Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3
Aprendizaje Automático

Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3
Aprendizaje Automático

Semana 8 Autoencoders

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
  - Bagging / Pasting —> Random Forests
  - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
  - Generative Adversarial Networks (GAN)
  - Reinforcement Learning

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
  - Bagging / Pasting —> Random Forests
  - Boosting
- Support Vector Machines

Deep Learning

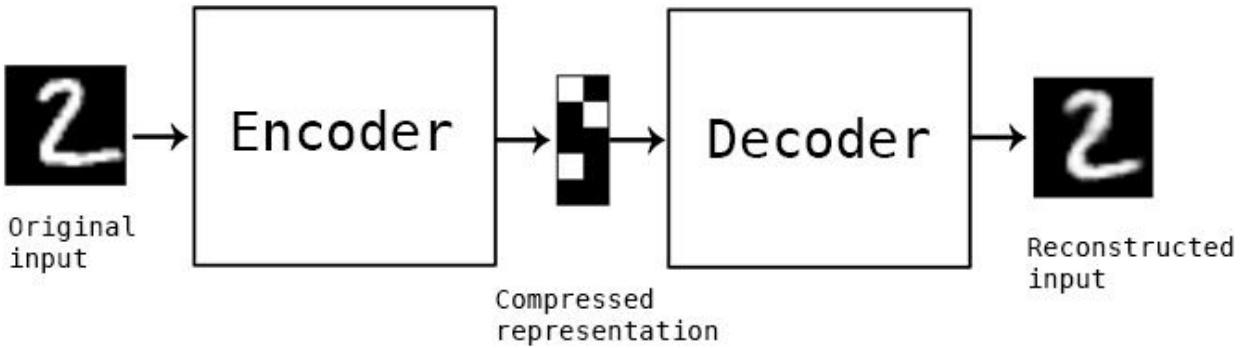
- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
  - Generative Adversarial Networks (GAN)
  - Reinforcement Learning

Autoencoders

- A lo largo del curso se mostraron varios algoritmos y técnicas que utilizaban diferentes formas de aprendizaje
- Pero, en las redes neuronales que vimos, siempre se hizo **aprendizaje supervisado.** Siempre se buscaba mapear un dato de entrada a una variable target
- Hoy no vamos a ver un nuevo tipo de red (o por lo menos en el sentido más estricto del término), vamos a ver una arquitectura junto a una forma de entrenamiento no supervisado (en realidad auto-supervisado). Los **autoencoders**

Autoencoders

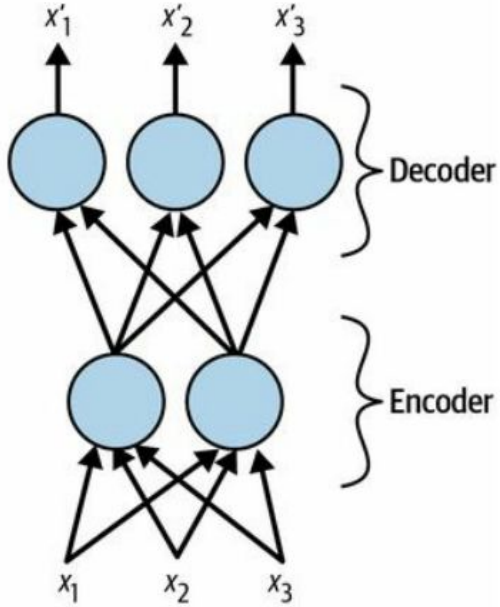
- Lo que busca un autoencoder es reproducir el dato de entrada
- El truco está en que aprende a hacerlo pasando por un espacio de dimensión menor, o sea, **codifica** los datos en un *espacio latente* (que se puede lograr forzando un *cuello de botella* en la arquitectura) para luego **decodificarlos** en el espacio original buscando reconstruir el dato de entrada



