

# Auto-encoders y variational auto-encoders

---

Auto-encoders y variational auto-encoders

---

## Introducción

# Introducción

## Aprendizaje no supervisado

Hasta ahora hemos estado hablando siempre de **aprendizaje profundo supervisado**, pero también podemos resolver problemas **no supervisados**.

### Objetivo:

- Extraer patrones directamente de los datos “sin etiquetar”, solo tenemos  $x$  y no  $y$ .

### Tareas comunes:

- Modelos generativos: Entender la distribución de  $x$  y generar nuevas muestras.
- Autoencoders: “Comprimir”  $x$  proyectándolo en un espacio de menor dimensión.

# Introducción

## Aprendizaje no supervisado



*The brain has about  $10^{14}$  synapses and we only live for about  $10^9$  seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of **unsupervised learning** since the perceptual input (including proprioception) is the only place we can get  $10^5$  dimensions of constraint per second.*

Geoffrey Hinton, 2014

# Introducción

## Aprendizaje no supervisado



*We need tremendous amount of information to build machines that have common sense and generalize.*

Yann LeCun, 2016

### ■ "Pure" Reinforcement Learning (cherry)

- ▶ The machine predicts a scalar reward given once in a while.
- ▶ **A few bits for some samples**



### ■ Supervised Learning (icing)

- ▶ The machine predicts a category or a few numbers for each input
- ▶ Predicting human-supplied data
- ▶ **10→10,000 bits per sample**

### ■ Unsupervised/Predictive Learning (cake)

- ▶ The machine predicts any part of its input for any observed part.
- ▶ Predicts future frames in videos
- ▶ **Millions of bits per sample**

■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

# Introducción

## Modelos generativos

Un **modelo generativo** es un modelo probabilístico  $p$  que puede ser utilizado como un “simulador de datos”.

Su propósito es generar datos sintéticos pero realistas de alta dimensión

$$\mathbf{x} \sim p_{\theta}(\mathbf{x}),$$

que se asemejen lo más posible a la distribución desconocida de datos  $p(\mathbf{x})$ .

# Introducción

## Modelos generativos



Ley de Moore de los modelos generativos de imágenes

# Introducción

## Modelos generativos

Algunas aplicaciones:



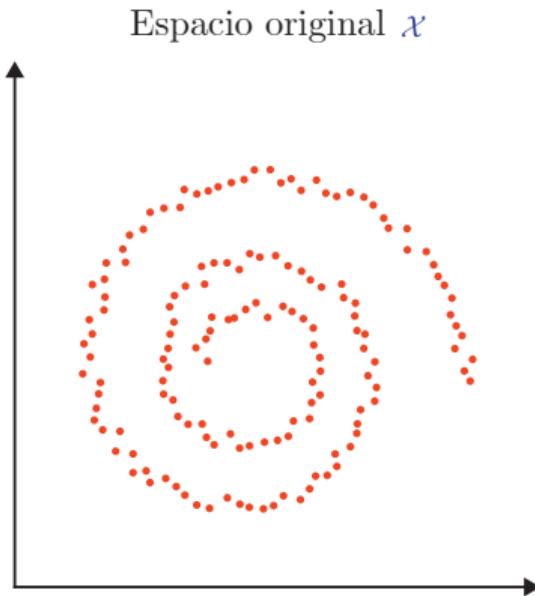
Los modelos generativos tienen un rol muy importante en muchos problemas actuales

