

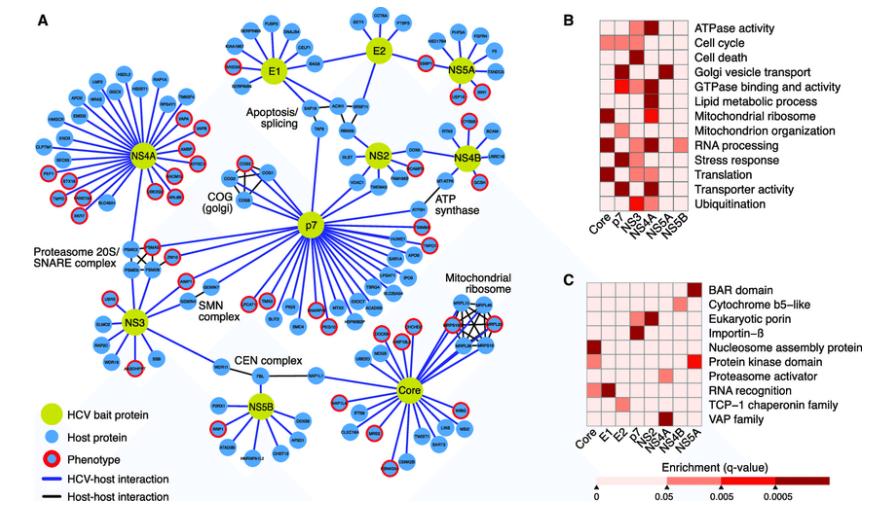
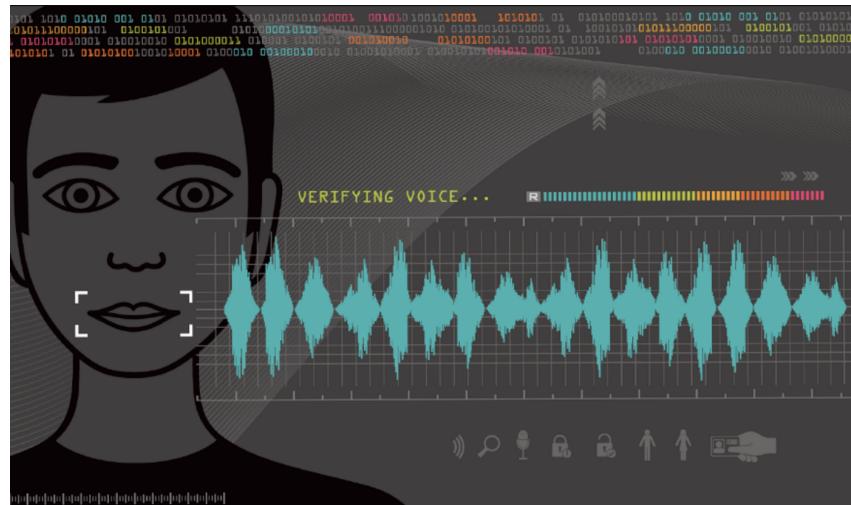
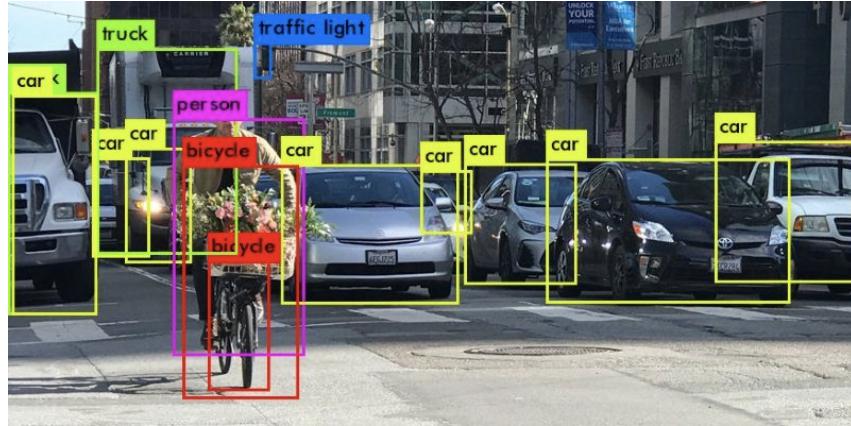
# Intro al Aprendizaje No-Supervisado

Redes de Hopfield y Redes de Boltzmann



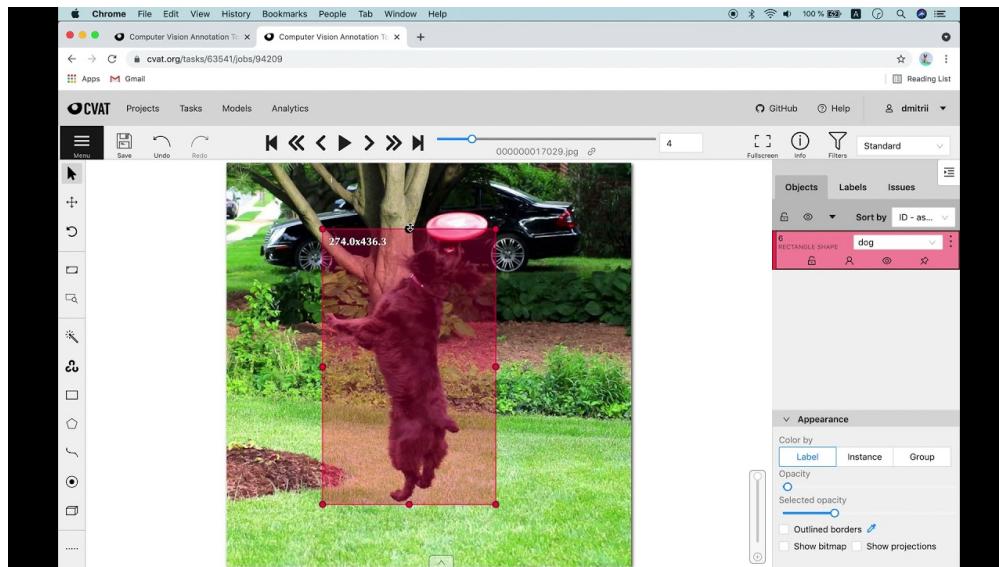
Prof. Ricardo Ñanculef - Departamento de Informática UTFSM 2022

# Introducción



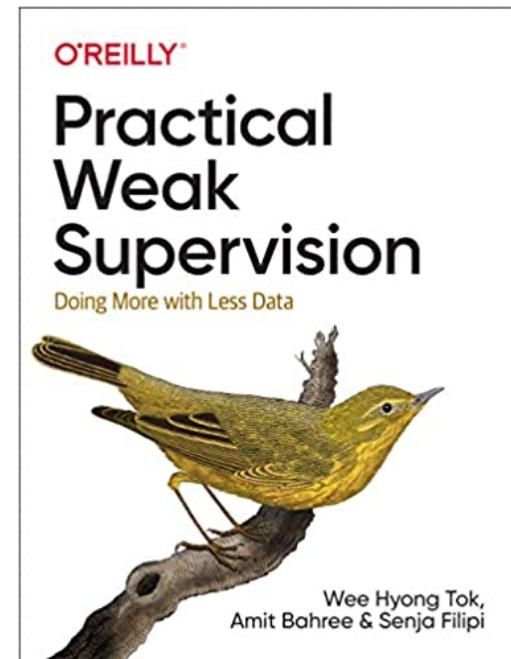
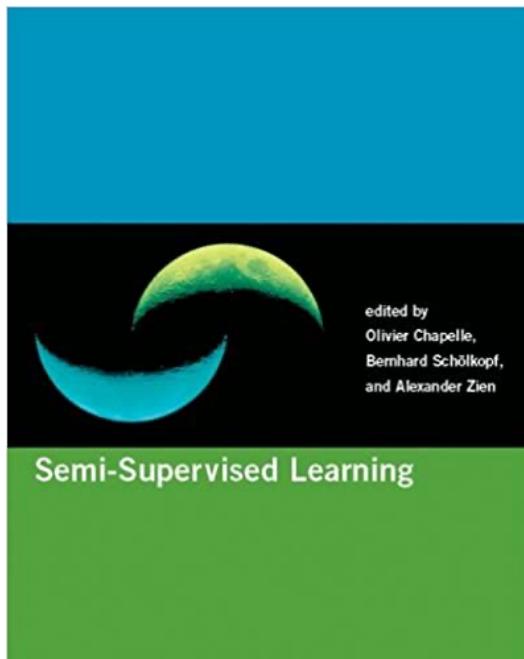
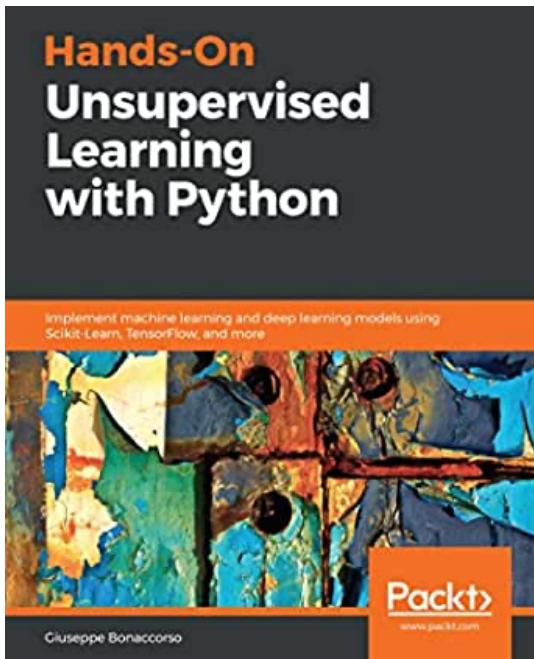
# Introducción

- Buena parte de estos progresos se han producido dentro del paradigma supervisado de aprendizaje: los modelos requieren etiquetas o anotaciones explícitas para guiar el entrenamiento.
- En la práctica obtener estas anotaciones es caro, lento, puede generar problemas éticos, y a veces es simplemente imposible.



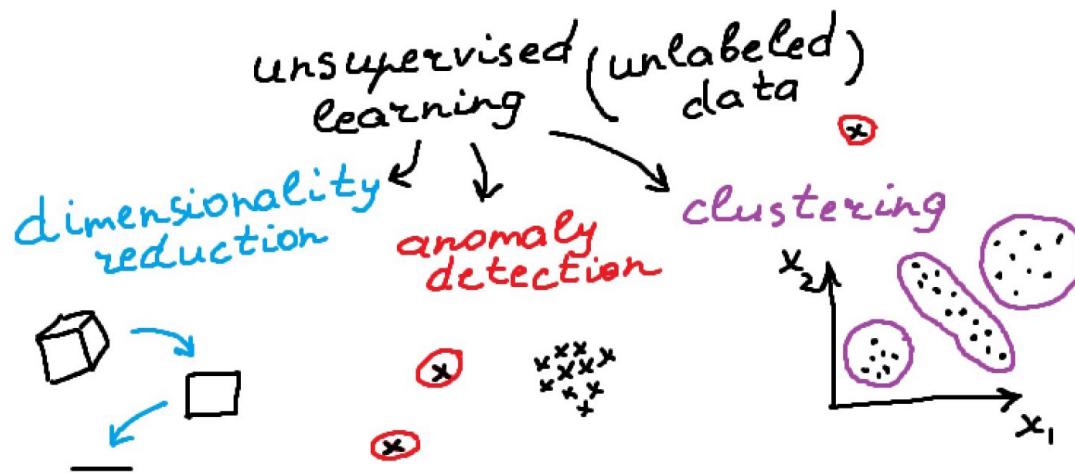
# Alternativas a la Supervisión Explícita

- Esto ha aumentado el interés por temas/áreas como el aprendizaje no-supervisado, semi-supervisado, auto-supervisado, débilmente supervisado, etc. así como de técnicas que permitan compensar la falta o escasez de datos etiquetados (pre-entrenamiento, transfer learning, etc).



# Aprendizaje No-Supervisado Clásico

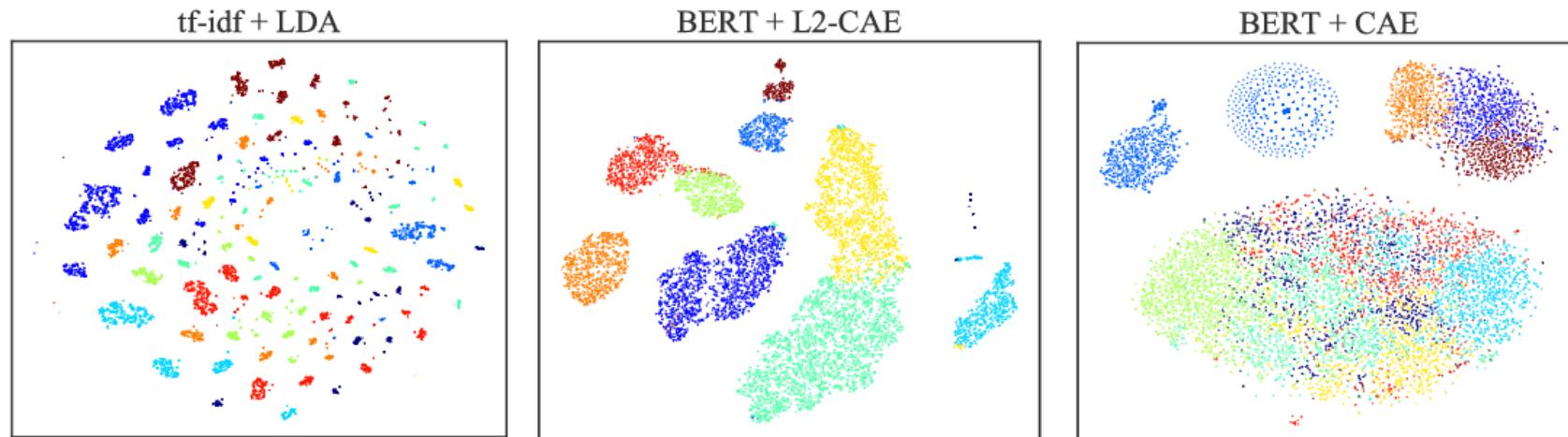
- Formas clásicas de aprendizaje no-supervisado: clustering, detección de anomalías, reducción de dimensionalidad, reglas de asociación.



- Estas técnicas están diseñadas para una tarea específica y no nos permitan suplir la falta de anotaciones en **cualquier** problema aplicado.
- Ejemplo: Necesitamos construir un clasificador de memes que decida automáticamente si se trata o no de un caso de **hate speech**.

