Modelos Generativos

Nahuel Costa

Grado en Ciencia e Ingeniería de Datos

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Flow-based models

Modelos generativos como las redes GAN o los VAEs no aprenden explícitamente la función de densidad de los datos, $p(x)$.

Como ya vimos, esto es entendible ya que si tomamos como ejemplo un modelo generativo típico con variables latentes, $p(x) = \int p(x|z)p(z)$, difícilmente se puede llegar a computar de forma explícita porque es prácticamente imposible recorrer todos los valores de z.

Flow-based models

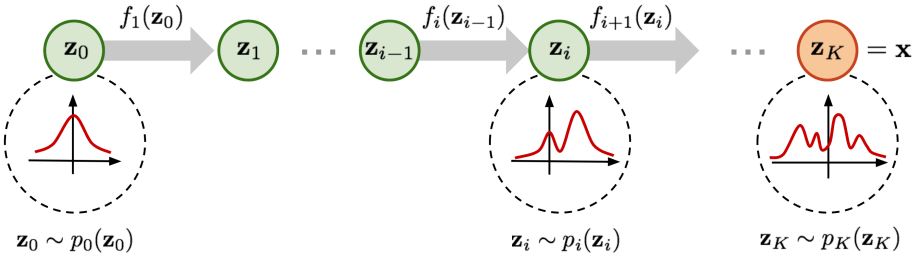
Los "flow-based models" aproximan este problema por medio de los **normalizing flows**, un método que permite la estimación de densidad de los datos. Una buena estimación de $p(x)$ hace posible realizar eficientemente muchas tareas como muestrear nuevos puntos de datos no observados pero realistas (generación de datos), predecir la rareza de eventos futuros, inferir variables latentes, rellenar muestras de datos incompletas, etc.

Normalizing flows

Dado que para entrenar modelos de aprendizaje profundo utilizamos backpropagation, se espera que la distribución de probabilidad a posteriori $p(z|x)$ sea lo suficientemente simple como para calcular la derivada de forma fácil y eficiente. Por eso se suele utilizar la distribución gaussiana en los modelos generativos de variables latentes, aunque la mayoría de las distribuciones del mundo real sean mucho más complicadas que la gaussiana.

Normalizing flows

Un normalizing flow transforma una distribución simple, $p_0(z_0)$, en una compleja, $p_K(z_k)$, aplicando una secuencia de funciones de transformación invertibles. "Fluyendo" a través de una cadena de transformaciones, se sustituye repetidamente la variable z_i por la nueva z_{i+1} para finalmente obtener una distribución de probabilidad acorde a la variable objetivo final.



Normalizing flows

Tanto la transformación directa como su inversa se pueden calcular exactamente. Esto permite realizar la estimación de densidad. Para ello, se deben tener en cuenta dos cosas:

- La densidad de la muestra transformada inversamente: para obtener la muestra transformada se aplica la secuencia de transformaciones inversas a la muestra original. A continuación, se evalua la densidad de esta muestra transformada bajo la distribución simple original.

Normalizing flows

- El cambio de volumen debido a las transformaciones: a medida que se realizan las transformaciones, el espacio de la muestra se distorsiona. El cambio de volumen se calcula multiplicando los valores absolutos de los determinantes de las matrices jacobianas de cada transformación.

Multiplicando estos dos valores (la densidad de la muestra transformada y el cambio de volumen), se obtiene la densidad de la muestra original bajo la distribución compleja.

Aplicaciones

Las aplicaciones más directas de los normalizing flows son:

- Estimación de densidad: para calcular la densidad exacta de los datos. Se pueden aplicar para ajustar densidades multimodales a los datos observados. También, pueden utilizarse como modelos híbridos que modelan la densidad conjunta de entradas y objetivos $p(x|y)$, a diferencia de los modelos de clasificación que sólo modelan $p(y|x)$ y los modelos de densidad que sólo modelan $p(x)$. Esto es útil para tareas como la detección de anomalías.

- Generación de datos: para diferentes modalidades de datos, incluyendo imágenes, vídeo, audio, texto y objetos estructurados como gráficos y nubes de puntos.

Aplicaciones

- Inferencia: para modelar distribuciones posteriores variacionales en modelos de variables latentes. También se pueden utilizar para guiar simulaciones con el fin de hacer la inferencia más eficiente. Este enfoque se ha utilizado para la inferencia de modelos de simulación en cosmología [1] y neurociencia computacional [2].

Matriz jacobiana

Dada una función de mapeo de un vector de entrada n-dimensional a un vector de salida m-dimensional, la matriz de todas las derivadas parciales de primer orden de esta función se denomina matriz jacobiana:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Se puede entender como un traductor de un espacio vectorial a otro.

Determinante

El valor absoluto del determinante (sólo existe para matrices cuadradas) puede considerarse como una medida de "cuánto expande o contrae el espacio la multiplicación por la matriz".

El determinante de una matriz cuadrada M detecta si es invertible: si $det(M) = 0$, entonces no es invertible (una matriz singular con filas o columnas linealmente dependientes; o cualquier fila o columna es toda 0); en caso contrario, si $det(M) \neq 0$, entonces M es invertible.

El determinante del producto es equivalente al producto de los determinantes: $det(AB) = det(A)det(B)$.

