

Sistemas de Inteligencia Artificial

Autoencoders

Centro de Inteligencia Computacional

2022

Autoencoders

Autoencoders

- Arquitectura de Redes Neuronales No Supervisadas.
- Reducción de la Dimensionalidad.
- Son la base de algunos modelos de redes neuronales generativas.

¿Que pasa si la salida final de una MLP es una entrada para otra MLP inversa?

Autoencoder

Arquitectura

- Dos redes neuronales artificiales de perceptrones multicapa, donde la salida de la primera red se conecta con la entrada de la segunda.
- La segunda red tiene la distribución invertida de neuronas en las capas y como salida tiene la misma dimensión que la entrada de la primera red.

Formulación General

- $Z = f(X) = h(XW + b)$
- $X' = g(Z) = h(ZV + p)$

Esto debe satisfacerse de manera que $L(X, X') = \|X - X'\|^2$.

Autoencoder

Aprendizaje

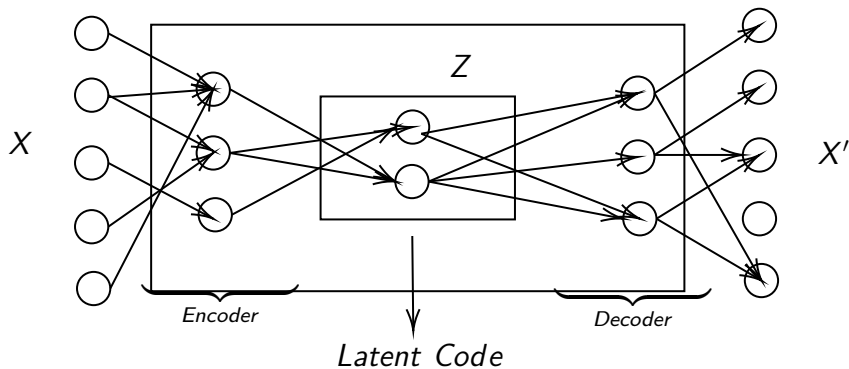
- La red es entrenada por cualquier método de aprendizaje que sirva para un Multi-Layer Perceptron, pero para cada patrón de entrada X se pone como salida esperada el mismo patrón X .
- La red por lo tanto aprende los pesos sinápticos que generan en la salida X' , el mismo valor presentado a la entrada.

Formulación General

- $Z = f(X) = h(XW + b)$
- $X' = g(Z) = h(ZV + b)$

Esto debe satisfacerse de manera que $L(X, X') = \|X - X'\|^2$.

Función identidad



Autoencoder

¿Qué utilidad puede representar en definitiva una función identidad?

Restricciones

El chiste está justamente en que es posible agregar restricciones que fuercen al autoencoder a aprender algo útil de la distribución del set de datos. La restricción natural que tiene por la arquitectura es que al utilizar menos neuronas en la capa central, fuerza a encontrar una codificación más eficiente del conjunto de datos.

Autoencoder lineal

Descomposición Espectral

Repasando la clase de Aprendizaje No supervisado y PCA, si una matriz X es invertible y sus autovalores son reales, entonces la matriz puede ser diagonalizable y factorizarse

$$X = ELE^T \quad (1)$$

Donde L es la matriz diagonal formada por los autovalores y E es la matriz de filas formada por sus autovectores.

Autoencoder lineal

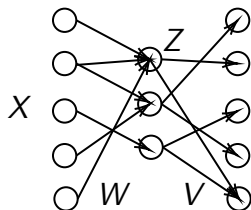
Descomposición en valores singulares

Para el caso de matrices no cuadradas, existe una operación similar pero más general que es la descomposición en valores singulares, SVD. Dada una matriz X de $n \times d$,

$$SVD(X) = \tilde{Z}\Sigma V^T \quad (2)$$

Donde \tilde{Z} son los vectores columnas singulares de izquierda, Σ es una matriz diagonal positiva y V son los vectores singulares derechos. El cálculo de SVD se realiza minimizando $\|X - \tilde{Z}\Sigma V^T\|$.

Autoencoder lineal



Una vez que el autoencoder aprende minimiza

$$J = ||X - ZV^T|| \quad (3)$$

$$X \approx ZV^T \quad (4)$$

- X : La matriz de datos de $n \times d$. Hay n datos de dimensión d .
- Z : La salida de la capa interna del autoencoder, del código de $n \times k$.
- V : La matriz de pesos sinápticos asociada al Decodificador de $k \times d$.

Autoencoder lineal

Lemma: PCA

La salida del código interno Z del autoencoder lineal son las salidas de las proyecciones de los datos en los componentes principales.

Autoencoder lineal

SVD

Si V^T , que tiene los pesos del decoder del Autoencoder, es ortonormal, entonces $X \approx ZV^T$ del Autoencoder puede verse como la descomposición en valores singulares $X \approx SVD(X) = \tilde{Z} \Sigma V^T$ con $\tilde{Z}\Sigma = Z$, ya que la descomposición de SVD surge de resolver la minimización de la Ecuación 3.

Autoencoder lineal

PCA

A su vez, por la descomposición espectral, con los autovectores de la matriz de covarianza $\frac{X^T X}{n-1}$, se puede obtener $\frac{X^T X}{n-1} = E L E^T$ donde E es la matriz de transformación de PCA con los autovectores.

