Autoencoders

César Olivares

Pontificia Universidad Católica del Perú
Maestría en Informática
INF659 - Técnicas avanzadas de data mining y sistemas inteligentes

2018

Aprendizaje No Supervisado / Motivación

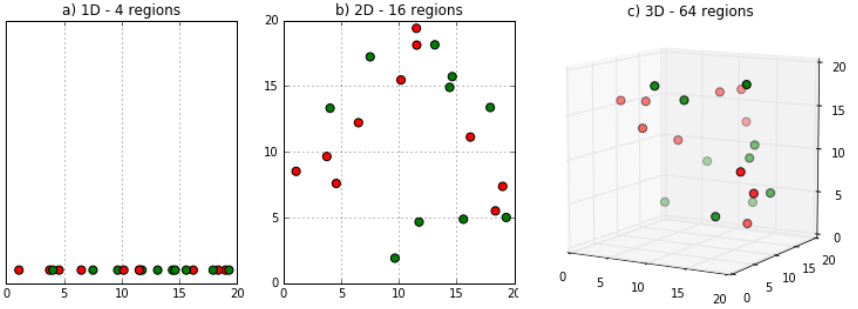
- El **Aprendizaje No Supervisado** no tiene como objetivo predecir valores ni clases, sino *descubrir patrones y estructura* en conjuntos de datos **no etiquetados**.
- A diferencia del aprendizaje supervisado, no tenemos una etiqueta o variable dependiente y
- ¿Podemos identificar grupos de datos o de características?
- ¿Cómo podemos representar o incluso visualizar de manera compacta muchos datos con muchas características?
- ¿Podemos en general aprender una buena representación de los datos?
- ¿Podemos generar nuevos ejemplos?
- ¿Podemos determinar qué tan probable es un punto x particular?
- En el actual estado del arte, los algoritmos de aprendizaje supervisado requieren grandes cantidades de datos etiquetados para alcanzar una buena exactitud.

Reducción de la dimensionalidad / Motivación

- A mayor dimensionalidad, mayor tiempo de aprendizaje y espacio de almacenamiento.
- El número de posibles configuraciones crece exponencialmente y dificulta la generalización.
- La complejidad computacional de modelar la distribución probabilística de los datos crece también exponencialmente.
- A menudo se tiene mucha redundancia de información en las características
- Eliminar la colinealidad de las características puede mejorar el rendimiento del modelo de aprendizaje.
- Es más fácil visualizar datos en bajas dimensiones (2D, 3D).

Maldición de la dimensionalidad

- En espacios con altas dimensiones, cada punto termina estando muy lejos de prácticamente todos los demás.
- Conforme aumenta el número de dimensiones o características, el número de datos requeridos para generalizar con exactitud crece exponencialmente.
- La maldición de la dimensionalidad afecta severamente a los modelos de aprendizaje, sobre todo a los que dependen de medidas de distancia entre los puntos.
- Algunas características irrelevantes podrían estar introduciendo ruido en las distancias relevantes entre los puntos.



Fuente: http://www.kdnuggets.com/2017/04/must-know-curse-dimensionality.html

Análisis de componentes principales (PCA)

- El Análisis de Componentes Principales (PCA por sus siglas en inglés *Principal Component Analysis*) es una técnica de aprendizaje no supervisado que permite transformar un conjunto de observaciones con características numéricas **correlacionadas** entre sí en un conjunto de valores compuestos de variables **no correlacionadas** entre sí.
- Esta transformación se realiza mediante la identificación de un conjunto de ejes ortogonales de rotación, llamados **componentes principales**, que colectivamente explican al máximo la varianza del conjunto de datos original.
- El Análisis de Componentes Principales hace referencia al procedimiento para calcular estos componentes principales y a su posterior uso en la comprensión de los datos.
- Los componentes principales obtenidos se ordenan según la proporción de la varianza total explicada por cada uno.