Auto-encoders y variational auto-encoders

---

## Auto-encoders

# Auto-encoders



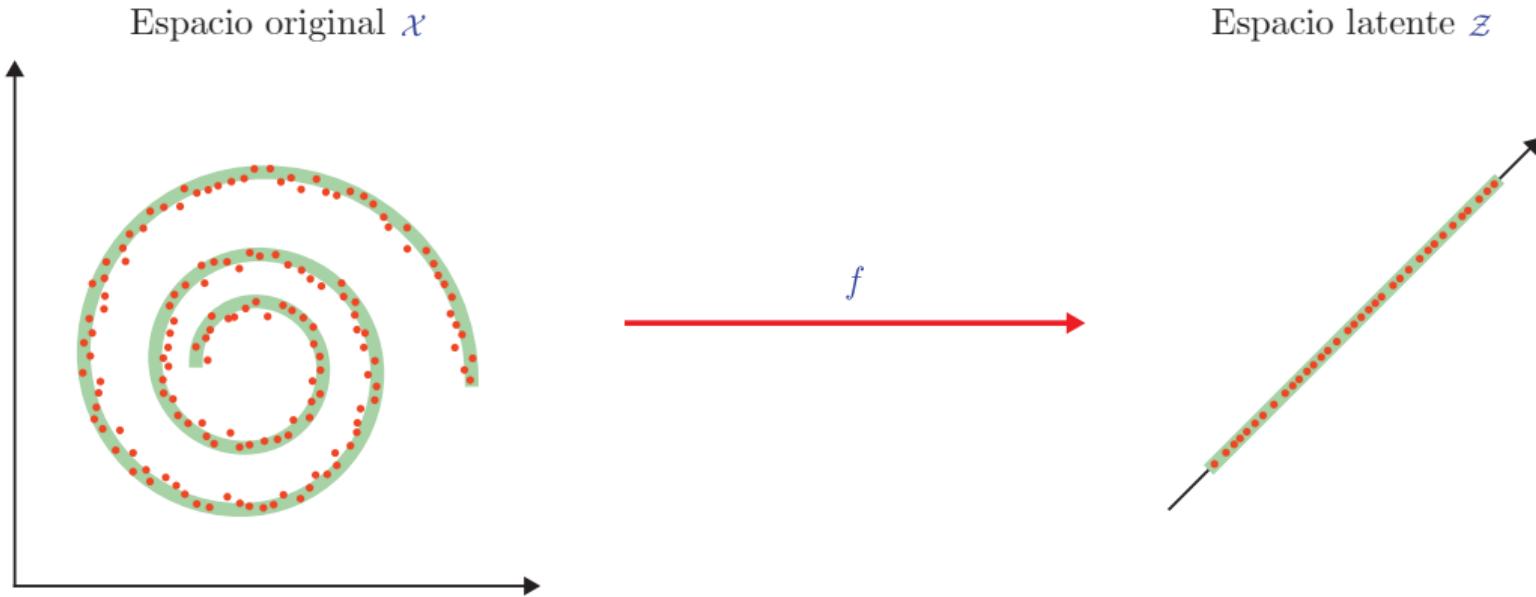
# Auto-encoders



# Auto-encoders



# Auto-encoders



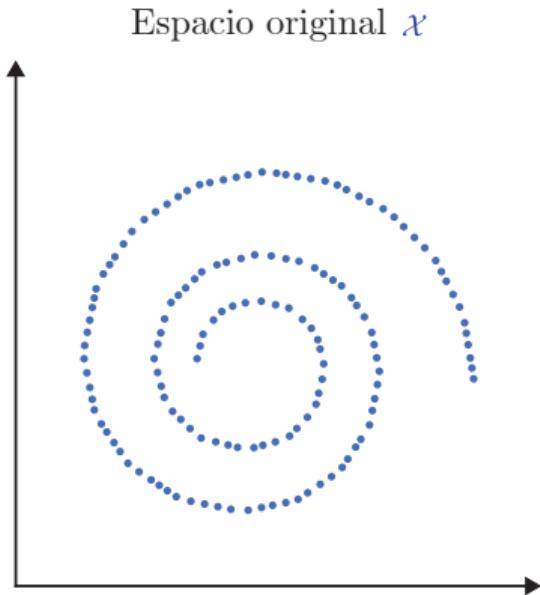
# Auto-encoders



# Auto-encoders



# Auto-encoders

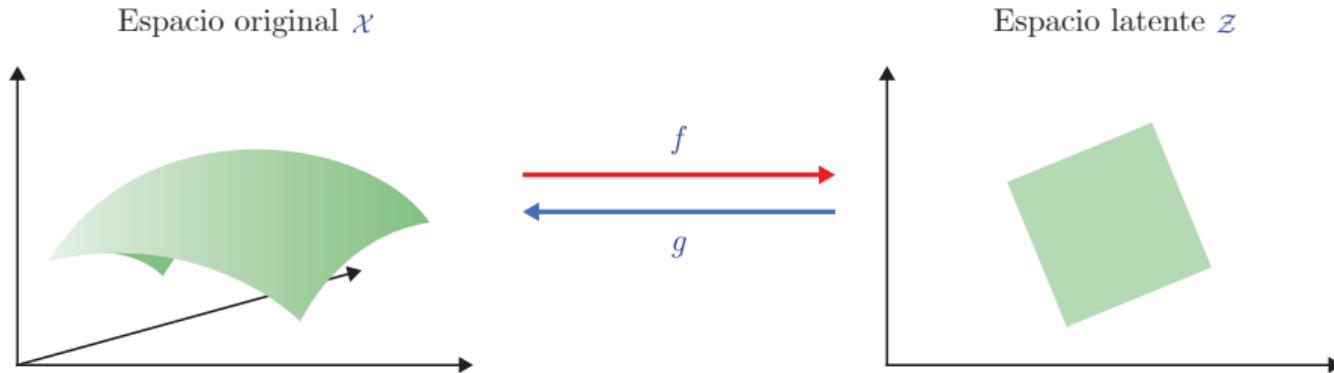


# Auto-encoders

Un **auto-encoder** es una función compuesta a partir de:

- Un **encoder**  $f$  que proyecta del espacio original  $\mathcal{X}$  al espacio latente  $\mathcal{Z}$ .
- Un **decoder**  $g$  que proyecta de vuelta al espacio original.

El objetivo es que  $g \circ f$ , es decir, que la composición de funciones se aproxime lo máximo posible a los datos originales o función identidad.



## Auto-encoders

Siendo  $p(\mathbf{x})$  la distribución de los datos en  $\mathcal{X}$ , un buen auto-encoder puede caracterizarse con la *reconstruction loss*:

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [||\mathbf{x} - g \circ f(\mathbf{x})||^2] \approx 0.$$

Esta función de pérdida mide como de bien el auto-encoder puede reconstruir los datos originales.

