

Parcial IA



2.1. Modelo, función de costo y optimización por gradiente automático, de los autoencoders regularizados, autoencoders variacionales y las redes generativas adversarias (GANs).

Autoencoder Regularizado

Autoencoder:

Codificador: Encoder, decodificador (decoder).

El codificador toma una entrada $x \in \mathbb{R}^d$ y la transforma en una representación codificada $z \in \mathbb{R}^m$, donde $m < d$. Se expresa como:

$$z = f(x; \theta)$$

Representación matemática.

Codificación

$$z = f(x; w_e, b_e) = \phi(w_e x + b_e)$$

Espacio latente.

Donde:

w_e : matriz de pesos

b_e : vector de sesgos.

ϕ : función de activación

z : representación latente.

Decodificación.

$$\hat{x} = g(z, w_d, b_d) = \varphi(w_d z + b_d)$$



Donde:

wd : matriz de pesos del decodificador.

bd : vector de sesgos del decodificador.

ψ : función de activación

\hat{x} : reconstrucción de la entrada.

La salida \hat{x} del autoencoder es el resultado de la composición de las funciones f (codificador) y g (decodificador).

$$\hat{x} = (g \cdot f)(x) = g(f(x; we, be); wd, bd).$$

de esta forma el autoencoder completo puede ser visto como una función compuesta.

$$h(x; we, be, wd, bd) :$$

Modelo.

$$- h(x; we, be, wd, bd) = g(f(x; we, be), wd, bd) = \hat{x} -$$

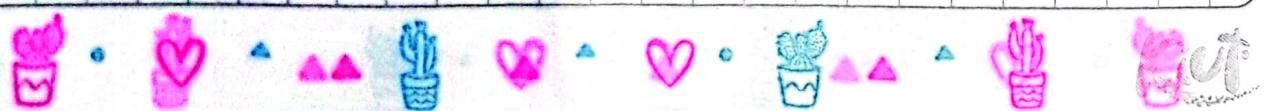
- función de costo -

La función de costo mide la diferencia entre la entrada x y la salida reconstruida \hat{x} .

$$J(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^N \|x_i - h(x_i; we, be, wd, bd)\|^2$$

$$J(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^N \|x - \hat{x}\|^2$$

$$E_x \{ J(x, \hat{x}) \} = E_x \{ \frac{1}{N} \sum_{i=1}^N \|x - h(x_i; we, be, wd, bd)\|^2 \}.$$





-Optimización-

$$\Theta = \{w_c, w_d, b_c, b_d\}$$

$$\hat{\Theta} = \arg \min_{\Theta} E_x \{ L(x, (g, f(x))) \}$$

-Autoencoder regularizado. - NO supervisado

No etiquetas, No target.

Incluye un término adicional en su función de costo, para penalizar ciertas propiedades de los pesos, con el fin de mejorar la generalización del modelo y evitar y paliar problemas como el sobreajuste. Es decir, no generaliza bien a datos nuevos que no ha visto antes.

Se toma el autoencoder y agrega una regularización a los pesos del modelo.

-función de Costo regularizada.-

$$L(w_c, b_c, w_d, b_d) = E_x \{ \|x - h(x; w_c, b_c, w_d, b_d)\|^2\} + \lambda \mathcal{R}(w_c, w_d)$$

donde

- $E \{ \|x - \hat{x}\|^2 \}$: Esperanza del error cuadrático medio MSE entre la entrada x y la salida reconstruida \hat{x} .

- $\mathcal{R}(w_c, w_d)$: Término de regularización, p.e.

$$\cdot L_1: \mathcal{R}_{L_1}(w) = \sum |w_i|$$

$$\cdot L_2: \mathcal{R}_{L_2}(w) = \|w\|^2$$

- λ : es un hiperparámetro, que controla la importancia



de la regularización.

-minimizar

- Optimización -

$$\Theta = \{w_e, w_d, b_e, b_d\}$$

$$\hat{\Theta} = \arg \min_{\Theta} \mathbb{E}_x \{ J(x, g_e(x)) \} + \lambda \mathcal{R}(\Theta)$$

Autoencoder regularizado (función de costo CE).

