Nombre | Fecha día mes año

Profesor | Materia

Institución | Curso | Nota

Funcion de costo.

Existen varios modelos de la funcion de costo:

1. MSE: Se basa en la diferencia entre los valores reales y las predicciones. Para cada muestra la diferencia se eleva al cuadrado y despues se toma el promedio de estos valores, dando como resultado la magnitud promedio del error.

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

- n: # ejemplos del conjunto de datos
- Y_i: Valor real de la i-esima muestra
- $\hat{Y}_i$: Valor predicho por el modelo

2. Entropia cruzada

$$loss = -\frac{1}{n} \sum_{i=1}^{n} \sum_{c=1}^{C} Y_{i,c} \log(\hat{Y}_{i,c})$$

- C # total de clases
- $Y_{i,c}$: es un valor binario que indica la i-esimo muestra pertenece a la clase c (1 si pertenece, 0 sino)
- $\hat{Y}_{i,c}$: es la probabilidad predicha para la clase c para la i esimo muestra

3 Kullback-Leibler Divergence (KL) mide como se diferencia una distribucion de probabilidad Q de una distribución de probabilidad P. Para distribuciones de probabilidad discretas:

$$KL(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Para continua: $KL(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{P(x)}{q(x)} dx$

- $P(x)$ y $p(x)$ funciones de masa de probabilidad
- $\log \frac{P(x)}{Q(x)}$ el el logaritmo de la ration entre los probabilidades $P(x)$ y $Q(x)$.

PAPYER

Scanned with CamScanner

Función de costo General

De manera más general, la función de costo $J(\theta)$ para una red neuronal puede estar definida en términos de una función de pérdida L que mide la discrepancia entre los valores reales 'y' y los valores predichos $\hat{y}$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y}_i)$$

- $L(y_i, \hat{y}_i)$: Función de pérdida específica
- N: # de muestras en el conjunto de datos
- θ: Parámetros del modelo (Pesos y bioses)

Función de costo y descenso del gradiente

$$\nabla_\theta J(\theta) = \left[\frac{\partial J(\theta)}{\partial \theta_1}, \frac{\partial J(\theta)}{\partial \theta_2}, \dots \frac{\partial J(\theta)}{\partial \theta_n} \right]^T$$

- $J(\theta)$: Función de costo, θ representa los parámetros
- $\nabla_\theta J(\theta)$: gradiente

El algoritmo de descenso de gradiente actualiza los parámetros θ iterativamente para minimizar la función de costo. En cada iteración t, los parámetros se actualizan según la siguiente regla:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta J(\theta_t)$$

- θ_t son los parámetros en la iteración t.
- α es la tasa de aprendizaje, un parámetro escalar que controla el tamaño del paso en la dirección opuesta al gradiente
- $\nabla_\theta J(\theta_t)$: gradiente en función de costo, este apunta en la dirección de la mayor tasa de incremento de función de costo. Para minimizar la función de costo, los parámetros se actualizan en la dirección opuesto al gradiente.

Scanned with CamScanner

Autoencoder

- Encoder

Dado un vector de entrada $x \in \mathbb{R}^n$, el codificador lo transforma en una representación latente $z \in \mathbb{R}^m$ usando la función $f_{\theta_e}(x)$, donde θ_e son los parámetros

$$z = f_{\theta_e}(x) = \sigma(W_e x + b_e)$$

- $W_e \in \mathbb{R}^{m \times n}$ matriz de pesos del codificador
- $b_e \in \mathbb{R}^m$ vector de sesgos
- σ : es la función de activación
- z : es el vector latente de dimensión reducida.

- Decoder

El decodificador toma el vector latente z y lo mapea de nuevo al espacio de entrada x usando la función $g(\theta_d(z))$ donde θ_d son los parámetros del codificador:

$$\hat{x} = g_{\theta_d}(z) = \sigma(W_d z + b_d)$$

- $W_d \in \mathbb{R}^{n \times m}$ matriz de pesos del decodificador
- $b_d \in \mathbb{R}^n$ vector de sesgos
- $\hat{x}$: reconstrucción de la entrada original.

- Función de pérdida de reconstrucción

El objetivo del autoencoder es minimizar la diferencia entre la entrada original x y la reconstrucción $\hat{x}$. Una función de pérdida MSE

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

- Regularización

Se incorpora al autoencoder para mejorar su capacidad de generalización y prevenir sobre ajuste.

- Penaliza los valores grande de los pesos

$$L_{L_2} = \lambda \sum_{j=1}^{p} \|W_j\|_2^2$$

- λ coeficiente de regularización que controla la fuerza de la penalización
- $\|W_j\|_2^2$ norma L_2 de los pesos

En general

$$z = f_{encode}(x)$$

$$\hat{x} = g_{decoder}(z)$$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[L(x_i, \hat{x}_i) + \lambda R(z_i)\right]$$

- $L(x_i, \hat{x}_i)$: Función de pérdida de reconstrucción
- $R(z_i)$: término de regularización
- λ: hiperparámetro que controla la importancia de regularización

Autoencoder Variacional (VAE)

1. Distribución latente

Se compara la distribución latente $P(z|x)$ se compara con la distribución posterior $q(z|x)$, con el fin de minimizar la diferencia entre estas dos distribuciones.

2 Bayes:

Usando el teorema de Bayes sabemos que:

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

- $p(x|z)$: verosimilitud de la reconstrucción, osea que tan bien el modelo puede reconstruir x a partir de z)
- $P(z)$ la distribución previa conocida
- $p(x)$ es la evidencia, difícil de calcular por su naturaleza.