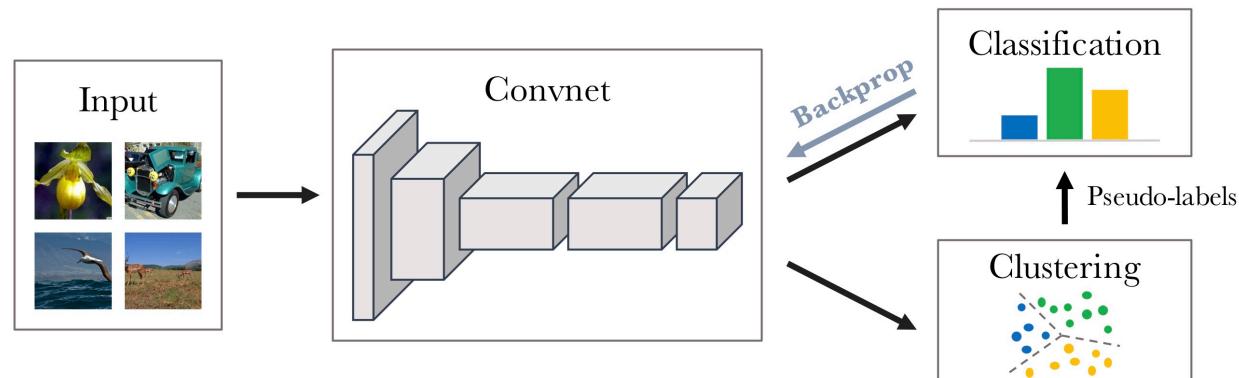
# Aprendizaje No-Supervisado Clásico

- Al igual que las técnicas clásicas de aprendizaje supervisado, las técnicas clásicas de aprendizaje no-supervisado son extremadamente sensibles a la **representación** que hagamos de los datos, es decir, al **espacio de características** sobre el que entrenemos el método. Abajo: clustering de texto usando diferentes representaciones.



# Aprendizaje No-Supervisado Profundo

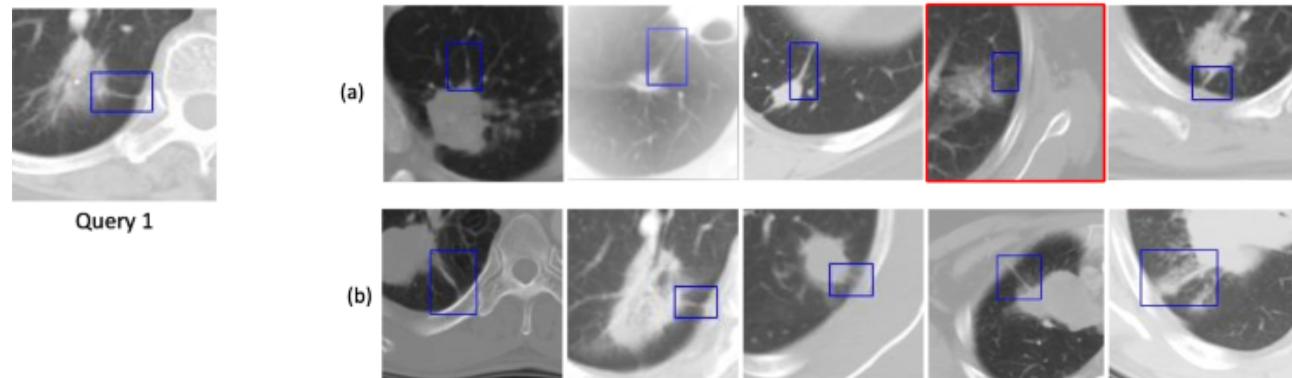
- Al igual que en el caso supervisado, el uso de arquitecturas profundas con múltiples niveles internos de representación (capas) permiten que el modelo refine o aprenda su propia representación para el problema.
- Por ejemplo, si nuestra tarea fuese descubrir clusters de imágenes, podríamos combinar un método de clásico de clustering con una CNN.



Caron, Mathilde, et al. "Deep clustering for unsupervised learning of visual features." Proceedings of the European conference on computer vision (ECCV). 2018.

# Aprendizaje de Representaciones

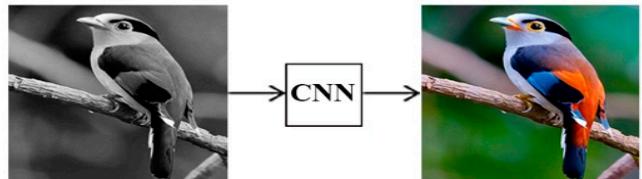
- Con frecuencia, el objetivo real al momento de entrenar un modelo como el anterior, no es resolver la tarea no-supervisada que guía el entrenamiento. El objetivo real es aprender una representación, es decir, obtener un extractor de características que luego puede ser **transferido** a otros problemas (en que **hay escasez o ausencia total** de etiquetas).



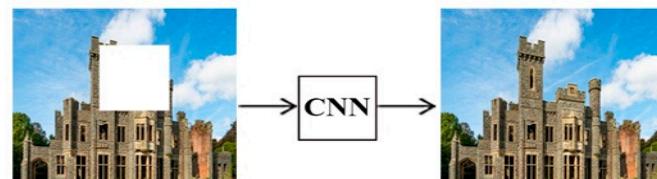
Molina, Gabriel, et al. "A New Content-Based Image Retrieval System for SARS-CoV-2 Computer-Aided Diagnosis." Int. Conf. on Medical Imaging and Computer-Aided Diagnosis. Springer, Singapore, 2021.

# Auto-Supervisión y Tareas de Pretexto

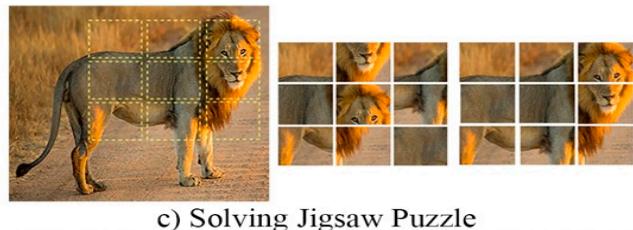
- Si el objetivo es la representación, porqué no emplear otras tareas (más allá de las clásicas tareas auto-supervisadas) para guiar el entrenamiento?



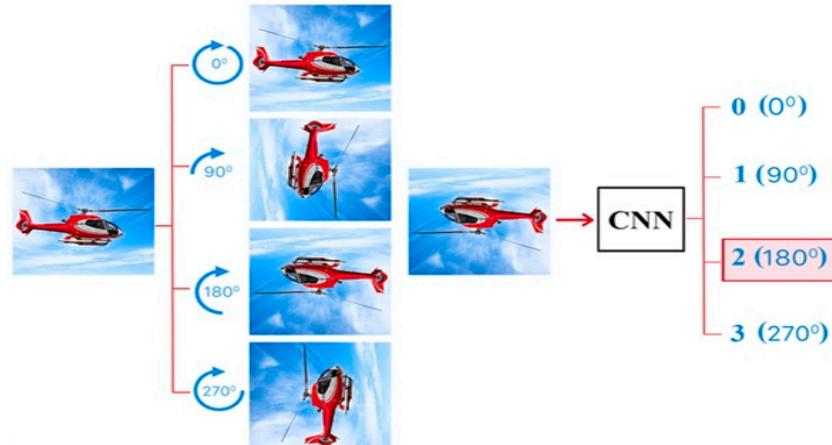
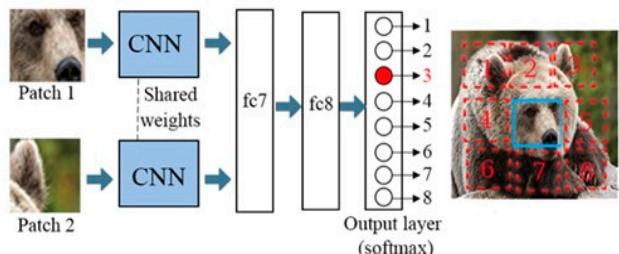
a) Colorizing an image



b) Inpainting



c) Solving Jigsaw Puzzle

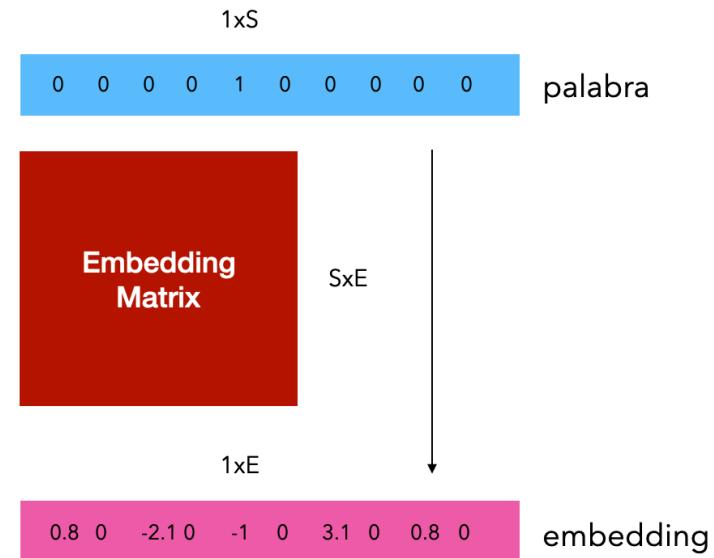
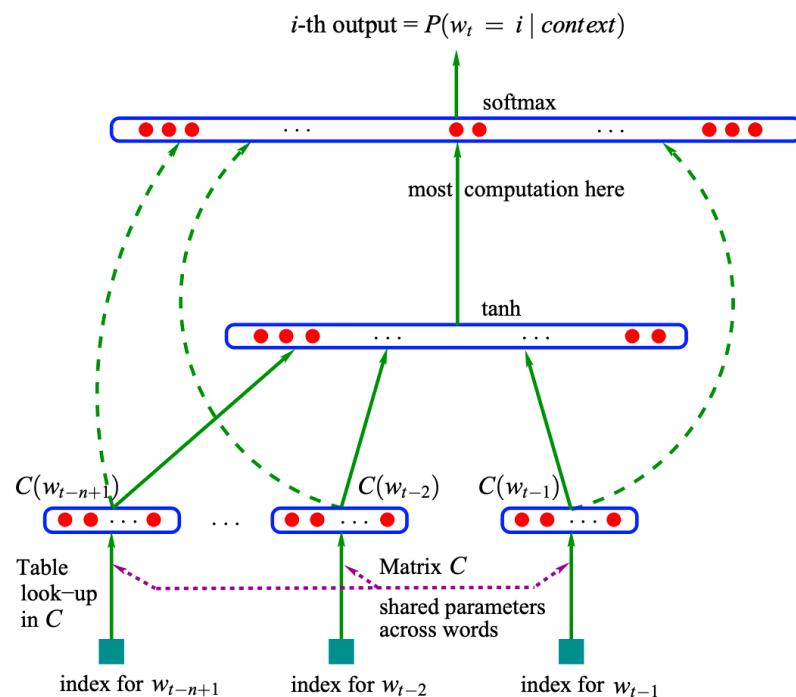


e) Estimating the rotation angle

Albelwi, Saleh. "Survey on Self-Supervised Learning: Auxiliary Pretext Tasks and Contrastive Learning Methods in Imaging." *Entropy* 24.4 (2022): 551.

# Auto-Supervisión en Lenguaje Natural

- Los modelos neuronales de lenguaje (natural) hoy en día en pleno apogeo se entrena en efecto de modo “auto-supervisado” usando tareas de pretexto.  
Abajo: **NNLM** model

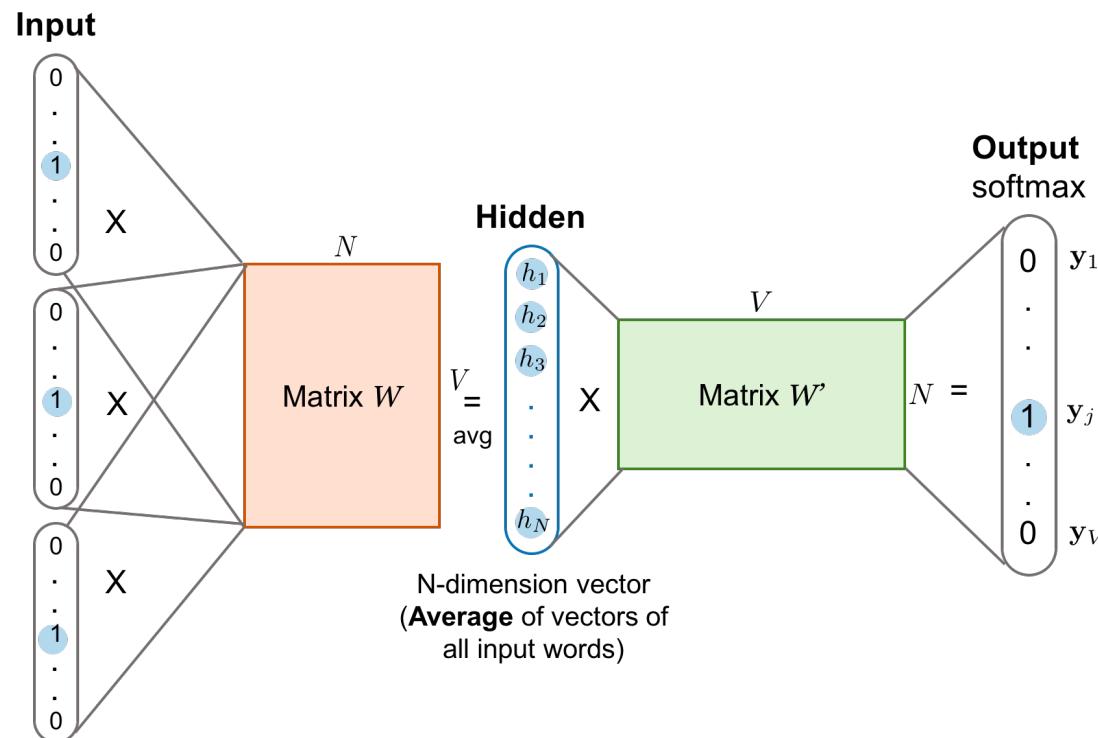


Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." *Advances in neural information processing systems* 13 (2000).



