

## PARCIAL IA

2.1. Presente el modelo, función de costo y optimización por gradiente automático, de los autoencoders regulados, autoencoders variacionales y las redes generativas adversarias (GANs).

### → Autoencoder

Es una red neuronal diseñada para aprender una representación compacta y eficiente de los datos, capaz de reconstruirlos lo más fielmente posible.

El autoencoder está compuesto por:

1. Codificador (Encoder): Mapea la entrada  $x$  a una representación latente  $z$ .

2. decodificador (Decoder): Mapea la representación latente  $z$  de vuelta a una reconstrucción de la entrada  $\hat{x}$ .

Representación matemática.

Codificación:

$$z = f(x; W_e, b_e) = \phi(W_e x + b_e) \rightarrow \text{espacio latente.}$$

Donde:

$W$ : matriz de pesos.

$b$ : vector de sesgos.

$\phi$ : función de activación.

$z$ : representación latente.

## Decodificación

$$\hat{x} = g(z; W_d, b_d) = \varphi(W_d z + b_d).$$

Donde:

$W_d$ : matriz de pesos del decodificador.

$b_d$ : vector de sesgos del decodificador

$\varphi$ : 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).

$$\hat{x} = (g \circ f)(x) = g(f(x; W_e, b_e); W_d, b_d).$$

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

$$h(x; W_e, b_e, W_d, b_d):$$

$$h(x; W_e, b_e, W_d, b_d) = g(f(x; W_e, b_e); W_d, b_d) = \hat{x}$$

## Funcióñ de Costo

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

$$f(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^N \|x_i - h(x_i; W_e, b_e, W_d, b_d)\|^2.$$

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

Si lo hacemos sobre toda la distribución de datos posible tenemos:  $E\hat{\sigma}^2$ ?

$$E\{f(x, \hat{x})\} = E_x\{f(x, (g \circ f(x)))\},$$

## Optimización.

$$\Theta = \{W_e, W_d, b_d, b_e\}$$

$$\hat{\Theta} = \arg_{\Theta} \min \mathbb{E}_x \{ f(x, g \circ f(x)) \}$$

### -> Autoencoder Regularizado

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 problemas como el sobreajuste (overfitting), es decir no generalizar bien a datos nuevos que no ha visto antes.

Retomamos el autoencoder y agregamos una regularización a los pesos del modelo.

### Función de costo regularizada

$$f(W_e, b_e, W_d, b_d) = \mathbb{E}_x \{ \|x - h(x; W_e, b_e, W_d, b_d)\|^2\} + \lambda \Omega(W_e, W_d)$$

donde

- $\mathbb{E}_x \{ \|x - \hat{x}\|^2 \}$ : Es la esperanza del error cuadrático medio MSE entre la entrada  $x$  y la salida reconstruida  $\hat{x}$

- $\Omega(W_e, W_d)$ : Término de regularización, por ejemplo:

- L1 :  $\Omega_{L1}(W) = \sum_i |W_i|$

- L2 :  $\Omega_{L2}(W) = \|W\|^2$

- $\lambda$ : es un hiperparámetro que controla la importancia de la regularización

## Optimización

$$\Theta = \{W_c, W_d, b_c, b_d\}$$

$$\hat{\Theta} = \arg \min_{\Theta} \text{Eff}(x, (g \circ f(x))) + \lambda \varphi(\Theta)$$

## Autoencoder regularizado (función de costo $J(\theta)$ )

• Codificador:  $f(x; \theta_e)$ :

$$z = f(x; \theta_e) = \phi(W_e x + b_e)$$

• Decodificador:  $g(z; \theta_d)$ :

$$\hat{x} = g(z; \theta_d) = \phi(W_d z + b_d).$$

### Función de costo:

$$J(\theta) = E_x [ -l(x, g(f(x; \theta_e); \theta_d))] + \lambda \cdot R(\theta).$$

donde

•  $\theta = (\theta_e, \theta_d) = (W_e, b_e, W_d, b_d) \rightarrow$  todos los parámetros del autoencoder.

•  $l(x, \hat{x}) = -\sum_{i=1}^n (x_i \log(\hat{x}_i) + (1-x_i) \log(1-\hat{x}_i))$  es la entropía cruzada entre la entrada  $x$  y la salida reconstruida  $\hat{x}$ .

•  $R(\theta) = \frac{1}{2} (\|W_e\|^2 + \|W_d\|^2)$  es el término de regularización L2.

### Función de optimización

$$\theta = \arg \min_{\theta} [E_x [-l(x, g(f(x; \theta_e); \theta_d))] + \lambda \cdot R(\theta)]$$

### Actualización de parámetros

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

$\nabla_{\theta} J(\theta)$ : gradiente de la función de costo  $J(\theta)$  respecto a  $\theta$ .

$\eta$ : tasa de aprendizaje.

$t$ : indica la iteración actual.

## Autoencoder variacional (VAE)

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

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

- Codificador probabilístico:  $f(x; \theta_e)$ .