Autoencoder lineal

Entonces, reemplazando X como la factorización por $X \approx SVD(X) = \tilde{Z} \Sigma V^T$:

Matriz de covarianza de X

$$\begin{aligned}\frac{X^T X}{n-1} &= (\tilde{Z} \Sigma V^T)^T (\tilde{Z} \Sigma V^T) \frac{1}{n-1} \\ \frac{X^T X}{n-1} &= (V^T)^T \Sigma^T \tilde{Z}^T \tilde{Z} \Sigma V^T \frac{1}{n-1} \\ \frac{X^T X}{n-1} &= V \Sigma^T \Sigma V^T \frac{1}{n-1} = V \Sigma^2 V^T \frac{1}{n-1}\end{aligned}$$

Autoencoder lineal

Matriz de covarianza de X

$$\frac{X^T X}{n-1} = V \Sigma^2 V^T \frac{1}{n-1} = V \left(\frac{\Sigma^2}{n-1} \right) V^T$$

Pero por la descomposición espectral, esto es también igual a $E(L) E^T = \frac{X^T X}{n-1} = V \left(\frac{\Sigma^2}{n-1} \right) V^T$ por lo que si X tiene media cero,

- $\lambda_i = S_i^2 / n - 1$, con S_i los valores singulares de Σ .
- $V = E$

Entonces al transformar los datos con PCA

$$T_{PCA}(X) = X E = \underbrace{(\tilde{Z} \Sigma V^T)}_I E = \tilde{Z} \Sigma = Z \text{ que corresponde}$$

a la salida del espacio latente.

Autoencoder lineal

Lemma: PCA

La salida del código interno Z del autoencoder lineal son las salidas de las proyecciones de los datos en los componentes principales.

$$T_{PCA}(X) = X E = Z \quad (5)$$

Autoencoder lineal

Reducción no lineal de la dimensionalidad

Esto no es una manera muy eficiente de obtener las proyecciones en los componentes principales, pero lo que permite es pensar al autoencoder como una extensión no lineal de la descomposición en componentes principales (cuando se usan funciones de activación no lineales). Es una suposición fuerte.

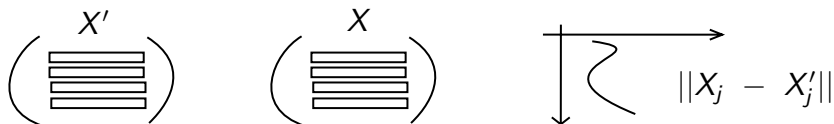
DAE - Denoising autoencoder

Usos

El anglicanismo **Denoising** hace referencia a la eliminación del ruido. Como el autoencoder genera una aproximación de la matriz de datos X en base a $X \approx X' = Z V^T$ entonces esto permite utilizarse para dos tareas.

- Identificador de outliers
- Eliminación del ruido.

DAE - Denosing Autoencoder



Detección de outliers

- Se entrena al autoencoder con el conjunto de entrenamiento.
- Luego se lo somete a nuevas muestras que no necesariamente se encuentran en el conjunto.
- Se mide con alguna medida la diferencia entre cada valor de entrada del autoencoder y los valores obtenidos en la salida.

DAE - Denoising autoencoder

Eliminación de ruido

Ruido se le llama a perturbaciones sobre los datos que contaminan el contenido de información que se quiere recuperar. Hay muchos tipos diferentes de ruidos y son inherentes a cualquier proceso de obtención de información. Es el mal de todos, imposible de eliminar por completo.

Eliminación de ruido

Los DAE pueden utilizarse para eliminar ruido ya que la estructura interna del Encoder/Decoder intenta preservar el contenido de información más relevante en el conjunto de datos.

DAE - Denoising autoencoder

Ruido

La idea entonces es una vez aprendido un conjunto de datos, cuando aparecen nuevas muestras que pueden estar contaminadas con ruido, reemplazar la muestra ruidosa por la muestra obtenida en la salida del Decodificador.

Para esto, en vez de entrenar con los elementos obtenidos directamente del conjunto de datos, se modela el ruido, es decir, se lo caracteriza numericamente (e.g. salt-and-pepper) o en general con una función de distribución de probabilidad (e.g. Gaussiano, Rayleigh) y se agrega a los datos X generando una nueva instancia \tilde{X} . Sin embargo, la salida que se pone en la red corresponde al dato X sin alterar.

Regularización

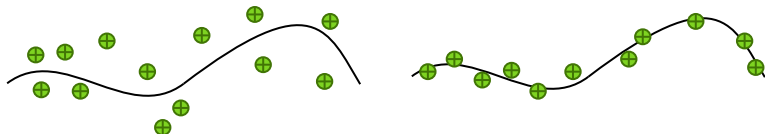
Regularizar implica agregar una condición extra R sobre la función objetivo L que limita el espacio de búsqueda de las soluciones.

$$J = L + \lambda R = \min_{\phi} \frac{1}{N} \|Y - X\phi\| \underbrace{(+\lambda \|f(\phi)\|)}_{\text{Término Regularizador}} \quad (6)$$

Desalienta las soluciones complejas o extremas en un problema de optimización. El parámetro de regularización λ se usa justamente para ajustar la importancia que se le da a al término regularizador (Tikhonov Regularization).

