



Parcial IA

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

Autoencoder Regularizado

Sea $X \in \mathbb{R}^{N \times H \times W \times C}$

con: N: muestra

H: altura de la imagen

W: Ancho de la imagen

C: Profundidad o canales

Para una muestra $\tilde{X} \in \mathcal{X}$ definimos el modelo como.

Input: $X \in \mathbb{R}^{H \times W \times C}$

Output: $\tilde{X} \in \mathbb{R}^{H \times W \times C}$

$$\hat{X} = f(X|\theta) = (l_C \circ l_{C-1} \circ \dots \circ l_1)(X|\theta)$$

Sea una salida parcial $Z_L = \hat{X}$

$$Z_L = l_L(Z_{L-1}) = V_L(Z_{L-1} \otimes \tilde{W}_L + b_L) \in \mathbb{R}^{H_L \times W_L \times D_L}$$

con $V_L: \mathbb{R}^{H_L \times W_L \times D_L} \rightarrow \mathbb{R}^{H_L \times W_L \times D_L}$

como la función de Activación en
la capa L-ésima y el vector
de representación en el espacio latente
lo definimos como

; D_L : cantidad de ritmos de
la capa L-ésima

\tilde{W}_L : pesos de la capa
L-ésima

$$Z_h = V_h(Z_{h-1} \otimes \tilde{W}_h + b_h)$$

b_h : bias de la capa
L-ésima

Para este caso $\theta \in \{\tilde{W}_L, b_L\}$ de tal manera que la
función de costo se expresa así:

$$E \{ \hat{f}(X, f(X|\theta)) \}$$

Como nuestro caso es una versión Regularizada entonces:

$$E \{ \hat{f}(X, f(X|\theta)) \} + \lambda \| \tilde{W} \|_F^2$$

Donde $\lambda(\cdot)$ es una función de costo, $\lambda \in \mathbb{R}^+$ es un coeficiente de regularización y p significa la norma p de los pesos

Para encontrar los θ^* que minimicen nuestra función de costo hacemos

$$\theta^* = \underset{\theta}{\operatorname{argmin}} E_{\mathbf{x}} \{ \lambda(\mathbf{x}, f(\mathbf{x}|\theta)) \} + \lambda \| \mathbf{W} \|_p^p$$

Por aproximación con media muestral tenemos:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^N \lambda(\mathbf{x}_n, f(\mathbf{x}_n|\theta)) + \lambda \| \mathbf{W} \|_p^p$$

Autoencoder Variacional:

Partamos del modelo del autoencoder Regularizado

Input $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$

Output $\tilde{\mathbf{x}} \in \mathbb{R}^{H \times W \times C}$

Modelo

$$E_{\mathbf{x}} \{ \lambda(\mathbf{x}, f(\mathbf{x}|\theta)) \} + \lambda \| \tilde{\mathbf{W}} \|_p^p$$

Para este caso modificamos el componente de regularización de la siguiente manera:

$$E_{\mathbf{x}} \{ \lambda(\mathbf{x}, f(\mathbf{x}|\theta)) \} + E_{\mathbf{x}_h} \{ I(x=\mathbf{x}) - I(x=\mathbf{x}, z=z_h) \}$$

$$E_{\mathbf{x}} \{ \lambda(\mathbf{x}, f(\mathbf{x}|\theta)) \} + E_{p(\mathbf{x}, \mathbf{z}_h)} \left\{ \log \left(\frac{p(\mathbf{x}, \mathbf{z}_h)}{p(\mathbf{x}) p(\mathbf{z}_h)} \right) \right\}$$

donde $P(\mathbf{x}, \mathbf{z}_h) = P(\mathbf{z}_h | \mathbf{x}) P(\mathbf{x})$

$$E_{\mathbf{x}} \{ \lambda(\mathbf{x}, f(\mathbf{x}|\theta)) \} + E_{p(\mathbf{x}, \mathbf{z}_h)} \left\{ \log \left(\frac{P(\mathbf{z}_h | \mathbf{x})}{P(\mathbf{z})} \right) \right\}$$