Análisis de componentes principales (PCA) / Ejemplo

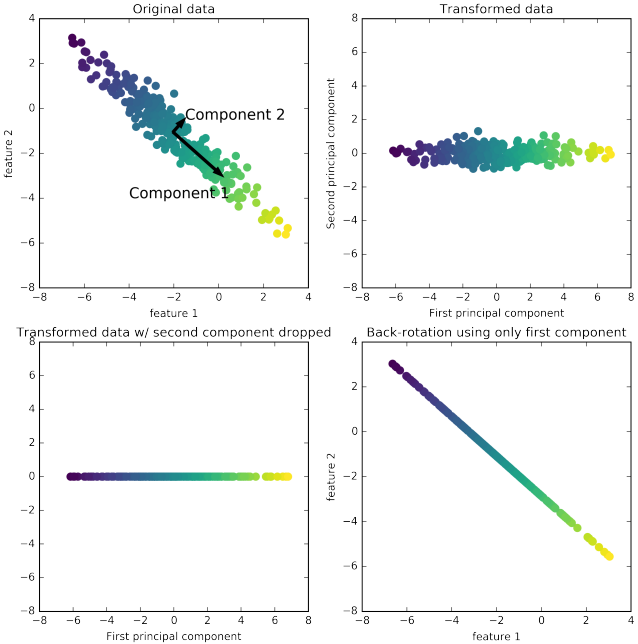


Figura 1: Transformación de datos con PCA (Mueller 2016)

Análisis de componentes principales / Visualización 3D

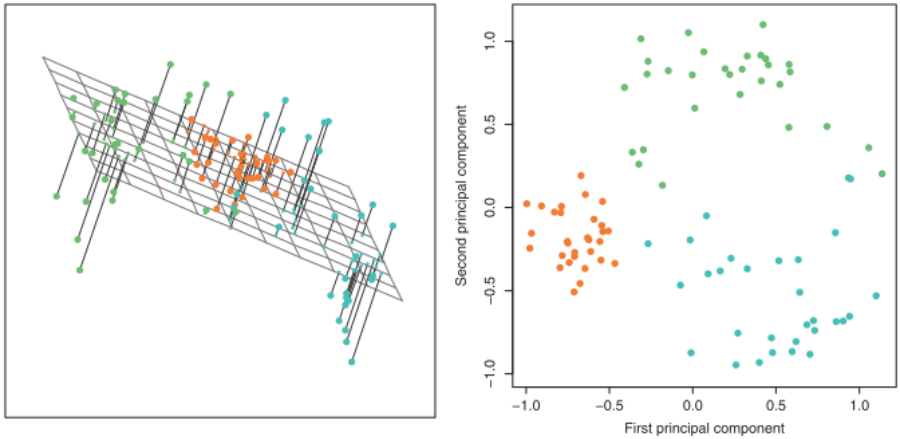


Figura 2: Noventa observaciones simuladas en tres dimensiones. Izquierda: las direcciones de los dos componentes principales definen el plano que mejor se ajusta a los datos y minimiza la suma de distancias cuadradas de cada punto al plano. Derecha: los vectores de puntajes de los dos componentes principales dan las coordenadas de proyección de las 90 observaciones sobre el plano. La varianza en el plano es máxima. (James 2013)

Análisis de componentes principales / Explicación con SVD

- PCA se puede explicar de diversas maneras. Aquí lo mostraremos a partir de la Descomposición en valores singulares (SVD).
- Toda matriz $n \times p$ puede ser descompuesta de manera única (salvo el signo de los componentes principales) como el producto de tres matrices con propiedades especiales:

$$X = U\Sigma V^\top$$

- U es una matriz $n \times r$
- *Sigma* es una matriz $r \times r$
- $V^\top$ es una matriz $r \times p$
- Si $p < n$, entonces $r = p$.
- U y V son ortogonales (por lo tanto, rotaciones).
- *Sigma* es diagonal (por lo tanto, un ajuste de tamaño).
- Las filas de $V^\top$ corresponden a los **vectores de cargas** de los principales componentes de X.
- Las columnas de $U\Sigma$ corresponden a los **vectores de puntajes** de los principales componentes de X.

Modelos de factores lineales

Modelos de factores lineales

- La investigación en aprendizaje profundo requiere construir buenos modelos probabilísticos de las entradas, $p_{model}(x)$.
- Modelos de este tipo pueden ser usados para inferir cualquier variable en su entorno dada cualquier otra de las variables.
- A menudo estos modelos incluyen variables latentes h, de manera que $p_{model}(x) = \mathbb{E}_h p_{model}(x|h)$
- Estas variables latentes brindan una nueva manera de representar los datos.
- Los modelos de factores lineales son algunos de los más simples modelos probabilísticos con variables latentes.
- Un **modelo de factores lineales** se caracteriza por el uso de una función de decodificación lineal y estocástica que genera x añadiendo ruido a una transformación lineal de h.

