Modelos Generativos 1

Material auxiliar

Máster en Inteligencia Artificial aplicada a Mercados Financieros

Jorge del Val

Índice

1. Introducción y motivación
2. Breve revisión de probabilidad
3. Reducción de dimensionalidad
   - PCA. Aplicación a bolsa
   - Autoencoders
4. Modelos generativos
   - Entrenamiento. Maximum Likelihood.
   - GMM

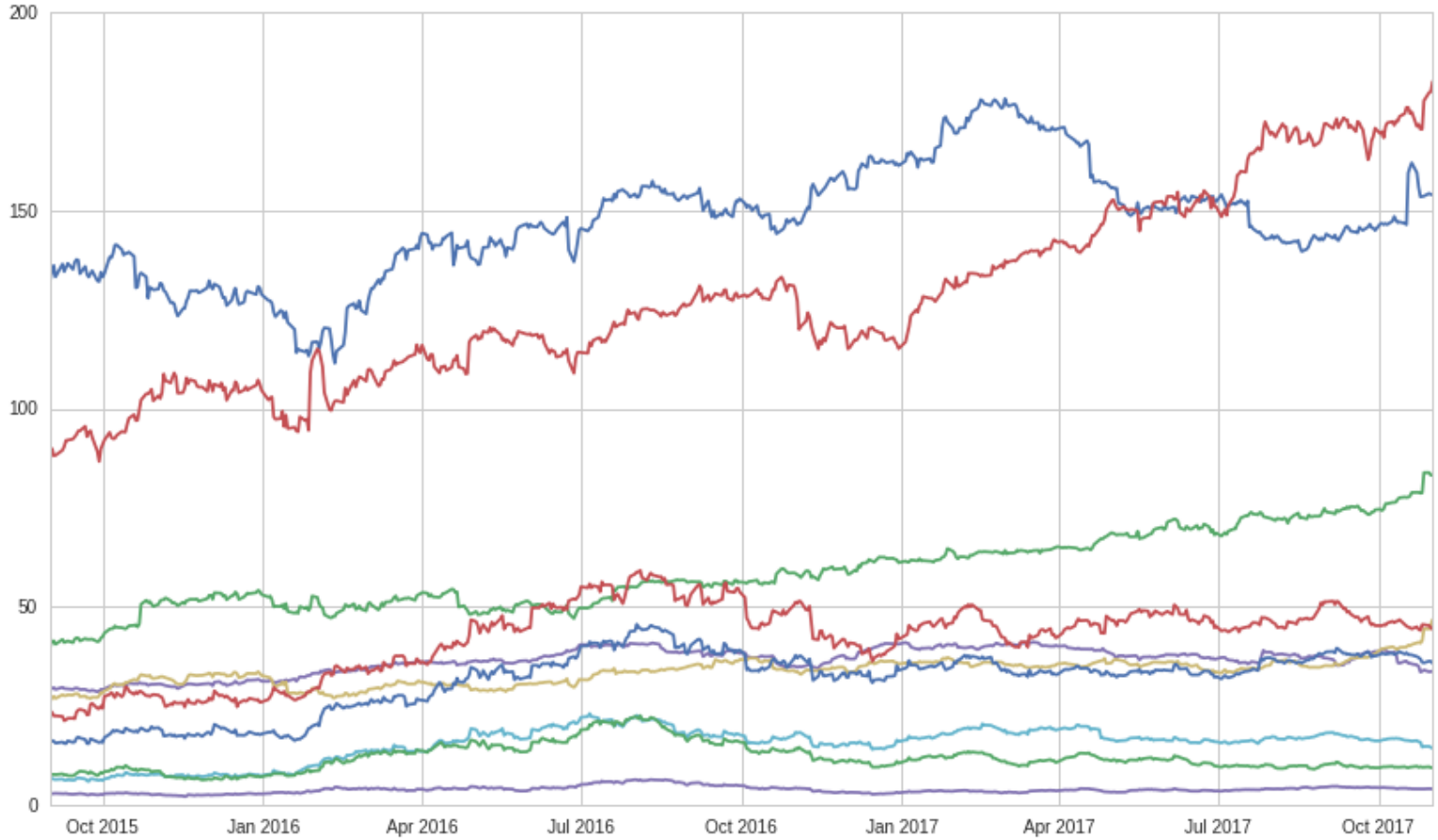
In a sentence…

Modelos que generan o manipulan cosas

In a better sentence…

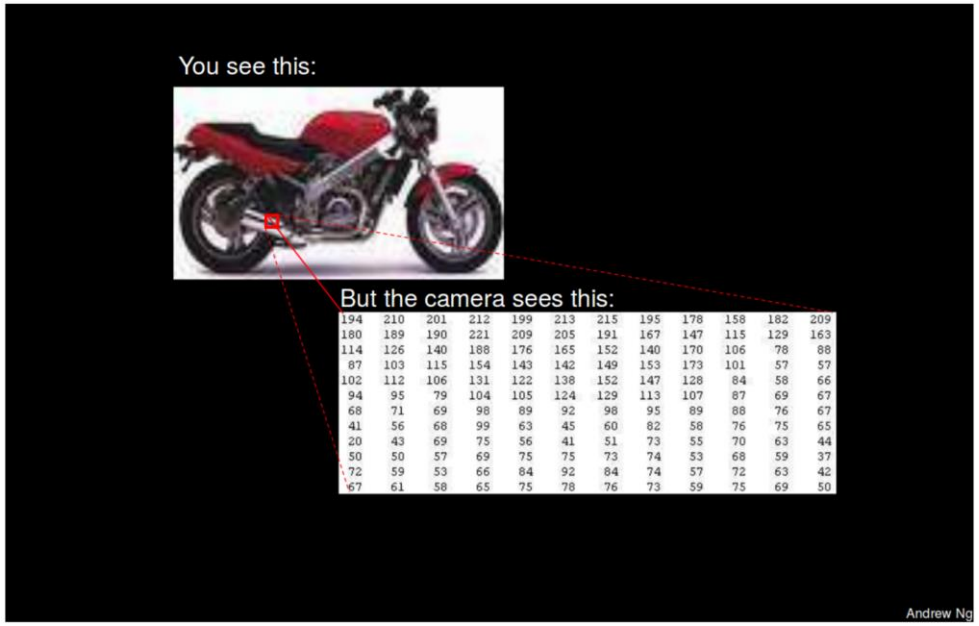
Modelos que aprenden la distribución de probabilidad de los datos y son capaces de sacar muestras

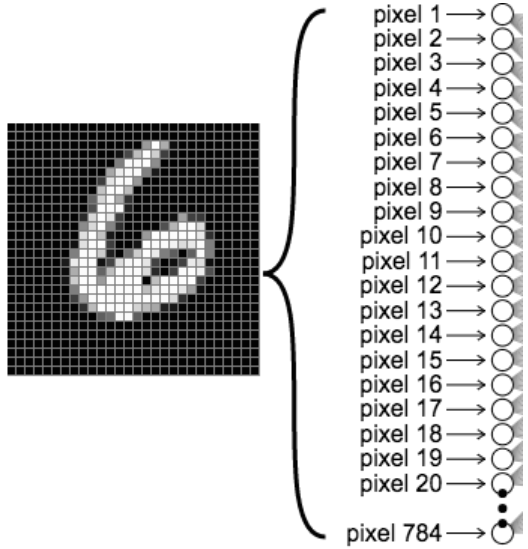
Pero… qué hacen realmente?


200
150
100
50
0
Oct 2015
Jan 2016
Apr 2016
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Jul 2017
Oct 2017