# Auto-Supervisión en Lenguaje Natural

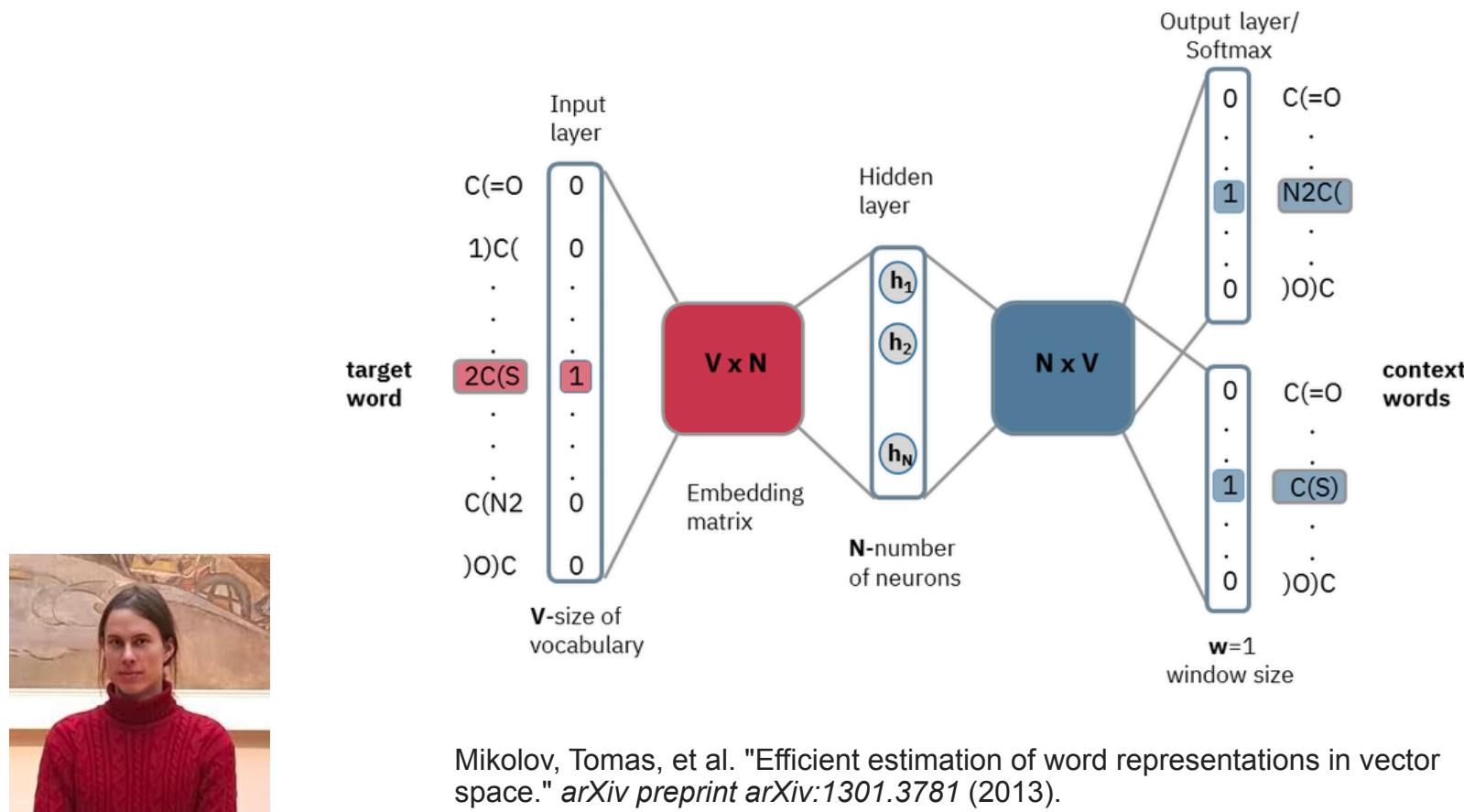
- Word2Vec entrenado en modo **CBOW**: dado el contexto de una palabra (enmascarada), el modelo debe predecir la palabra.



Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

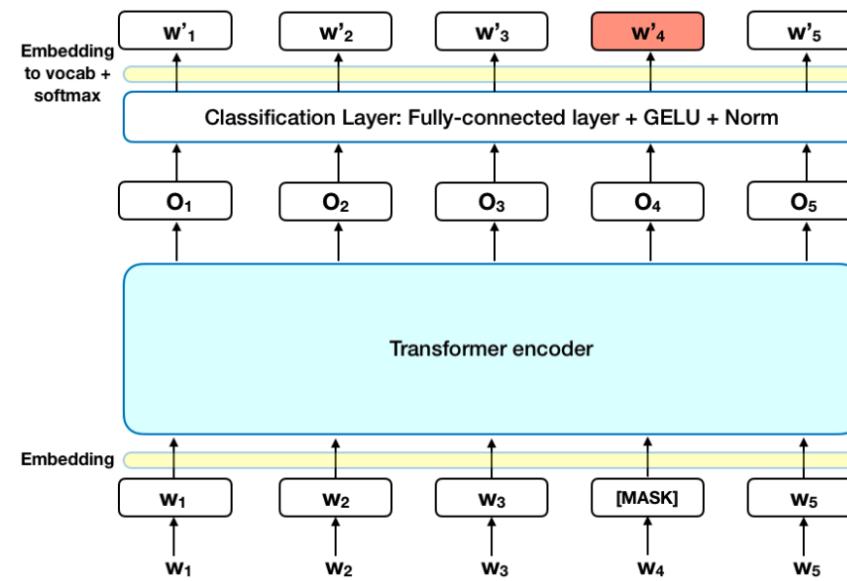
# Auto-Supervisión en Lenguaje Natural

- Word2Vec entrenado en modo **Skip-gram**: dada una palabra, el modelo debe predecir el contexto de la palabra.



# Auto-Supervisión en Lenguaje Natural

- **BERT:** dadas dos frases con palabras enmascaradas, el modelo debe predecir (1) las palabras enmascaradas y (2) con qué probabilidad la segunda frase sigue a la primera en el corpus. Se usa un transformer como arquitectura base.



Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

# Esto es Aprendizaje No-Supervisado?

- Notemos que las tareas que un modelo utiliza para “auto-supervisarse” dependen mucho del área de aplicación (tipo de dato) y de la creatividad del humano que la diseña (aún así mejor que diseñar los atributos directamente).
- Si bien estos métodos nos permiten escapar a la falta/escasez de etiquetas, los principios de aprendizaje utilizados son esencialmente supervisados: creamos una salida deseada (tarea de pretexto) que el modelo intenta aproximar explícitamente: un truco para *aprender sin supervisión con supervisión*.
- **¿Existe algún principio de propósito general que permita a un modelo aprender realmente sin supervisión (sin salida y)? ¿Qué significa aprender sin un objetivo o tarea (y) dada externamente?**
- Una de las respuestas más antiguas a estas preguntas la ofrecen los **Métodos Basados en Energía (EBM)**, cuyos exponentes más conocidos son las Redes de Hopfield y las Redes de Boltzmann.



# Redes de Hopfield

Proc. Natl. Acad. Sci. USA  
Vol. 79, pp. 2554–2558, April 1982  
Biophysics

## Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

J. J. HOPFIELD

Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125; and Bell Laboratories, Murray Hill, New Jersey 07974

Contributed by John J. Hopfield, January 15, 1982

**ABSTRACT** Computational properties of use to biological organisms or to the construction of computers can emerge as collective properties of systems having a large number of simple equivalent components (or neurons). The physical meaning of content-addressable memory is described by an appropriate phase space flow of the state of a system. A model of such a system is given, based on aspects of neurobiology but readily adapted to integrated circuits. The collective properties of this model produce a content-addressable memory which correctly yields an entire memory from any subpart of sufficient size. The algorithm for the time evolution of the state of the system is based on asynchronous parallel processing. Additional emergent collective properties include some capacity for generalization, familiarity recognition, categorization, error correction, and time sequence retention. The collective properties are only weakly sensitive to details of the modeling or the failure of individual devices.

calized content-addressable memory or categorizer using extensive asynchronous parallel processing.

### The general content-addressable memory of a physical system

Suppose that an item stored in memory is "H. A. Kramers & G. H. Wannier *Phys. Rev.* **60**, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "H. A. Kramers & G. H. Wannier, (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Vannier, (1941)". In computers, only relatively simple forms of content-addressable memory have been made in hardware (10, 11). Sophisticated ideas like error correction in accessing information are usually introduced as software (10).

There are classes of physical systems whose spontaneous be-

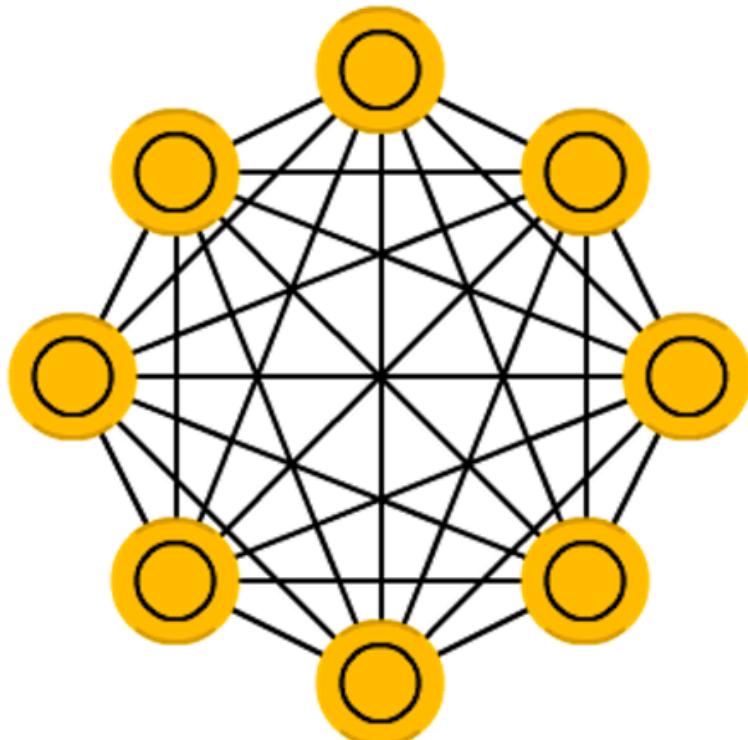


Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the national academy of sciences* 79.8 (1982): 2554-2558.



# Redes de Hopfield

- En una *Red de Hopfield* tradicional tenemos **d neuronas binarias totalmente conectadas** entre sí mediante **arcos no-dirigidos** (reversibles).



- Denotaremos por  $x_i$  el estado de la neurona  $i$  y por  $W_{ij}$  el peso de conexión entre la neurona  $i$  y la neurona  $j$  (de modo que  $W$  es la matriz de pesos).
- Las conexiones son simétricas y no se permiten "loops" entre neuronas, es decir  $W_{ij} = W_{ji}$  y  $W_{ii} = 0 \forall i$ .
- Por comodidad, representaremos la activación de una neurona como  $+1$  y su no activación como  $-1$ .

# Redes de Hopfield

- Como  $x_i \in \pm 1$ , la ecuación de actualización de estado para cada neurona luce como sigue:

$$x_i = \text{sign}\left(\sum_j W_{ij}x_j + b_j\right) = \begin{cases} +1 & \sum_j W_{ij}x_j + b_j \geq 0 \\ -1 & \sum_j W_{ij}x_j + b_j < 0 \end{cases}$$

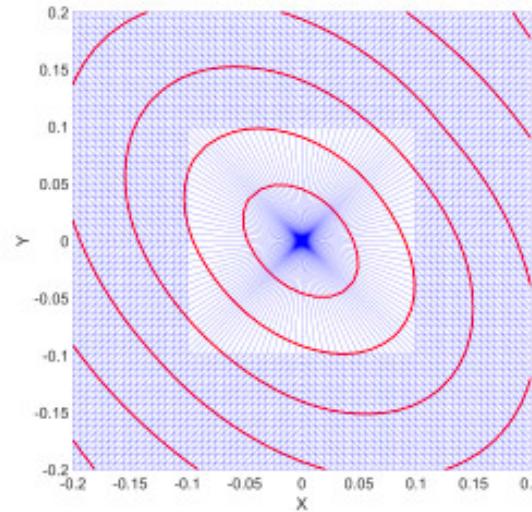
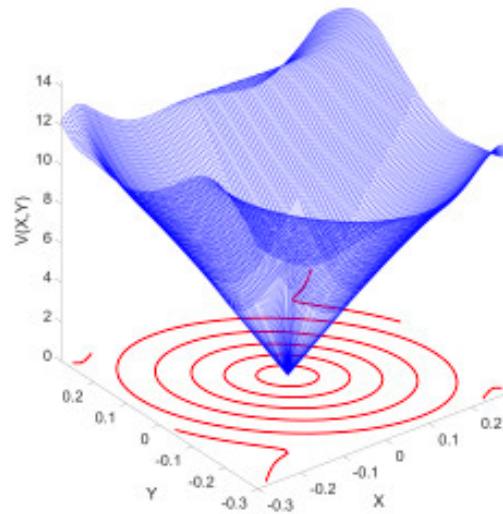
- Los biases (umbrales de excitación) se pueden omitir de la ecuación agregando 1 neurona que están siempre activa.
- Dado un estado de toda la red en el tiempo  $t$ , digamos  $x(t)$ , la ecuación anterior determina un nuevo estado en el tiempo  $t+1$ . En otras palabras una red de Hopfield, es una **red recurrente**, con una cierta **dinámica**.
- Hopfield descubrió que esta dinámica admite una **función de Lyapunov E y por lo tanto tiende a un cierto equilibrio a medida que  $t \rightarrow \infty$** .

# Redes de Hopfield

- Esta función  $E$  se conoce hoy en día como la **Energía** del sistema

$$E = -\frac{1}{2}\mathbf{x}^T \mathbf{W} \mathbf{x} - \mathbf{x}^T \mathbf{b} = -\frac{1}{2} \sum_{i,j} W_{ij} x_i x_j - b_i x_i$$

- Actualizando el sistema desde una configuración inicial, una neurona a la vez, la energía sólo puede disminuir. Como se trata de una función acotada por debajo, el sistema converge tarde o temprano **a un punto de equilibrio**.



# Redes de Hopfield

- Demostrar lo anterior es muy simple examinando lo que sucede cuando actualizamos 1 neurona (usualmente se elige de modo aleatorio)

$$E = -\frac{1}{2}\mathbf{x}^T \mathbf{W} \mathbf{x} - \mathbf{x}^T \mathbf{b} = -\frac{1}{2} \sum_{i,j} W_{ij} x_i x_j - b_i x_i$$

$$\Rightarrow \Delta E(t) = -\Delta x_i(t) \left( \sum_j W_{ij} x_j(t-1) + b_j \right)$$

$$x_i(t) = \begin{cases} +1 & \sum_j W_{ij} x_j(t-1) + b_j \geq 0 \text{ Podía estar en } +1 \text{ o } -1 \text{ pero termina en } +1 \\ -1 & \sum_j W_{ij} x_j(t-1) + b_j < 0 \text{ Podía estar en } +1 \text{ o } -1 \text{ pero termina en } -1 \end{cases}$$

$$\Rightarrow \begin{cases} \Delta x_i(t) \geq 0 & \sum_j W_{ij} x_j(t-1) + b_j \geq 0 \\ \Delta x_i(t) \leq 0 & \sum_j W_{ij} x_j(t-1) + b_j < 0 \end{cases}$$

# Redes de Hopfield

- Tenemos entonces que

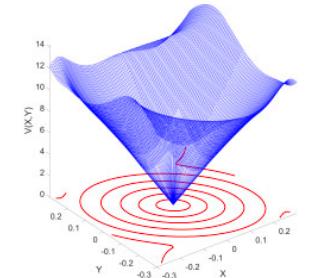
●  $\Delta E(t) = -\Delta x_i(t) \left( \sum_j W_{ij} x_j(t-1) + b_j \right)$

●  $\begin{cases} \Delta x_i(t) \geq 0 & \sum_j W_{ij} x_j(t-1) + b_j \geq 0 \\ \Delta x_i(t) \leq 0 & \sum_j W_{ij} x_j(t-1) + b_j < 0 \end{cases}$

$$\Rightarrow \Delta E(t) \geq 0$$

con  $\Delta E(t) = 0$  ssi  $\Delta x(t) = 0$

- Por lo tanto, si el sistema se itera lo suficiente, converge siempre a una configuración de **mínima energía**.