En lugar de mapear la entrada  $x$  directamente a una representación  $z$ , el codificador en un VAE mapea  $x$  a los parámetros de una distribución gaussiana, típicamente la media  $\mu$  y la desviación estándar  $\sigma^2$ , que definen una distribución de probabilidad en el espacio latente.

$$\mu, \log \sigma^2 = f(x; \theta_e); \quad x \xrightarrow{f(x; \theta_e)} \mu, \log \sigma^2$$

- $\mu$ : media del espacio latente.
- $\log \sigma^2$ : es logaritmo de la varianza.
- $z$ : muestra tomada de la distribución  $N(\mu, \sigma^2)$ .
- $\theta_e$ : 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.

$$\hat{x} = g(z; \theta_d); \quad \theta_d \text{ son los parámetros del decoder.}$$

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

por bayes tenemos

$$P(z|x) = \frac{P(x|z)P(z)}{P(x)} \rightarrow \text{(calcular } P(x) \text{)} \rightarrow \text{muy difícil.}$$

reescribimos la 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 la desigualdad de Jensen:

$$\log P(x) \geq E_{z \sim q(z|x)} \left[ \log \frac{P(x|z)P(z)}{q(z|x)} \right]$$

Evidence lower Bound (ELBO).

$$\text{ELBO} = E_{z \sim q(z|x)} [\log p(x|z)] - \text{KL}(q(z|x) \| P(z)).$$

Descomposición del ELBO.

El ELBO se descompone en dos términos.

1. Perdida de Reconstrucción.

$$\text{L}_{\text{recon}} = -E_z [\log P(x|z)]$$

Este término mide qué tan bien el modelo puede reconstruir la entrada  $x$  a partir de una muestra  $z$  del espacio latente.

2. Divergencia KL.

$$\text{KL}(q(z|x) \| P(z)) = E_z \left[ \log \frac{q(z|x)}{P(z)} \right]$$

Este término mide la diferenciatractable, el VAE utiliza la trick de reparametrización. En lugar de muestrear directamente de  $q(z|x)$ , reparametrizamos  $z$  de la siguiente manera.

$$z = M + L \odot \epsilon, \text{ donde } \epsilon \sim N(0, I).$$

## Funcióñ de costo

$$L(\theta_e, \theta_d) = -E_z [\log P(x|z)] + KL(q(z|x)\|P(z)).$$

Perdida de  
Reconstrucción

Regularización KL.

$$\Theta = \theta_e, \theta_d.$$

Optimizaciones

$$\Theta^* = \arg \min_{\Theta} L(\Theta)$$

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta (-\nabla_{\Theta} E_z [\log P(x|z|\Theta)] + \nabla_{\Theta} KL(q(z|x_j|\Theta)\|P(z)))$$

donde

$\Theta$ : parámetros del modelo.

$$((5)(x)(p)) + B = (5)$$

$$(2)(x)(p) + B = (2)$$

$$(3)(x)(p) + B = (3)$$

$$[(5)(x)(p)] + B = (5)$$

$$((5)(x)(p)) + B = (5)$$

$$(2)(x)(p) + B = (2)$$

$$\int (x)(p) dx + B = (5)$$

$$((5)(x)(p)) + B = (5)$$

$$(5)(x)(p) + B = (5)$$

# GANs

## Generative adversarial

Son un tipo de modelo generativo, donde 2 redes neuronales, el generador y el discriminador, se entrenan conjuntamente en un marco adversarial.

Estructura del modelo GAN.

- En una GAN hay 2 componentes principales.

### 1. Generador (G)

- Toma una muestra  $z$  de una distribución latente (normalmente una distribución normal estandarizada  $N(0, I)$ ) y genera una muestra  $\hat{x} = G(z; \theta_G)$  que intenta imitar la distribución de los datos reales  $P_{\text{data}}(x)$ .

### 2. Discriminador (D)

- Toma una muestra, ya sea una muestra real  $x$  del conjunto de datos o una muestra generada  $\hat{x}$  del generador, y produce un valor que representa la probabilidad de que la muestra provenga de los datos reales  $P_{\text{data}}(x)$ .

### Función de costo

- Discriminador: El discriminador se entrena para maximizar su capacidad de distinguir entre muestras reales y generadas.

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

- $x$  es una muestra real
- $G(z; \theta_G)$ : muestra generada por el generador
- $D(x; \theta_D)$ : probabilidad de que  $x$  sea una muestra real
- $\theta_D$ : parámetros del discriminador

• **Generador:  $G$**  El generador se entrena para "enganchar" el discriminador, es decir, para minimizar la capacidad del discriminador de distinguir entre las muestras reales y generadas.

$$L_G(\theta_G) = -\mathbb{E}_z [\log D(G(z; \theta_G); \theta_D)],$$

donde

$\theta_G$ : son los parámetros del generador.

**Optimización** la optimización de las GANs se realiza en 2 etapas alternas: una para actualizar los parámetros del discriminador  $\theta_D$  y otra para actualizar los del generador  $\theta_G$ .