Modelos de factores lineales / Generación de datos

- Para generar datos se realiza dos pasos:
  1. Se muestrea los factores explicatorios h de una distribución

     $$h \sim p(h),$$

     donde $p(h)$ es una distribución factorial, y $p(h) = \Pi_i p(h_i)$
  2. Se muestrea los valores observados x dados los factores h:

     $$x = Wh + b + \text{ruido},$$

     donde el ruido suele ser gausiano y diagonal (independiente entre sus dimensiones)

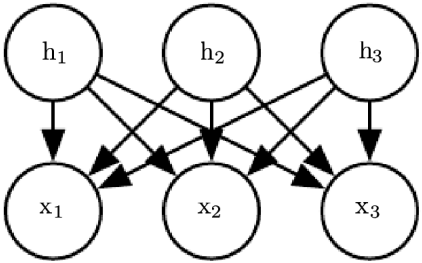


Figura 3: Modelo gráfico dirigido que describe a la familia de modelos de factores lineales, donde $x = Wh + b +$ ruido. (Goodfellow 2016)

Modelos de factores lineales / Ejemplos

- Los principales modelos de factores lineales se diferencian en la forma específica del ruido y de la distribución a priori $p(h)$:
  - Análisis de factores
    - La distribución a priori de h es gausiana con varianza 1: $h \sim \mathcal{N}(h; 0, I)$.
    - Se asume que las variables observadas x_i son condicionalmente independientes dada h.
    - El ruido es muestreado de un distribución de covarianza gausiana con matriz de covarianza $\psi = diag(\boldsymbol{\sigma}^2)$
    - Las variables latentes capturan las dependencias entre las variables observadas x_i
  - PCA probabilístico
    - Modifica el análisis de factores asumiendo que las varianzas condicionales σ_i^2 son iguales entre sí, con lo que $x = Wh + b + \sigma z$, donde $z \sim \mathcal{N}(z; 0, I)$.
    - Se convierte en PCA (determinístico) conforme $\sigma \to 0$.
    - Dada una entrada x, estima una distribución sobre h, en vez de un valor determinístico.
    - Estima una función de densidad de probabilidad.
    - Puede generar muestras.
  - Otros modelos
    - Análisis de componentes independientes (ICA)
    - Análisis de características lentas (SFA)
    - Codificación dispersa

PCA interpretado como variedad

- Se puede interpretar que los modelos de factores lineales, incluidos PCA y el análisis de factores aprenden una variedad en un espacio hiperdimensional.
- Se podría decir que el PCA probabilístico define un región de alta probabilidad en forma de «panqueque», una distribución gausiana muy delgada en algunos de sus ejes.
- En este contexto, el PCA determinístico estaría alineando este «panqueque» con una variedad lineal en ese espacio hiperdimensional.

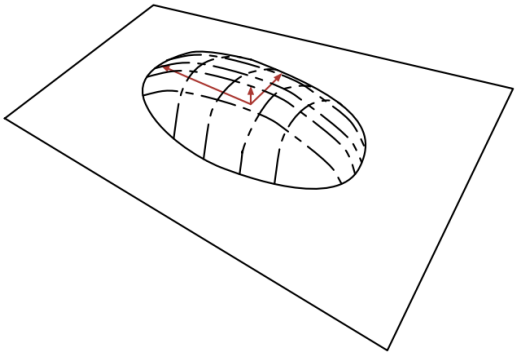


Figura 4: Interpretación de PCA como una variedad. (Goodfellow 2016)

Autoencoders

Autoencoders

- Un *autoencoder* es una red neuronal en la que se desea obtener como salida un vector idéntico al de entrada, con el objetivo de que las unidades ocultas **h** en una capa intermedia aprendan buenos «códigos» de representación.
- No requieren supervisión alguna (los datos no necesitan estar etiquetados)
- La red se compone de dos partes: una función de codificación $h = f(x)$ y una de decodificación que produce una reconstrucción $r = g(h)$.
- Dependiendo del tipo de representaciones deseadas, y para evitar que el autoencoder simplemente copie las entradas, se restringe las capas ocultas, p.ej, limitando su tamaño.
- Son muy importantes para tareas tales como reducción de la dimensionalidad, extracción de características, pre-entrenamiento no supervisado, modelos generativos, recuperación de información, entre otras.