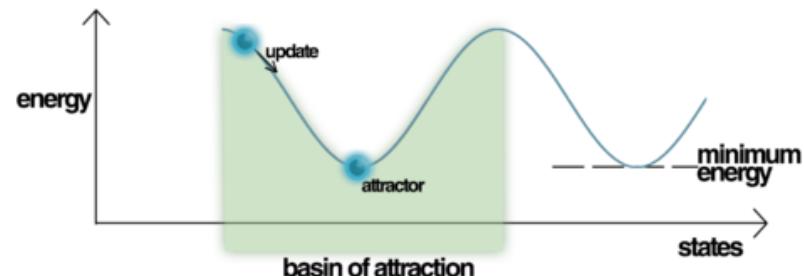
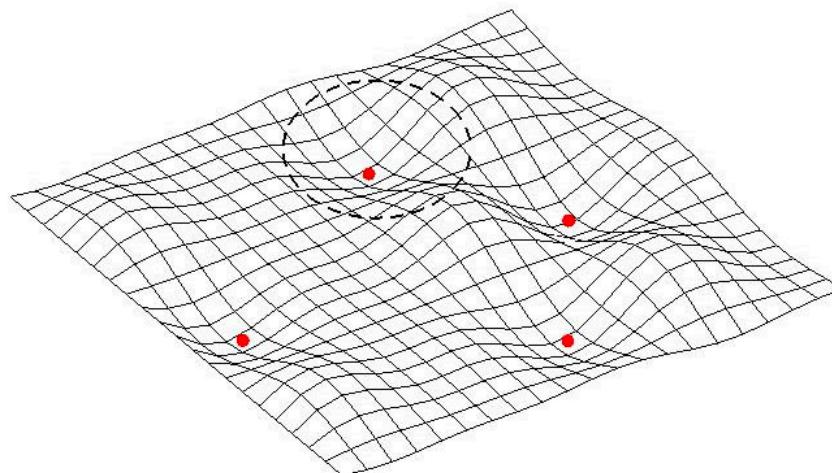


# Redes de Hopfield

- Cualquiera sea la configuración a la que converge el sistema, debe tratarse de un **punto fijo**, es decir, un estado que satisface

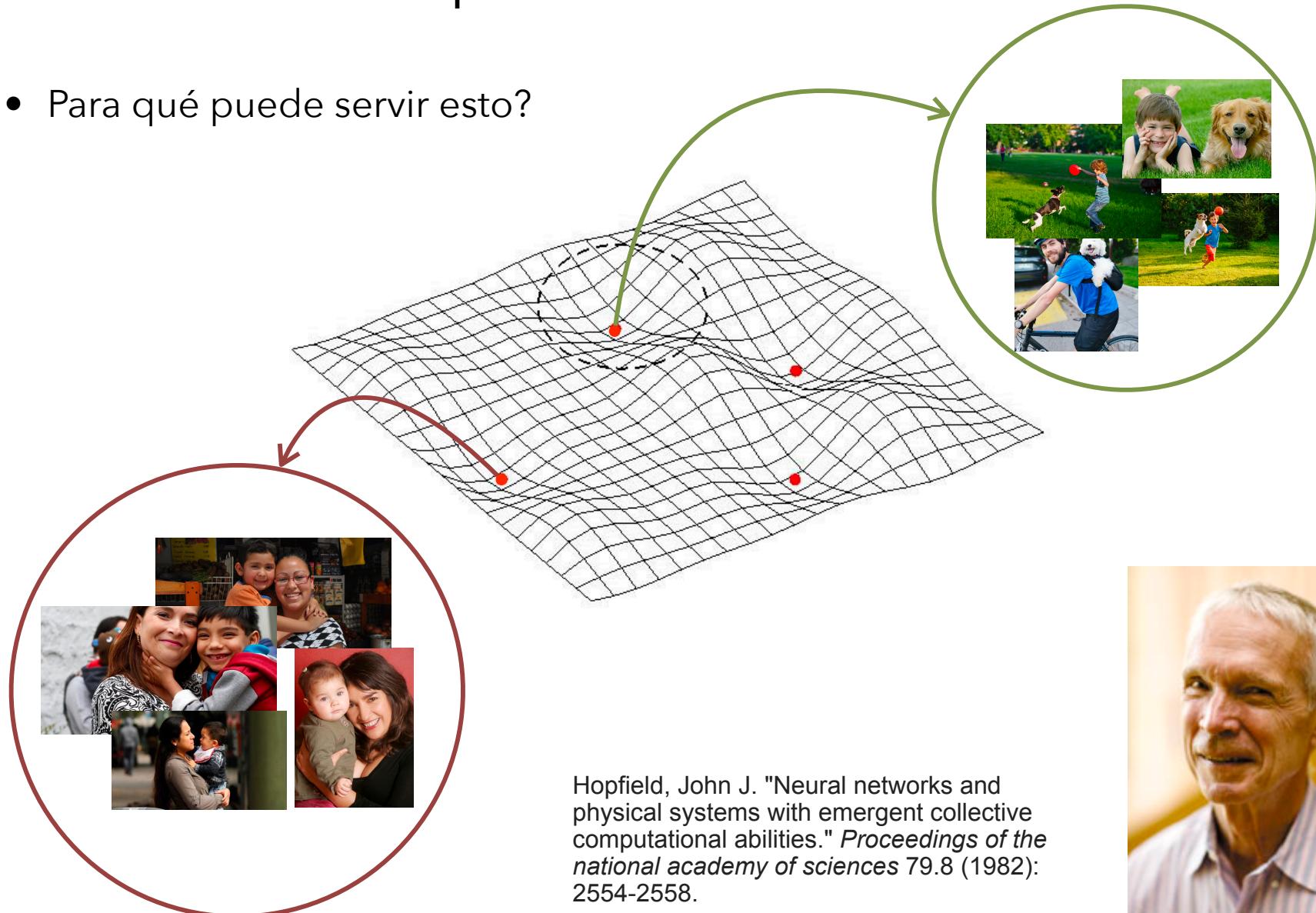
$$\mathbf{x}(t+1) = \mathbf{x}(t) = \text{sign}(\mathbf{W}\mathbf{x}(t) + \mathbf{b})$$

- Estos puntos fijos son de hecho **atractores** de la función de energía. Si se comienza a iterar el sistema lo suficientemente cerca de un punto fijo  $\mathbf{x}^*$  el sistema convergerá a ese punto de equilibrio.

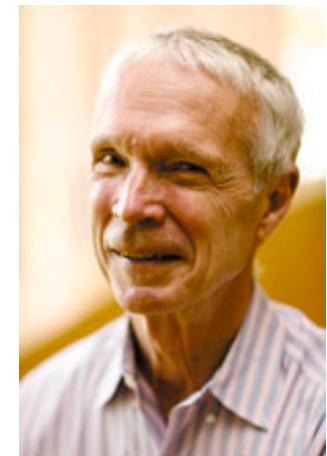


# Redes de Hopfield

- Para qué puede servir esto?

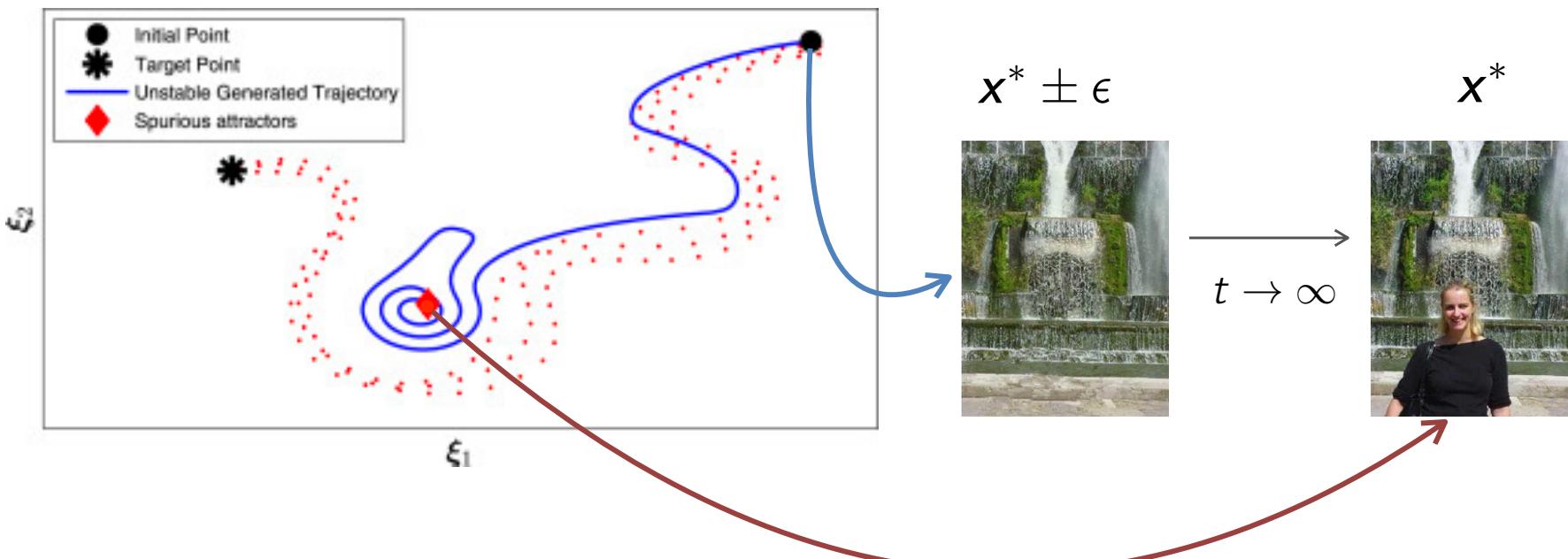


Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the national academy of sciences* 79.8 (1982): 2554-2558.



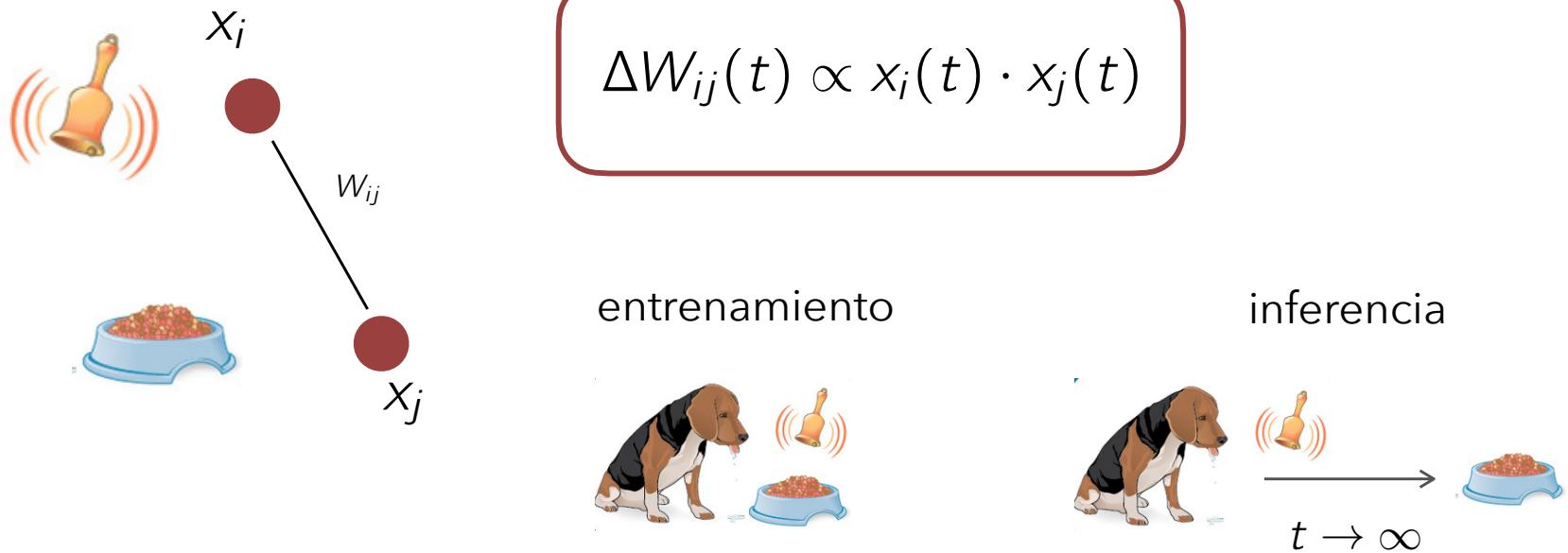
# Redes de Hopfield

- Podemos usar esta red para crear memorias asociativas, es decir **almacenar y recuperar recuerdos** de eventos que hemos observado en el ambiente  $\{x^{(\ell)}\}_{\ell=1}^n$ .
- Esta memoria será robusta en el sentido de que un evento similar pero posiblemente incompleto o distorsionado respecto de un evento del pasado gatillará una dinámica que convergerá a evocar el recuerdo completo.



# Redes de Hopfield

- ¿Como entrenamos esta red para recordar? La idea básica la propuso Hebb hacia 1949 (primer principio documentado de aprendizaje no supervisado).
- Principio básico: "Si, al presentarse un estímulo, dos neuronas se activan juntas, la conexión entre ellas se aumenta".
- Principio extendido: "Si, al presentarse un estímulo, dos neuronas se activan igual, la conexión entre ellas se aumenta. De lo contrario disminuye."