NEWS
Latest news
Economic Plan

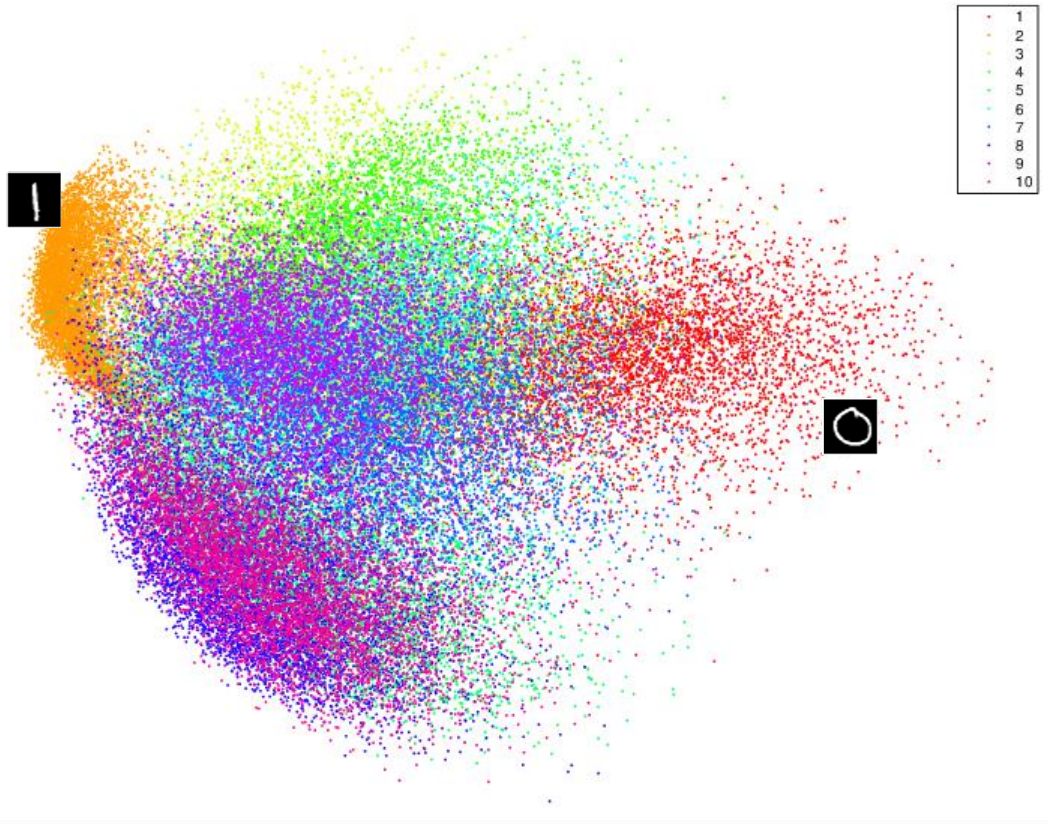
Al final… todo son números

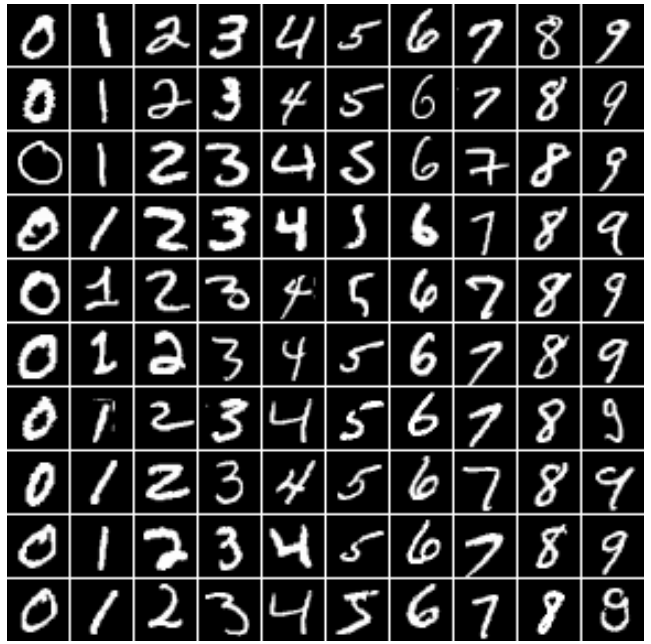




… en particular, vectores *M*-dimensionales en $\mathcal{X} \subseteq \mathbb{R}^M$.

Los datos están lejos de ser aleatorios

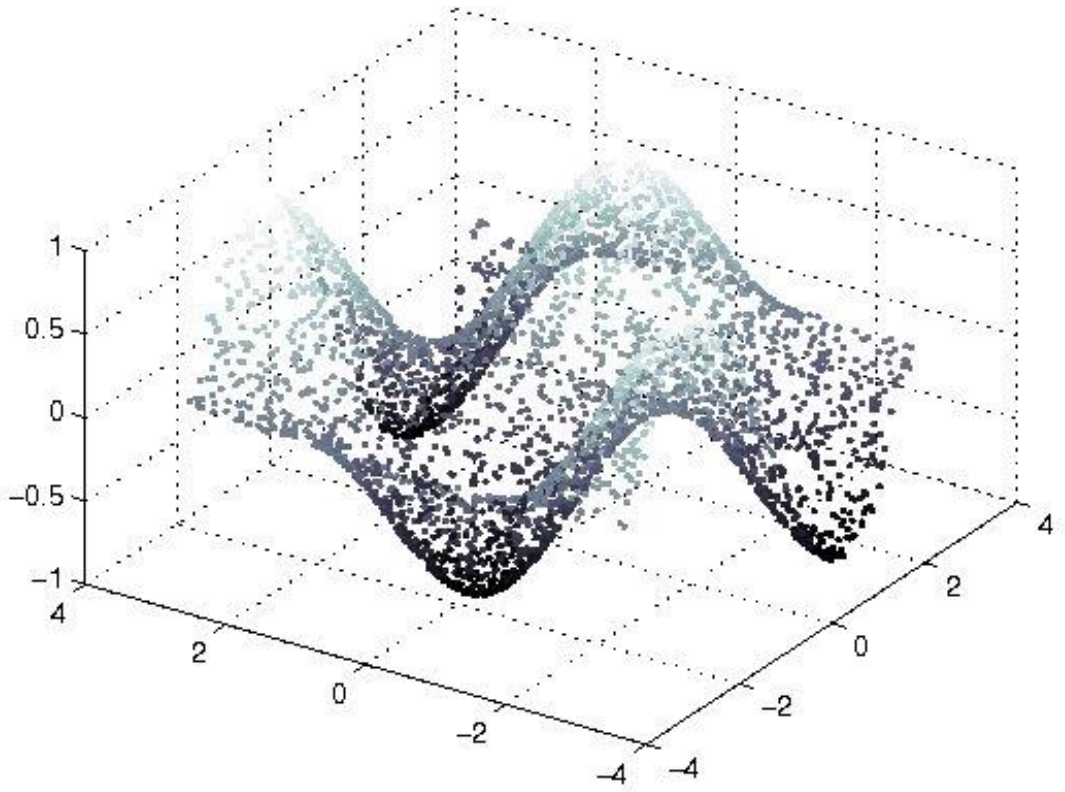




Necesitamos *M* pixels para representar una cara?

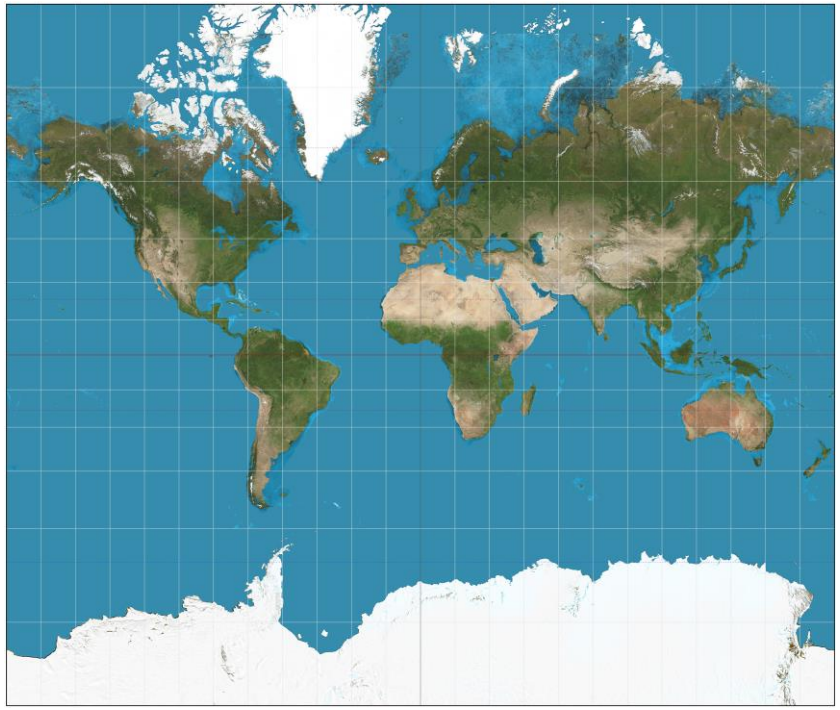
M=1000000 pixels!

Los datos no son realmente *M*-dimensionales

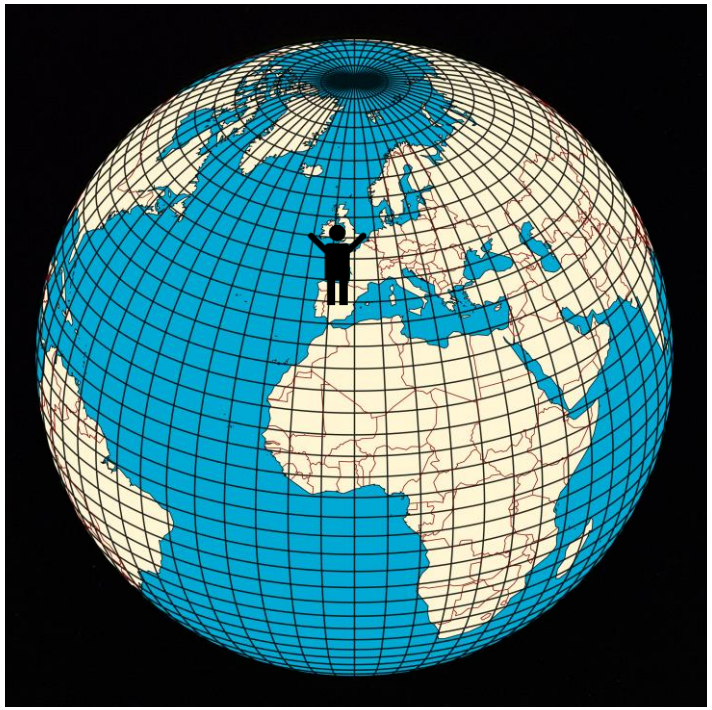


Se encuentran más bien en un *manifold* de menos dimensiones!

Manifold?

