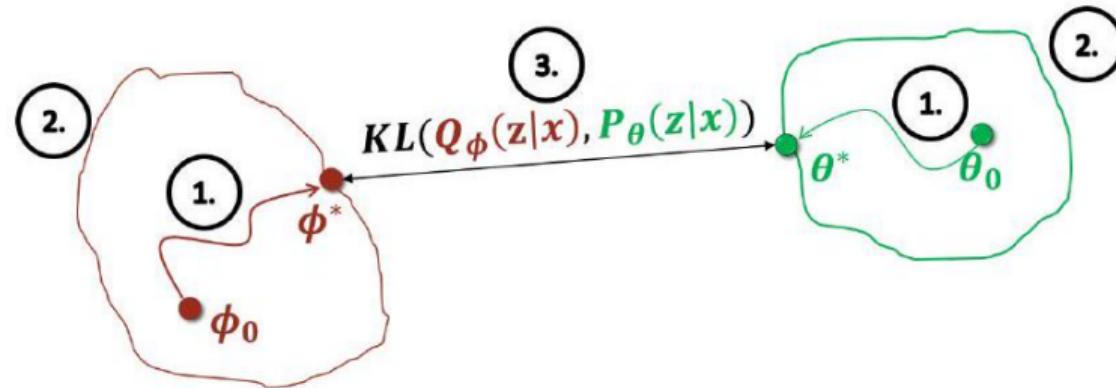
Ventajas:

- Combinar modelos simples para obtener uno más flexible (e.g., mezcla de gaussianas)
- El modelo dirigido permite muestreo ancestral (generación eficiente):
 $\mathbf{z} \sim p(\mathbf{z}), \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}; \theta)$
- Las representaciones latentes para cualquier \mathbf{x} pueden ser inferidas mediante $q_\phi(\mathbf{z}|\mathbf{x})$

Desventajas:

- La log-verosimilitud es generalmente intratable, por lo tanto el aprendizaje es difícil
- Aprendizaje conjunto de un modelo (θ) y un componente de inferencia amortizada (ϕ) para optimizar una cota inferior de la verosimilitud (ELBO).

Direcciones de Investigación



Distintas formas de mejorar el aprendizaje variacional:

- ① Mejores técnicas de optimización
- ② Familias de aproximación más expresivas
- ③ Funciones de pérdida alternativas.

Familias de Modelos - Encoder

Amortización (Gershman & Goodman, 2015; Kingma; Rezende; ..)

- Escalabilidad: Aprendizaje e inferencia eficientes en conjuntos de datos masivos
- Efecto de regularización: Debido al entrenamiento conjunto, también regulariza implícitamente el modelo θ (Shu et al., 2018)

Aumentando posteriors variacionales

- Métodos de Monte Carlo: Muestreo por Importancia (Burda et al., 2015), MCMC (Salimans et al., 2015, Hoffman, 2017, Levy et al., 2018), Monte Carlo Secuencial (Maddison et al., 2017, Le et al., 2018, Naesseth et al., 2018), Muestreo por Rechazo (Grover et al., 2018)
- Flujos de Normalización (Rezende & Mohammed, 2015, Kingma et al., 2016)

Familias de Modelos - Decoder

- Decodificadores poderosos $p(\mathbf{x}|\mathbf{z}; \theta)$ como DRAW (Gregor et al., 2015), PixelCNN (Gulrajani et al., 2016)
- Priors parametrizados y aprendidos $p(\mathbf{z}; \theta)$ (Nalusnick et al., 2016, Tomczak & Welling, 2018, Graves et al., 2018)
- Modelos jerárquicos donde múltiples VAEs están apilados uno encima del otro (Modelos de Difusión)

Objetivos Variacionales

Una ELBO más ajustada no implica:

- Mejores muestras: La calidad de la muestra y las verosimilitudes no están correlacionadas (Theis et al., 2016)
- Códigos latentes informativos: Los decodificadores poderosos pueden ignorar los códigos latentes debido a la compensación en la minimización del error de reconstrucción frente a la penalización previa de KL (Bowman et al., 2015, Chen et al., 2016, Zhao et al., 2017, Alemi et al., 2018)

Alternativas a la divergencia KL:

- Las alfa-divergencias de Renyi (Li & Turner, 2016)
- Métricas de probabilidad integral como la discrepancia de media máxima, distancia de Wasserstein (Dziugaite et al., 2015; Zhao et. al, 2017; Tolstikhin et al., 2018)

Referencias

-  C. M. Bishop, *Pattern Recognition and Machine Learning*.
Springer, 2006.
-  Stanford, “CS236 Deep Generative Models.” <https://deepgenerativemodels.github.ioLecture>, 2024.
-  J. M. Tomczak, *Deep Generative Modeling*.
Springer Cham, 2024.