CAE - Contractive Autoencoder

Los CAE agregan un término regularizador a las funciones de costo, que le resta sensibilidad a la entrada, intentando aprender representaciones más simples y crudas de los datos. De esa manera por ejemplo, datos en dos dimensiones pueden reducirse a una dimensión, como una representación de una función de dos dimensiones, que pasa a una representación paramétrica (con un solo parámetro t).



SAE - Sparse Autoencoder

Los SAE por otro lado, utilizan una arquitectura donde el espacio latente tiene más dimensiones de la entrada. Para lograr imponer restricciones que permitan adquirir información de la estructura de los datos, imponen un término regularizador que favorece la presencia de esparcidad, esto es, que un número de neuronas del espacio latente estén en 0 anuladas. Los SAE se pueden utilizar para realizar **Feature Learning**, es decir para identificar caracterizaciones o transformaciones de los datos que son útiles para discriminarlos. La Esparcidad se impone mediante una restricción umbralizada, que de no superarse desactiva todas las neuronas involucradas.

Deep Generative Model

Deep Generative Model

Modelado Generativo o Discriminativo

- **Modelo Discriminativo:** es crudo en los datos. Por ejemplo, una ANN busca la hiper-sábana que separa los datos en dos clases sin importarle cómo se generan (o como se podrían) generar esos datos.
- **Modelo Generativo:** establece cierta relación causal. El modelo hipotetiza cómo se generan los datos en sí.

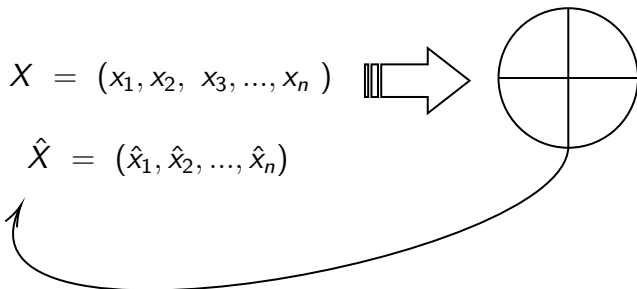
Deep Generative Model

Modelo Generativo

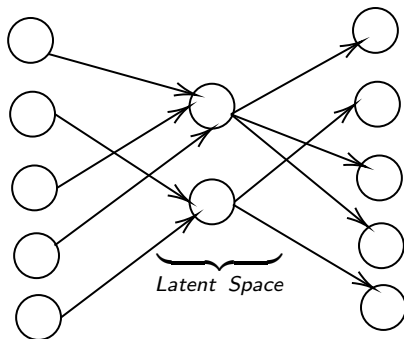
La pregunta que surge entonces es si es posible utilizando una red neuronal, implementar un mecanismo generativo causal, que me permita obtener muestras que comparten características representativas de un conjunto de datos.

¿Con lo que vieron hasta ahora, cómo se imaginan un mecanismo así?

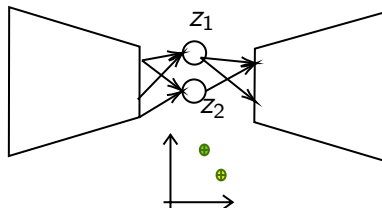
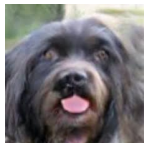
Modelo Generativo



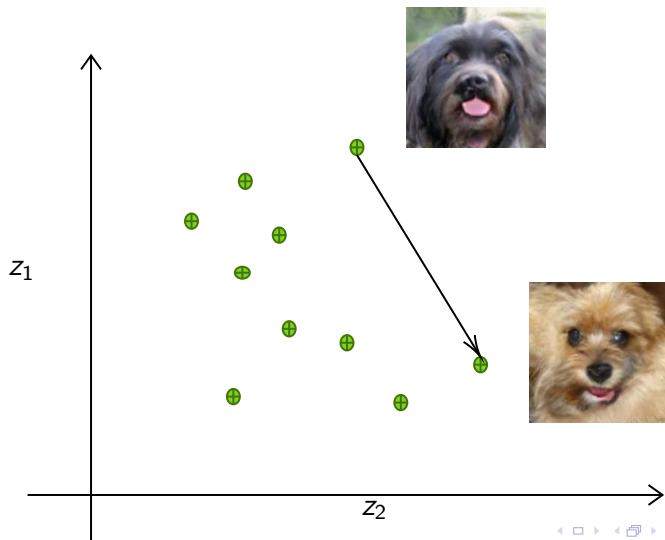
Generative Autoencoder



Generative Autoencoder



Generative Autoencoder: Concept Vector

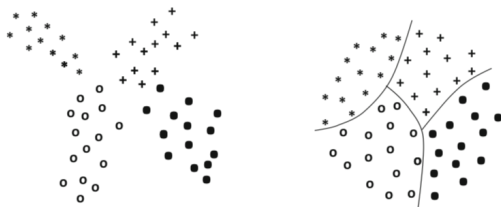


Generative Autoencoder

Algoritmo del Autoencoder Generativo

- Utilizamos un Autoencoder para codificar en el Espacio Latente todos los patrones.
- Luego dejamos de lado el Encoder, para movernos dentro de un vector de representación.
- Generamos muestras sucesivas especificando de manera directa los valores de z_1 y z_2 .
- Para cada tupla de valores de z_i obtenemos una nueva muestra generada por el Decodificador.