Codificador: $f(x; \Theta_e) :$

$$z = f(x; \Theta_e) = \phi(w_e x + b_e)$$

decodificador: $y(z; \Theta_d) :$

$$x' = g(z; \Theta_d) = p(w_d z + b_d)$$

- función de costo:

$$L(\Theta) = \mathbb{E}_x \{ J(x, g(f(x; \Theta_e); \Theta_d)) \}$$

$$+ \lambda \mathcal{R}(\Theta)$$

donde,

$\Theta = (\Theta_e, \Theta_d) = (w_e, b_e, w_d, b_d) \rightarrow$ parámetros del autoencoder.

$$J(x, \tilde{x}) = - \sum_{i=1}^n (x_i \log(\tilde{x}_i) + (1-x_i) \log(1-\tilde{x}_i))$$

es la entropía cruzada entre la entrada x y salida reconstruida \tilde{x} .

$$\mathcal{R}(\Theta) = \frac{1}{2} (||w_e||^2 + ||w_d||^2) \text{ es el término de regularización.}$$



Optimización

$$\theta = \arg \min_{\theta} [E_x [f(x, g(f(x, \theta_c); \theta_d))] + \lambda \cdot \Omega(\theta)]$$

Actualización parámetros

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} I(\theta) \text{ donde}$$

$$\nabla_{\theta} I(\theta)$$

gradiente de la función de costo

$I(\theta)$ respecto a θ

MSE

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2$$

η = tasa aprendizaje

$$\Omega(\theta, \theta') \geq \frac{1}{2} (\|\theta\|^2 + \|\theta'\|^2)$$

λ , hiperp. t : indica iteración

Nota: También es comúnmente utilizado MSE.

Autoencoder Variacional (VAE) (No supervisado).

Es una extensión del autoencoder tradicional que introduce una interpretación probabilística en la codificación, permitiendo la generación de nuevas muestras a partir del espacio latente.

Se modela mediante:

$$P(x) = \int P(x, z) dz = \int P(x|z) P(z) dz$$

↑ pnor.

Codificador probabilístico.

$$f_c(x; \theta_c)$$



En lugar de mapear la entrada x directamente a una representación z , el codificador en VAE mapea x a los parámetros de una distribución Gaussiana, típicamente la μ y la desviación estándar σ , que definen una distribución de probabilidad en el espacio latente.

$$\mu, \log \sigma^2 = f(x; \theta_e), x \rightarrow \mu, \log \sigma^2$$

μ : media del espacio latente

$\log \sigma^2$: es logaritmo de la varianza.

z muestra tomada de la distribución $N(\mu, \sigma^2)$.

θ_e : parámetros del decodificador.

• Decodificador (Decoder)

El decodificador toma una muestra z del espacio latente y la transforma de vuelta a una reconstrucción \hat{x} de la entrada.

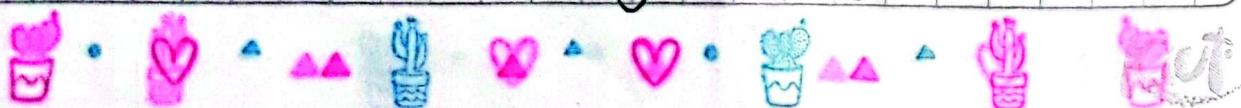
$$\hat{x} = g(z; \theta_d); \theta_d \text{ son parámetros del decodificador}$$

$$z \xrightarrow{g(z; \theta_d)} \hat{x}$$

Por bayes tenemos:

$$P(z|x) = \frac{P(x|z) P(z)}{P(x)} \rightarrow \text{Calcular } p(x) \rightarrow \text{Difícil}$$

Rescribir evidencia marginal logarítmica



log p(x) utilizando $q(z|x)$ como:

$$\log p(x) = \log \int p(x|z) p(z) dz$$

$$= \log \int \frac{q(z|x)}{q(z|x)} \cdot p(x|z) p(z) dz.$$

Por igualdad de Jensen:

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} \left[\log \frac{p(x|z) p(z)}{q(z|x)} \right]$$

Evidence Lower Bound (ELBO)

$$\text{ELBO} = \mathbb{E}_{z \sim q(z|x)} \left[\log p(x|z) \right] - \text{KL}(q(z|x) || p(z))$$

Descomposición del ELBO.

1. Pérdida reconstrucción.

$$\text{Jrecon} = -\mathbb{E}_z [\log p(x|z)]$$

Mide reconstrucción modelo entrada x a partir de una muestra z del espacio latente.

