

# El éxito de Deep Learning

→ Las ideas principales sobre feed forward networks no han cambiado mucho en el periodo 1980 - 2015.

→ Factores de cambio después de 2015:

- \* Grandes bases de datos disponibles reducen el estrés sobre generalización estadística
- \* Más poder de cómputo = redes más grandes.

→ Pequeños pero notables avances

\* error cuadrático medio a entropía cruzada

$$mse(y, \hat{y}) = \frac{1}{N} \sum_{n=1}^N \|y_n - \hat{y}_n\|_2^2$$

$$D_{KL}(p(x) || q(x)) = E_p \{ I(q(x)) - I(p(x)) \}$$

$$D_{KL}(p(x) || q(x)) = \int p(x) \log \left( \frac{p(x)}{q(x)} \right) dx$$

$$= \int p(x) \log(p(x)) dx - \int p(x) \log(q(x)) dx$$

$$D_{KL}(p || q) = -H(p) + H(p; q)$$

Cross-entropy:  $H(p; q) = - \int p(x) \log(q(x)) dx$

\* 1980-1990 → MSE

\* 1990 →  $H(p; q)$ ;  $p(y | f)$   
cross-entropy      likelihood → Bayes

• Otros cambios: sigmoid, tanh  $\rightarrow$  ReLU y  
varianter

**NOTA:** De 2006-2012 se creía que las FFN no funcionaban bien a menos que se les ayudara con otros tipos de modelos, ej: probabilísticos.

→ Hoy en día se sabe que bajo unos buenos recursos (Colab, Kaggle, Azure, AWS) y buenas prácticas, FFN funcionan bien (autograd, ReLU, Inicializadores, Batch-normalization, Regularizadores)

→ Incluso, métodos basados en gradiente descendente y FFN se usan para desarrollar modelos probabilísticos. (Variational autoencoders, Generative adversarial networks, Deep Gaussian Process)

→ **Deep Learning clásico:** Supervised learning  
 $f: \mathbb{R}^P \rightarrow \mathbb{R}^M$ ; aprender  $f$  desde  $\{x_n \in \mathbb{R}^P, y_n \in \mathbb{R}^M\}_{n=1}^N$  con  $N \rightarrow \infty$ .

→ **Deep Learning Research:**

- Generar nuevas muestras
- Determinar  $P(x|f, \theta)$
- Tratar con valores perdidos.
- Relacionar tareas
- ⋮

## → Claves en Deep Learning Research

- \* Incluir unsupervised / semi-supervised
- \* Reducir la cantidad de datos etiquetados requeridos.

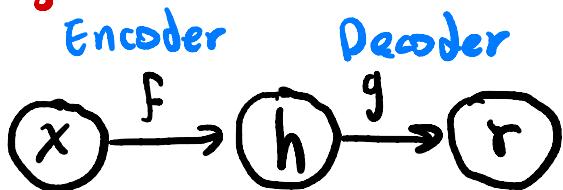
→ **Problema principal:** unsupervised learning con alta-dimensionaldad ( $P \rightarrow \infty$ ) de las variables aleatorias.

→ **Desafíos:** estadístico/computacional

$\downarrow$   
Generalización

$\downarrow$   
tratar con muchas dimensiones.

→ **Algoritmo inicial - Autoencoders.**



$$f^*, g^* = \arg \min_{f, g} L(x, g(f(x)))$$

→ Autoencoder modernos extienden la idea de funciones determinísticas a funciones estocásticas  
 $P_{\text{encoder}}(h|x)$ ;  $P_{\text{decoder}}(x|h)$ .

## → Sparse Autoencoder. (Regularized)

$$f^*, g^* = \arg \min_{f,g} L(x, g(f(x))) + \mathcal{R}(h)$$



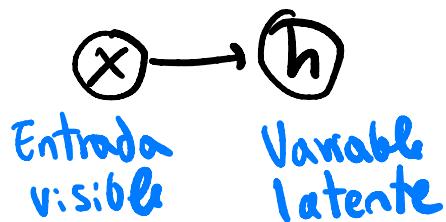
↓  
Modelo:  
 - ralo  
 - ▽ pequeñas  
 - ruido  
 - Entradas perdidas

**NOTA:** Cualquier modelo generativo  $\hat{x} = g(f(x))$  con variables latentes  $h$  y procedimiento de inferencia puede entenderse como un autoencoder.

**Modelos generativos derivados de autoencoders:**

- Helmholtz machine → variational autoencoder
- generative stochastic network

→ Sparse (regularized) autoencoder como approx. maximum likelihood of generative models.



$$p_{\text{model}}(x, h) = p(x|h) p(h)$$

↓  
Prior (creencias)

Podemos que:  $p(x) = \sum_h p(x, h)$

$$\log(p(x)) = \log\left(\sum_h p(x, h)\right) \quad (*)$$

Autoencoder: aprox. de  $(*)$  por estimación puntual

Así:

$$\begin{aligned}\log(p(h, x)) &= \log(p(x|h)p(h)) \\ &= \log(p(x|h)) + \log(p(h))\end{aligned}$$

Por ejemplo: para inducir prior sparse, se puede fijar pdf Laplaciana:

$$p(h_i) = \frac{\lambda}{2} e^{-\lambda|h_i|}$$

$$p(h) = \prod_i p(h_i) = \prod_i \frac{\lambda}{2} e^{-\lambda|h_i|}$$

$$\log(p(h)) = \log\left(\frac{\lambda}{2} \prod_i e^{-\lambda|h_i|}\right) = \underbrace{\log\left(\frac{\lambda}{2}\right)}_{\text{cte}} - \underbrace{\sum_i \lambda|h_i|}_{\Omega(h)}$$

**NOTA:** El término de penalidad en el autoencoder se puede entender como el efecto de  $p(h)$  sobre una aprox. de maximum likelihood.

↪ Autoencoder como aprox. de un modelo generativo.

Denoising autoencoders (DAE):

$$f^*, g^* = \arg \min_{f, g} L(x, g(f(\tilde{x})))$$

$$\tilde{x} = x + \eta ; \quad \eta : \text{ruido.}$$

DAE permite aprox.  $p(x)$

## Regularización por penalidad en derivadas

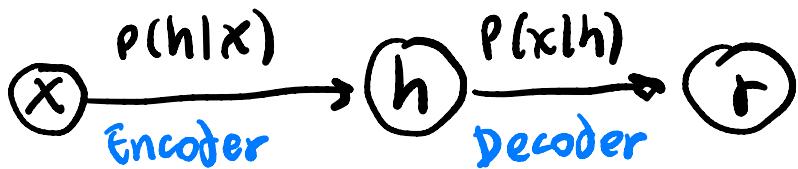
$$f^*, g^* = \arg \min_{f, g} L(x, g(f(x))) + \Omega(h, x)$$

$$\Omega(h, x) = \lambda \sum_i \|\nabla_x h(x)\|$$

→ se obliga a que el modelo no cambie mucho ante pequeñas variaciones de  $x$ .

contractive autoencoder - (CAE).

## Stochastic encoders and decoders



Se extiende el concepto de función a distribución:

$$f(x) \rightarrow p(h|x)$$

$$g(h) = g(f(x)) \rightarrow p(x|h)$$