Generative Autoencoder



Generative Autoencoder

La conquista del espacio latente

Necesitamos entonces algo más que nos permita **capturar** lo que pasa en el espacio latente, para poder generar nuevas muestras **válidas** a medida que nos movemos en ese espacio latente.

Aproximación de una pdf

¿Si tengo un generador de números pseudoaleatorios que se corresponde con una distribución de probabilidad uniforme, y tengo una p.d.f., ¿Cómo puedo usar el generador para generar números aleatorios que se corresponden con la p.d.f. ?

Inverse Sampling Theorem

$$Y \sim U[0, 1], F \text{ invertible y } F = CDF(pdf()), \\ X = F^{-1}(Y) \text{ tiene } CDF(X) = F$$

$$Y \sim U[0, 1], F = CDF(pdf()), X = \inf_x \{x : F(x) \geq Y\}$$

Aproximación de una pdf

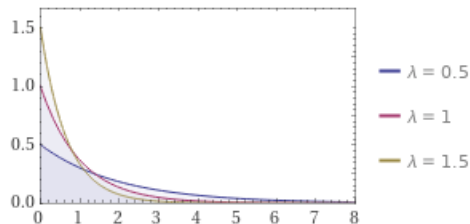
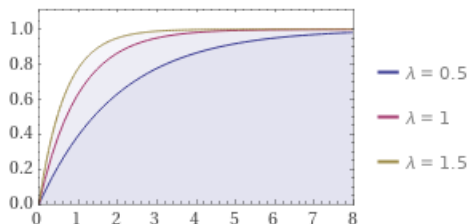
Eligen un valor de $Y \sim U[0, 1]$, se fijan que les da en $X = F^{-1}(Y)$. La distribución de esos X va a corresponder con la *pdf* en cuestión.

Inverse Sampling Theorem

$Y \sim U[0, 1]$, F invertible y $F = CDF(pdf())$,
 $X = F^{-1}(Y)$ tiene $CDF(X) = F$

$Y \sim U[0, 1]$, $F = CDF(pdf())$, $X = \inf_x \{x : F(x) \geq Y\}$

Aproximación de una p.d.f.



Aproximación de una p.d.f.

Teorema Central del Limite

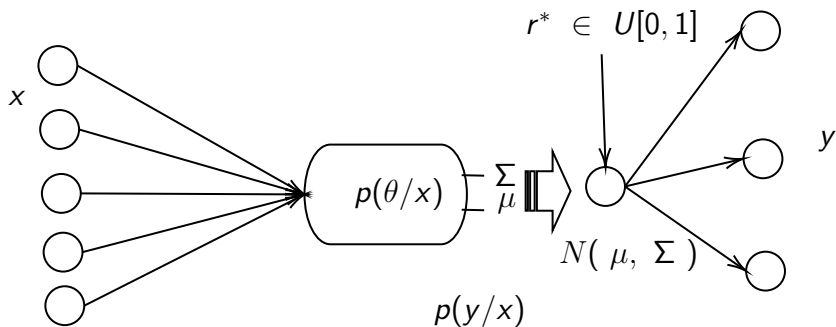
Para el caso de Gaussianas, se puede utilizar el TCL, obtener un número grande de $y \in U[0, 1]$ y con eso generar correspondiente $N(\frac{n}{2}, \frac{n}{12})$.

Stochastic Feedforward Neural Networks

Ahora, ¿ Cómo podríamos hacer eso con una red neuronal ?

Una red de una capa oculta puede representar cualquier función computable (Arbitrary Continuous Functions, Cybenki's Theorem). Entre otras cosas, puede representar una función de probabilidad, una p.d.f.

Stochastic Feedforward Neural Networks



Teoría de la Información

Variable aleatoria: $X = x_i$

Información: $I = -\log p(x_i)$

Entropía: Más o menos, cuanta info tienen todos los estados posibles que puede tomar x . Promedio ponderado.

$$H = \mathbb{E}_{p(x_i)} I(x_i) = -\sum p(x_i) \log p(x_i)$$

Información Mutua:

$$I(x, y) = H(y) - H(y/x) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

$H(x) = I(x, x)$, y se tiene que,

- $I = 0$ implica que $p(x) = 1$ Certeza !
- $I > 0$ algo de información
- $I(x_k) > I(x_i) \Rightarrow P_k < P_i$, lo más raro (menos probable) aporta más información.

Teoría de la Información

La entropía es mínima, cero, cuando es una situación de total certeza (teniendo en cuenta que $\lim_{p \rightarrow 0^+} p \log p = 0$), y es máxima, cuando todos los eventos son equiprobables (probabilidad uniforme).

$$0 \leq H(x) \leq \log(2k + 1)$$

Entropía condicional: $H(x/y) = H(x, y) - H(y)$

Entropía conjunta: $H(x, y) = \sum_{i=1}^m \sum_{j=1}^m p(x_i, y_j) \log \frac{1}{p(x_i, y_j)}$

Cross Entropy

Cross Entropy

$$H_p(q) = - \sum_{k=1} q(y_k) \log p(y_k)$$

Binary Cross Entropy

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N \{(y_i) \log p(y_i) + (1 - y_i) \log(1 - p(y_i))\}$$

$q(y_k) = y_i = 0|1$, y representa el label, $p(y_i)$, $1..N$ es la probabilidad asignada a ese label para la muestra i . Binary Cross Entropy sirve como una función para estimar cuál es la efectividad de un clasificador que asigna el label y a cada elemento de $1..N$ con probabilidad $p(y_k)$. Un clasificador perfecto que estima y_i con probabilidad 1, genera un valor de binary cross entropy de cero.