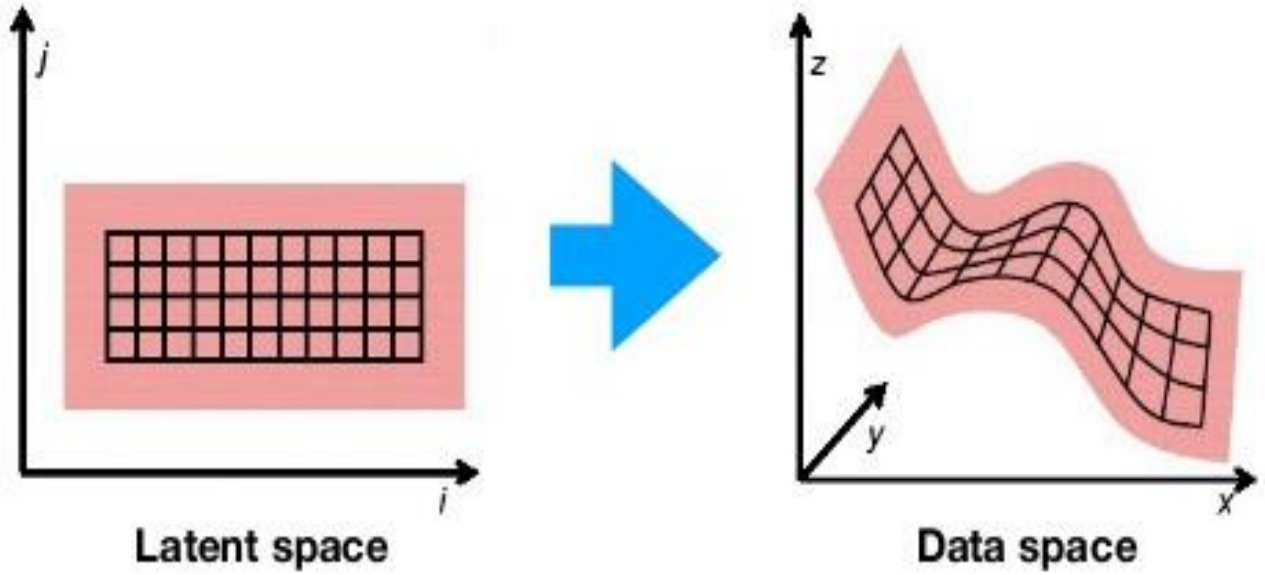


$z = (z_1, z_2)$



$x = (x_1, x_2, x_3)$

Dimensiones latentes de los datos

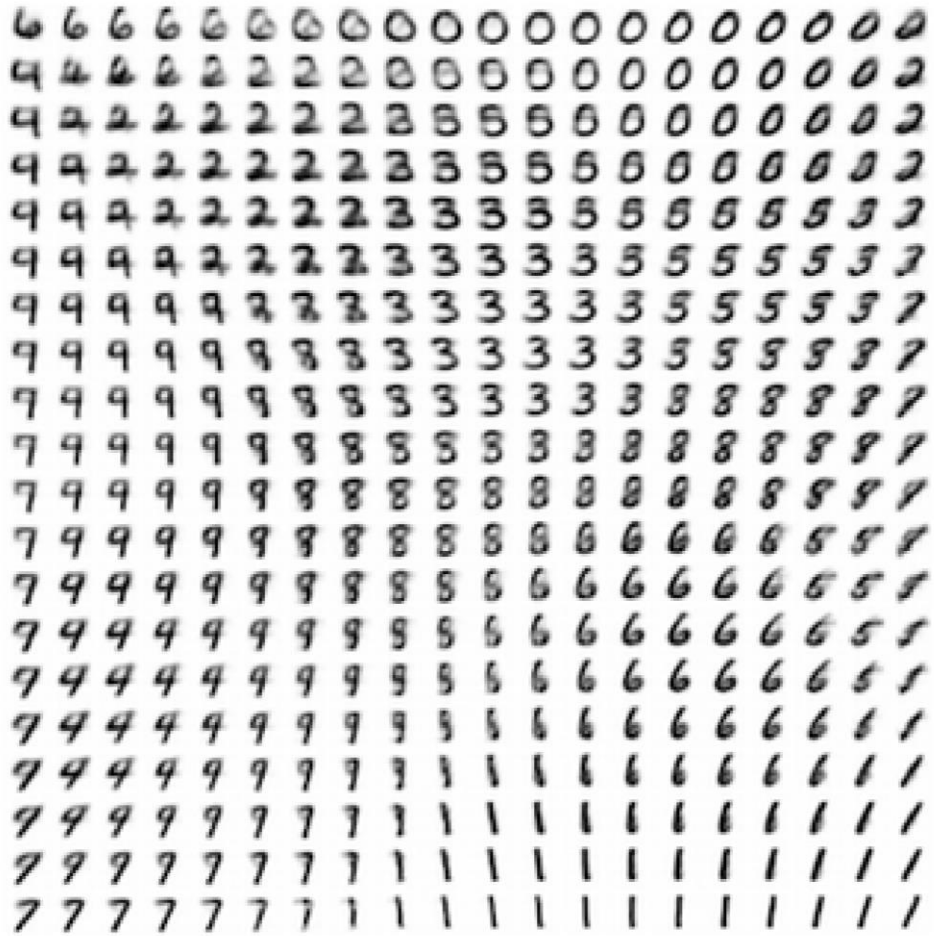


*__Spoiler__: Los modelos generativos aprenden ambas cosas: la geometría intrínseca de los datos y la distribución de probabilidad!

Images credit Ward A.D. et al 2007

z_2

z_1



z_2

z_1

Auto-Encoding Variational Bayes. Kingma et al. 2013

Un paseo por el espacio latente

A Style-Based Generator Architecture for Generative Adversarial Networks. Karras et al. 2018 (NVIDIA)

Breve revisión de probabilidad

… a modo de consulta

Variables aleatorias

Una variable aleatoria $\mathbf{x}$ es un objeto matemático que se puede *muestrear/realizar,* obteniendo resultados aleatorios x_i en un espacio muestral Ω.

$$\mathbf{x} \sim p(x), x \in \Omega$$

Comúnmente $\Omega \subseteq \mathbb{R}^N$, por lo que podemos sumar o hacer operaciones entre muestras.

¿Por qué variables aleatorias?

Para nosotros, cada dato x_i no es nada más que una realización de una variable aleatoria subyacente $\mathbf{x}$ *de la cual* ***no conocemos*** $p(x)$*!!*

El aprendizaje no supervisado y los modelos generativos intentan estimar la "realidad" subyacente $\mathbf{x}$ a partir de las muestras x_i

Probabilidad

Si Ω es discreto, e.g., categorías ($\Omega = \{\text{Gato}, \text{Perro}, \dots\}$), entonces

$$\sum_{x \in \Omega} p(x) = 1$$

Si Ω es continuo, e.g., precios de bolsa ($\Omega = \mathbb{R}$), entonces

$$\int_{x \in \Omega} p(x) = 1$$

Probabilidad conjunta

- La probabilidad de que pasen dos sucesos $\mathbf{x} = x$ e $\mathbf{y} = y$ "a la vez" se denomina probabilidad conjunta $p(x, y)$.

- La probabilidad condicional $p(x|y)$ es la distribución de $\mathbf{x}$ si se observa que $\mathbf{y} = y$.

- La probabilidad conjunta se puede expresar como una cadena de sucesos, i.e., $p(x, y) = p(y|x)p(x) = p(x|y)p(y)$.

Independencia

- Si los sucesos son *independientes* entonces $p(x|y) = p(y)$, i.e., la distribución de $\mathbf{x}$ no cambia al observar $\mathbf{y}$.

- …por lo tanto $p(x, y) = p(x)p(y)$.

Estadísticos: Valor esperado

$$\mathbb{E}[\mathbf{x}] = \int x\, p(x)\, dx$$

El valor esperado es un operador lineal! Esto es:

$$\mathbb{E}[\mathbf{x} + \mathbf{y}] = \mathbb{E}[\mathbf{x}] + \mathbb{E}[\mathbf{y}]$$

Estadísticos: Varianza y desviación típica

$$\mathrm{Var}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \mu_x)^2]$$

$$\sigma_x = \sqrt{\mathrm{Var}[\mathbf{x}]}$$

Estadísticos: Entropía

$$\mathrm{H}[\mathbf{x}] = \mathbb{E}[-\log_2(\mathbf{x})]$$

*Mide la cantidad de información contenida en una variable aleatoria en bits**

*Nerd add-on: cota inferior de la longitud media en bits de un código prefijo que comprimiera **x**

Estimando los estadísticos de las muestras

Si no tenemos $p(x)$ no podemos calcular nada, pero… y si tenemos *N* muestras $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}$ en su lugar?

Los estimadores se acercan al estadístico a medida que *N* crece. El más importante: la media como estimador del valor esperado:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \rightarrow \mathbb{E}[\mathbf{x}]$$

Covarianza

$$\mathrm{Cov}[\mathbf{x}, \mathbf{y}] = \mathbb{E}\big[(\mathbf{x} - \mu_x)(\mathbf{y} - \mu_y)\big]$$

...notad que $\mathrm{Var}[\mathbf{x}] = \mathrm{Cov}[\mathbf{x}, \mathbf{x}]$.