Dadas dos funciones de proyección con parámetros  $f(\cdot; \theta_f)$  and  $g(\cdot; \theta_g)$ , el entrenamiento consiste aprender los parámetros que minimicen dicha loss:

$$\theta_f, \theta_g = \arg \min_{\theta_f, \theta_g} \frac{1}{N} \sum_{i=1}^N ||\mathbf{x}_i - g(f(\mathbf{x}_i, \theta_f), \theta_g)||^2.$$

# Auto-encoders

## Ejemplo

Imaginemos, por ejemplo, un auto-encoder lineal con

$$f : \mathbf{z} = \mathbf{U}^T \mathbf{x}$$

$$g : \hat{\mathbf{x}} = \mathbf{U}\mathbf{z},$$

con  $\mathbf{U} \in \mathbb{R}^{p \times d}$ , el *reconstruction loss* se reduce a

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[ \|\mathbf{x} - \mathbf{U}\mathbf{U}^T \mathbf{x}\|^2 \right].$$

Auto-encoders y variational auto-encoders

---

## Deep Auto-encoders

# Deep Auto-encoders

## Mayor profundidad

Para obtener mejores resultados, en vez de proyecciones lineales se suelen utilizar redes neuronales profundas en  $f$  y  $g$ .

Algunos ejemplos:

- Combinando un MLP encoder  $f : \mathbb{R}^p \rightarrow \mathbb{R}^d$  con un MLP decoder  $g : \mathbb{R}^d \rightarrow \mathbb{R}^p$ .
- Combinando un convolutional network encoder  $f : \mathbb{R}^{w \times h \times c} \rightarrow \mathbb{R}^d$  con un decoder decoder  $g : \mathbb{R}^d \rightarrow \mathbb{R}^{w \times h \times c}$  compuesto de capas convolucionales reciprocas.



# Deep Auto-encoders

Ejemplo MNIST

Datos originales  $\mathbf{x}$  con  $d = 784$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 3 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de CNN con  $d = 2$ .

7 2 1 0 9 1 9 9 8 9 0 6  
9 0 1 5 9 7 5 9 9 6 6 5  
9 0 7 9 0 1 5 1 5 9 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de PCA con  $d = 2$ .

9 3 1 0 9 1 9 9 8 9 0 8  
9 0 1 8 9 9 8 9 9 8 9 8  
9 0 9 9 0 1 8 1 8 9 9 8

# Deep Auto-encoders

Ejemplo MNIST

Datos originales  $\mathbf{x}$  con  $d = 784$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 3 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de CNN con  $d = 4$ .

7 2 1 0 4 1 4 9 9 9 0 6  
9 0 1 5 9 7 3 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 0 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de PCA con  $d = 4$ .

9 2 1 0 9 1 9 9 0 9 0 0  
9 0 1 3 9 9 3 9 9 0 6 5  
9 0 9 9 0 1 3 1 3 0 9 0

# Deep Auto-encoders

Ejemplo MNIST

Datos originales  $\mathbf{x}$  con  $d = 784$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 3 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de CNN con  $d = 8$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 3 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de PCA con  $d = 8$ .

7 3 1 0 4 1 9 9 0 7 0 0  
9 0 1 0 9 7 3 4 9 6 0 5  
4 0 7 4 0 1 3 1 3 4 7 0

# Deep Auto-encoders

Ejemplo MNIST

Datos originales  $\mathbf{x}$  con  $d = 784$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de CNN con  $d = 16$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de PCA con  $d = 16$ .

7 2 1 0 9 1 4 9 6 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

# Deep Auto-encoders

Ejemplo MNIST

Datos originales  $\mathbf{x}$  con  $d = 784$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de CNN con  $d = 32$ .

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Resultado de auto-encoder  $g \circ f$  creado a partir de PCA con  $d = 32$ .

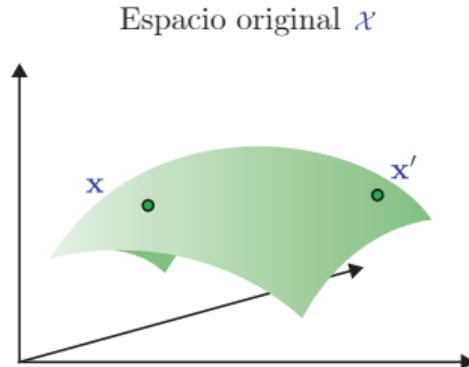
7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

# Deep Auto-encoders

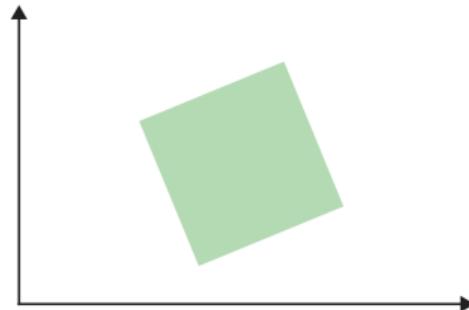
## Interpolación

### Interpolación de puntos

Para comprender la representación latente aprendida, podemos elegir dos muestras  $x$  y  $x'$  al azar e interpolar muestras a lo largo de la línea en el espacio latente.



Espacio latente  $\mathcal{Z}$



# Deep Auto-encoders

## Interpolación

### Interpolación de puntos

Para comprender la representación latente aprendida, podemos elegir dos muestras  $\mathbf{x}$  y  $\mathbf{x}'$  al azar e interpolar muestras a lo largo de la línea en el espacio latente.