Fuente

Autoencoders

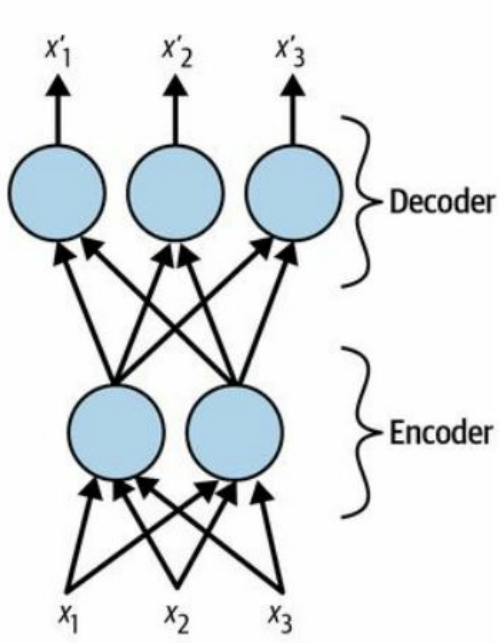
- En su forma más sencilla es una red totalmente conectada con una capa oculta:
  - La cantidad de neuronas de la entrada tiene que ser **igual** a la de salida
  - La cantidad de neuronas de la capa oculta tiene que ser **menor** que las de la entrada/salida
- En el entrenamiento se pone como target a aprender al mismo dato que se le presentó a la red
- De esta forma, cuando el autoencoder busca reproducir la entrada lo forzamos a *comprimir la información* aprendiendo sus características más relevantes



Imágen del Géron

Autoencoders

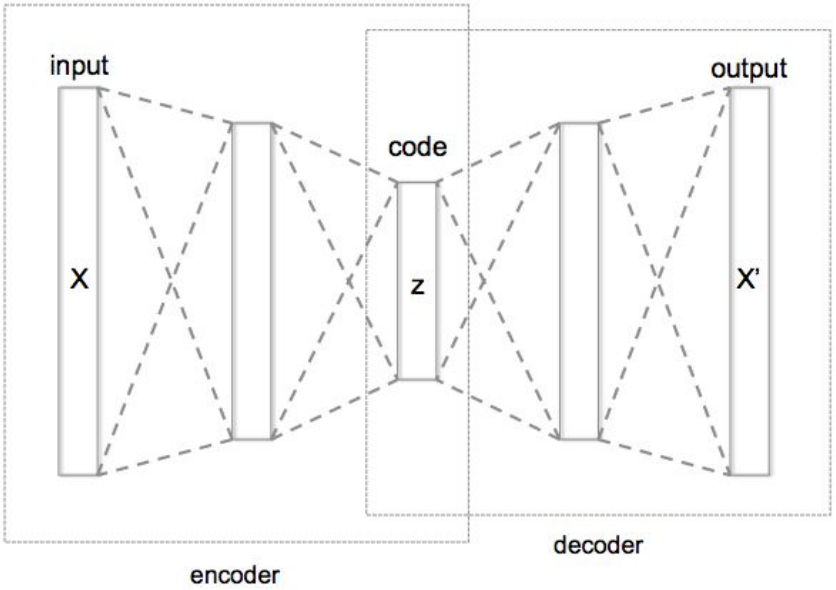
- Se van a distinguir dos partes de la red, la parte que efectivamente *comprime* la información llevándola al espacio latente y la parte que toma una representación en este espacio y busca regenerar el dato
- La primera parte es el **encoder** y la segunda el **decoder**



Imágen del Géron

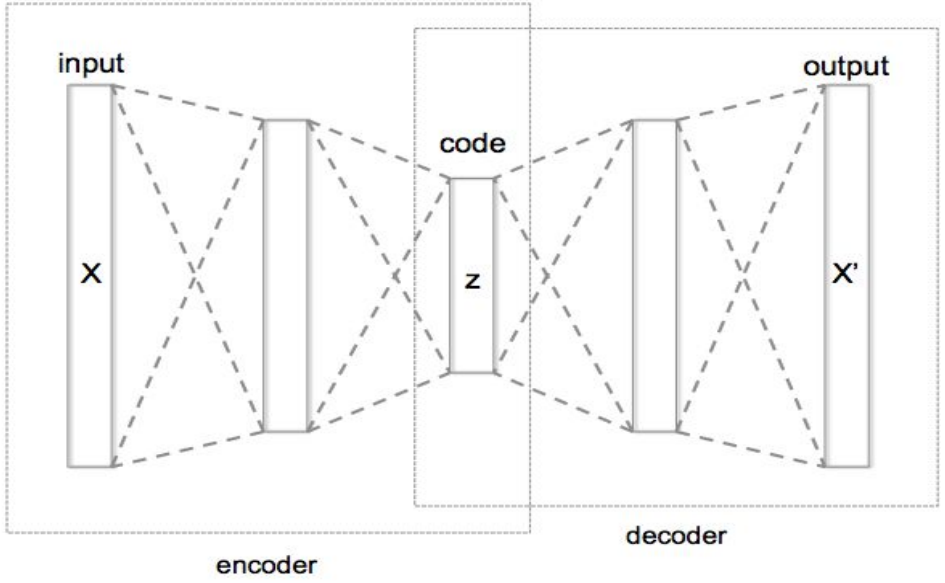
Autoencoders

- Por supuesto, esto se puede generalizar haciendo arquitecturas más profundas.
- Esto se suele llamar *Stacked Autoencoders* o *Deep Autoencoders*
- La idea es que añadir más capas permite aprender codificaciones más complejas
- Ojo, hay que ser cuidadoso y no hacer al AE demasiado poderoso, no queremos que simplemente “mapee una entrada a una salida” queremos forzarlo a que **codifique** y **decodifique**