Teorema del cambio de variable

Dada una variable aleatoria z y su función de densidad de probabilidad conocida $z \sim \pi(z)$ queremos construir una nueva variable aleatoria utilizando una función de asignación 1-1, $x = f(z)$. La función f es invertible, por lo que $z = f^{-1}(x)$. Ahora, la cuestión es cómo inferir la función de densidad de probabilidad desconocida de la nueva variable, $p(x)$?

$$\int p(x)dx = \int \pi(z)dz = 1;$$

$$p(x) = \pi(z)\left|\frac{dz}{dx}\right| = \pi(f^{-1}(x))\left|\frac{df^{-1}}{dx}\right| = \pi(f^{-1}(x))\left|(f^{-1})'(x)\right|$$

Teorema del cambio de variable

Por definición, la integral $\int \pi(z)dz$ es la suma de un número infinito de rectángulos de anchura infinitesimal Δz. La altura de dicho rectángulo en la posición z es el valor de la función de densidad $\pi(z)$. Cuando sustituimos la variable, $z = f^{-1}(x)$ da $\frac{\Delta z}{\Delta x} = (f^{-1}(x))'$ y $\Delta z = (f^{-1}(x))'\Delta x$.

Aquí $\left|(f^{-1}(x))'\right|$ indica el cociente entre el área de rectángulos definidos en dos coordenadas distintas de variables z y x respectivamente.

Aplicación a normalizing flows

De acuerdo a la figura 5:

$$z_{i-1} \sim p_{i-1}(z_{i-1})$$

$$z_i = f_i(z_{i-1}), \text{ por lo que } z_{i-1} = f_i^{-1}(z_i)$$

$$p_i(z_i) = p_{i-1}(f_i^{-1}(z_i)) \left| \det \frac{df_i^{-1}}{dz_i} \right|$$

Aplicación a normalizing flows

Convertiendo la ecuación para que sea una función de z_i se puede hacer inferencia con la distribución base:

$$p_i(z_i) = p_{i-1}(f_i^{-1}(z_i)) \left| \det \frac{df_i^{-1}}{dz_i} \right|$$

$$= p_{i-1}(z_{i-1}) \left| \det \left(\frac{df_i}{dz_{i-1}} \right)^{-1} \right| \quad ; \text{Según el teorema de la función inversa.}$$

$$= p_{i-1}(z_{i-1}) \left| \det \frac{df_i}{dz_{i-1}} \right|^{-1} \quad ; \text{Según las propiedades de los jacobianos.}$$

$$\log p_i(z_i) = \log p_{i-1}(z_{i-1}) - \log \left| \det \frac{df_i}{dz_{i-1}} \right|$$

Aplicación a normalizing flows

Dada tal cadena de funciones de densidad de probabilidad, conocemos la relación entre cada par de variables consecutivas. Podemos ampliar la ecuación de x paso a paso hasta llegar a la distribución inicial z_0:

$$x = z_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)$$

Aplicación a normalizing flows

$$
\begin{aligned}
\log p(x) &= \log \pi_K(z_K) = \log \pi_{K-1}(z_{K-1}) - \log \left| \det \frac{df_K}{dz_{K-1}} \right| \\
&= \log \pi_{K-2}(z_{K-2}) - \log \left| \det \frac{df_{K-1}}{dz_{K-2}} \right| - \log \left| \det \frac{df_K}{dz_{K-1}} \right| \\
&= \cdots \\
&= \log \pi_0(z_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{dz_{i-1}} \right|
\end{aligned}
$$

Aplicación a normalizing flows

El camino recorrido por las variables aleatorias $z_i = f_i(z_{i-1})$ es el flujo y la cadena completa formada por las distribuciones sucesivas π_i se denomina **normalizing flow**.

Requerido por los cálculos que se necesitan hacer, una función de transformación f_i debe satisfacer dos propiedades:

1. Que sea invertible
2. Que su determinante jacobiano sea fácil de calcular

Modelos basados en Normalizing flows

Algunas arquitecturas populares de normalizing flows son Real NVP [3] (2016), Masked Autoregressive Flows [4], Glow [5] (2018, por Kingma - mismo autor que VAE y del optimizador Adam), SurVAE [6] (mezcla de VAEs y normalizing flows).

Las diferencias entre estas arquitecturas residen en las transformaciones que aplican y diseños de red específicos, pero todas comparten el objetivo común de transformar una distribución simple en una más compleja.

Recursos

Aquí puedes encontrar más recursos sobre normalizing flows.

[1] J. Alsing, T. Charnock, S. Feeney, and B. Wandelt, "Fast likelihood-free cosmology with neural density estimators and active learning," *Monthly Notices of the Royal Astronomical Society*, vol. 488, no. 3, pp. 4440–4458, 2019.

[2] P. J. Gonçalves, J.-M. Lueckmann, M. Deistler, M. Nonnenmacher, K. Öcal, G. Bassetto, C. Chintaluri, W. F. Podlaski, S. A. Haddad, T. P. Vogels, *et al.*, "Training deep neural density estimators to identify mechanistic models of neural dynamics. biorxiv," 2019.

[3] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[4] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," *Advances in neural information processing systems*, vol. 30, 2017.

[5] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.

[6] D. Nielsen, P. Jaini, E. Hoogeboom, O. Winther, and M. Welling, "Survae flows: Surjections to bridge the gap between vaes and flows," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12685–12696, 2020.