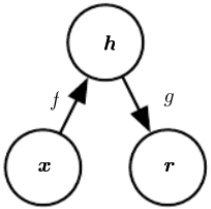


Figura 5: Estructura general de un autoencoder. (Goodfellow 2016)

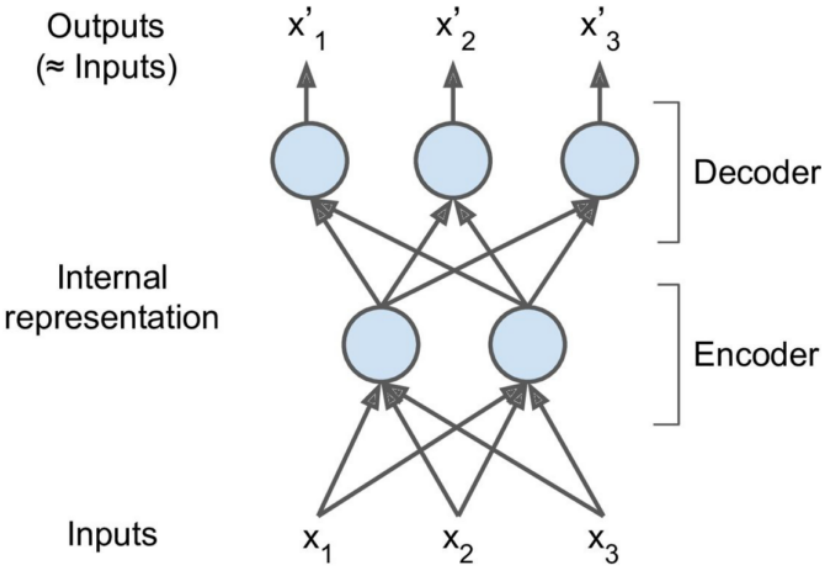


Figura 6: El experimento de memoria de Chase & Simon y un autoencoder simple. (Géron 2017)

Autoencoders subcompletos

- Son aquellos cuyo código tiene menor dimensionalidad que las entradas.
- Restringir el tamaño del código fuerza al autoencoder a capturar las características más relevantes de las entradas.
- Si la función de decodificación es lineal y se mide la pérdida con el error cuadrático medio, el autoencoder aprende a describir el mismo supespacio lineal que PCA.
- Si se usa funciones no lineales, el autoencoder tiene mucha más capacidad y generaliza PCA al aprendizaje de subespacios curvos.

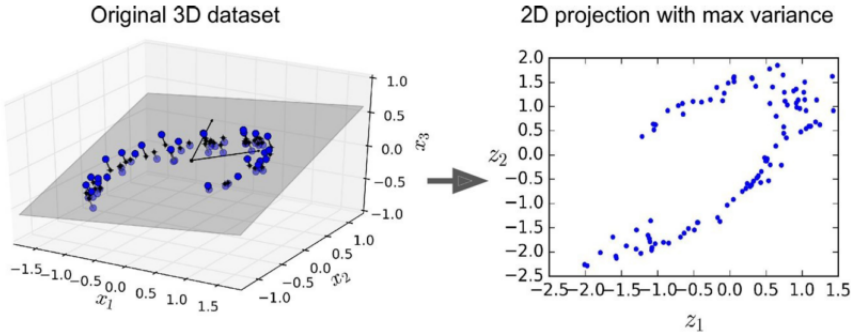


Figura 7: PCA realizado por un autoencoder subcompleto lineal. (Géron 2017)