# Deep Auto-encoders

## Interpolación

### Interpolación de puntos

Para comprender la representación latente aprendida, podemos elegir dos muestras  $\mathbf{x}$  y  $\mathbf{x}'$  al azar e interpolar muestras a lo largo de la línea en el espacio latente.



# Deep Auto-encoders

## Interpolación

### Interpolación de puntos

Para comprender la representación latente aprendida, podemos elegir dos muestras  $\mathbf{x}$  y  $\mathbf{x}'$  al azar e interpolar muestras a lo largo de la línea en el espacio latente.



# Deep Auto-encoders

## Interpolación

Interpolación de PCA ( $d = 32$ ).



# Deep Auto-encoders

## Interpolación

Interpolación de Auto-encoder ( $d = 32$ ).

5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6

8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9

6 6 6 6 6 0 0 0 0 0 0 0 0 0 0 0

6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4

3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7

2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

# Deep Auto-encoders

## Generación

### Generar nuevos datos

Además de generar valores intermedios entre dos datos existentes, también podemos utilizar el decoder para obtener la representación en  $\mathcal{X}$  de valores desconocidos en  $\mathcal{Z}$ .

Se introduce un modelo de densidad  $q$  en  $\mathcal{Z}$  y se utiliza  $g$  para mapear en  $\mathcal{X}$ .



# Deep Auto-encoders

## Generación

### Generar nuevos datos

Además de generar valores intermedios entre dos datos existentes, también podemos utilizar el decoder para obtener la representación en  $\mathcal{X}$  de valores desconocidos en  $\mathcal{Z}$ .

Se introduce un modelo de densidad  $q$  en  $\mathcal{Z}$  y se utiliza  $g$  para mapear en  $\mathcal{X}$ .



# Deep Auto-encoders

## Generación

### Generar nuevos datos

Además de generar valores intermedios entre dos datos existentes, también podemos utilizar el decoder para obtener la representación en  $\mathcal{X}$  de valores desconocidos en  $\mathcal{Z}$ .

Se introduce un modelo de densidad  $q$  en  $\mathcal{Z}$  y se utiliza  $g$  para mapear en  $\mathcal{X}$ .



# Deep Auto-encoders

## Generación

### Generar nuevos datos

Además de generar valores intermedios entre dos datos existentes, también podemos utilizar el decoder para obtener la representación en  $\mathcal{X}$  de valores desconocidos en  $\mathcal{Z}$ .

Se introduce un modelo de densidad  $q$  en  $\mathcal{Z}$  y se utiliza  $g$  para mapear en  $\mathcal{X}$ .



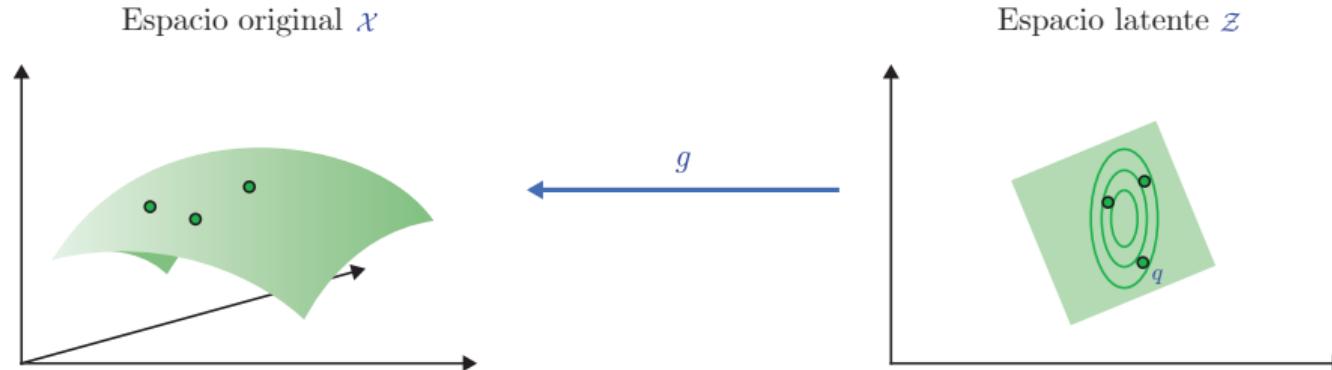
# Deep Auto-encoders

## Generación

### Generar nuevos datos

Además de generar valores intermedios entre dos datos existentes, también podemos utilizar el decoder para obtener la representación en  $\mathcal{X}$  de valores desconocidos en  $\mathcal{Z}$ .

Se introduce un modelo de densidad  $q$  en  $\mathcal{Z}$  y se utiliza  $g$  para mapear en  $\mathcal{X}$ .



# Deep Auto-encoders

## Generación

Por ejemplo, si utilizamos una distribución gaussiana  $q(\mathbf{z}) = \mathcal{N}(\hat{\mu}, \hat{\Sigma})$ , donde  $\hat{\mu}$  y  $\hat{\Sigma}$  se estiman a partir de los datos de entrenamiento:

Generación de datos con auto-encoder ( $d = 16$ ).