KL: Kullback-Leibler Divergencia ó Entropía relativa

Es una "norma" (en rigor no) que permite ver la distancia entre distribuciones de probabilidad, i.e. si tiende a cero implica que las distribuciones son parecidas. Una medida de similaridad.

KL

$$KL(q||p) = - \sum_{x_i} q(x) \log \frac{q(x)}{p(x)}$$

donde $p(x)$ es la referencia a la que se compara $q(x)$.

KL: Kullback-Leibler Divergencia ó Entropía relativa

$$KL(q||p) = - \sum_{x_i} q(x) \log \frac{q(x)}{p(x)}$$

Propiedades:

- $KL(q||p) \neq KL(p||q)$
- $KL(q||q) = 0$
- $KL(q||p) \geq 0$
- $KL(q||p) = H_p(q) - H(q) \geq 0$

Deep Generative Model

VAE - Variational Autoencoder

Los VAE son la convergencia de dos ideas.

- Una estimación variacional de un modelo generativo[14, 10, 7, 11].
- Una solución a esa estimación basada en dos redes neuronales feedforward acopladas, un Encoder y un Decoder, que conforman un Autoencoder.

Inferencia Variacional

Inferencia Variacional

Peterson, Anderson 1987 (Mean Field Theory) [2]

Jordan 1999 [18]

Hinton, Van Camp 1993 [9]

Estimación Bayesiana[6]

Estimación Bayesiana

Sea S el espacio de muestreo, z_i , $i = 1, \dots, n$ una partición de S y x otro evento en S

$$\underbrace{P(z_i | x)}_{\text{Posterior}} = \frac{\overbrace{P(x | z_i)}^{\text{Likelihood}} \overbrace{P(z_i)}^{\text{prior}}}{P(x)} = \frac{P(x | z_i) P(z_i)}{\sum_{i=1}^n P(x | z_i) P(z_i)}$$

El teorema de bayes permite calcular una probabilidad condicional $P(z_i | x)$, en base a su conjugada $P(x | z_i)$.

Estimación Bayesiana

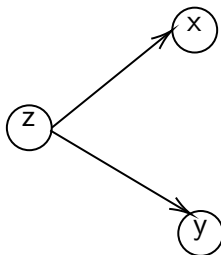
Por ejemplo, asumiendo que x es una característica observable, puedo empíricamente obtener los likelihood (contar las muestras de una clase particular z_i que tienen x). Luego esa información usarla para estimar la posterior.

Naive Bayes

Con esto construyo un clasificador simple, basado justamente en evaluar esa probabilidad condicional. Entonces cuando se observa x , determinar de que tipo es, z_1 o z_2 .

Podemos chequear si $P(z_1/x) \stackrel{?}{\geq} P(z_2/x)$

Redes Bayesianas



Teorema de la factorización

$$p(x, y, z) = p(x/z)p(y/z)p(z)$$

Inferencia



Problema de Inferencia

$$p(z/x) = \frac{p(x/z)p(z)}{p(x)}$$

¿Qué cosas podría tener en z (no observable) en base a lo que observo en x ?

El problema es que $p(x)$, la evidencia marginal, es muy difícil de calcular. Por probabilidad total: $p(x) = \int p(x/z)p(z)$. Esto puede ser una integral/sumatoria en cada una de las dimensiones de z . Puede que no tenga una representación analítica (closed-form) o incluso que sea intractable (e.g. requiere tiempo exponencial para computarse).

Inferencia



Problema de Inferencia

$$p(z/x) = \frac{p(x/z)p(z)}{p(x)}$$

Otra alternativa, podría ser mediante MCMC [16], Markov Chain Monte Carlo, que es una técnica que estima estas distribuciones realizando un sampleo de todos los valores, pero puede demorar mucho tiempo.

Inferencia Variacional

La inferencia variacional intenta resolver este problema mediante la solución de un problema de optimización.

Optimización del Problema de Inferencia

Objetivo

Tratar de aproximar $p(z/x)$ con una $q(z)$, donde esta última la podemos armar para que sea una función que queremos (e.g. una Gaussiana).

Optimización del Problema de Inferencia

Objetivo

Tratar de aproximar $p(z/x)$ con una $q(z)$, donde esta última la podemos armar para que sea una función que queremos (e.g. una Gaussiana).