Autoencoders profundos

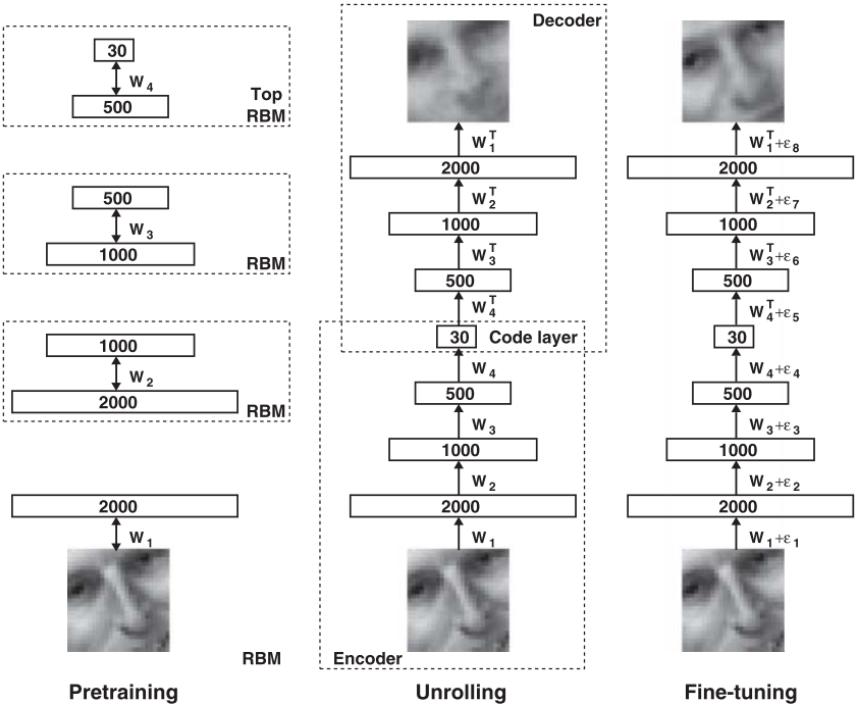


Figura 8: Autoencoder subcompleto profundo. (Hinton 2006)

Autoencoders profundos

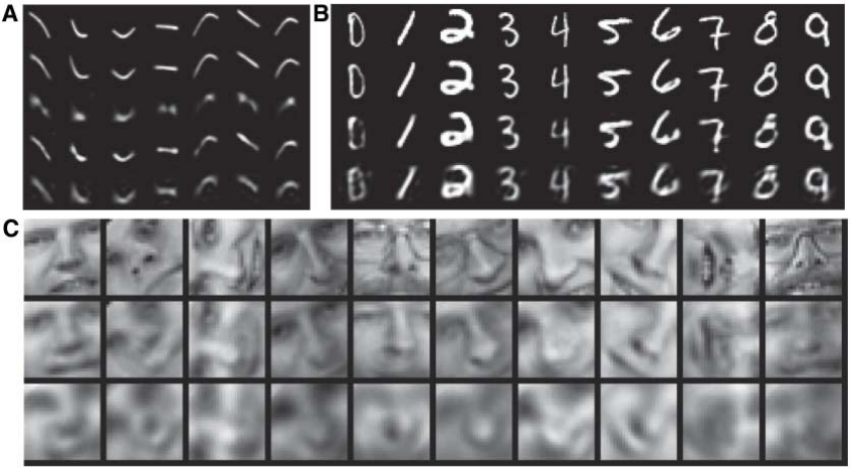


Figura 9: (A) De arriba a abajo: Imágenes originales; reconstrucciones de un autoencoder profundo de 6 dimensiones; PCA logístico de 6 componentes; PCA logístico y PCA estándar con 18 componentes. (B) De arriba a abajo: Imágenes originales; reconstrucciones de un autoencoder de 30 dimensiones; PCA logístico y PCA estándar de 30 dimensiones. (C) De arriba a abajo: Imágenes originales; reconstrucciones de un autoencoder de 30 dimensiones; PCA estándar de 30 dimensiones. (Hinton 2006)