Variables aleatorias en múltiples dimensiones

- Si $\Omega(\mathbf{x})$ es un espacio de *N>1* dimensiones, $\mathbf{x}$ es simplemente un vector de *N* variables aleatorias:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix}$$

Variables aleatorias en múltiples dimensiones

$$\mathbb{E}[\mathbf{x}] = \begin{pmatrix} \mathbb{E}[\mathbf{x}_1] \\ \vdots \\ \mathbb{E}[\mathbf{x}_N] \end{pmatrix}$$

- El equivalente a “varianza” es ahora una matriz de covarianzas:

$$\Sigma = \begin{pmatrix} \mathrm{Cov}[\mathbf{x}_1, \mathbf{x}_1] & \cdots & \mathrm{Cov}[\mathbf{x}_1, \mathbf{x}_N] \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}[\mathbf{x}_N, \mathbf{x}_1] & \cdots & \mathrm{Cov}[\mathbf{x}_N, \mathbf{x}_N] \end{pmatrix}$$

Variables aleatorias en múltiples dimensiones

La varianza en la dirección de un vector (unitario) u es simplemente

$$\mathrm{Var}_u[\mathbf{x}] = u^T \Sigma u$$

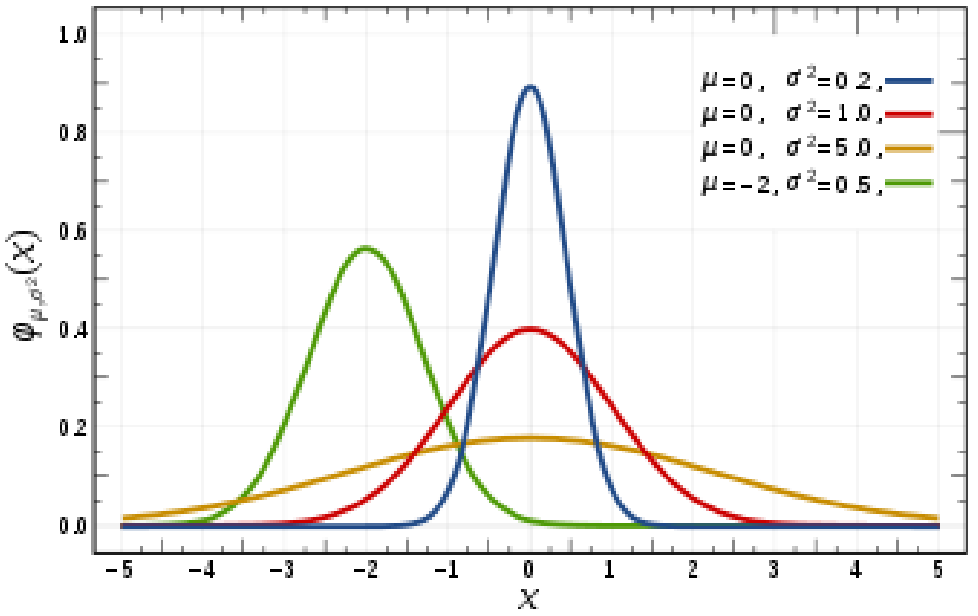
Por lo que la matriz de covarianzas nos permite calcular la varianza en cualquier dirección.

* Fun stuff: podéis demostrar esto? :)

Distribuciones: normal unidimensional

$$\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$$

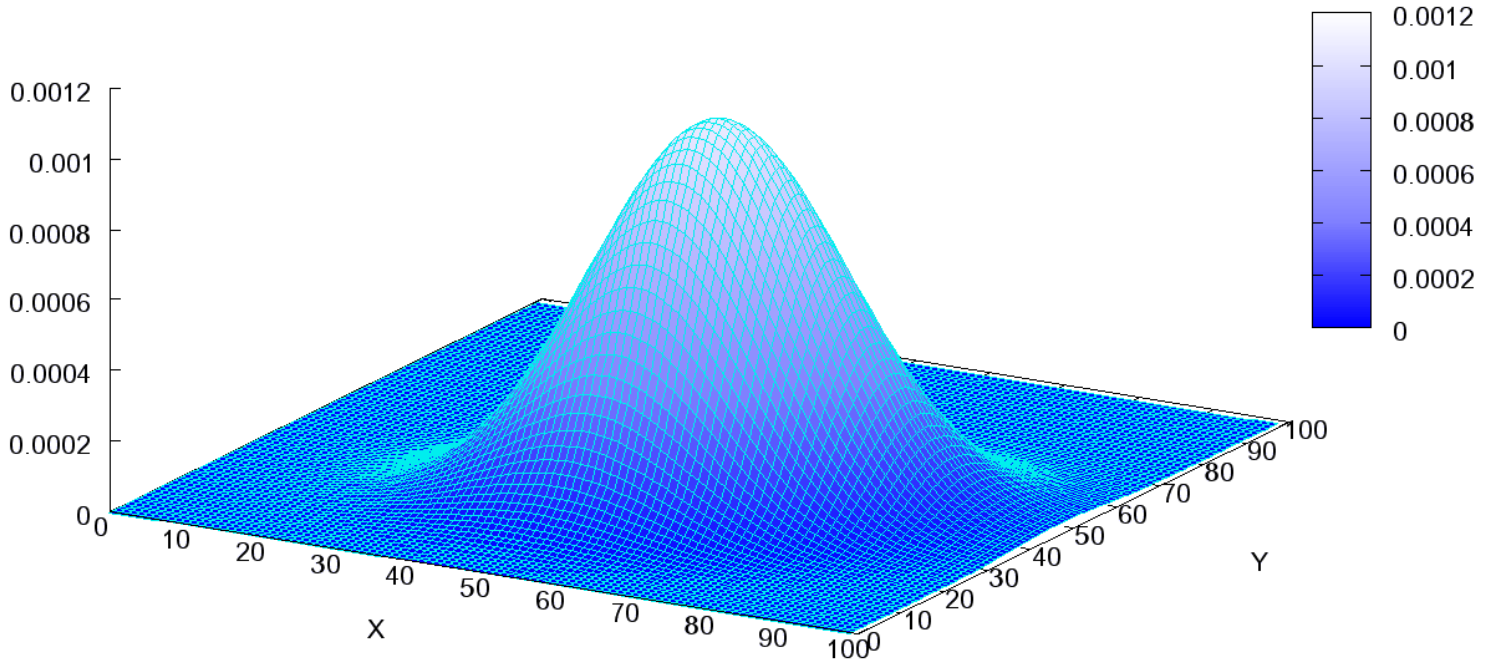
- $\mathbb{E}[\mathbf{x}] = \mu$
- $\mathrm{Var}[\mathbf{x}] = \sigma^2$



Distribuciones: normal multidimensional

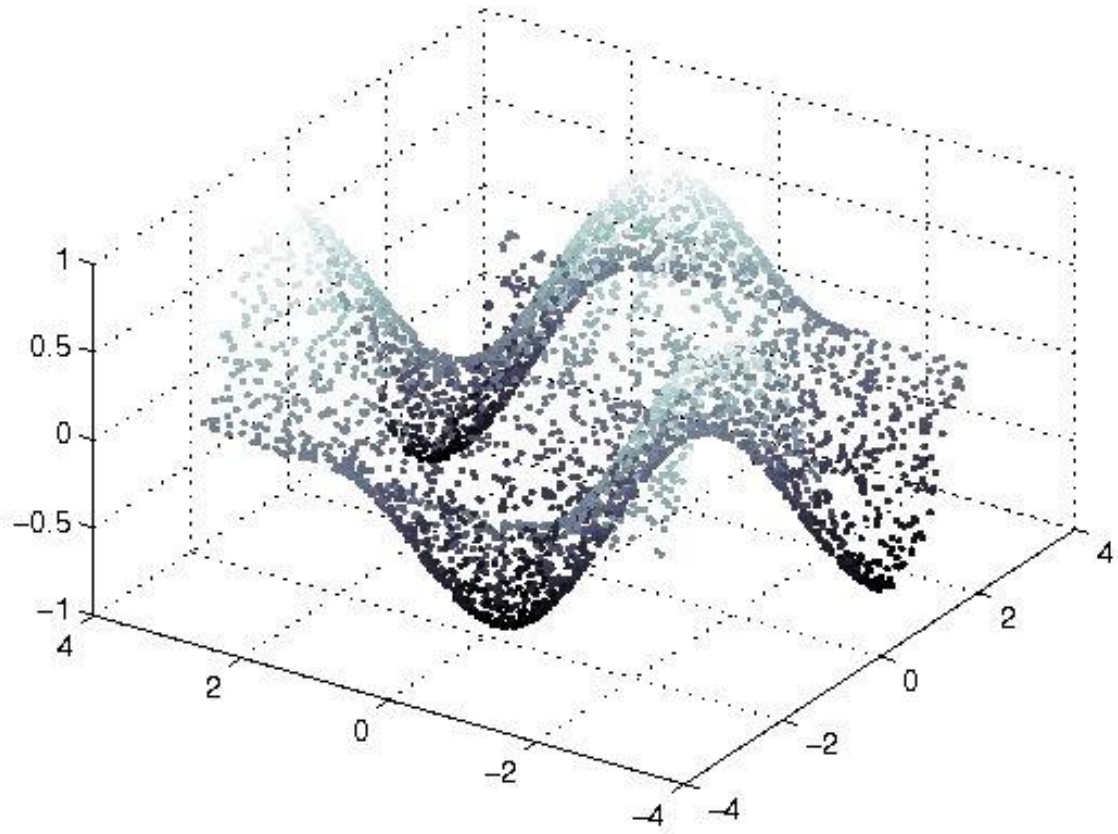
$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$$

- $\mathbb{E}[\mathbf{x}] = \mu \in \mathbb{R}^N$
- $\mathrm{Cov}[\mathbf{x}] = \Sigma \in \mathbb{R}^{N\mathrm{x}N}$