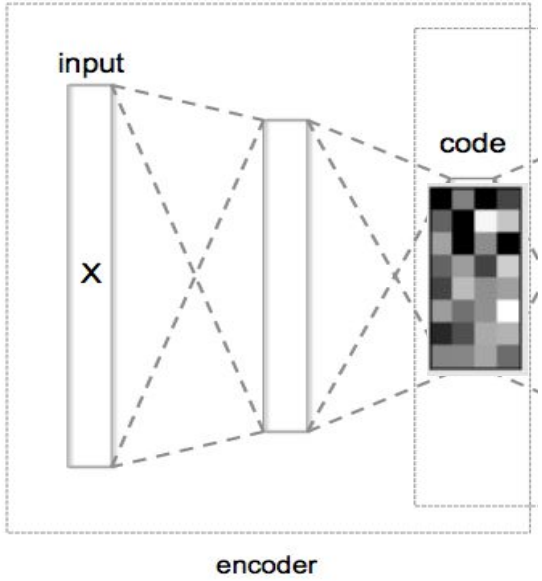


Imágen de Wikipedia

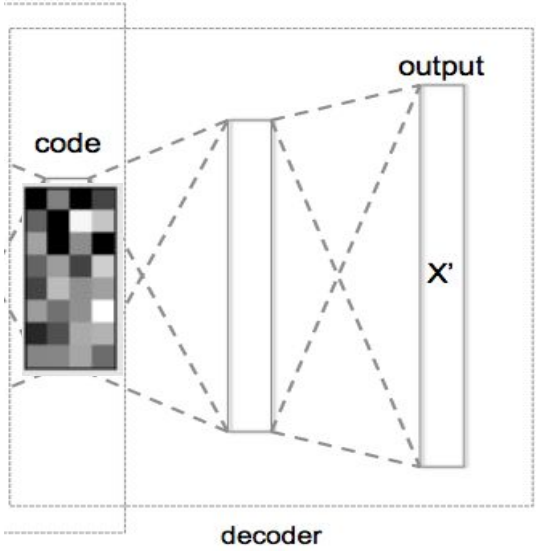
Autoencoders



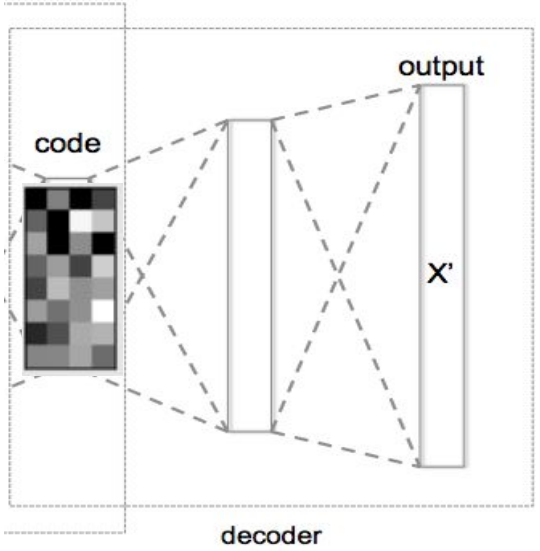
Autoencoders



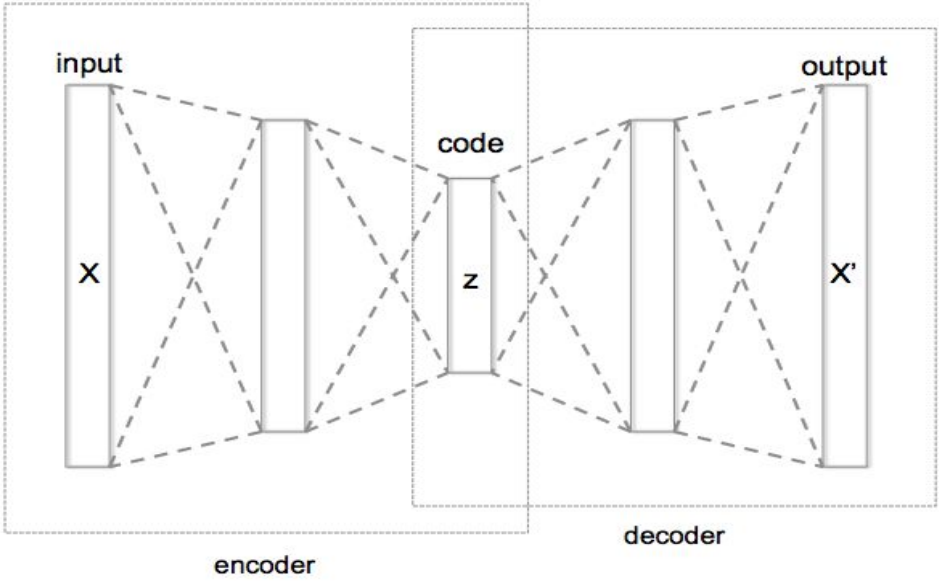
Autoencoders



Autoencoders

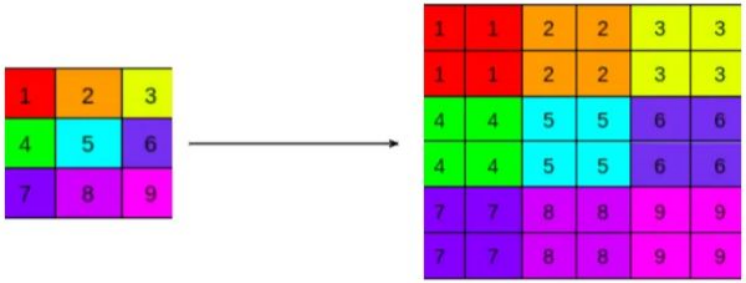


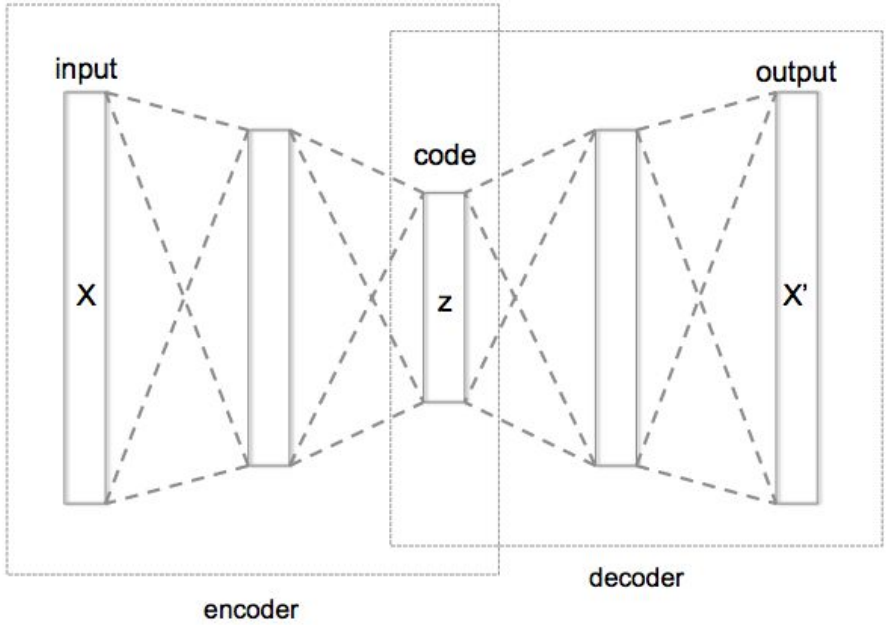
Autoencoders



Autoencoders

- También se pueden usar capas convolucionales, en este caso estaríamos hablando de un *Convolutional Autoencoder*
- La sutileza que aparece al incorporar capas convolucionales es que necesitamos regresar al tamaño original usando capas de *up-sampling*





Fuente

Autoencoders

- ¿Aplicaciones?, varias

1. Reducción de la dimensionalidad
2. Detección de características
3. Pre-entrenamiento no supervisado
4. Reducción de ruido
5. Detección de anomalias
6. Generación de datos
7. …

Fuente

Autoencoders

- Vamos a ver algunas aplicaciones y variantes en el notebook, lo importante es quedarse con la idea que una vez entrenado tenemos…
  a. Una red que aprendió características de los datos de forma tal que puede **codificarlos** (o comprimirlos, si quieren) en un espacio de menor dimensión
  b. Una red que toma un dato en este espacio latente y lo **decodifica** para llegar a un elemento en el espacio original que se parece mucho al dato de entrada

Notebook

Vamos al notebook!

`Notebook_Semana_8_Autoencoders.ipynb`