Autoencoders profundos

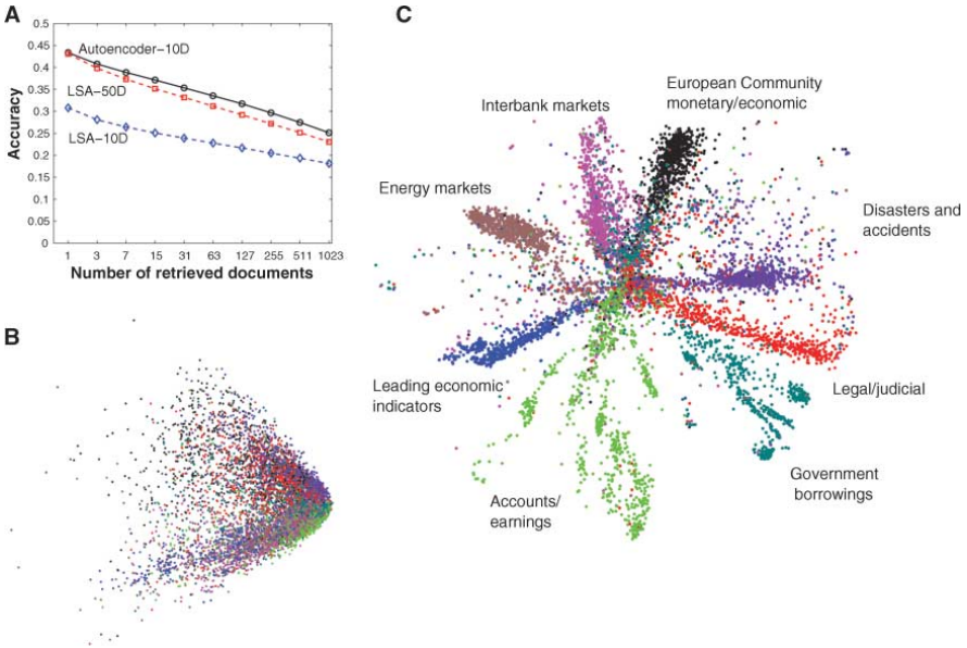


Figura 10: (A) Recuperación de información con un autoencoder 2000-500-250-125-10. (B) Códigos producidos por LSA de 2 dimensiones. (C) Códigos producidos por un autoencoder 2000-500-250-125-2. (Hinton 2006)

Autoencoders dispersos

- Otra manera de restringir las unidades ocultas de un autoencoder es reduciendo el número de unidades activas, p.ej., que en promedio sólo 5 % de las unidades de la capa h estén activas.
- El tamaño de cada lote de entrenamiento (*batch*) no debe ser muy pequeño para que la media pueda ser precisa.
- En la función de costo se añade un término que penaliza las unidades más activas que lo deseado. En lugar del error cuadrático medio se suele usar la divergencia de Kullback-Leibler (entre la activación promedio deseada ρ y la activación promedio $\hat{\rho}_j$ de cada unidad j de la capa h:

$$D_{KL}(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{(1-\rho)}{(1-\hat{\rho}_j)}$$

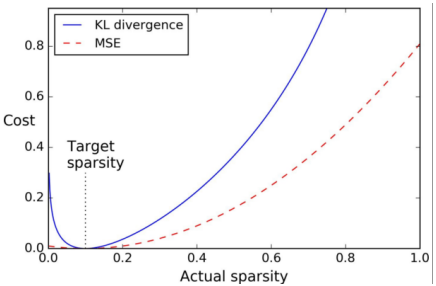


Figura 11: Medidas de pérdida para dispersión (Géron 2017)

Autoencoders dispersos altamente sobrecompletos

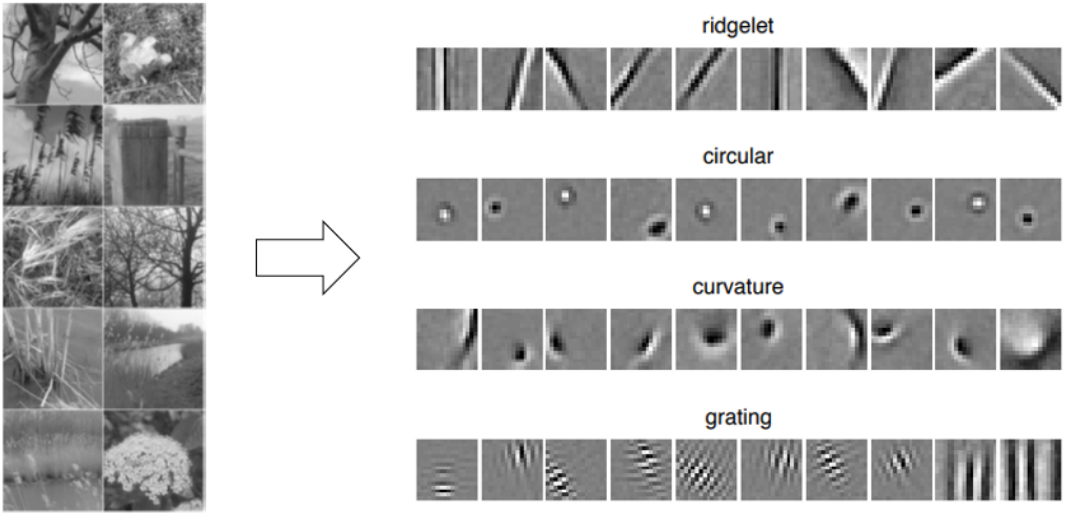


Figura 12: Ejemplos representantivos de cuatro tipos de características base aprendidas por un autoencoder disperso sobrecompleto 10x. (Olshausen 2013)

Denoising autoencoders (eliminadores de ruido)

- Otra manera de restringir las unidades ocultas de un autoencoder es añadiendo ruido a las entradas y entrenándolo a recuperar las entradas originales sin ruido.
- El ruido puede ser simplemente gausiano, o se puede «apagar» entradas como cuando se usa *dropout* (*masking noise*).