Al final tenemos que nuestro modelo se resume en:

$$E_{\mathbb{X}} \left\{ \mathcal{L}(\mathbb{X}, f(\mathbb{X}|\theta)) \right\} + \lambda E_{p(\mathbb{X}, \mathbb{Z}_h)} \left\{ KL(p(\mathbb{Z}_h|\mathbb{X})||P(\mathbb{Z}_h)) \right\}$$

Donde KL es la divergencia kullback-leibler

Recordemos que $\mathbb{Z}_h = V_h (\mathbb{Z}_{h-1} \otimes \tilde{W}_h + b_h)$ y necesitamos encontrar θ^* para este caso tenemos

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad E_{\mathbb{X}} \left\{ \mathcal{L}(\mathbb{X}, f(\mathbb{X}|\theta)) + \lambda E_{p(\mathbb{X}, \mathbb{Z}_h)} \left\{ KL(p(\mathbb{Z}_h|\mathbb{X})||P(\mathbb{Z}_h)) \right\} \right\}$$

Nuevamente por media muestral se tiene

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbb{X}_n, f(\mathbb{X}_n|\theta)) + \lambda \frac{1}{N} \sum_{n=1}^N KL(p(\mathbb{Z}_h|\mathbb{X}_n)||P(\mathbb{Z}_h))$$

Redes Generativas Adversarias
Generador

Input: $\mathbb{Z} \in \mathbb{R}^d$

Output: $\tilde{\mathbb{X}}^G \in \mathbb{R}^{H \times W \times C}$

$$\tilde{\mathbb{X}}^G = G(\mathbb{Z}|\theta^G)$$

$$= U_0^G U_{-1}^G \cdots U_1^G (\mathbb{Z}|\theta^G)$$

$$\tilde{\mathbb{X}}^G = \tilde{\mathbb{X}}^G_L = U_L^G (\tilde{\mathbb{X}}^G_{L-1})$$

$$= V_L^G (\tilde{\mathbb{X}}^G_{L-1} \otimes \tilde{W}^G + b^G)$$

$$\theta^G = \{\tilde{W}^G, b^G\}$$

para una muestra tenemos

Discriminador

Input: $\mathbb{X} \in \mathbb{R}^{H \times W \times C}$

Output: $\tilde{Y} \in \{0, 1\}$

$$\tilde{Y} = D(\mathbb{X}|\theta^D)$$

$$= U_0^D U_{-1}^D \cdots U_1^D (\mathbb{X}|\theta^D)$$

$$\tilde{Y} = \tilde{Y}_L^D = U_L^D (\tilde{Y}_{L-1}^D)$$

$$= V_L^D (\tilde{Y}_{L-1}^D \otimes \tilde{W}^D + b^D)$$

$$\theta^D = \{\tilde{W}^D, b^D\}$$

$\tilde{\mathbb{X}}^G_L, \tilde{Y}_L^D$ son salidas parciales de cada modelo

V_L^G, V_L^D son funciones de activación de la L -ésima capa

$$\theta = \{\theta^G, \theta^D\}$$

Si asumimos que $Z \sim P(Z)$ y que los datos reales $X \in \mathbb{R}$ siguen $P(X)$ al igual que $X \sim P(X)$

Para la función de costo la dividimos en dos partes

Discriminador: Este modelo se encarga de entregarnos la probabilidad de que la muestra X provenga de la distribución real $D(X|θ^0)$ o que la imagen generada X' provenga de la distribución de los datos generados $D(X|θ^G)$. En este modelo queremos aumentar la confianza del discriminador para identificar las muestras reales como reales y las falsas como falsas.

Por lo tanto

$$E_{X \sim P(X)} \{ \log(D(X|\theta^0)) \} + E_{Z \sim P(Z)} \{ 1 - \log(D(G(Z|\theta^G)|\theta^0)) \}$$

cada término es 1 si el Discriminador Acierta de lo contrario es 0

Generador: En el caso del Generador Este modelo busca que el término $E_{Z \sim P(Z)} \{ 1 - \log(D(G(Z|\theta^G)|\theta^0)) \}$ sea cero por lo tanto el problema de optimización es:

$$\theta^G = \arg \min_{\theta^G} \arg \max_{\theta^0} E_{X \sim P(X)} \{ \log(D(X|\theta^0)) \} + E_{Z \sim P(Z)} \{ 1 - \log(D(G(Z|\theta^G)|\theta^0)) \}$$

2.3 Consulte en qué consisten los class activation maps (CAM) y su aplicabilidad para interpretar los resultados de clasificación de imágenes con redes neuronales. Proponga un modelo de clasificación de la base de datos Fashion Mnist con regularización por autoencoders. Presente una simulación para el método GradCam++ en al menos tres capas de la red propuesta para dos clases de prueba. Nota: se sugiere utilizar el paquete keras-vis: <https://raghakot.github.io/keras-vis/>.