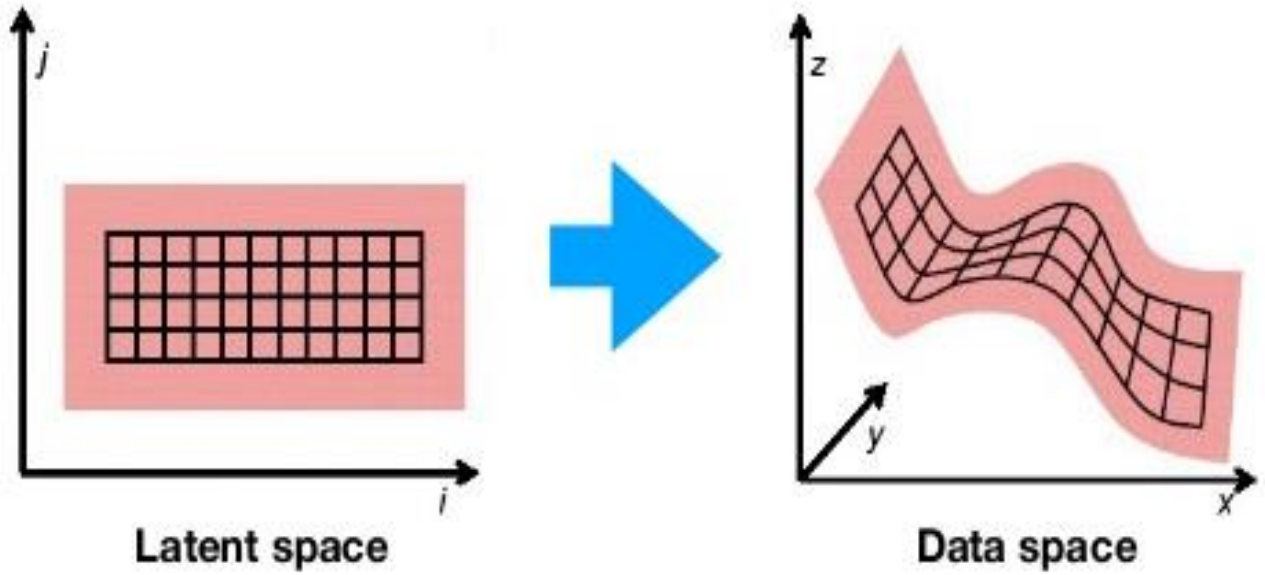


Reducción de dimensionalidad

… recordando

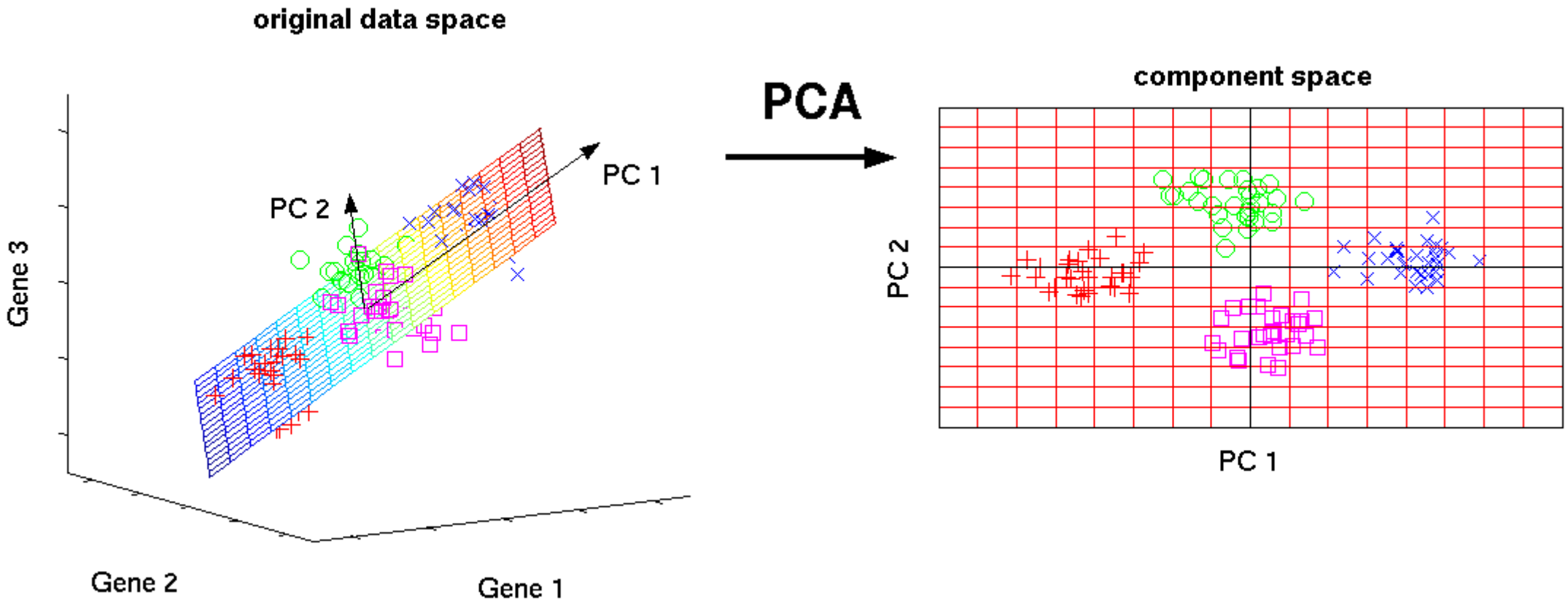


Dimensiones latentes de los datos



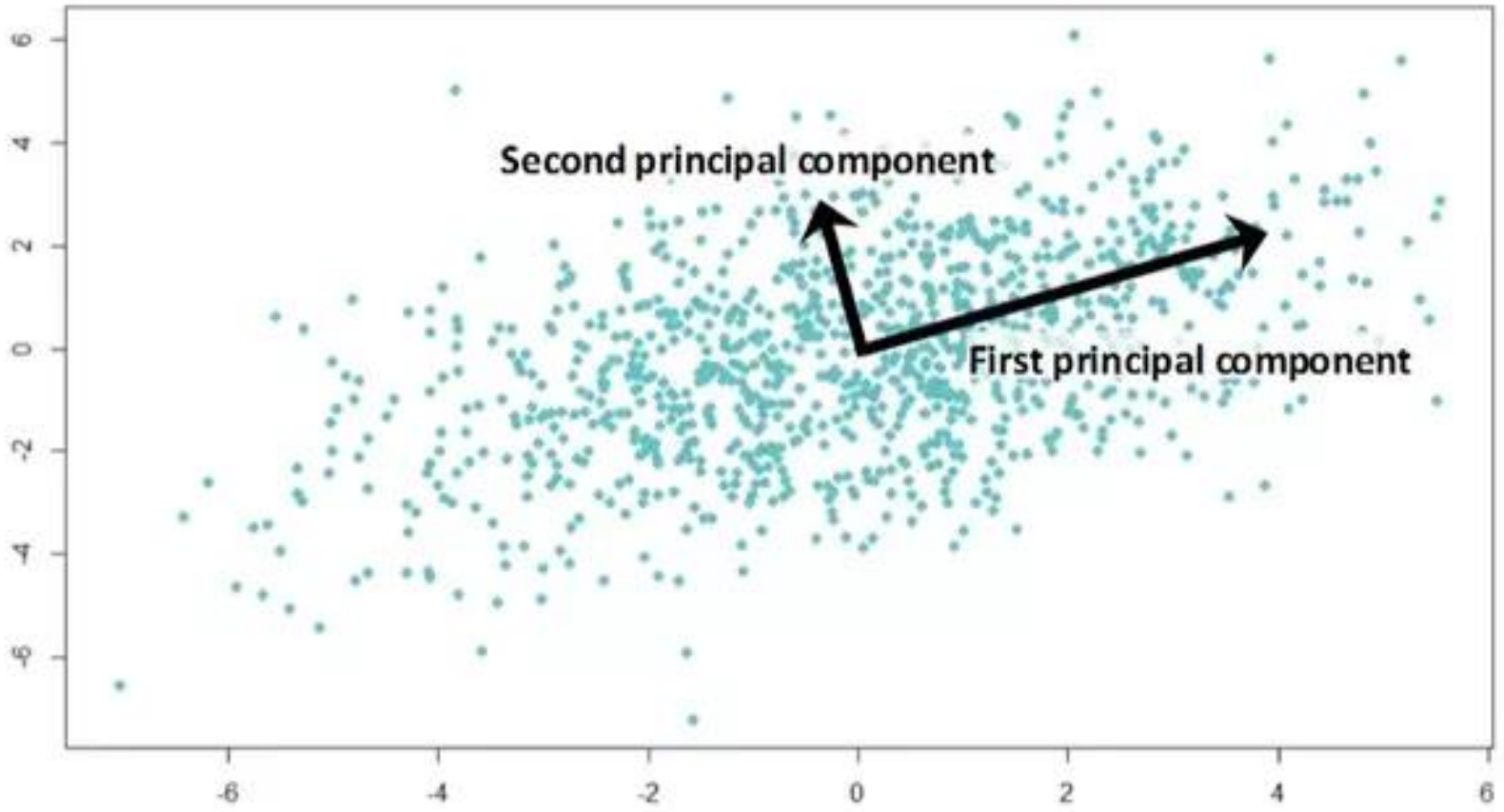
Images credit Ward A.D. et al 2007

Principal Component Analysis



PCA asume que el manifold es lineal!

Principal Component Analysis



¿Qué es realmente un componente?

Un componente principal es un vector unitario u que maximiza localmente la varianza en esa dirección. I.e.,

$$\max_{u} u^T \Sigma u$$
$$s.t. \; ||u|| = 1$$

Resulta que si resolvemos este problema tenemos la ecuación:

$$\Sigma u = \lambda u$$

…es un autovector de la matriz de covarianza!

En finanzas...