# Redes de Hopfield

- Si iteramos esta regla presentando un ejemplo/dato a la vez del conjunto de datos disponibles  $\{x^{(\ell)}\}_{\ell=1}^n$  obtendremos los "pesos de Hopfield" (corregimos por  $I$  para que los loops efectivamente no existan)

$$\Delta W(t) = x \cdot x^T - I$$



$$W(n) = \sum_{\ell} x^{(\ell)} \cdot x^{(\ell)T} - nl$$

$$b(n) = \sum_{\ell} x^{(\ell)}$$

- ¿Qué sucede si en el futuro presentamos a la red uno estos datos?
- Si asumimos que los datos (que queremos recordar) están suficientemente separados entre sí ...

# Redes de Hopfield

- Si asumimos que los datos son ortogonales ..

$$\begin{aligned}\mathbf{W}x^{(\ell*)} + \mathbf{b} &= \sum_{\ell} x^{(\ell)} \cdot x^{(\ell)}^T x^{(\ell*)} - nx^{(\ell*)} + \sum_{\ell} x^{(\ell)} \\ &= \|x^{(\ell*)}\|^2 \cdot x^{(\ell*)} - nx^{(\ell*)} + \sum_{\ell} x^{(\ell)} \\ &= d \cdot x^{(\ell*)} - nx^{(\ell*)} + \sum_{\ell} x^{(\ell)} \\ &= (d - (n - 1)) \cdot x^{(\ell*)} + \sum_{\ell \neq \ell*} x^{(\ell)}\end{aligned}$$

$$\begin{aligned}x_i^{(\ell*)} = +1 \Rightarrow (\mathbf{W}x^{(\ell*)} + \mathbf{b})_i &\geq d - 2(n - 1) \Rightarrow \text{sign}(\mathbf{W}x_i^{(\ell*)} + \mathbf{b}_i) = +1 \\ x_i^{(\ell*)} = -1 \Rightarrow (\mathbf{W}x^{(\ell*)} + \mathbf{b})_i &\leq -d + 2(n - 1) \Rightarrow \text{sign}(\mathbf{W}x_i^{(\ell*)} + \mathbf{b}_i) = -1 \\ &d > 2n\end{aligned}$$



# Redes de Hopfield

- Si asumimos que los datos son ortogonales y la red tiene capacidad suficiente capacidad ( $d > 2n$ ), cada dato de entrenamiento  $x^{(\ell)}$  se vuelve un **punto fijo** de la función de Energía.

$$x^{(\ell)} = \text{sign}(\mathbf{W}x^{(\ell)} + \mathbf{b})$$

$$x^{(\ell)}(t+1) = \text{sign}(\mathbf{W}x^{(\ell)}(t) + \mathbf{b}) = x^{(\ell)}(t)$$

- Notar que asumimos que la red tiene suficiente memoria ( $n < 0.5d$ ). En la práctica diferentes simulaciones demuestran que los datos se vuelven puntos fijos si  $n < 0.15d$ .
- El resultado clásico se suele obtener considerando datos elegidos aleatoriamente del espacio de Hamming  $\{\pm 1\}^d$ . Notar que en este caso,

$$\mathbb{E}(x_i^{(\ell)}) = 0, \mathbb{E}(x_i^{(\ell)}x_i^{(\ell')}) = 0, \mathbb{E}(x_i^{(\ell)}x_j^{(\ell)}) = 0 \quad \forall \ell \neq \ell', i \neq j$$



# Redes de Hopfield

- Tenemos entonces que

$$\begin{aligned} (\mathbf{W}x^{(\ell*)} + \mathbf{b})_i &= \sum_{j \neq i} W_{ij} x_j^{(\ell*)} + b_i \\ &= \sum_{j \neq i} \sum_{\ell} x_i^{(\ell)} x_j^{(\ell)} x_j^{(\ell*)} + \sum_{\ell} x_i^{(\ell)} \\ &= (d - 1)x_i^{(\ell*)} + \sum_{j \neq i} \sum_{\ell \neq \ell*} x_i^{(\ell)} x_j^{(\ell)} x_i^{(\ell*)} + \sum_{\ell} x_i^{(\ell)} \\ &= dx_i^{(\ell*)} + \sum_{j \neq i} \sum_{\ell \neq \ell*} x_i^{(\ell)} x_j^{(\ell)} x_i^{(\ell*)} + \sum_{\ell \neq \ell*} x_i^{(\ell)} \end{aligned}$$

$$\Rightarrow \mathbb{E}(\mathbf{W}x^{(\ell*)} + \mathbf{b} \mid x^{(\ell*)})_i = dx_i^{(\ell*)} + \sum_{j \neq i} \sum_{\ell \neq \ell*} \mathbb{E}(x_i^{(\ell)} x_j^{(\ell)}) x_i^{(\ell*)} + \sum_{\ell \neq \ell*} \mathbb{E}(x_i^{(\ell)})$$

$$\boxed{\mathbb{E}(\mathbf{W}x^{(\ell*)} + \mathbf{b} \mid x^{(\ell*)})_i = dx_i^{(\ell*)}}$$



# Redes de Hopfield

- Tenemos entonces que

$$\mathbb{E}(\mathbf{W}x^{(\ell*)} + \mathbf{b} | x^{(\ell*)}) = dx^{(\ell*)}$$

- En valor esperado ...

$$\text{sign}(\mathbf{W}x^{(\ell*)} + \mathbf{b}) = x^{(\ell*)}$$

**Es decir, los datos son los puntos fijos de la función de Energía.**

- A partir de este tipo de análisis “probabilísticos”, McEliece concluye que el número máximo de memorias que una red de Hopfield puede recordar es  $\mathcal{O}(n/4\log(n))$ . Si queremos que la red recuerde la mayoría pero no necesariamente todos los datos, este número sube al doble.



McEliece, R. et al. "The capacity of the Hopfield associative memory." *IEEE transactions on Information Theory* 33.4 (1987): 461-482.

# Hopfield is All You Need?

- Recientemente (2021) se ha propuesto una extensión continua del modelo de Hopfield con una regla de actualización equivalente al modelo atencional empleado por el Transformer. Esta regla garantiza la convergencia a puntos estacionarios y tiene una capacidad de memoria exponencialmente grande (en la dimensionalidad del espacio asociativo). Las “capas de Hopfield” propuestas permitieron mejorar el estado del arte en 4 problemas diferentes.



2021

## HOPFIELD NETWORKS IS ALL YOU NEED

Hubert Ramsauer\* Bernhard Schäfl\* Johannes Lehner\* Philipp Seidl\*  
Michael Widrich\* Thomas Adler\* Lukas Gruber\* Markus Holzleitner\*  
Milena Pavlovic<sup>†,§</sup> Geir Kjetil Sandve<sup>§</sup> Victor Greiff<sup>‡</sup> David Kreil<sup>†</sup>  
Michael Kopp<sup>†</sup> Günter Klambauer\* Johannes Brandstetter\* Sepp Hochreiter\*,†

\*ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning,  
Johannes Kepler University Linz, Austria

<sup>†</sup>Institute of Advanced Research in Artificial Intelligence (IARAI)

<sup>‡</sup>Department of Immunology, University of Oslo, Norway

<sup>§</sup>Department of Informatics, University of Oslo, Norway

Ramsauer, Hubert, et al. "Hopfield networks is all you need." *arXiv preprint arXiv:2008.02217* (2020).

# Demo

PRO INF395-Memories(Hopfield).ipynb ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se guardaron todos los cambios

Comentar Compartir Editando

Índice + Código + Texto Reconectar Editando

Memories (Demo Redes de Hopfield)

Créditos

Modelo

Utils

MNIST

SkImage Data

Sección



↑ ↓ ↻ ↺ ↻ ↻ ↻ ↻

**i**

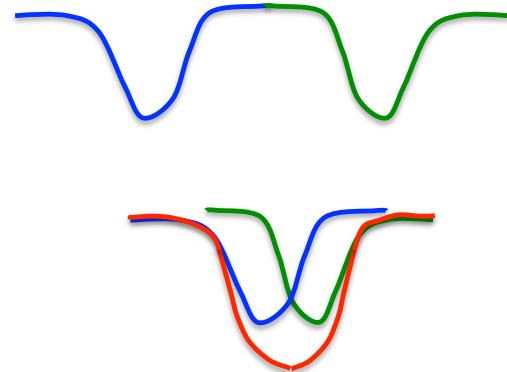
# Problemas de una Red Hopfield

- La red tiene una capacidad bastante limitada y hace un uso ineficiente de sus parámetros.
  - Con  $d$  unidades sólo puede almacenar del orden de  $0.15d$  memorias. Como cada memoria es de  $d$  bits, la red sólo recuerda del orden de  $0.15d^2$  bits.
  - Como la red tiene  $d$  unidades completamente conectadas usa  $d^2$  pesos. Como cada peso puede tomar un valor en  $[-n, n]$ , necesitamos  $d^2 \log(n)$  para almacenar la red.
- Cuando supera su capacidad, la red comienza a olvidar el pasado para poder almacenar la información nueva.
- La red tiende a recordar todo aquello que ve sin una preferencia explícita (sólo la frecuencia) por piezas más “relevantes” de información.
- La máquina es sensible a correlaciones entre datos.



# Problemas de una Red Hopfield

- A causa de lo último, el entrenamiento puede crear memorias falsas (fake memories o spurious memories): cosas que el modelo nunca vio pero que recuerda ...



**Datos:** (Sofía, Sweater Negro) ●  
(Ramón, Sweater Rojo)  
(Pedro, Sweater Azul)  
(Sofía, Sweater Blanco) ●

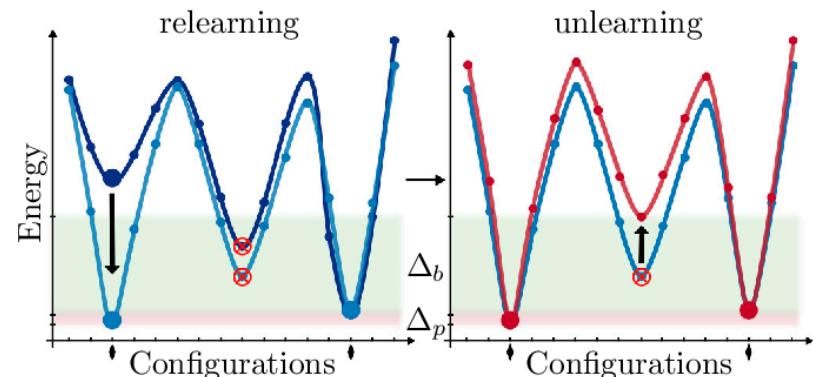
**Fake:** (Sofía, Sweater Gris) ●

# Problemas de una Red Hopfield

- Se ha propuesto (Hopfield et al.) resolver este problema induciendo una fase de “sueño” en la red que le permita des-aprender.
- Paso 1: Elegir una configuración aleatoria
- Paso 2: Iterar la red hasta alcanzar una memoria
- Paso 3: Des-aprendizaje parcial

$$\Delta W_{ij} = -\eta x_i x_j$$

- Paso 4: Terminar o volver a Paso 1.

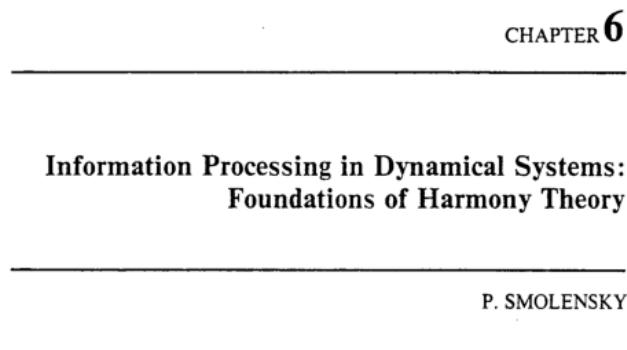


Hopfield, John J., David I. Feinstein, and Richard G. Palmer. "Unlearning'has a stabilizing effect in collective memories." *Nature* 304.5922 (1983): 158-159.



# Restricted Boltzmann Machines

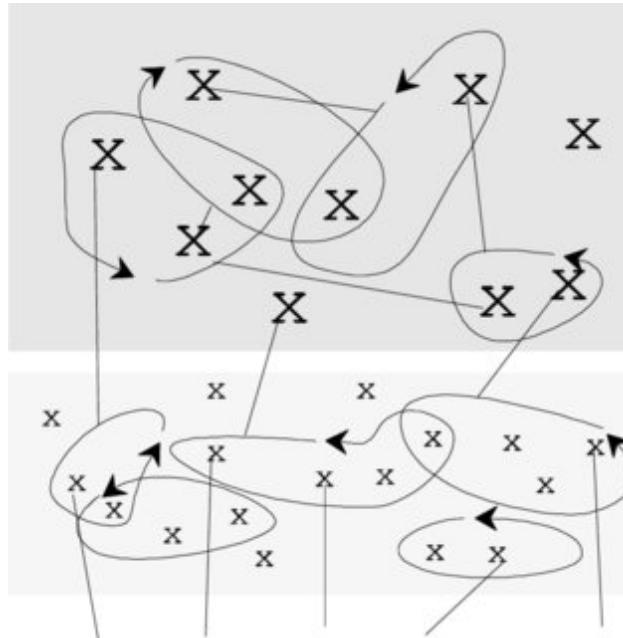
- A.K.A. "El Harmonium", hoy RBMs.



## INTRODUCTION

### The Theory of Information Processing

At this early stage in the development of cognitive science, methodological issues are both open and central. There may have been times when developments in neuroscience, artificial intelligence, or cognitive psychology seduced researchers into believing that their discipline was on the verge of discovering the secret of intelligence. But a humbling history of hopes disappointed has produced the realization that understanding the mind will challenge the power of all these methodologies combined.



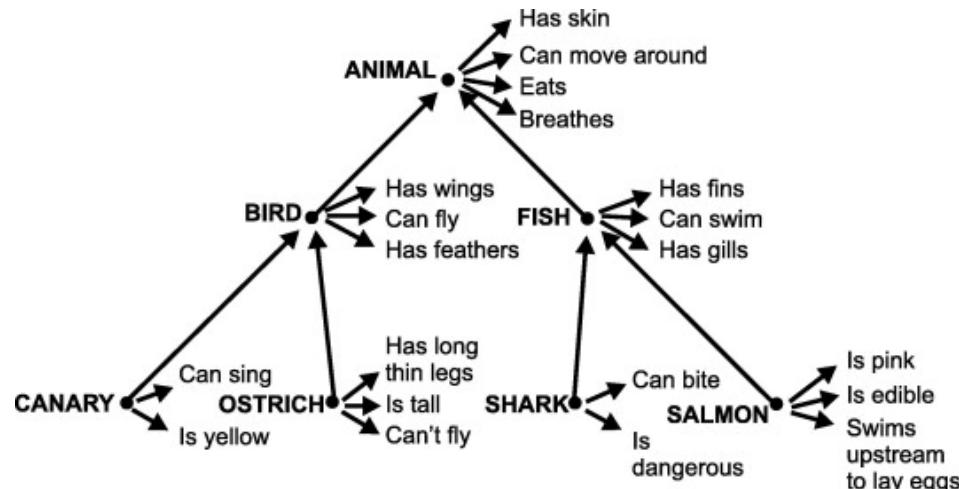
Smolensky, Paul (1986). "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory" (PDF). In Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press. pp. 194–281.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural Computation 18.7 (2006): 1527-1554.