Three rows of handwritten digits generated by a 16-dimensional auto-encoder. The digits are somewhat blurry and lack fine detail.

Row 1: 8 8 8 3 2 7 3 4 3 6 5 4  
Row 2: 0 9 3 4 8 5 8 5 3 1 6  
Row 3: 3 5 8 4 9 2 3 4 8 3 3 3

Generación de datos con auto-encoder ( $d = 32$ ).

Three rows of handwritten digits generated by a 32-dimensional auto-encoder. The digits are clearer than those from the 16-dimensional model, though they still lack some detail.

Row 1: 2 7 3 5 8 4 7 8 4 5 3 8 3  
Row 2: 4 5 8 2 0 4 9 3 2 8 8 7  
Row 3: 1 4 5 3 6 2 4 7 5 3 6 3 8

Los resultados son malos porque la distribución seleccionada es **simple e inadecuada**.

Auto-encoders y variational auto-encoders

---

## Denoising Auto-encoders

# Denoising Auto-encoders

## Eliminar ruido

Además de la ya mencionada reducción de dimensión, los auto-encoders también son capaces de **restaurar datos dañados o con ruido**.

En este caso particular podemos ignorar la estructura encoder/decoder, lo que nos dejaría con:

$$h : \mathcal{X} \rightarrow \mathcal{X}$$

Esta expresión hace referencia a un **denoising auto-encoder**.

El objetivo es optimizar  $h$  de tal forma que, cualquier perturbación  $\tilde{\mathbf{x}}$  de la señal  $\mathbf{x}$  sea restaurada a  $\mathbf{x}$  de nuevo, por tanto

$$h(\tilde{\mathbf{x}}) \approx \mathbf{x}.$$

# Denoising Auto-encoders

## Ejemplo 2D

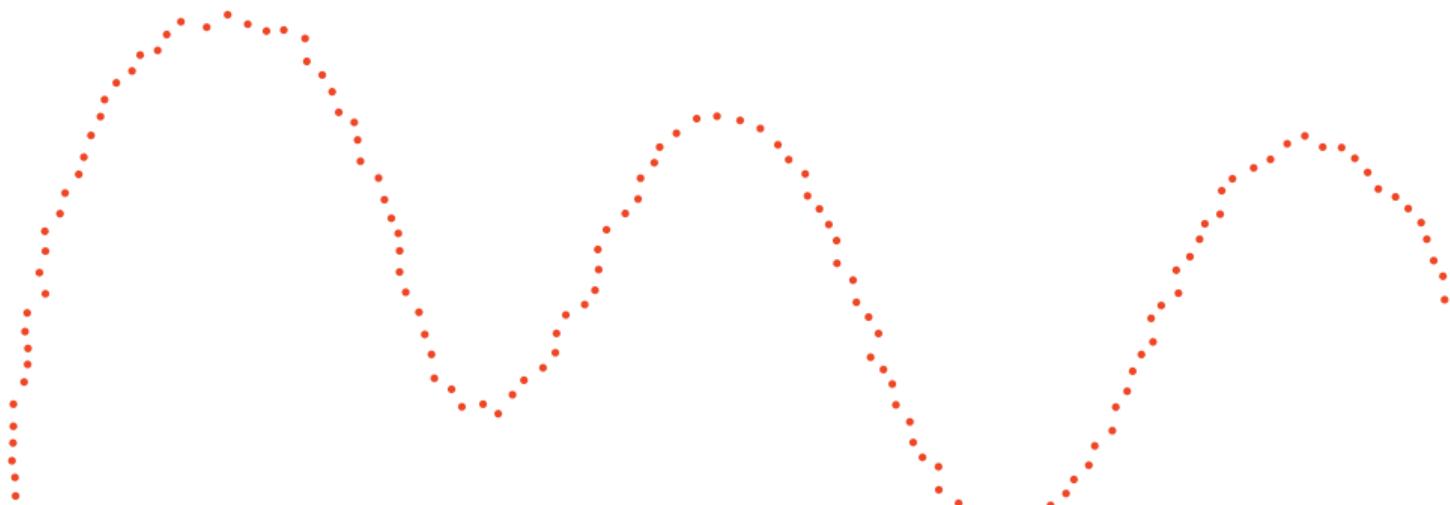
Podemos ilustrar este comportamiento con datos en 2D, añadiendo ruido Gaussiano y utilizando la MSE:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^N \|x_n - h(x_n + \epsilon_n; \theta)\|^2$$

donde  $x_n$  son los datos originales y  $\epsilon_n$  el ruido Gaussiano.

# Denoising Auto-encoders

## Ejemplo 2D



Conjunto de datos original  $\mathbf{x}$ .

# Denoising Auto-encoders

Ejemplo 2D



Datos con ruido adicional  $\tilde{x} = x + \epsilon$ .

# Denoising Auto-encoders

## Ejemplo 2D



Distribución aprendida por el denoising auto-encoder.

# Denoising Auto-encoders

## Ejemplo 2D



# Denoising Auto-encoders

## Ejemplo 2D



Resultado  $h(\tilde{x})$  del decoder.

# Denoising Auto-encoders

## Ejemplo MNIST

Podemos hacer lo mismo con datos más complejos, por ejemplo el conjunto MNIST.