2. Divergencia KL.

$$\text{KL}(q(z|x) || p(z)) = \mathbb{E}_z \left[\log \frac{q(z|x)}{p(z)} \right]$$

Mide diferenciatractable, VAE utiliza reparametrización. En lugar de mesturar directamente de $q(z|x)$, reparametriza z :

$$z = \mu + \sigma \odot \epsilon \quad \text{donde } \epsilon \sim N(0, I).$$



Funciòn de costo

$$J(\theta_e, \theta_d) = -E_z \underbrace{[\log p(x|z)]}_{\text{pérdida de reconstrucción}} + k_l \underbrace{(q(z|x_j|\theta) | p(z))}_{\text{regularización}}$$

$\theta = \theta_e, \theta_d$

$$\text{Optimización } \theta^* = \arg \min_{\theta} J(\theta)$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta \left(-\nabla_{\theta} E_z [\log p(x|z; \theta)] + \nabla_{\theta} k_l (q(z|x_j|\theta) | p(z)) \right)$$

donde

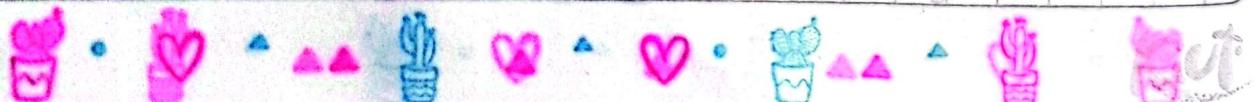
θ : paràmetros del modelo.

GANs. Red generativa adversativa. (NO supervisado)

Modo generativo, 2 redess neuronales, generador y discriminador, se entranan conjuntamente en un marco adversarial.

1. Generador (G)

Toma muestra z de una distribución latente $N(0, I)$ y genera una muestra




 $\hat{x} = G(z; \theta_G)$ que intenta imitar la distribución de datos reales. $p_{\text{data}}(x)$.

2. Discriminador (D).

Toma una muestra del conjunto de datos o una generada \hat{x} y produce un valor que representa la probabilidad de que la muestra provenga de los valores reales. $p_{\text{data}}(x)$.

función de Costo

Discriminador: Máxima capacidad distinguir muestras reales y generadas.

$$J_D(\theta_D) = -E_x \{ \log D(x; \theta_D) \} - E_z [\log (1-D(G(z; \theta_G); \theta_D))]$$

x muestra real

$G(x; \theta_G)$: muestra generada generador.

$D(x; \theta_D)$: probabilidad x sea muestra real.

θ_D : parámetros discriminador.

Generador: Se entrena para "engañar" el discriminador. Minimiza capacidad de distinguir entre muestras reales y generadas.

$$J_G(\theta_G) = -E_z [\log D(G(z; \theta_G); \theta_D)]$$

Donde

θ_G : Son los parámetros del generador.



Optimización: Se realizan en 2 etapas alternas

1. Actualizar parámetros del discriminador Θ_D y otra para actualizar los del generador Θ_G .

o. Discriminador.

Entrena maximizar función de costo $J_D(\Theta_D)$

$$\Theta_D^{(t+1)} = \Theta_D^{(t)} + \eta_D \nabla_{\Theta_D} J_D(\Theta_D).$$

Donde

$\nabla_{\Theta_D} J_D(\Theta_D)$: Gradiente función costo discriminador respecto parámetros Θ_D .

η_D : tasa aprendizaje

o. Generador.

Entrena maximizar función costo $J_G(\Theta_G)$

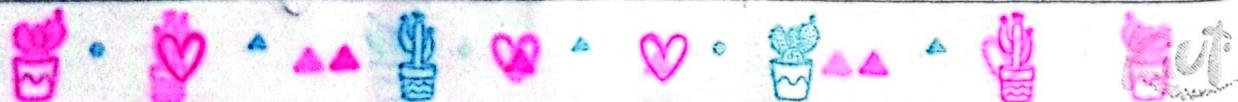
$$\Theta_G^{(t+1)} = \Theta_G^{(t)} - \eta_G \nabla_{\Theta_G} J_G(\Theta_G).$$

$\nabla_{\Theta_G} J_G(\Theta_G)$: Gradiente generador respecto parámetros Θ_G .

η_G : tasa aprendizaje.

o. Función Costo Combinada

$$\hat{\Theta} = \arg \min_{\Theta_G} \max_{\Theta_D} J(\Theta).$$



dónde

$$L(\theta) = -E_x [\log D(x; \theta_D)] - E_z [\log (1 - D(G(z; \theta_G); \theta_D))]$$

Generador toma una muestra latente z y genera una muestra $G(z; \theta_G)$.