- Factor analysis, portfolio management, algorithmic trading…

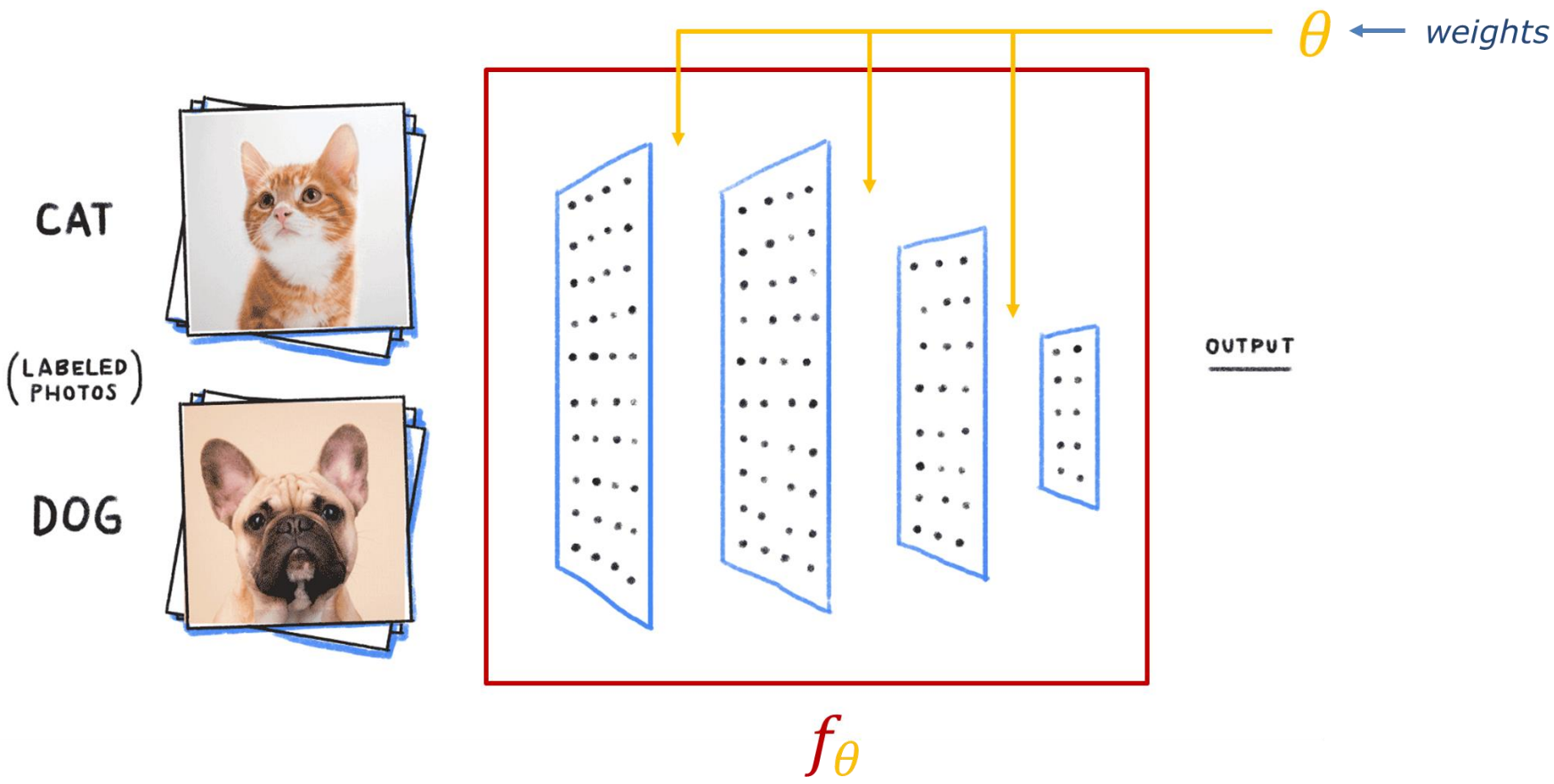
- *Statistical Arbitrage in the U.S. Equities Market.* M. Avellaneda, J.H. Lee. 2008. https://www.math.nyu.edu/faculty/avellane/AvellanedaLeeStatArb20090616.pdf

Hands on

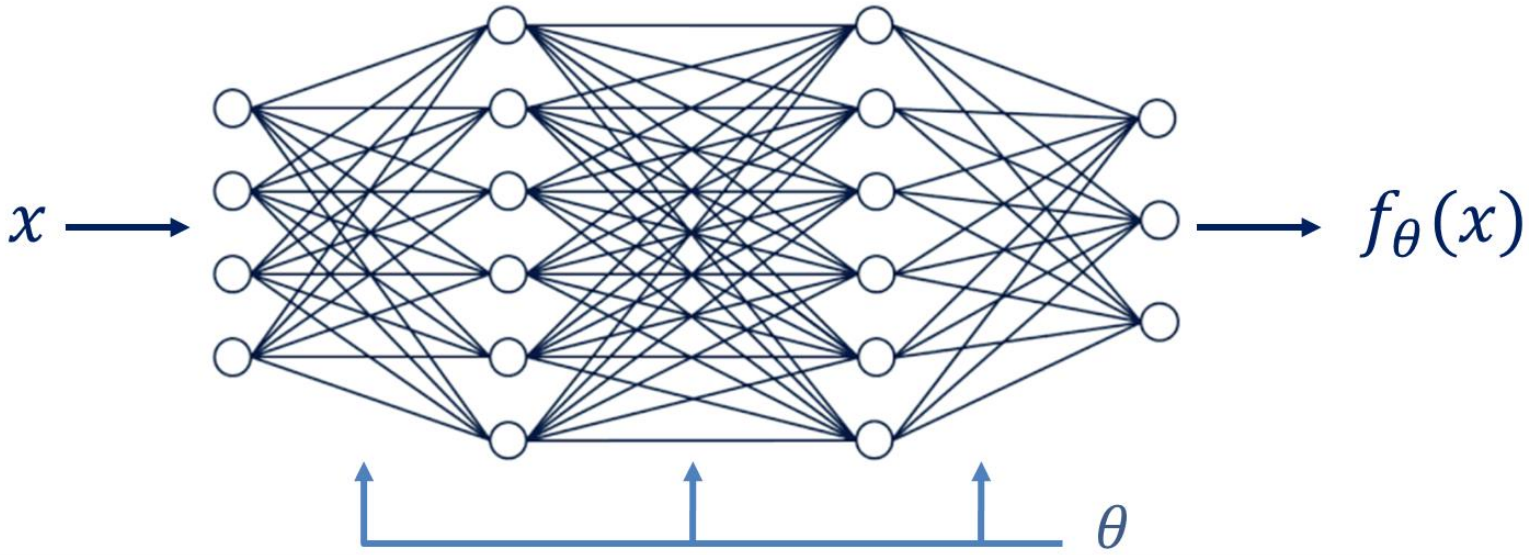
- Datos de precios del S&P500: https://www.kaggle.com/camnugent/sandp500

(Muy) breve revisión de deep learning

Redes neuronales



Redes neuronales



*Las redes neuronales pueden aproximar cualquier función $f(x)$ a cualquier precisión!**

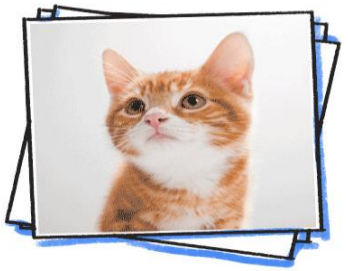
Pero cómo encontramos un buen θ (entrenamiento)?

Optimizamos una función de coste (error)!

$$\min_{\theta} L(\theta; \text{data})$$

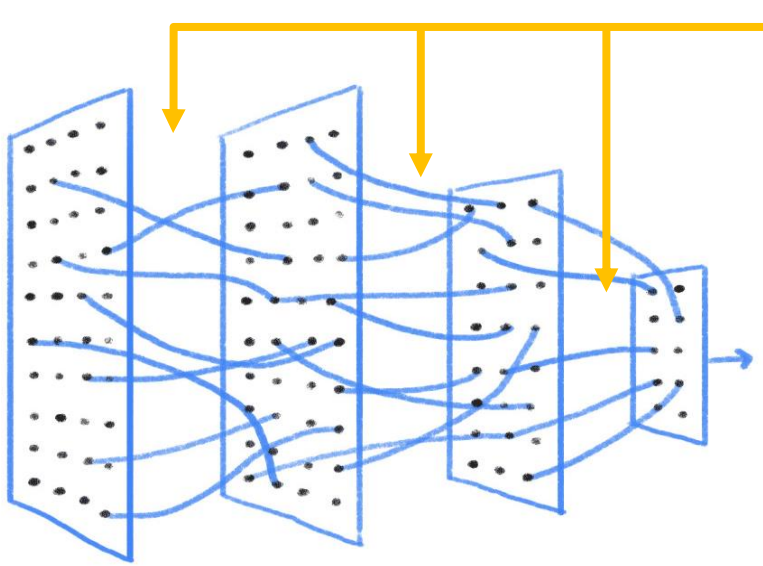
E.g. Classifier

$y_{cat} \in \{0,1\}$



x (LABELED PHOTOS)

$y_{dog} \in \{0,1\}$



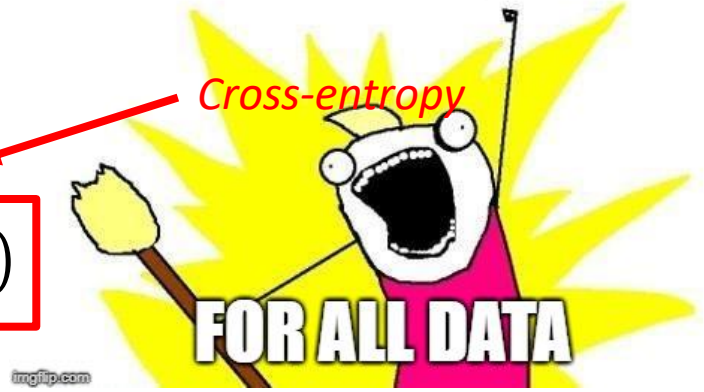
θ

$f_\theta(x)$ I want it to be... $p(cat)$

MINIMIZE

Cross-entropy

$$L(\theta; x, y) = -y_{cat} \log f_\theta(x) - y_{dog} \log\left(1 - f_\theta(x)\right)$$