Cuando trabajamos con imágenes, los tipos de *ruido* que podemos aplicar son más variados.

Original



Eliminar pixels



Desenfoque



Máscara de bloqueo



# Denoising Auto-encoders

Ejemplo MNIST

Datos originales.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos con el 50% de los pixels eliminados.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos reconstruidos.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

# Denoising Auto-encoders

Ejemplo MNIST

Datos originales.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 4 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos con el 90% de los pixels eliminados.

7 3 1 0 4 1 4 9 4 9 0 6  
9 0 1 5 9 7 8 4 9 6 4 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos reconstruidos.

7 3 1 0 4 1 4 9 4 9 0 6  
9 0 1 5 9 7 8 4 9 6 4 5  
4 0 7 4 0 1 3 1 3 4 7 2

# Denoising Auto-encoders

Ejemplo MNIST

Datos originales.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos con desenfoque ( $\sigma = 4$ ).

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos reconstruidos.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

# Denoising Auto-encoders

Ejemplo MNIST

Datos originales.

7 2 1 0 4 1 4 9 5 9 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 0 1 3 1 3 4 7 2

Datos con máscara de bloqueo de tamaño  $16 \times 16$  pixels.

1 0 0 0 0 0 0 0 0 0 0 0  
0 1 0 0 0 0 0 0 0 0 0 0  
0 0 1 0 0 0 0 0 0 0 0 0  
0 0 0 1 0 0 0 0 0 0 0 0

Datos reconstruidos.

7 2 1 0 4 1 4 9 5 7 0 6  
9 0 1 5 9 7 8 4 9 6 6 5  
4 0 7 4 3 1 3 1 3 4 7 2

# Denoising Auto-encoders

## Ejemplo MNIST

### Multiples soluciones

La distribución previa  $p(\mathbf{x}|\tilde{\mathbf{x}})$  (probabilidad de obtener  $\mathbf{x}$  dado  $\tilde{\mathbf{x}}$ ), **puede ser multimodal**, es decir, puede tener **múltiples soluciones posibles**.

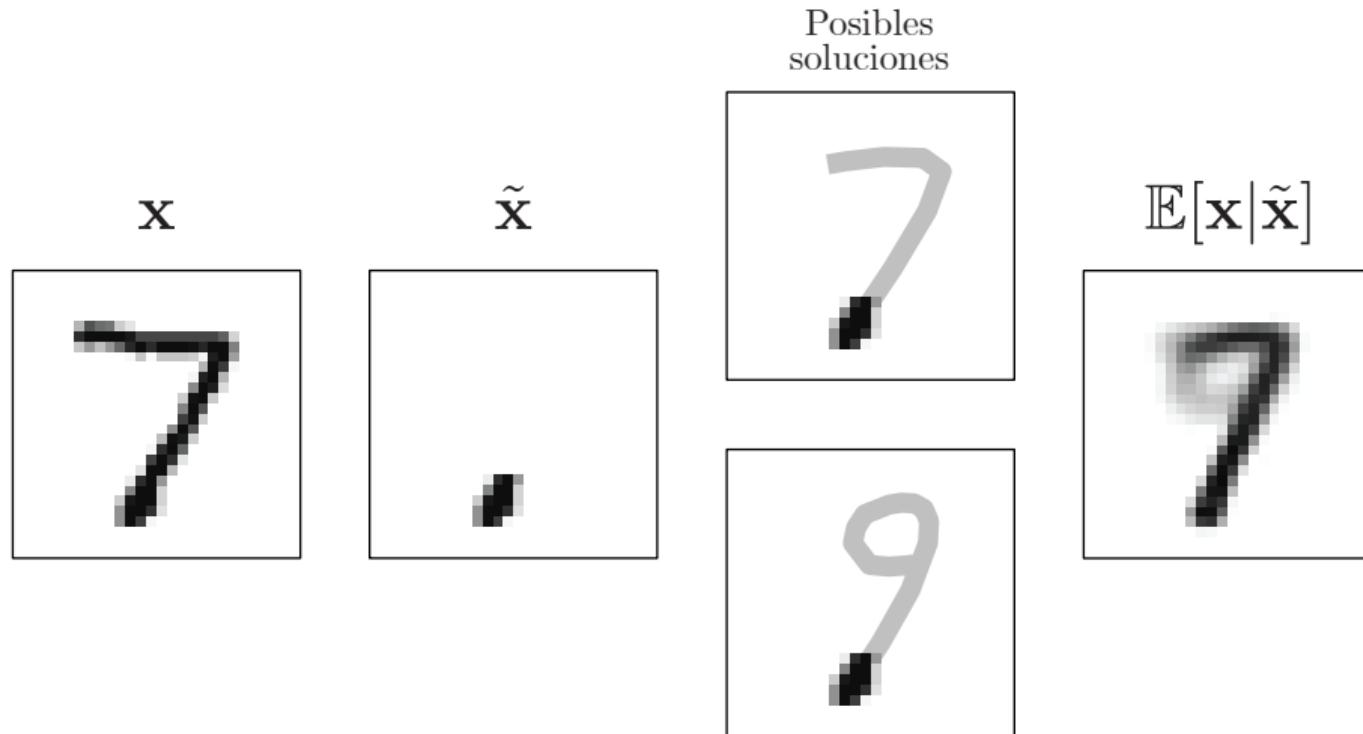
Si entrenamos un autoencoder utilizando MSE entonces la mejor reconstrucción que podemos esperar es el “promedio” de las posibles soluciones

$$h(\tilde{\mathbf{x}}) = \mathbb{E}[\mathbf{x}|\tilde{\mathbf{x}}],$$

lo que es improbable que coincida con  $p(\mathbf{x}|\tilde{\mathbf{x}})$ .