• **Optimización del discriminador.**

El discriminador  $D$  se entrena para maximizar su función de costo  $L_D(\theta_D)$ .

$$\theta_D^{(t+1)} = \theta_D^{(t)} + \eta_D \nabla_{\theta_D} L_D(\theta_D)$$

Donde

•  $\nabla_{\theta_D} L_D(\theta_D)$  es el gradiente de la función de costo del discriminador con respecto a sus parámetros  $\theta_D$ .

•  $\eta_D$ : tasa de aprendizaje.

• **Optimización del Generador:**

El generador  $G$  se entrena para minimizar su función de costo  $L_G(\theta_G)$ .

$$\theta_G^{(t+1)} = \theta_G^{(t)} - \eta_G \nabla_{\theta_G} L_G(\theta_G)$$

Donde:

- $\nabla_{\theta} \mathcal{L}_G(\theta_G)$ : gradiente de la función de costo del generador con respecto a sus parámetros  $\theta_G$
- $\gamma_G$  es la tasa de aprendizaje para el generador.

La función de costo combinada sería

$$\hat{\theta} = \arg \min_{\theta_G} \max_{\theta_D} \mathcal{J}(\theta).$$

donde

$$\mathcal{J}(\theta) = -\mathbb{E}_x [\log D(x; \theta_D)] - \mathbb{E}_z [\log (1 - D(G(z; \theta_G); \theta_D))].$$

—

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

• discriminador toma una muestra (real ó generada) y produce una probabilidad de que la muestra sea real.

## Deep fake.

- Generador ( $G$ ): Toma muestras de una distribución latente,  $N(0, I)$ , y genera una muestra.  
 $\hat{x} = G(z; \theta_G)$ , intenta imitar la distribución de datos.

## 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.  $p_{data}(x)$

función de costo.

### Discriminación.

$$L_D(\theta_D) = -\mathbb{E}_x [\log D(x; \theta_D)] - \mathbb{E}_{\hat{x}} [\log (1 - D(\hat{x}; \theta_D))]$$

$$L_G(\theta_G) = -\mathbb{E}_{\hat{x}} [\log D(G(\hat{x}; \theta_G); \theta_D)]$$

### Optimización.

$$\theta_D^{(t+1)} = \theta_D^{(t)} + \eta_D \nabla_{\theta_D} L_D(\theta_D)$$

$$\theta_G^{(t+1)} = \theta_G^{(t)} - \eta_G \nabla_{\theta_G} L_G(\theta_G)$$

$$L(\theta) = -\mathbb{E}_x [\log D(x; \theta_D)] - \mathbb{E}_{\hat{x}} [\log (1 - D(G(\hat{x}; \theta_G); \theta_D))]$$

$$L(\theta) = (1 - \alpha) L_D(\theta) + \alpha L_G(\theta)$$

$\alpha \in [0, 1]$

- Módulo inyección de identidad (IM).
- $\text{Ada} \setminus N \rightarrow$

$$\text{Ada} \setminus N (\text{feature}) = Q_S \left( \frac{\text{face}^T - \mu(\text{face})}{\sigma(\text{face})} \right) + \mu_S.$$

~~feature: mapa de características de rostro~~  
~~características de rostro~~  
~~objetivo~~

us:

vector de identidad en bruto del rostro frontal  
media y desviación para cada uno de los canales de características

- Perdida de Identidad. (Jenky Ross).

$$L_{ID} = \frac{1}{N} \sum_{i=1}^N \| \text{face}_i^T - \mu(\text{face}) \|_2^2 \quad \text{where } \mu_i: \text{vector de identidad extraido de la imagen}$$

- Perdida de Reconstrucción

$$L_{RE} = \| \text{face}_i^T - \text{face}_i \|_2^2 \quad \text{where } \text{face}_i: \text{vector rostro fuente}$$

$$\text{losses} = \| \text{IR} - \text{IT} \|, \quad \text{where } \text{IR}: \text{imagen resultado}$$

$\text{IT}$ : Imagen Objetivo

- Cantidad de características utilizadas.

$$L_{NUM} = \sum_{i=1}^N \| D_i^T (\text{IR}) - D_i^T (\text{IT}) \|$$

- Función de pérdida total -

$$\text{Pérdida total} = \lambda_{\text{adv}} L_{\text{adv}} + \lambda_{\text{reco}} L_{\text{reco}}$$

- Pérdida adversarial -

$$L_{\text{adv}} = -\mathbb{E}_{\text{real}} [\log D(\mathcal{I}_{\text{real}})]$$

$$-\mathbb{E}_{\text{fake}} [\log (1 - D(\mathcal{I}_{\text{fake}}))]$$

minimizar

Pérdida de penalización de gradiente (GP).

$$\text{Loss} = E_i [C \|\nabla_i D(\mathcal{I}_h - 1)\|^2]$$

L1 loss

Encoder  $\xrightarrow{\text{IN}}$  Decoder

Target image  $\xrightarrow{\text{O}}$

Identity encoder

$\xrightarrow{\text{Id loss}}$

\* source image.

GAN - Loss (Real)

Loss

Gan - Loss (Fake)