E.g. Classifier

$y_{cat} \in \{0,1\}$

x (LABELED PHOTOS)

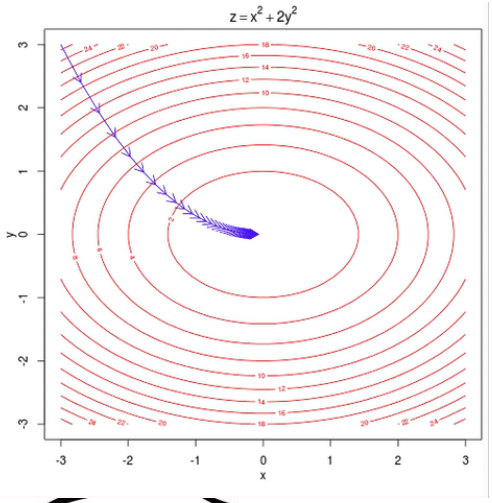
CAT

DOG

$p(cat)$

MINIMIZE

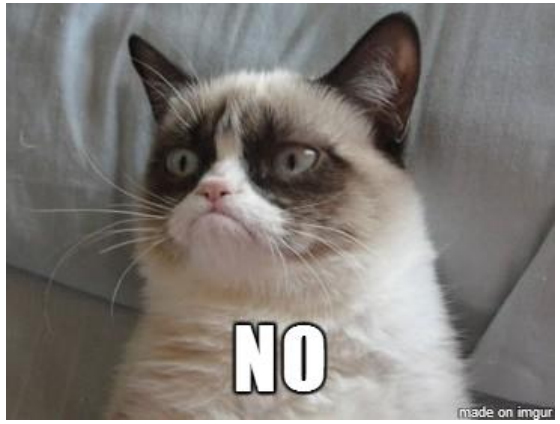
Pero cómo encontramos un buen θ (entrenamiento)?

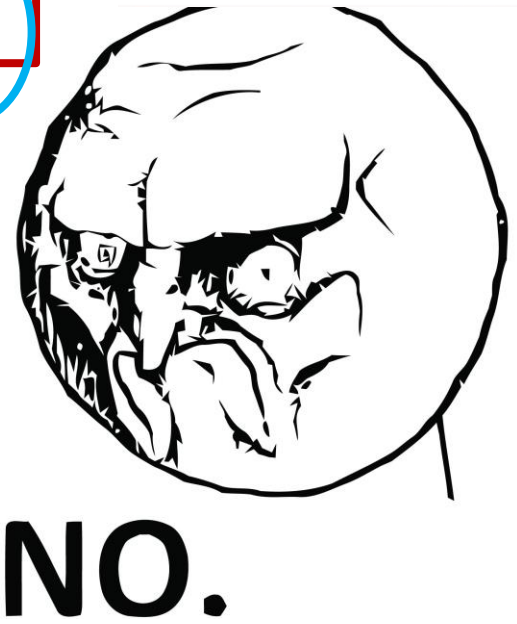


Optimizamos una función de coste (error)!

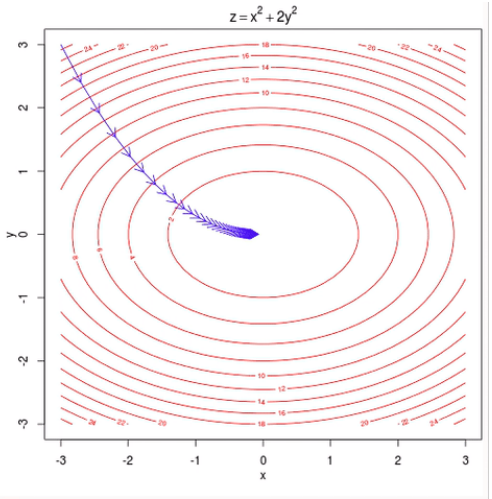
Stochastic Gradient Descent

$$\min_{\theta} L(\theta; \text{data}) \longrightarrow \theta_{t+1} = \theta_t - \alpha \nabla_\theta L$$





Pero cómo encontramos un buen θ (entrenamiento)?



Optimizamos una función de coste (error)!

Stochastic Gradient Descent

$$\min_{\theta} L(\theta; \text{data}) \rightarrow \theta_{t+1} = \theta_t - \alpha \nabla_\theta L$$

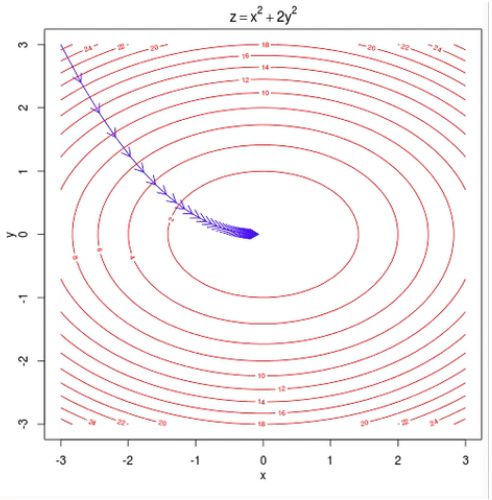
```
model = MyNetwork()
theta = model.parameters()
optimizer = torch.optim.Adam(theta, lr=0.001)

for x, y in dataloader:
    y_pred = model(x)
    loss = myloss(y, y_pred)
    loss.backwards()
    optimizer.step()
```

PyTorch

```
model = MyNetwork()
theta = model.trainable_variables
optimizer = tf.train.AdamOptimizer(lr = 0.001)

for x, y in dataset:
    with tf.GradientTape() as g
        y_pred = model(x)
        loss = myloss(y, y_pred)
    grads = g.gradient(loss, theta)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

TensorFlow

Pero cómo encontramos un buen θ (entrenamiento)?



Optimizamos una función de coste (error)!

Stochastic Gradient Descent

$$\min_{\theta} L(\theta; \text{data}) \rightarrow \theta_{t+1} = \theta_t - \alpha \nabla_\theta L$$

```
model = MyNetwork()
theta = model.parameters()
optimizer = torch.optim.Adam(theta, lr=0.001)