Discriminador toma muestra (real/generada) y produce probabilidad de que la muestra sea real.

Deep fake

Intercambio de rostros, manteniendo atributos rostro objetivo (expresión, postura), mientras se transfiere a la identidad del rostro fuente.

- Modulo mycción de Identidad (IM):

- AdaIN: Incrusta identidad del rostro fuente en el mapa de características del rostro objetivo.

$$\text{AdaIN}(\text{fea}, \text{us}) = \text{os} \left(\frac{\text{fea} - \mu(\text{fea})}{\sigma(\text{fea})} \right) + \text{us}$$

donde: fea es el mapa de características del rostro objetivo.

us es el vector de identidad extraido del



rostro fuente.

μ (F_{fa}) y σ (f_{fa}) Media y desviación por canal del mapa de características.

U_S y O_S variables a partir de VS utilizando capas conectadas.

-pérdida de identidad (Identity loss):

Sé utiliza para asegurar que la identidad del rostro generado sea similar a la del rostro fuente. Se calcula utilizando la similitud coseno:

$$L_{Id} = \frac{1 - \mathbf{VR} \cdot \mathbf{VS}}{\|\mathbf{VR}\|_2 \|\mathbf{VS}\|_2}$$

Donde,

\mathbf{VR} vector identidad extraido de la imagen generada.

\mathbf{VS} es el vector de identidad del rostro fuente.

-pérdida reconstrucción:

Cuando rostro fuente y rostro objetivo son

de la misma identidad, la imagen generada

debe parecerse a la imagen objetivo.

$$L_{Recon} = \|I_R - I_T\|_1$$

Donde:

I_R imagen resultado, I_T imagen objetivo.





- pérdida coincidencia de características débiles (weak feature matching loss): - - -

Alinear la imagen generada con la imagen objetivo en un nivel semántico alto, ayuda a preservar los atributos rostro objetivo.

$$L_{WFM}(P) = \sum_{i=m}^M \frac{1}{N_i} \| D^{(i)}(IR) - D^{(i)}(IT) \|_1$$

Donde:

$D^{(i)}$ extractor características capa i discriminador.

N_i número elementos capa i.

M número total de capas.

m capa donde se comienza a calcular la pérdida en la coincidencia de características débiles.

• ♡ • ▲ Cactus ▲ ♡ • -pérdida Adversarial (Adversarial Loss): - - - -

Mejora realismo de imagen generada.

$$L_{\text{Adv}} = -E_{I_{\text{real}}} [\log D(I_{\text{real}})] -$$

Discriminador acierta imágenes reales. $E_{I_{\text{fake}}} [\log(1-D(I_{\text{fake}}))]$ minimizar.

Discriminador es engañado Valor esperado pérdida por generador. $E_{I_{\text{fake}}} [\log(1-D(I_{\text{fake}}))]$ imágenes fake (generadas).

Mide imágenes clasificadas correctamente e incorrectamente por el discriminador. D .

pérdida de penalización de Gradiente (GP).

Asegura que discriminador tenga un gradiente suave en el espacio de imágenes.

$$L_{\text{GP}} = E_{\mathbb{I}} \left[\left(\|\nabla_i D(\mathbb{I})\|_2^2 - 1 \right)^2 \right]$$

Donde:

\mathbb{I} interpolación entre imágenes reales y generadas.

$$\mathbb{I} = M_1 w_1 I_{\text{real}} + M_2 w_2 I_{\text{fake}} + q_1 g_1 A_1 +$$

\vdots

$$\mathbb{I} = M_1 w_1 I_{\text{real}} + M_2 w_2 I_{\text{fake}} + q_1 g_1 A_1 + q_2 g_2 A_2 + \dots + q_n g_n A_n$$



débiles

-Función de pérdida Total (Overall Loss function) -

Combina todas las pérdidas para entrenar el modelo.

$$\text{pérdida total} = \lambda_{\text{id}} L_{\text{id}} + \lambda_{\text{recon}} L_{\text{recon}}$$

$$+ L_{\text{Adu}} + \lambda_{\text{Gp}} L_{\text{Gp}} + \lambda_{\text{wFM}} L_{\text{wFM}}$$

Donde:

L_{Adu} pérdida adversarial para mejor calidad imagen.

L_{Gp} es la penalización de Gradiente.

$\lambda_{\text{id}}, \lambda_{\text{recon}}, \lambda_{\text{Gp}}, \lambda_{\text{wFM}}$. Son pesos controlan contribución.