Los CAM son técnicas utilizadas para interpretar y visualizar la importancia de diferentes regiones de una imagen en la clasificación realizada por una Red convolucional

CAM proyecta las activaciones de la última capa convolucional en la clase objetivo utilizando los pesos de la capa de clasificación. Se definen como:

$$CAM = \sum_k w_{kc} \sum_{i,j} A_k(i,j)$$

como se observa en la ecuación la activación A_k en el mapa de características k junto con los pesos de w_{kc} que conectan A_k con la clase c se ponderan sin discriminar importancia o contribución para ello necesita como penultima capa un Average Pooling

Grad-CAM: Generalización de CAM que no se limita o se restringe a tener como penúltima capa un Average Pooling. Para ello calcula la importancia mediante el uso de gradiente de la siguiente manera

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_k(i,j)}$$

donde el mapa de calor se define como

$$C_{\text{Grad-CAM}} = \text{ReLU} \left(\sum_k \alpha_k^c A_k \right)$$

Finalmente **Grad-CAM++:** Mejora sobre Grad-CAM al considerar las contribuciones no lineales de cada neurona al resultado final.

El gradiente se estima de la siguiente manera

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \left[\frac{\partial^2 y^c}{\partial A_k(i,j)^2} + \frac{\partial y^c}{\partial A_k(i,j)} \right]$$

y el mapa de calor se estima como:

$$C_{\text{Grad-CAM++}} = \text{ReLU} \left(\sum_k \alpha_k^c A_k \right)$$

- 2.4 A partir del concepto de autoencoders y autoencoders variacionales, presente una aplicación de DeepFake sobre imágenes o vídeo. Discuta el modelo matemático empleado, los paquetes utilizados y un ejemplo ilustrativo del proceso DeepFake.

se $\mathbb{X} \in \mathcal{X}$ y $\mathbb{X} \in \mathbb{R}^{100 \times 100 \times 3}$

si definimos el encoder como

$$E(\mathbb{X} | \theta^E) = u_0^E \circ u_1^E \circ \dots \circ u_i^E (\mathbb{X} | \theta^E) = h \xrightarrow{\text{representación latente}}$$

el Decoder como

$$DE(h | \theta^{DE}) = u_0^{DE} \circ u_1^{DE} \circ \dots \circ u_i^{DE} (h | \theta^{DE}) = \hat{\mathbb{X}} \in \mathbb{R}^{100 \times 100 \times 3}$$

y un discriminador

$$D(\mathbb{X} | \theta^D) = u_0^D \circ u_1^D \circ \dots \circ u_i^D (\mathbb{X} | \theta^D) = Y \in [0, 1]$$

con la siguiente función de costo

$$E \{ L(\mathbb{X}, DE(E(\mathbb{X} | \theta^E)) | \theta^{DE})) \}$$

$$E \{ L(\mathbb{X}, DE(E(\mathbb{X} | \theta^E)) | \theta^{DE})) \} -$$

$$E_{h \sim p(h)} \{ \log(D(h)) \} - E_{h \sim p(h)} \{ \log(D(E(\mathbb{X} | \theta^E))) \}$$

Estos dos últimos términos obligan al AE a que los datos del Encoder sigan a ser semejantes a una distribución prior $p(h)$.

Para el caso de una Mezcla de Rostros se hace

$$h_A = E(\mathbb{X} | \theta^E), \quad h_B = E(\mathbb{X} | \theta^E)$$

$$h_c = h_A \oplus h_B$$

$$\hat{\mathbb{X}} = DE(h_c | \theta^{DE})$$

θ^E, θ^{DE} , son los pesos y u_i entrenados de cada modelo