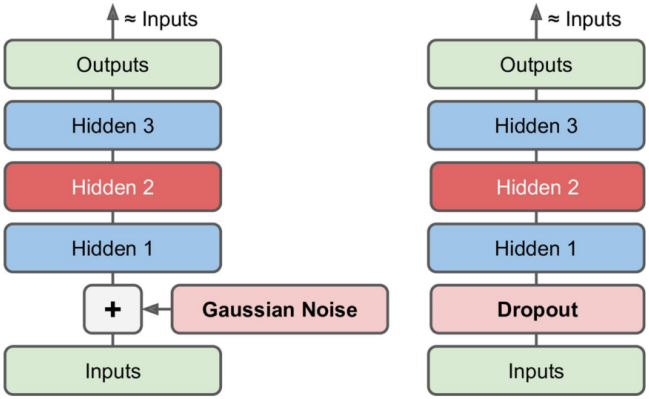


Figura 13: **Denoising autoencoders**, con ruido gausiano (izq.) o *dropout* (der.) (Géron 2017)

Denoising autoencoders (eliminadores de ruido)

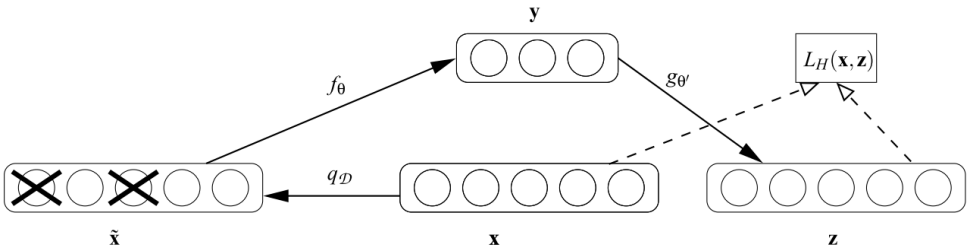


Figura 14: Arquitectura de un *denoising autoencoder*. Un ejemplo x es corrompido estocásticamente (mediante $q_\mathcal{D}$) produciéndose $\tilde{x}$. El autoencoder mapea $\tilde{x}$ a y (vía f_θ) e intenta reconstruir x (vía $g_{\theta'}$), produciendo la reconstrucción z. El error de reconstrucción se mide con la función de pérdida $L_H(x, z)$. (Vincent 2010)

Denoising autoencoders (eliminadores de ruido)

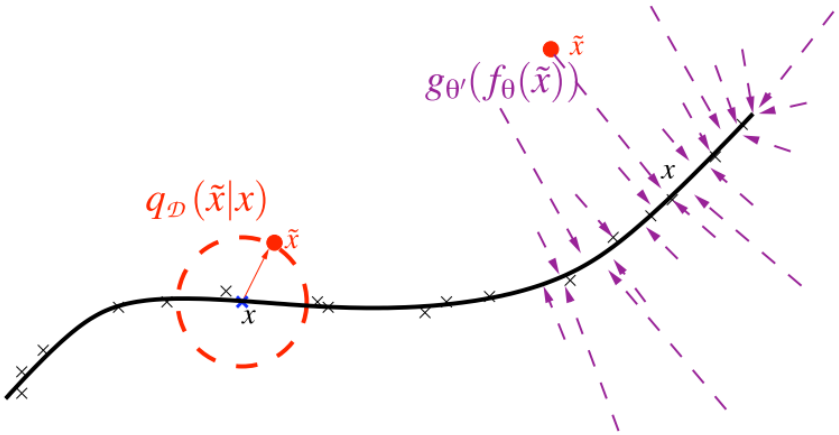


Figura 15: Aprendizaje de una variedad. Supongamos que los datos ($\times$) se concentran cerca a una variedad de bajas dimensiones. Los ejemplos $\tilde{x}$ ($\bullet$), corrompidos vía $q_{\mathcal{D}}(\tilde{x}|x)$, caerán generalmente lejos de la variedad. El modelo aprende $p(x|\tilde{x})$ para «proyectarlos de regreso» en la variedad vía $g_{\theta'}(f_\theta(\cdot))$. La representación intermedia $Y = f_\theta(x)$ puede ser interpretada como un sistema de coordenadas de los puntos x de la variedad. (Vincent 2010)

Autoencoders variacionales

- Los autoencoders variacionales tienen dos principales características que los diferencian de los anteriores:
  - Son modelos *probabilísticos* no sólo durante el entrenamiento sino también al realizar inferencia.
  - Son modelos *generativos*, es decir, pueden generar nuevas instancias parecidas a los datos de entrenamiento.
- Las capas de codificación no producen un código sino un valor medio μ y una desviación estándar σ para el código.
- El decodificador muestrea un valor de una distribución gausiana $\mathcal{N}(\mu, \sigma)$ y prosigue según lo usual.