# Denoising Auto-encoders

Ejemplo MNIST



Auto-encoders y variational auto-encoders

---

## Inferencia variacional

# Inferencia variacional

## Motivación

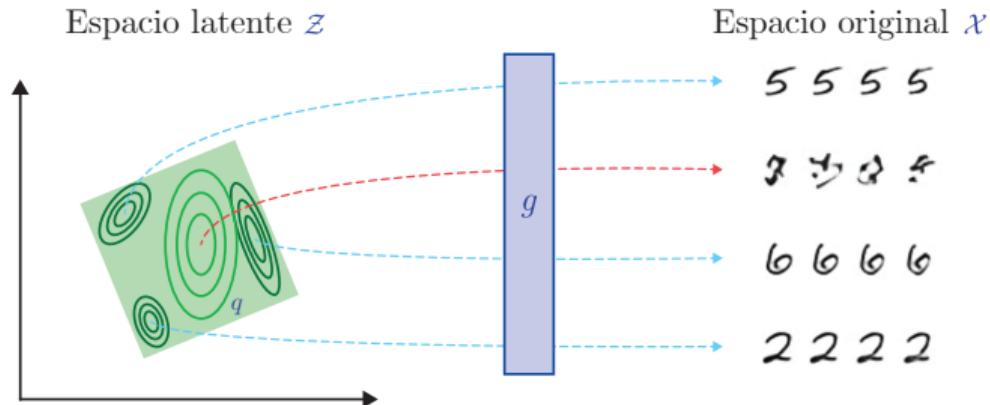
Como veíamos anteriormente (página 38), si intentamos **generar nuevos datos** en  $\mathcal{X}$  a partir de unos puntos en  $\mathcal{Z}$  asumiendo que siguen una distribución  $q$ , los resultados no son prometedores.



# Inferencia variacional

## Motivación

Este problema proviene de la distribución  $q$  seleccionada. Esta es **muy simple e inadecuada**, los datos en  $\mathcal{Z}$  se comportan de manera diferente.



El espacio latente tiene discontinuidades y al tomar una muestra en estas zonas intermedias el decodificador produce salidas poco realistas.

# Inferencia variacional

## Motivación

### Imponer distribución

No conocemos la distribución en  $\mathcal{Z}$  para poder generar ejemplos con sentido, pero podemos **forzar al auto-encoder para que aprenda una específica** durante el entrenamiento.

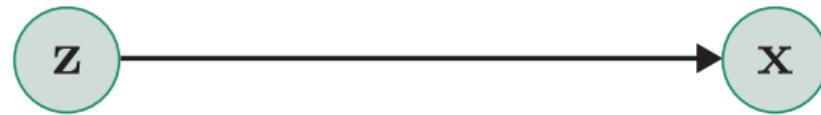
En vez de entrenar el auto-encoder y adivinar la distribución de  $\mathcal{Z}$  vamos a:

- 1 Imponer una distribución para  $\mathcal{Z}$ .
- 2 Entrenar el decoder  $g$  de tal forma que  $g(\mathcal{Z})$  coincida con los datos de entrenamiento.

# Inferencia variacional

## Modelo de variables latentes

Consideremos un modelo que relaciona un conjunto de variables observables  $\mathbf{x} \in \mathcal{X}$  con un conjunto de variables latentes  $\mathbf{z} \in \mathcal{Z}$ .



Como en cada modelo, tenemos que definir  $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$  donde:

- $p(\mathbf{x}|\mathbf{z})$ : Distribución de probabilidad de  $x$  dado un valor de  $z$ .
- $p(\mathbf{z})$ : Distribución estimada de probabilidad de  $z$ .

# Inferencia variacional

## Proceso generativo estocástico

Si asumimos que las variables latentes  $z$  influyen a la hora de generar las  $x$  esto implica que **podemos generar datos en  $\mathcal{X}$  siguiendo un proceso aleatorio controlado por las variables latentes  $z$ .**



# Inferencia variacional

## Proceso generativo estocástico

Si asumimos que las variables latentes  $z$  influyen a la hora de generar las  $x$  esto implica que **podemos generar datos en  $\mathcal{X}$  siguiendo un proceso aleatorio controlado por las variables latentes  $z$ .**



# Inferencia variacional

Proceso generativo estocástico

Si asumimos que las variables latentes  $z$  influyen a la hora de generar las  $x$  esto implica que **podemos generar datos en  $\mathcal{X}$  siguiendo un proceso aleatorio controlado por las variables latentes  $z$ .**



# Inferencia variacional

Proceso generativo estocástico

Si asumimos que las variables latentes  $z$  influyen a la hora de generar las  $x$  esto implica que **podemos generar datos en  $\mathcal{X}$  siguiendo un proceso aleatorio controlado por las variables latentes  $z$ .**