3. KL Divergence:

Para estimar la diferencia entre $q(z|x)$ y $P(z|x)$, se usa KL, es una medida de pseudo-distancia entre funciones de densidad de probabilidad

$$D_{KL}(q\|p) = E_q\left[\log\left(\frac{q(z|x)}{p(z|x)}\right)\right]$$

Dado que no se puede calcular la evidencia, se opera, sabemos que el logaritmo de un cociente es igual a la diferencia entre ellos

- $E_q[\log(p(x|z))]$: mide que tan bien el modelo puede reconstruir la entrada x a partir de la representacion latente z, y esta relacionado con el error de reconstruccion

- DKL: entre la distribucion latente aprendido $q(z|x)$ y la distribucion previa conocida $p(z)$ y representa que tan bie el model ajusta la distribucion latente previa

En general:

$$z \sim q_\phi(z|x)$$

$$\hat{x} \sim p_\theta(x|z)$$

$q_\phi(z|x)$: codificador que aproxima la distribucion posterior

$p_\theta(x|z)$: decodificador que genera la distribucion de entrada,

y su funcion de costo es

$$J(\theta, \phi) = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \| p(z))$$

Redes Neuronales adversarias (GANs)

En una GAN, se tienen dos modelos:

- Generador (G): Trata de generar datos que sean indistinguibles de datos reales.

- Discriminador (D): Intenta distinguir entre datos reales y datos generados

El objetivo del generador es engañar al discriminador para que clasifique incorrectamente los datos generados como reales

El objetivo del discriminador es mejorar su capacidad para distinguir entre datos reales y generados.

1 funcion de costo del discriminador:

Dado el conjunto de datos reales x y un conjunto de datos generados $G(z)$, el discriminador $D(x)$ trata de maximizar la probabilidad de clasificar correctamente los datos. su funcion de costo es:

$$J_D(\theta_D) = -E_{x \sim data}[\log D(x)] - E_{z \sim p_z}[\log(1 - D(G(z)))]$$

$E_{x \sim pdata(x)}$: Esperanza sobre los datos reales

$E_{z \sim p_z(z)}$: Esperanza sobre el espacio latente.

$D(x)$: la probabilidad asignada pa el discriminador de que x es un dato real

$G(z)$: la salida del generador para una entrada z del espacio latente

2 Funcion de costo del generador

El generador G busca maximizar la probabilidad del que el discriminador clasifique los datos generados como reales. Su funcion es:

$$J_G(\theta_G) = -E_{z\sim p_z}[\log D(G(z))]$$

3. Gradiente

- Discriminador:

$$\theta_D \leftarrow \theta_D - \alpha_D \nabla_{\theta_D} J_D(\theta_D)$$

- Generador

$$\theta_G \leftarrow \theta_G - \alpha_G \nabla_{\theta_G} J_G(\theta_G)$$

- α_D y α_G: tazos de opiendizaje

Modelo completo:

$$\min_G \max_D E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Class Activation Maps (CAM)

Tecnica utilizada para interpretar y visualizar que partes de una imagen influyen más en la decisión de una red neuronal al realizar una clasificación. CAM se emplea principalmente en el contexto de redes neuronales convolucionales (CNN) y permite identificar regiones claves de la imagen que contribuyen a predecir una determinada clase

funcionamiento

- Capas convolucionales: en una CNN extraen caracteristicas especiales de la imagen detectando patrones como bordes, texturas, formas, etc. estan caractenzadas en mapas caracteristicas (feature maps)
- Capa de clasificacion: Despues de las capas convolucionales, la salida de estas se conecta a una capa totalmente conectada (Fully connected) o una capa de prediccion, donde cada neurona representa una clase diferente

Scanned with CamScanner

- Generación del CAM: las Class Activation Maps identifican las regiones de la imagen activan más los mapas de características en las últimas capas convolucionales de la red.
Se realiza una ponderación de estos mapas de características utilizando los pesos de la capa de clasificación

Esto genera un mapa visual que indica qué partes de la imagen son más relevantes para la predicción de una clase en particular

El CAM para una clase c se calcula de manera matemáticamente:

$$CAM_c(x,y) = \sum_k w_k^c f_k(x,y).$$

$f_k(x,y)$ es el valor del mapa de características en la posición (x,y) en el canal k (salida de la última capa convolucional).

w_k^c son los pesos aprendidos que conectan el mapa de características k con la clase c

Grad-CAM

Gradient-weighted class Activation Maps es una extensión que permite aplicar la técnica a una mayor variedad de arquitecturas de CNN, incluidas aquellas sin una capa de promedio global. Utiliza gradientes de la clase objetivo con respecto a los mapas de características para ponderar la importancia de cada característica

Deepfakes

Están basados en modelos de aprendizaje profundo, especialmente en redes generativas adversarias (GANs) y autoencoders variacionales (VAEs).

los GANs utilizan enfoques adversarial donde el generador y el discriminador están bastante competencia. El generador crea imágenes que intentan engañar al discriminador, mientras que el discriminador intenta identificar las imágenes generadas como falsas. los VAEs y modelos de transformadores también se utilizan para tareas específicas relacionadas con deepfakes, como la codificación de identidades y la generación de contenido complejos. Estos modelos se entrenan para mejorar la calidad y el realismo de las imágenes o videos generados.

Finalmente se podrá decir que el modelo matemático mencionado Deepfakes se basa en Autoencoder variacional (VAE)

Scanned with CamScanner