# Restricted Boltzmann Machines

- El modelo aparece en el contexto de la histórica discusión sobre **IA simbólica** y **IA sub-simbólica**: ¿Podemos crear una IA que “descubra” automáticamente los aspectos/factores que caracterizan un conjunto de observaciones?



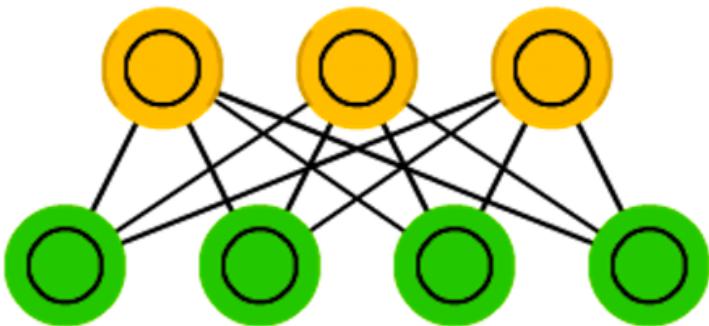
Smolensky, Paul (1986). "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory" (PDF). In Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press. pp. 194–281.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural Computation 18.7 (2006): 1527-1554.



# Restricted Boltzmann Machines

- Agregar al modelo **una capa oculta  $h$**  capaz de acomodar representaciones comprimidas de los datos que correspondan a **trazas o fragmentos compartidos de experiencias perceptivas.**



- Se mantiene la idea de usar **conexiones no dirigidas (reversibles)** entre las dos capas.
- Se restringen la conectividad de manera que sólo neuronas de capas diferentes están conectadas entre sí.

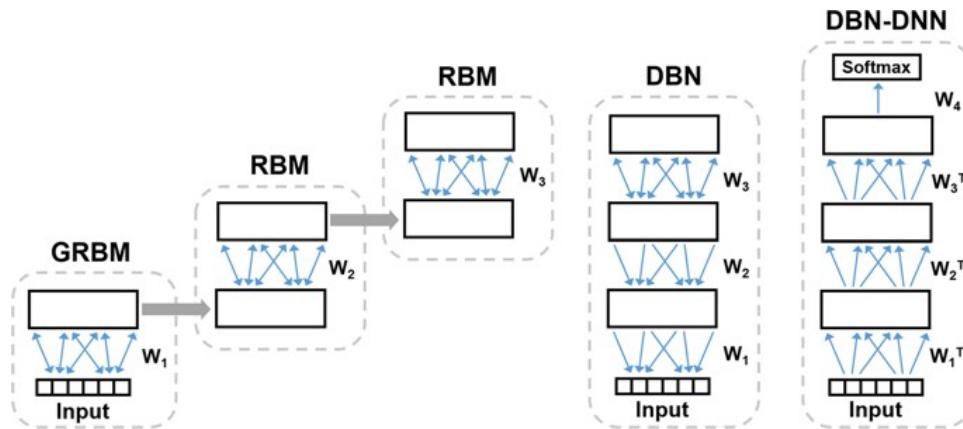
Smolensky, Paul (1986). "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory" (PDF). In Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press. pp. 194–281.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural Computation 18.7 (2006): 1527-1554.



# Restricted Boltzmann Machines

- Las redes entrenadas por Hinton tenían 2 a 3 capas ocultas y permitieron superar el estado del arte en algunas tareas clásicas como MNIST. La RBM se usó para pre-entrenar sin etiquetas esas capas ocultas.



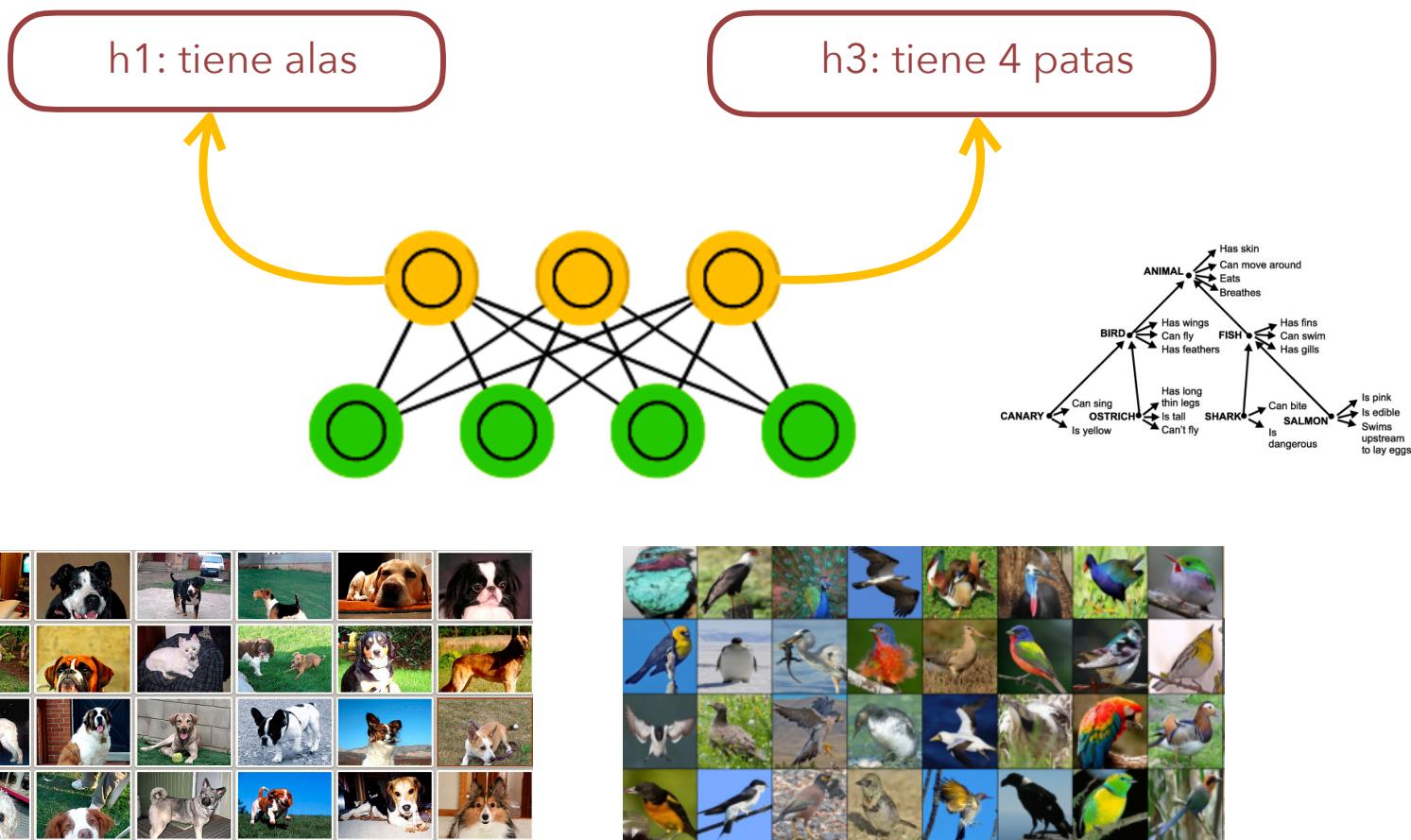
Version of MNIST Task	Learning Algorithm	Test Error %
Permutation invariant	Our generative model: 784 → 500 → 500 ↔ 2000 ↔ 10	1.25
Permutation invariant	Support vector machine: degree 9 polynomial kernel	1.4
Permutation invariant	Backprop: 784 → 500 → 300 → 10 cross-entropy and weight-decay	1.51
Permutation invariant	Backprop: 784 → 800 → 10 cross-entropy and early stopping	1.53
Permutation invariant	Backprop: 784 → 500 → 150 → 10 squared error and on-line updates	2.95



Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural Computation* 18.7 (2006): 1527-1554.

# Restricted Boltzmann Machines

- Estos fragmentos compartidos de memoria se puede interpretar como **factores** que la red crea y usa para explicar el mundo que observa.



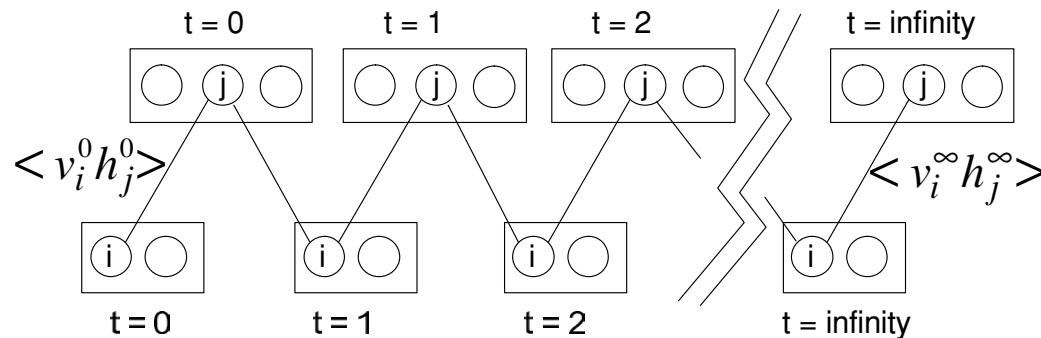
# Restricted Boltzmann Machines

- Estos fragmentos compartidos de memoria se puede interpretar como **factores** que la red crea y usa para explicar el mundo que observa.



# Restricted Boltzmann Machines

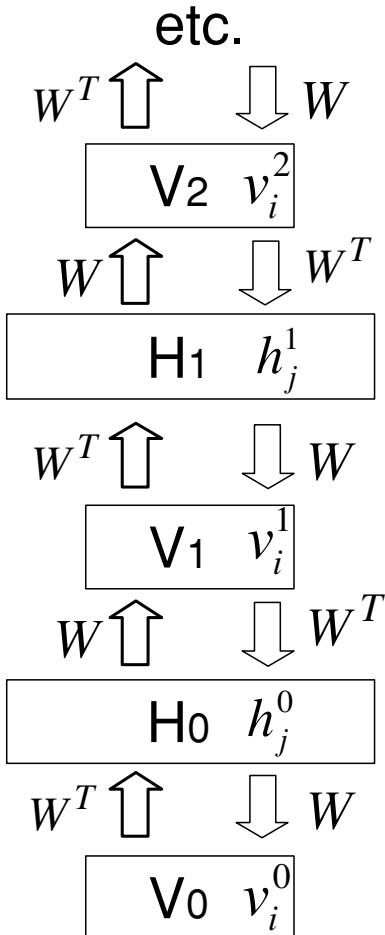
- Al igual que en el modelo de Hopfield, las conexiones reversibles permiten iterar “infinitamente” el modelo desde una configuración de partida (**estímulo**) hasta una configuración de equilibrio (**recuerdo** evocado por el modelo).
- La recuperación de esas memorias pasadas se realizará sin embargo permitiéndole a la red evocar sus propias **explicaciones** de los datos.



Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural Computation* 18.7 (2006): 1527-1554.



# Restricted Boltzmann Machines



- Desenrollando esta dinámica en el tiempo, es fácil ver que pese a su simplicidad, la red es **infinitamente profunda en el tiempo**.
- Los pesos de la red desenrollada son compartidos.

LETTER ————— Communicated by Yann Le Cun

## A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton  
[hinton@cs.toronto.edu](mailto:hinton@cs.toronto.edu)

Simon Osindero  
[osindero@cs.toronto.edu](mailto:osindero@cs.toronto.edu)  
Department of Computer Science, University of Toronto, Toronto, Canada M5S 3G4

Yee-Whye Teh  
[telyw@comp.nus.edu.sg](mailto:telyw@comp.nus.edu.sg)  
Department of Computer Science, National University of Singapore,  
Singapore 117543

We show how to use “complementary priors” to eliminate the explaining-away effects that make inference difficult in densely connected belief nets that have many hidden layers. Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory. The fast, greedy algorithm is used to initialize a slower learning procedure that fine-tunes the weights using a contrastive version of the wake-sleep algorithm. After fine-tuning, a network with three hidden layers forms a very good generative model of the joint distribution of handwritten digit images and their labels. This generative model gives better digit classification than the best discriminative learning al-

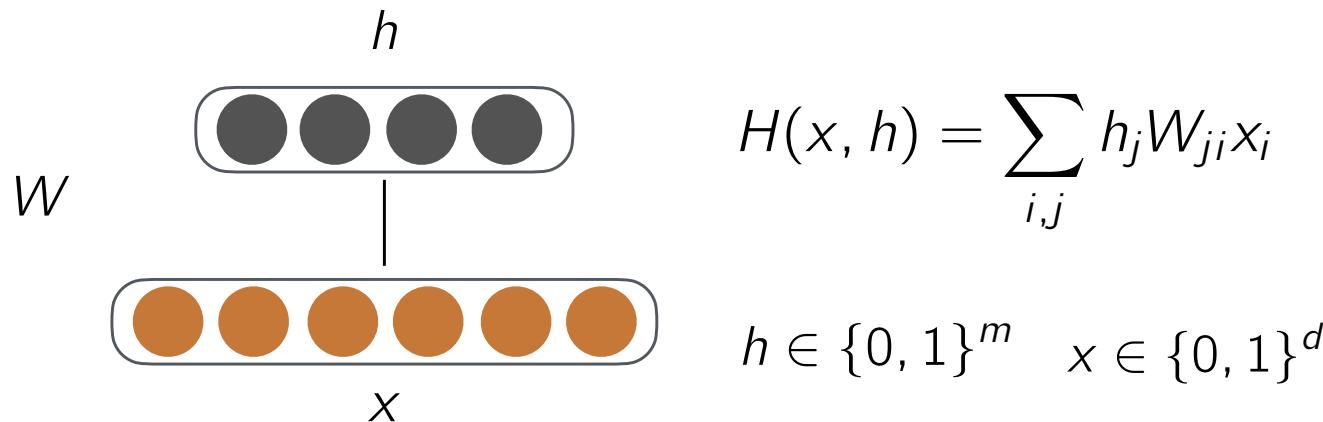


Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural Computation* 18.7 (2006): 1527-1554.