Mimización

$$\text{mín } KL(q||p) = - \sum q(x) \log \frac{p(x)}{q(x)}$$

Optimización del Problema de Inferencia

$$\begin{aligned}KL(q(z)||p(z/x)) &= - \sum q(z) \log \frac{p(z/x)}{q(z)} \\&= - \sum q(z) \log \frac{\frac{p(z,x)}{p(x)}}{\frac{q(z)}{1}} \\&= - \sum q(z) \log \frac{p(z,x)}{q(z)} \frac{1}{p(x)} \\&= - \sum q(z) \{ \log \frac{p(z,x)}{q(z)} - \log p(x) \}\end{aligned}$$

Optimización del Problema de Inferencia

$$KL(q(z)||p(z/x))$$

$$\begin{aligned} &= - \sum q(z) \{ \log \frac{p(z, x)}{q(z)} - \log p(x) \} \\ &= - \sum_z q(z) \log \frac{p(x, z)}{q(z)} + \sum_z q(z) \log p(x) \\ &= - \sum_z q(z) \log \frac{p(x, z)}{q(z)} + \log p(x) \underbrace{\sum_1 q(z)} \end{aligned}$$

Optimización del Problema de Inferencia

$$KL(q(z)||p(z/x)) = - \sum_z q(z) \log \frac{p(x, z)}{q(z)} + \log p(x) \quad (7)$$

$$\log p(x) = KL(q(z)||p(z/x)) + \sum_z q(z) \log \frac{p(x, z)}{q(z)} \quad (8)$$

Optimización del Problema de Inferencia

$$\underbrace{\log p(x)}_{\text{fijo para } x} = \underbrace{KL(q(z)||p(z/x))}_{\text{lo más pequeño posible}} + \underbrace{\sum_z q(z) \log \frac{p(x, z)}{q(z)}}_{\mathcal{L}, \text{ lo más grande posible}} \quad (9)$$

Optimización del Problema de Inferencia

$$\mathcal{L} \leq \log p(x) \quad (10)$$

Variational Lower Bound, tiene que ser lo más grande posible y cuando $KL \rightarrow 0$ entonces es exactamente el $\log p(x)$.

Optimización del Problema de Inferencia

$$\begin{aligned}\mathcal{L} &= \sum q(z) \log \frac{p(x, z)}{q(z)} \\ &= \sum q(z) \log \frac{p(x/z)p(z)}{q(z)} \\ &= \sum q(z) \{ \log p(x/z) + \log \frac{p(z)}{q(z)} \} \\ &= \sum q(z) \log p(x/z) + \sum q(z) \log \frac{p(z)}{q(z)}\end{aligned}$$

Optimización del Problema de Inferencia

$$\begin{aligned}\mathcal{L} &= \sum q(z) \log p(x/z) & + \sum q(z) \log \frac{p(z)}{q(z)} \\ \mathcal{L} &= \mathbb{E}_{q(z)} \log p(x/z) & - KL(q(z) || p(z))\end{aligned}$$

Optimización del Problema de Inferencia

Ingredientes:

$$p_{\theta}(x/z), p_{\theta}(z), q_{\phi}(z)$$

$$\theta^*, \phi^* = \arg \max \mathcal{L}(\theta, \phi, x)$$

Podemos elegir una distribución para $p_{\theta}(z)$ que queramos, luego estimar $p_{\theta}(x/z)$ sampleando, y finalmente elegir otra distribución para $q_{\phi}(z)$. Con eso obtenemos una expresión para \mathcal{L} y luego *optimizarla* encontrando los valores para los parámetros de esas distribuciones, ϕ^*, θ^* .

Resolución de un problema de optimización

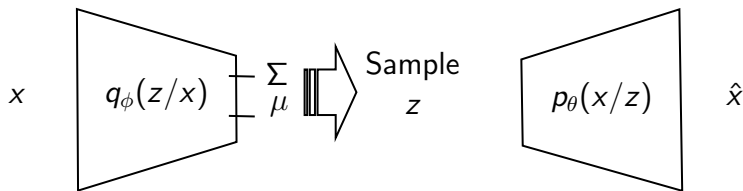
Ingredientes:

$$p_{\theta}(x/z), p_{\theta}(z), q_{\phi}(z)$$

$$\theta^*, \phi^* = \arg \max \mathcal{L}(\theta, \phi, x)$$

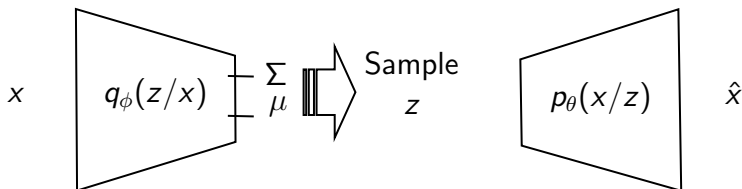
Es decir, necesito en \mathcal{L} optimizar al mismo tiempo $p_{\theta}(x/z)$ para que dado un z sampleado cualquiera, me genere un valor de x válido, y necesito optimizar $q_{\phi}(z/x)$ para que dada un valor de x real me genere parámetros para poder samplear z . Todo a la vez.

Autoencoder Variacional



Acá es donde se conecta con Autoencoders: puedo usar un Encoder para dado un x real, generar parametros para una $q(z)$, que se usa para generar valores de z que luego son sampleados y se usan en un Decoder, que los tiene que volver a convertir en un \hat{x} . Esos parámetros que se estiman θ^*, ϕ^* son los pesos de las dos redes.

Autoencoder Variacional[5]



Lo bueno es que el funcional \mathcal{L} ya incluye ambas cosas que se necesitan para optimizar la red.

$$-\mathcal{L} = - \underbrace{\mathbb{E}_{q(z)} \log p(x/z)}_{\text{Error de reconstrucción}} + \underbrace{KL(q(z)||p(z))}_{\text{Término regularizador}}$$

(maximizar \mathcal{L} es equivalente a minimizar $-\mathcal{L}$).