for x, y in dataloader:
    y_pred = model(x)
    loss = myloss(y, y_pred)
    loss.backwards()
    optimizer.step()
```

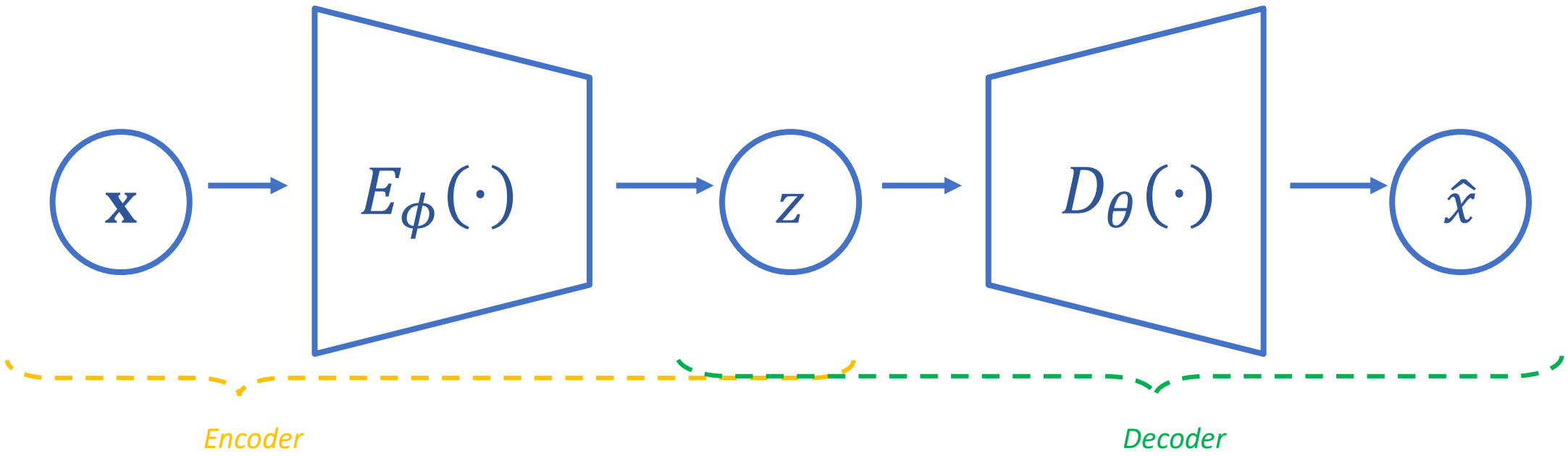
PyTorch

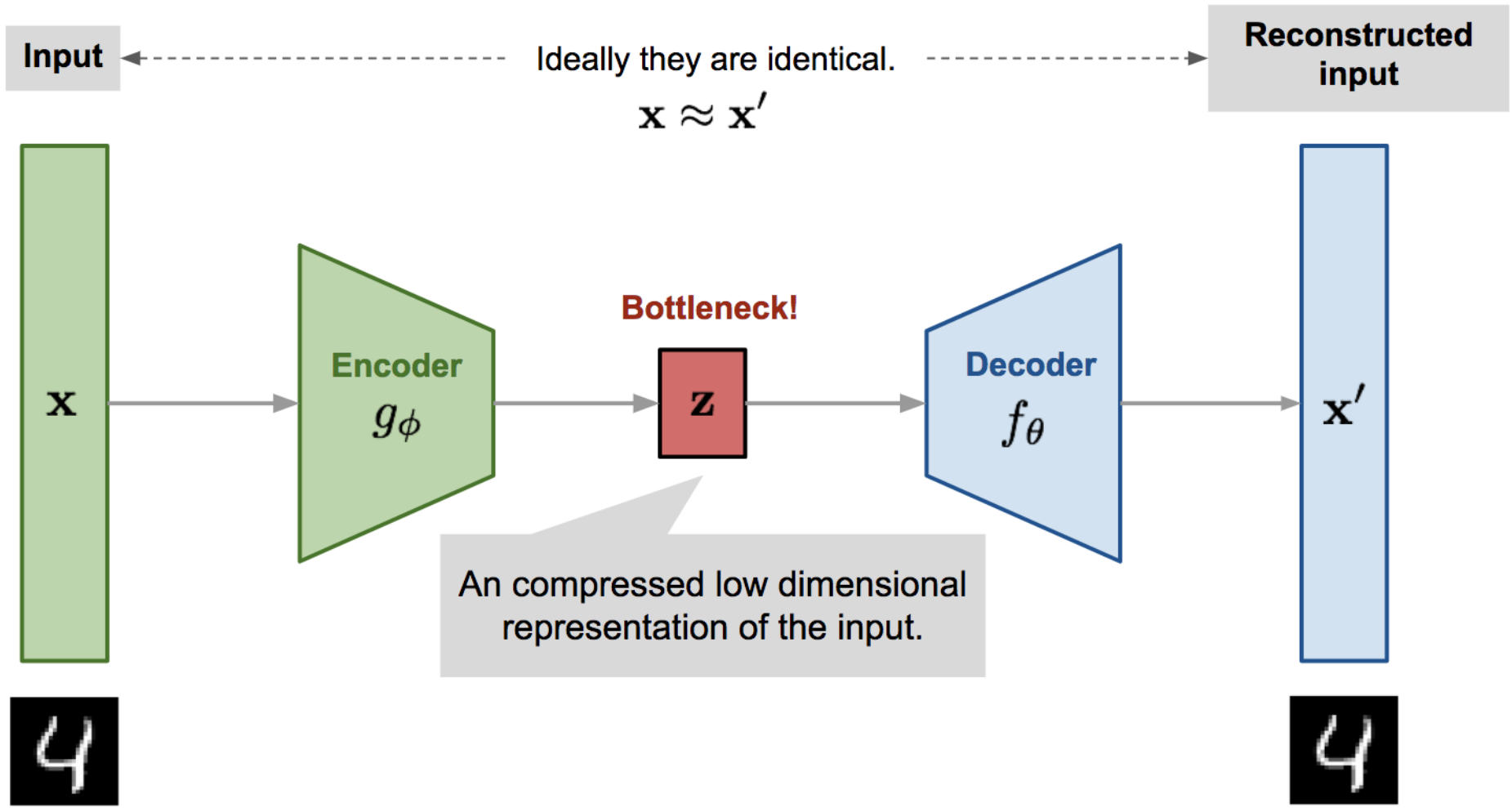
```
model = MyNetwork()
theta = model.trainable_variables
optimizer = tf.train.AdamOptimizer(lr = 0.001)

    loss = myloss(y, y_pred)
grads = g.gradient(loss, theta)
optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

TensorFlow

Easy to gradient descent any function with current frameworks!! ☺

Autoencoders


$\mathbf{x}$
$E_\phi(\cdot)$
z
$D_\theta(\cdot)$
$\hat{x}$
Encoder
Decoder


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Autoencoders

- Hacemos que la variable z tenga menos dimensiones que x.
- Si podemos reconstruir x sólo con z, entonces z guarda toda la información de x a pesar de tener menos dimensiones.
- El proceso de optimización fuerza a que así sea y, por lo tanto, z sea una *representación* o *código* de x.

Hands-on con Keras…

Hands-on con Keras...

- Diseñemos un autoencoder de dígitos!

Modelos generativos

Recordando…

Para nosotros, cada dato x_i es una *realización* de una variable aleatoria subyacente $\mathbf{x}$, con una distribución de probabilidad $p(x)$ desconocida

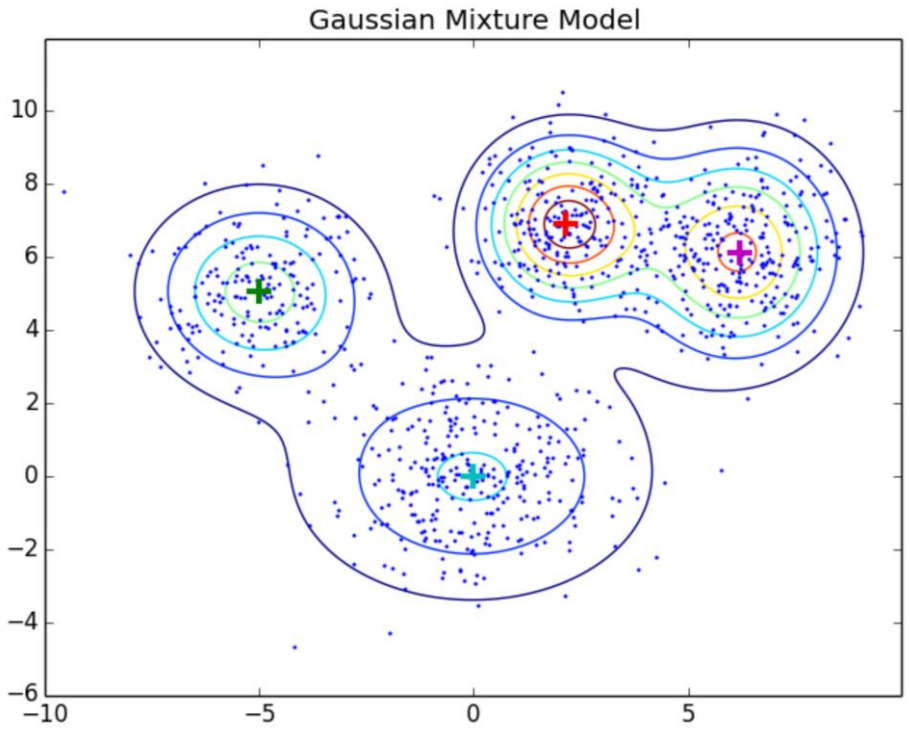
$$\mathbf{x} \sim p(x)$$

- El *aprendizaje no supervisado* es el campo que intenta inferir propiedades de $\mathbf{x}$ sólo con las muestras (datos).
- Los *modelos generativos* son un subconjunto del aprendizaje no supervisado que pretende *aproximar* $\mathbf{x}$ como una combinación de variables aleatorias "simples" que se puedan muestrear:

$$\mathbf{x} \approx G_\theta(\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_k}) \triangleq \hat{\mathbf{x}}$$

Ejemplo: Gaussian Mixture Models

Aquí cada z_i son gaussianas $\mathcal{N}(\mu_i, \Sigma_i)$ y la combinación $f_\theta(\cdot)$ es una *mezcla*.



Entrenamiento

Queremos aproximar $\mathbf{x}$ como $\hat{\mathbf{x}} = G_\theta(\mathbf{z})$. ¿Cómo encontramos los parámetros θ óptimos?

Maximiza la verosimilitud (likelihood) de tu modelo!!

Probabilidad de que tu modelo generara x_i

$$\max_\theta \mathcal{L}(\theta|x_{train}) = \max_\theta \prod_{i=1}^{N} p_\theta(x_i)$$

$$\max_\theta \log \mathcal{L}(\theta|x_{train}) = \max_\theta \sum_{i=1}^{N} \log p_\theta(x_i)$$

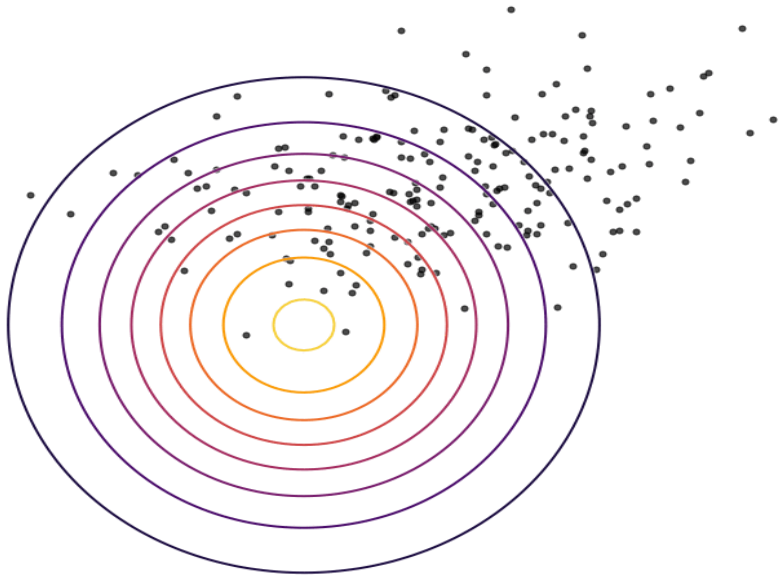


Image credit: Colin Raffel

Necesitamos $p_\theta(x)$ explícitamente!

Hands-on!

- Combinemos PCA con GMM para tener un modelo probabilístico tratable.