Autoencoders variacionales

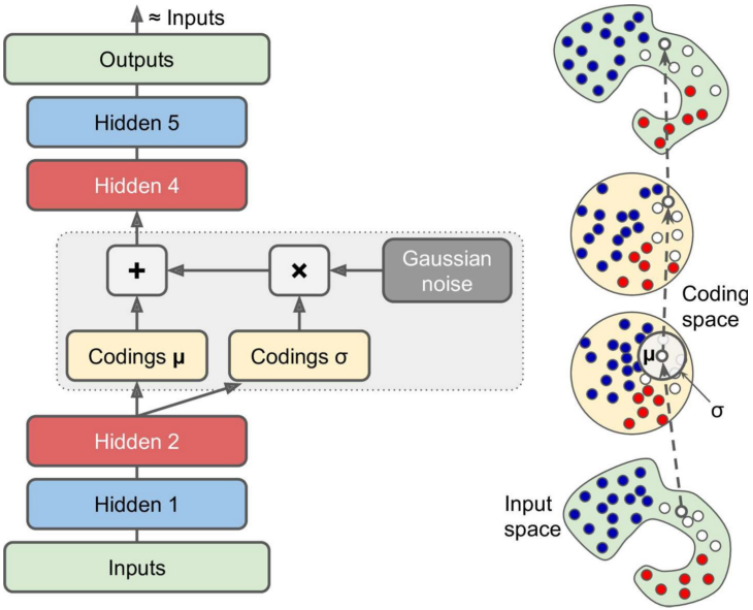
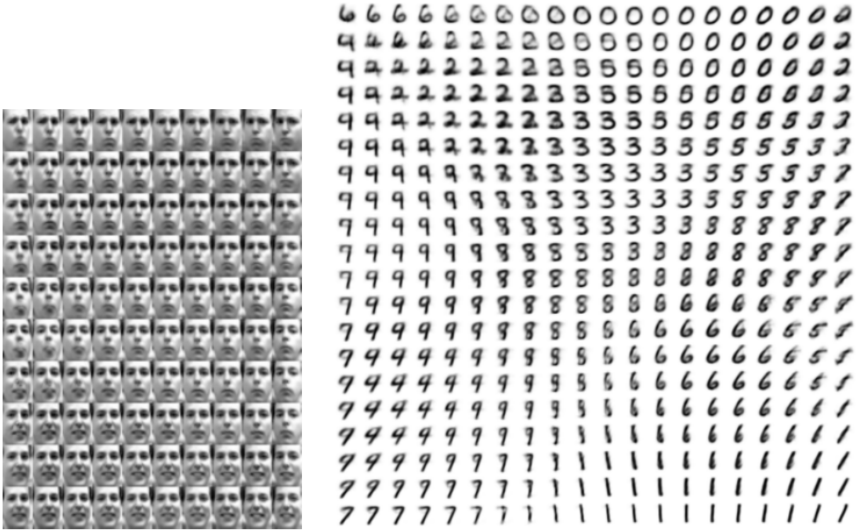


Figura 16: Arquitectura de un autoencoder variacional (izq.) y el paso de una instancia a través de él (der.) (Géron 2017)

Autoencoders variacionales



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figura 17: Visualizaciones del aprendizaje de variedades de dos dimensiones realizado por un autoencoder variacional. (Kingma 2013)

Bibliografía

- Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems O'Reilly Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. MIT Press. Retrieved from http://www.deeplearningbook.org/
- Hinton, G. E., & McClelland, J. L. (1988). Learning representations by recirculation. In Neural information processing systems (pp. 358–366).
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504–507.
- James, G., Witten, D., Hastie, T., & Tibishirani, R. (2013). An Introduction to Statistical Learning. Springer Texts in Statistics.
- Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv Preprint arXiv:1312.6114.
- Mueller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.
- Olshausen, B. A. (2013). Highly overcomplete sparse coding. In Human Vision and Electronic Imaging XVIII. Vol. 8651
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11(Dec), 3371–3408.