# Modelo

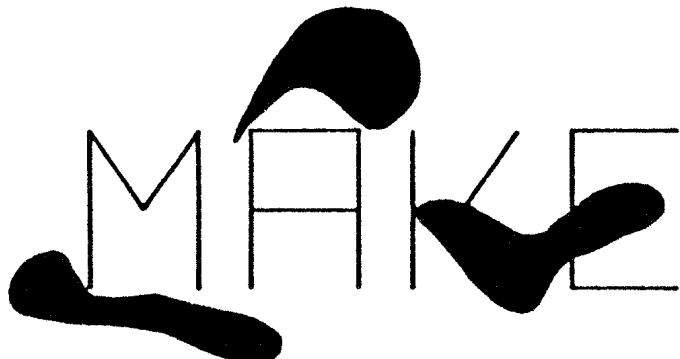
- Si  $x$  denota un posible estado de la primera capa y  $h$  denota un posible estado de la segunda capa, la máquina funciona utilizando una **función de armonía**  $H(x, h)$  que establece el nivel de **consistencia** entre ambos estados.



- Si consideramos primero el caso binario, tenemos que cuando  $W_{ij} > 0$ , una activación de  $x_i$  se considera más consistente con una activación de  $h_j$ . Si  $W_{ij} < 0$ , una activación de  $x_i$  se considera más consistente con una inactivación de  $h_j$ .

# Modelo

- Si la máquina recibiera una representación incompleta de  $x$ , la función  $H$  podría permitirle encontrar una configuración de **máxima consistencia** entre  $x$  y  $h$  que **complete** la percepción original.



$$H(x, h) = \sum_{i,j} h_j W_{ji} x_i$$

Smolensky, Paul (1986). "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory" (PDF). In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press. pp. 194–281.

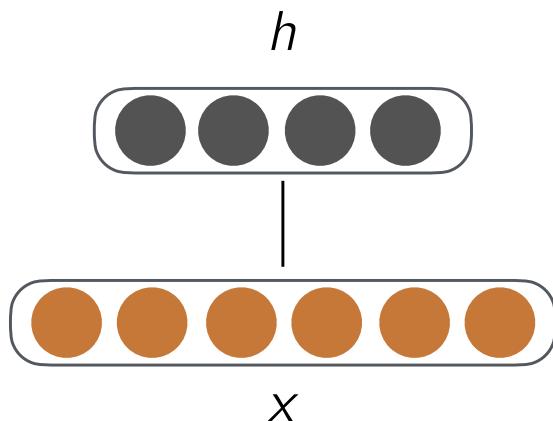


# Modelo

- La función de armonía puede interpretarse como el negativo de la energía del sistema de modo que una configuración de **máxima consistencia** corresponda a una configuración de **mínima energía**.

$$E(x, h) = -H(x, h)$$

- Es posible agregar términos lineales en  $x$  y en  $h$  a la función de armonía que representan la preferencia *a-priori* por observar esos factores de la representaciones latente y observable.

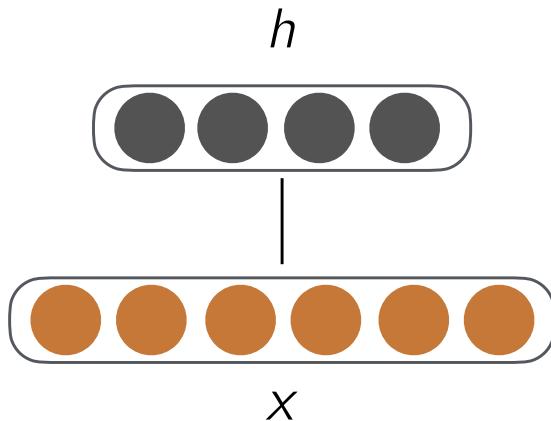


$$H(x, h) = \sum_{i,j} h_j W_{ji} x_i + \sum_j b_j h_j + \sum_i c_i x_i$$

$$h \in \{0, 1\}^m \quad x \in \{0, 1\}^d$$

# Modelo Probabilístico

- Matemáticamente, una RBM especifica un modelo de probabilidad para el ambiente. En el caso binario  $h \in \{0, 1\}^m$ ,  $x \in \{0, 1\}^d$  el modelo queda especificado mediante las siguientes ecuaciones:



$$p(x) = \sum_h p(x, h) \quad p(x, h) = \frac{\exp(-E(x, h))}{Z}$$

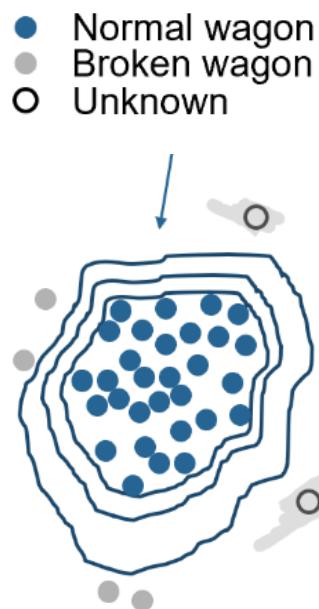
$$\begin{aligned} E(x, h) &= -H(x, h) = -\sum_{ij} h_j W_{ji} x_i - \sum_j b_j h_j - \sum_i c_i x_i \\ &= -(h^T W x + b^T h + c^T x) \end{aligned}$$

- Z se denomina la función de partición.

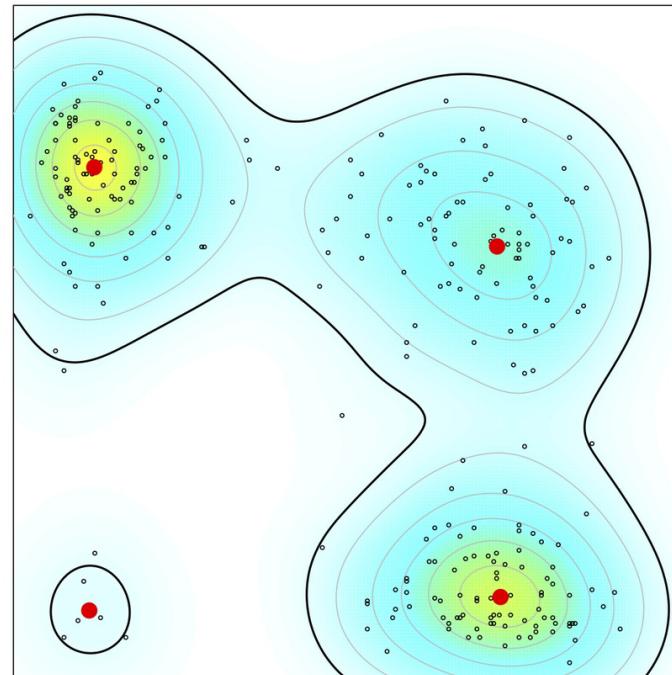
Alta armonía = Baja Energía = Alta Probabilidad

# Porqué un Modelo Probabilístico?

- Aprendiendo un modelo de  $p(x)$  podemos abordar muchas de las formas clásicas de aprendizaje no-supervisado.



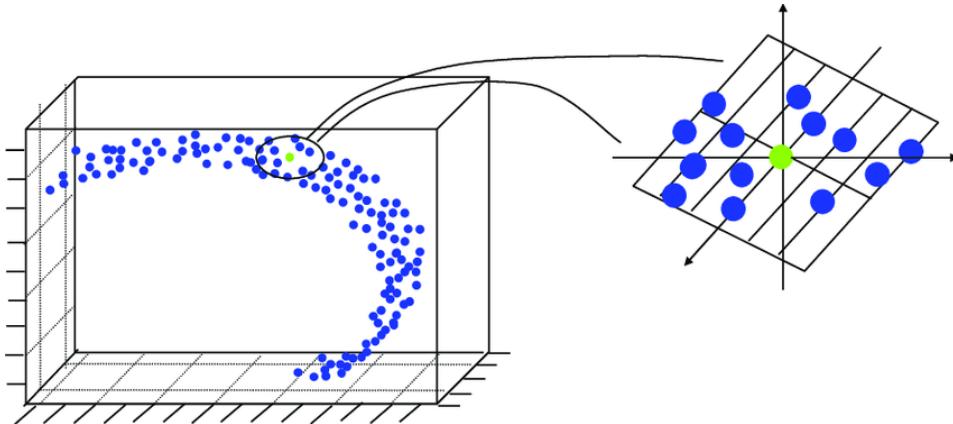
El problema de detectar anomalías puede verse como el de buscar outliers.



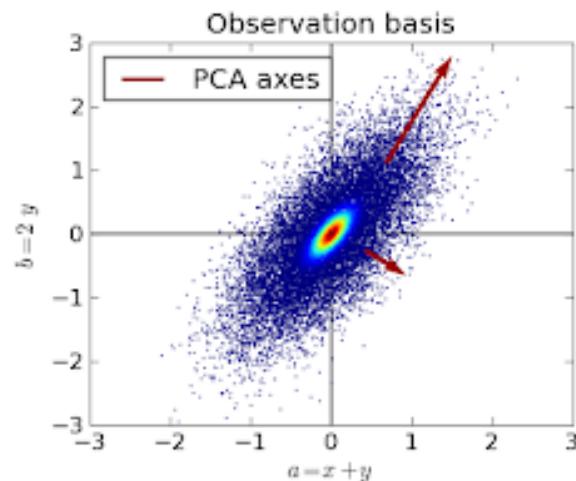
Clustering puede verse como la búsqueda de las modas de una distribución

# Porqué un Modelo Probabilístico?

- Aprendiendo un modelo de  $p(x)$  podemos abordar muchas de las formas clásicas de aprendizaje no-supervisado.



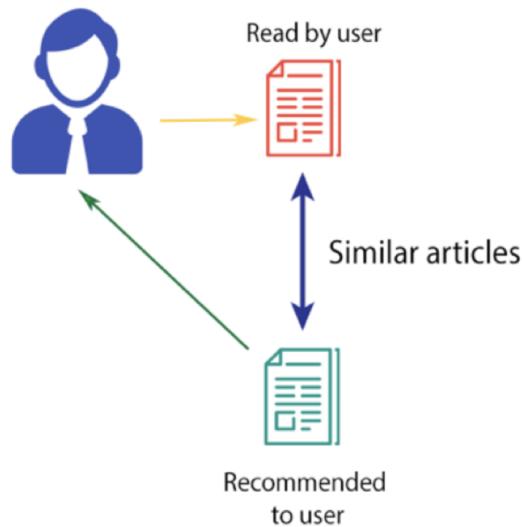
El problema de aprender una representación de baja dimensionalidad puede asociarse al problema de encontrar las direcciones en las que  $p(x)$  cambia más bruscamente.



PCA como la ortogonalización de la matriz de covarianzas de la data previamente centrada. Esencialmente un fit gaussiano.

# Porqué un Modelo Probabilístico?

- Aprendiendo un modelo de  $p(x)$  podemos abordar muchas de las formas clásicas de aprendizaje no-supervisado.



El problema de recomendar ítems no vistos puede verse como un problema de completación (de matrices en el caso de múltiples usuarios).



El problema de completar una representación puede verse como el de encontrar los valores más probables de ciertos atributos dados los demás.

# Porqué un Modelo Probabilístico?

- Probablemente, el aspecto más interesante de un modelo probabilístico sea su **capacidad generativa**. Si tenemos un modelo de  $p(x)$  podemos buscar un modo de muestra (simular) el proceso probabilístico para generar datos que se distribuyen como los datos reales pero no han sido observados (aún).

2006 (RBM)

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

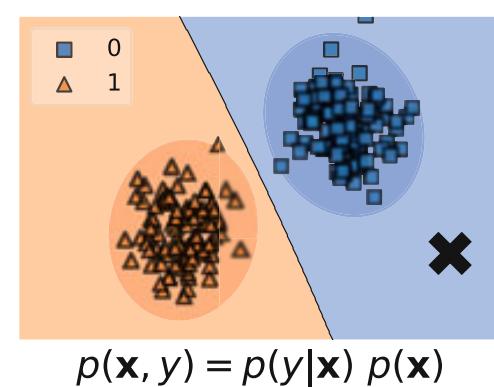
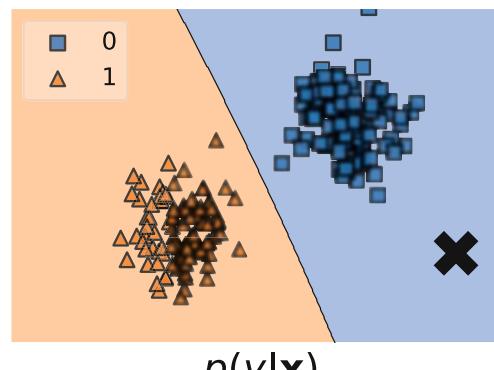
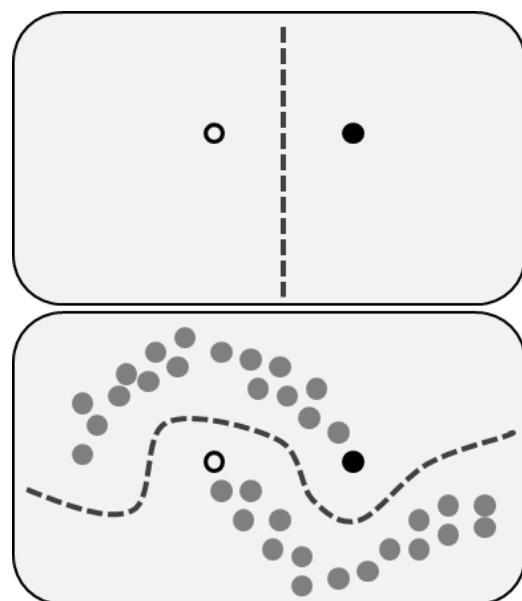
hoy (GAN)