# Inferencia variacional

## Proceso generativo estocástico

Si asumimos que las variables latentes  $\mathbf{z}$  influyen a la hora de generar las  $\mathbf{x}$  esto implica que **podemos generar datos en  $\mathcal{X}$  siguiendo un proceso aleatorio controlado por las variables latentes  $\mathbf{z}$ .**

En realidad no conocemos  $\mathbf{z}$  pero si  $\mathbf{x}$ , por lo que tratamos de resolver (probabilidad a posteriori):

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}.$$

Obtener  $p(\mathbf{x})$  es un **problema intratable** especialmente en modelos complejos dado que requiere una integrar sobre todas las posibles combinaciones de variables latentes

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}.$$

# Inferencia variacional

## Inferencia variacional

Como no podemos resolver  $p(\mathbf{z}|\mathbf{x})$  vamos a intentar aproximarla mediante un problema de optimización.

- Consideraremos una familia de distribuciones  $q_v(\mathbf{z}|\mathbf{x})$  que aproximan la probabilidad a posteriori  $p(\mathbf{z}|\mathbf{x})$  donde  $v$  es el índice de la familia de distribuciones.
- Los parámetros  $v$  se aprenden para minimizar la *Kullback Leibler (KL) divergence* entre la aproximación  $q_v(\mathbf{z}|\mathbf{x})$  y la probabilidad a posteriori  $p(\mathbf{z}|\mathbf{x})$ .
- Queremos resolver  $\arg \min_v \text{KL}(q_v(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x}))$ .

En otras palabras, queremos minimizar la diferencia entre la distribución variacional  $q_v(\mathbf{z}|\mathbf{x})$  y la verdadera distribución posterior  $p(\mathbf{z}|\mathbf{x})$ .

# Inferencia variacional

Gráficamente:



# Inferencia variacional

Si desarollamos la KL:

$$\arg \min_{\nu} \text{KL}(q_{\nu}(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x}))$$

$$= \arg \min_{\nu} \mathbb{E}_{q_{\nu}(\mathbf{z}|\mathbf{x})} \left[ \log \frac{q_{\nu}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right]$$

$$= \arg \min_{\nu} \mathbb{E}_{q_{\nu}(\mathbf{z}|\mathbf{x})} [\log q_{\nu}(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}, \mathbf{z})] + \log p(\mathbf{x}).$$

No podemos minimizar directamente la KL divergence debido al mismo problema:  $\log p(\mathbf{x})$ .

# Inferencia variacional

Sin embargo podemos escribir

$$\arg \min_{\nu} \text{KL}(q_{\nu}(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x}))$$

$$= \arg \min_{\nu} \mathbb{E}_{q_{\nu}(\mathbf{z}|\mathbf{x})} [\log q_{\nu}(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}, \mathbf{z})] + \log p(\mathbf{x})$$

$$= \arg \max_{\nu} \underbrace{\mathbb{E}_{q_{\nu}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}, \mathbf{z}) - \log q_{\nu}(\mathbf{z}|\mathbf{x})]}_{\text{ELBO}(\mathbf{x}; \nu)}$$

donde  $\text{ELBO}(\mathbf{x}; \nu)$  se conoce como *evidence lower bound objective*.

- Dado que  $\log p(\mathbf{x})$  no depende de  $\nu$  se puede considerar una constante.
- Así, minimizar la KL es equivalente a maximizar ELBO, algo que si es computable.
- Dado un conjunto  $\mathbf{d} = \{\mathbf{x}_i | i = 1, \dots, N\}$ , el objetivo final es  $\sum_{\{\mathbf{x}_i \in \mathbf{d}\}} \text{ELBO}(\mathbf{x}_i; \nu)$ .

# Inferencia variacional

Nótese que

$$\begin{aligned}\text{ELBO}(\mathbf{x}; \nu) &= \mathbb{E}_{q(\mathbf{z}; |\mathbf{x}\nu)} [\log p(\mathbf{x}, \mathbf{z}) - \log q_\nu(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\nu(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) - \log q_\nu(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_\nu(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q_\nu(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\end{aligned}$$

Por tanto, maximizar ELBO

- Fomenta que las distribuciones coloquen su masa en las configuraciones de variables latentes que explican los datos observados (primer término).
- Fomenta las distribuciones cercanas al anterior (segundo término).

# Inferencia variacional

## Optimización

$$\nu^* = \arg \max_{\nu} \text{ELBO}(\mathbf{x}; \nu).$$

Podemos utilizar el ascenso gradiente, siempre y cuando podamos evaluar  $\nabla_{\nu} \text{ELBO}(\mathbf{x}; \nu)$ . En general, este gradiente es difícil de calcular porque la expectativa es desconocida y los parámetros  $\nu$  son parámetros de la distribución  $q_{\nu}(\mathbf{z}|\mathbf{x})$  sobre la que integramos.

Auto-encoders y variational auto-encoders

---

## Variational Auto-encoders

# Variational Auto-encoders

Auto-encoders y variational auto-encoders

---

## Referencias

# Referencias

- ① **Lecture 11: Auto-encoders and variational auto-encoders**
- ② **Deep Learning Course**