Variational Autoencoder

Asumimos que:

- $p_{\theta}(z) = \mathcal{N}(0, \mathcal{I})$
- $q_{\phi}(z) = \mathcal{N}(\mu(x), \Sigma(x))$, con $\Sigma(x)$ diagonal.

Variational Autoencoder

Error de Reconstrucción:

$$p_{\theta}(x/z) = \frac{1}{\sqrt{(2\pi)^k} \sqrt{\Sigma(z)}} \exp\{(x - \mu(z))^T \Sigma(z)^{-1} (x - \mu(z))\}$$

$$\log p_{\theta}(x/z) \propto (x - \underbrace{\mu(z)}_{\hat{x}})^T \Sigma(z)^{-1} (x - \underbrace{\mu(z)}_{\hat{x}})$$

Variational Autoencoder

Término Regularizador:

$$KL(q_{\phi}(z) = \mathcal{N}(\mu(x), \Sigma(x)) || p_{\theta}(z) = \mathcal{N}(0, \mathcal{I}))$$

$$\begin{aligned} KL &= \frac{1}{2} \{ \text{trace}(\Sigma(x)) + \mu^T \mu(x) - k - \log(|\Sigma(x)|) \} \\ &= \frac{1}{2} \left\{ \sum_k \Sigma(x) + \sum_k (\mu(x))^2 - \sum_k 1 - \log(\prod_k \Sigma(x)) \right\} \\ &= \frac{1}{2} \left\{ \sum_k \Sigma(x) + \sum_k (\mu(x))^2 - \sum_k 1 - \sum_k (\log \Sigma(x)) \right\} \\ &= \frac{1}{2} \sum_k (\Sigma(x) + (\mu(x))^2 - 1 - \log \Sigma(x)) \end{aligned}$$

Variational Autoencoder

Término Regularizador:

$$KL = \frac{1}{2} \sum_k (\Sigma(x) + (\mu(x))^2 - 1 - \log \Sigma(x))$$

Por razones numéricas, se reemplaza $\Sigma(x) = \exp(\Sigma(x))$.

$$KL = \frac{1}{2} \sum_k (\exp \Sigma(x) + (\mu(x))^2 - 1 - \Sigma(x))$$

Variational Autoencoder[15]

$$J = L + \lambda R$$

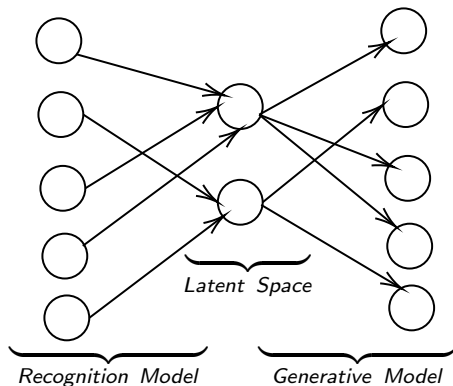
$$\text{máx } \mathcal{L} = \mathbb{E}_{q(z)} \log p(x/z) - KL(q(z)||p(z))$$

$$\text{mín } \mathcal{L} = -\mathbb{E}_{q(z)} \log p(x/z) + KL(q(z)||p(z))$$

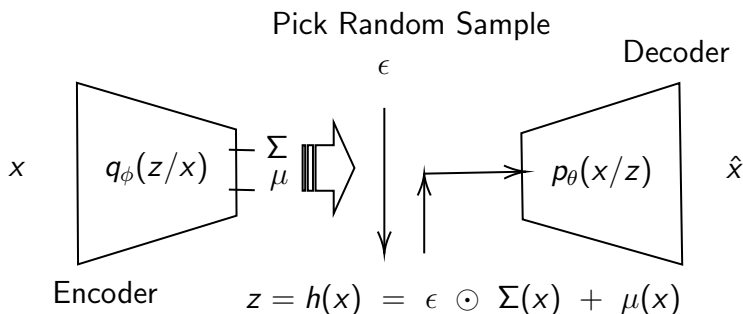
$$\text{mín } \mathcal{L} = \|\bar{X} - \bar{X}'\| + \frac{1}{2} \sum_k (\exp \Sigma(x) + (\mu(x))^2 - 1 - \Sigma(x))$$

$$\text{mín } \mathcal{L} = \|\bar{X} - \bar{X}'\| - \frac{1}{2} \sum_k (1 + \Sigma(x) - (\mu(x))^2 - \exp \Sigma(x))$$

Variational Autoencoder

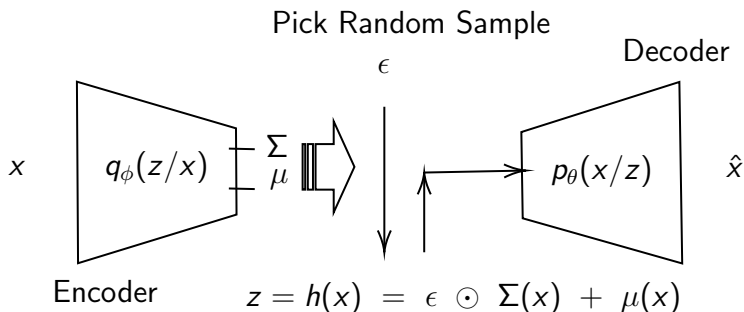


Algoritmo del Autoencoder Variacional



Recapitulando(1/2): Tomamos un x y lo ponemos en la entrada de la red. Con ese x vemos que valores de Σ y μ obtenemos en la salida del Encoder. Luego sampleamos un z de $p(z) = \mathcal{N}(0, \mathcal{I})$ (esto es equivalente a obtener un ϵ multiplicarlo por la varianza p.a.p. y sumarle la media).

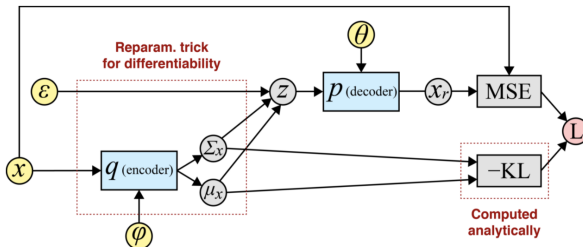
Algoritmo del Autoencoder Variacional



Recapitulando(2/2): Con ese z lo pasamos por el Decoder y obtenemos un \hat{x} . Entrenamos al Autoencoder ahora para que minimice \mathcal{L} actualizando primeros los pesos del Decoder, luego pasando por la función $h(x)$ y de ahí retropropagando el error obtenido en μ y Σ hacia los pesos del Encoder.

Variational Autoencoder

El *Parametrization Trick* de la capa estocástica de la red, permite poder retropropagar el error manejando esa capa que tiene una entrada extra que es una variable aleatoria [8]:

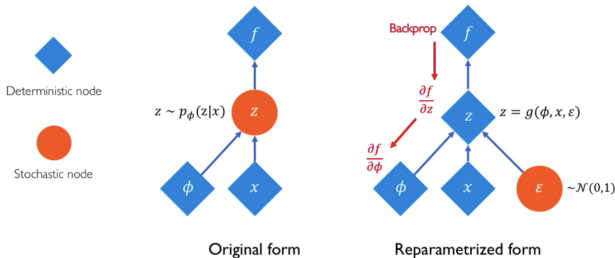


Fijense que el término regularizador, NO depende de la salida del decoder.

Variational Autoencoder

El valor de z se calcula en función de $\bar{\mu}$ y de $\bar{\Sigma}$.

$$z = h(\bar{X}) = \epsilon \odot \bar{\Sigma}(\bar{X}) + \bar{\mu}(\bar{X}) \quad (11)$$



Reparametrization Trick

¿Cómo propagar[17] el error por el nodo estocástico del encoder?

Supongamos que queremos actualizar un peso particular del encoder ω_e . Vamos a suponer que z tiene dimensión 1. Con una dimensión las dos salidas del encoder, actúan como perceptrones de salida.

$$J = L + \lambda R$$

$$\frac{\partial J}{\partial \omega_e} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial \mu} \frac{\partial \mu}{\partial \omega_e} + \lambda \frac{\partial R}{\partial \mu} \frac{\partial \mu}{\partial \omega_e}$$

$$\frac{\partial J}{\partial \omega_e} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial \sigma} \frac{\partial \sigma}{\partial \omega_e} + \lambda \frac{\partial R}{\partial \sigma} \frac{\partial \sigma}{\partial \omega_e}$$

Colab

`https://colab.research.google.com/drive/
1Hobi6plfrrUQ9DCiFPZ6vazt0u5Q05ND`

Referencias

- Inferencia Variacional [3].
- Metodos Variacionales [13].
- VAE [4].
- Autoencoders [1, 12].

Referencias I

- [1] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018.
- [2] James R Anderson and Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.
- [3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Feb 2017.
- [4] Carl Doersch. Tutorial on variational autoencoders, 2021.
- [5] ErmonGroup. *Variational Autoencoder*, 2020 (accessed April 3, 2021).
<https://ermongroup.github.io/cs228-notes/extras/vae/>.
- [6] Evjang. *Variational Bayes*, 2016 (accessed April 3, 2021). <https://blog.evjang.com/2016/08/variational-bayes.html>.

Referencias II

- [7] Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.
- [8] Gregoy Gundersen. *Reparametrization Trick*, 2021 (accessed October 2, 2021). <https://gregorygundersen.com/blog/2018/04/29/reparameterization/>.
- [9] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.
- [10] jhu edu. *Variational Tutorial*, 2011 (accessed September 3, 2018). <http://www.cs.jhu.edu/~jason/tutorials/variational.html>.
- [11] Mohammad Emtiyaz Khan and Guillaume Bouchard. Variational em algorithms for correlated topic models. Technical report, Technical report, University of British Columbia, 2009.

Referencias III

- [12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.
- [13] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.
- [14] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [15] Stephen Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function, 2019.
- [16] Baptise Rocca. *MCMC*, 2021 (accessed October 2, 2021). <https://towardsdatascience.com/bayesian-inference-problem-mcmc-and-variational-inference-2>

Referencias IV

- [17] Baptise Rocca. *Stack Exchange, Machine Learning*, 2021 (accessed October 2, 2021).
<https://stats.stackexchange.com/questions/420974/backpropagation-on-variational-autoencoders>.
- [18] Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.