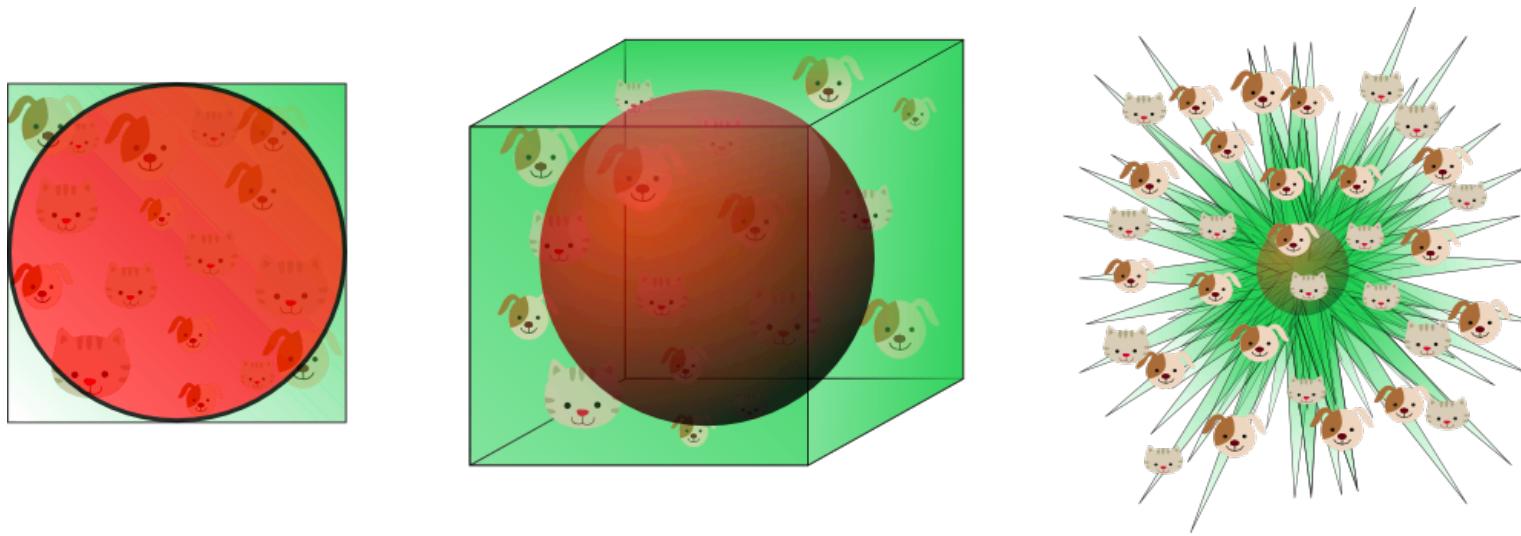
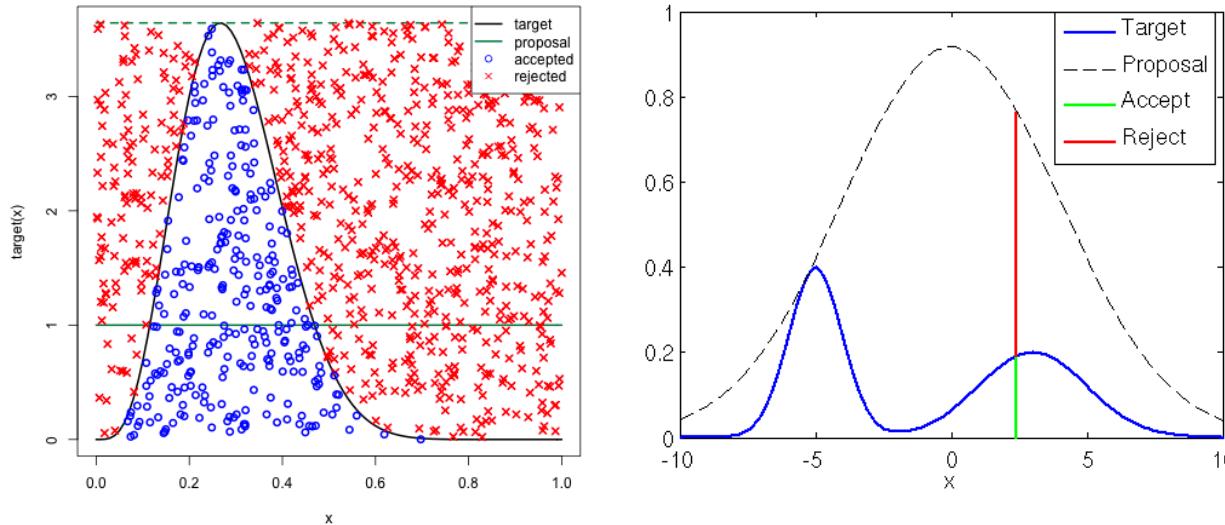
Imágenes creadas artificialmente por el modelo (no contenidas en el conjunto de entrenamiento).

# Porqué un Modelo Probabilístico?

- Si existe una aproximación general al problema de aprendizaje no supervisado es probable que ésta sea probabilística: **aprender sin supervisión consiste en aprender  $p(x)$ .**

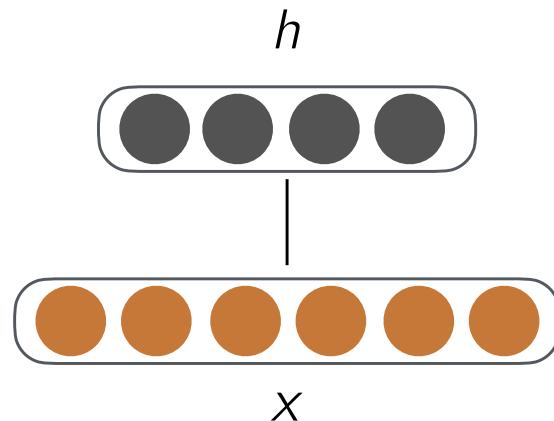


# Porqué un Modelo Probabilístico?



# Modelo Probabilístico

- El objetivo de una RBM es entonces aprender la **estructura de los datos** modelando su distribución de probabilidad.



$$p(x) = \sum_h p(x, h) \quad p(x, h) = \frac{\exp(-E(x, h))}{Z}$$

$$\begin{aligned} E(x, h) &= -H(x, h) = -\sum_{ij} h_j W_{ji} x_i - \sum_j b_j h_j - \sum_i c_i x_i \\ &= -(h^T W x + b^T h + c^T x) \end{aligned}$$

- Esperamos que la variable latente  $h \in \{0, 1\}^m$  capture factores importantes de variación de la distribución de probabilidad.

# Factorización Condicional

Teorema

$$p(x|h) = \prod_i p(x_i|h) \quad p(h|x) = \prod_j p(h_j|x)$$

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural Computation* 18.7 (2006): 1527-1554.



# Factorización Condicional

Teorema

$$p(x|h) = \prod_i p(x_i|h) \quad p(h|x) = \prod_j p(h_j|x)$$

Demo:

$$\begin{aligned} p(x|h) &= \frac{p(x, h)}{p(h)} = \frac{\exp(-E(x, h))}{\sum_x \exp(-E(x, h))} = \frac{\exp(h^T Wx + b^T h + c^T x)}{\sum_x \exp(h^T Wx + b^T h + c^T x)} \\ &= \frac{\exp(b^T h) \exp(h^T Wx + c^T x)}{\exp(b^T h) \sum_x \exp(h^T Wx + c^T x)} \\ &= \frac{\exp(\sum_i h^T W_{\cdot i} x_i + c_i x_i)}{\sum_x \exp(\sum_i h^T W_{\cdot i} x_i + c_i x_i)} \end{aligned}$$



# Factorización Condicional

$$\begin{aligned} \dots &= \frac{\exp\left(\sum_i h^T W_{\cdot i} x_i + c_i x_i\right)}{\sum_x \exp\left(\sum_i h^T W_{\cdot i} x_i + c_i x_i\right)} \\ &= \frac{\prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)}{\sum_x \prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)} \\ &= \frac{\prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)}{\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)} \\ &= \frac{\prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)}{\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{i \neq d} \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right) \exp\left(h^T W_{\cdot d} x_d + c_d x_d\right)} \\ &= \frac{\prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)}{\sum_{x_1} \sum_{x_2} \cdots \prod_{i \neq d} \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right) \sum_{x_d} \exp\left(h^T W_{\cdot d} x_d + c_d x_d\right)} \\ &= \frac{\prod_i \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)}{\left(\sum_{x_d} \exp\left(h^T W_{\cdot d} x_d + c_d x_d\right)\right) \sum_{x_1} \sum_{x_2} \cdots \prod_{i \neq d} \exp\left(h^T W_{\cdot i} x_i + c_i x_i\right)} \end{aligned}$$



# Factorización Condicional

$$p(x|h) = \dots$$

$$\begin{aligned} &= \frac{\prod_i \exp(h^T W_{\cdot i} x_i + c_i x_i)}{\prod_i (\sum_{x_i} \exp(h^T W_{\cdot d} x_i + c_i x_i))} \\ &= \prod_i \frac{\exp(h^T W_{\cdot i} x_i + c_i x_i)}{\sum_{x_i} \exp(h^T W_{\cdot d} x_i + c_i x_i)} = \prod_i p(x_i|h) \end{aligned}$$

$$\text{con } p(x_i|h) = \frac{\exp(h^T W_{\cdot i} x_i + c_i x_i)}{\sum_{x_i} \exp(h^T W_{\cdot d} x_i + c_i x_i)}$$

Análogamente con la otra:

$$p(h|x) = \prod_j p(h_j|x)$$

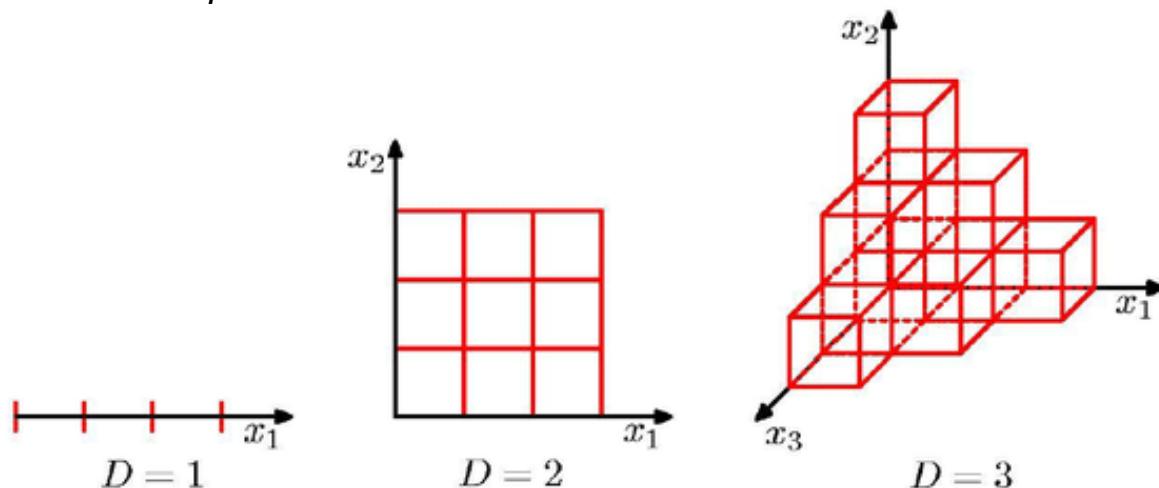
$$\text{con } p(h_j|x) = \frac{\exp(h_j W_{j \cdot} x + b_j h_j)}{\sum_{h_j} (h_j W_{j \cdot} x + b_j h_j)}$$



# RBM & Bayes Ingenuo

- Recordemos que el *Modelo Bayesiano Ingenuo* asume independencia de todos los atributos para romper la complejidad de modelar una distribución multi-variada (curse of dimensionality).

$$p_y(x) = \prod_i p_y(x_i)$$



Bayes, Thomas. "LII. An essay towards solving a problem in the doctrine of chances. FRS communicated by Mr. Price, in a letter to John Canton. *Philosophical Transactions of the Royal Society of London* 53 (1763): 370-418.

# Implicancias de la Factorización

- La RBM logra romper dicha complejidad a través de las variables latentes. Conociendo los valores de  $h$ , los atributos son independientes entre sí.

$$p(x|h) = \prod_i p(x_i|h)$$

- Explicabilidad: Esto confirma la interpretación de  $h_j$  como un factor que explica los valores que toma  $x$ . Por ejemplo, si  $x$  es la distribución de las palabras en un texto,  $h_j$  podría representar la presencia de un tópico de conversación.
- Otra implicancia del resultado es la eficiencia que tiene calcular  $p(x|h)$  (y por lo tanto simular  $x$  a partir de  $h$ ): gracias a la factorización, el costo es lineal en el número de atributos de  $x$  e vez de ser exponencial.



# Implicancias de la Factorización

- Lo más interesante es la reversibilidad del modelo: a partir de  $h$  es posible computar la probabilidad de los estados de  $x$  de manera extremadamente eficiente, pero también vale lo contrario!

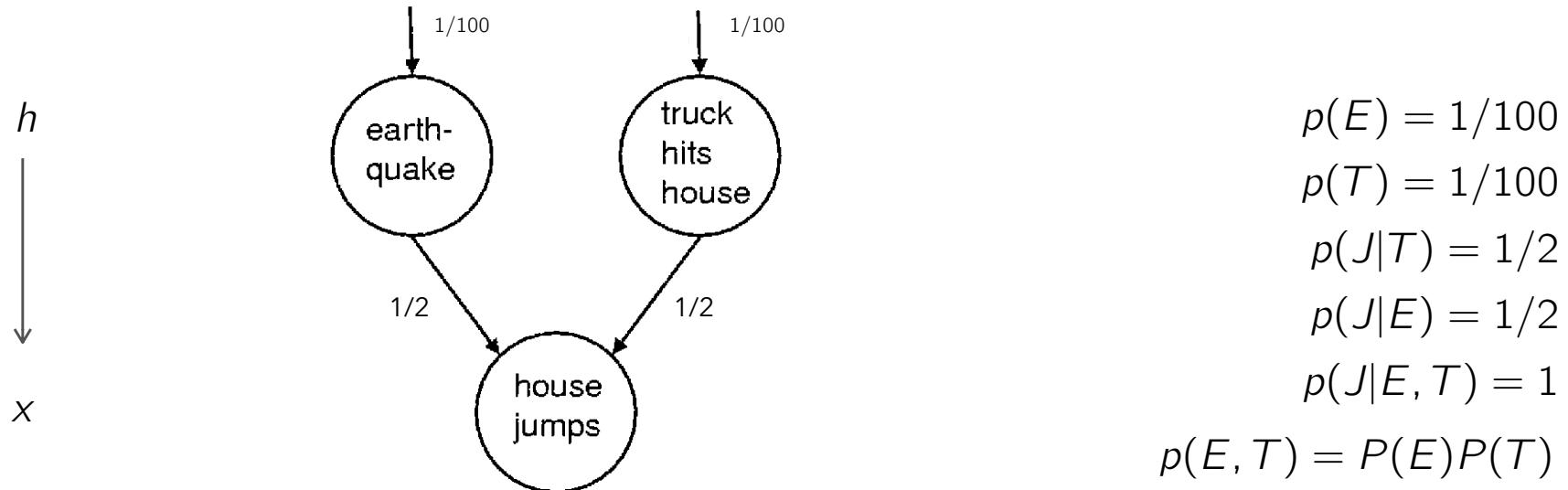
$$p(h|x) = \prod_j p(h_j|x)$$

- Eficiencia estadística: El resultado implica que los factores que modela la máquina son independientes uno del otro dado  $x$ , es decir, mínimamente redundantes entre sí (conexión clara con ICA).
- Eficiencia computacional: El resultado también implica que a partir de  $x$  podemos hacer inferencias sobre  $h$  de manera computacionalmente eficiente: el costo es lineal en el número de variables latentes en vez de exponencial.



# Explaining Away

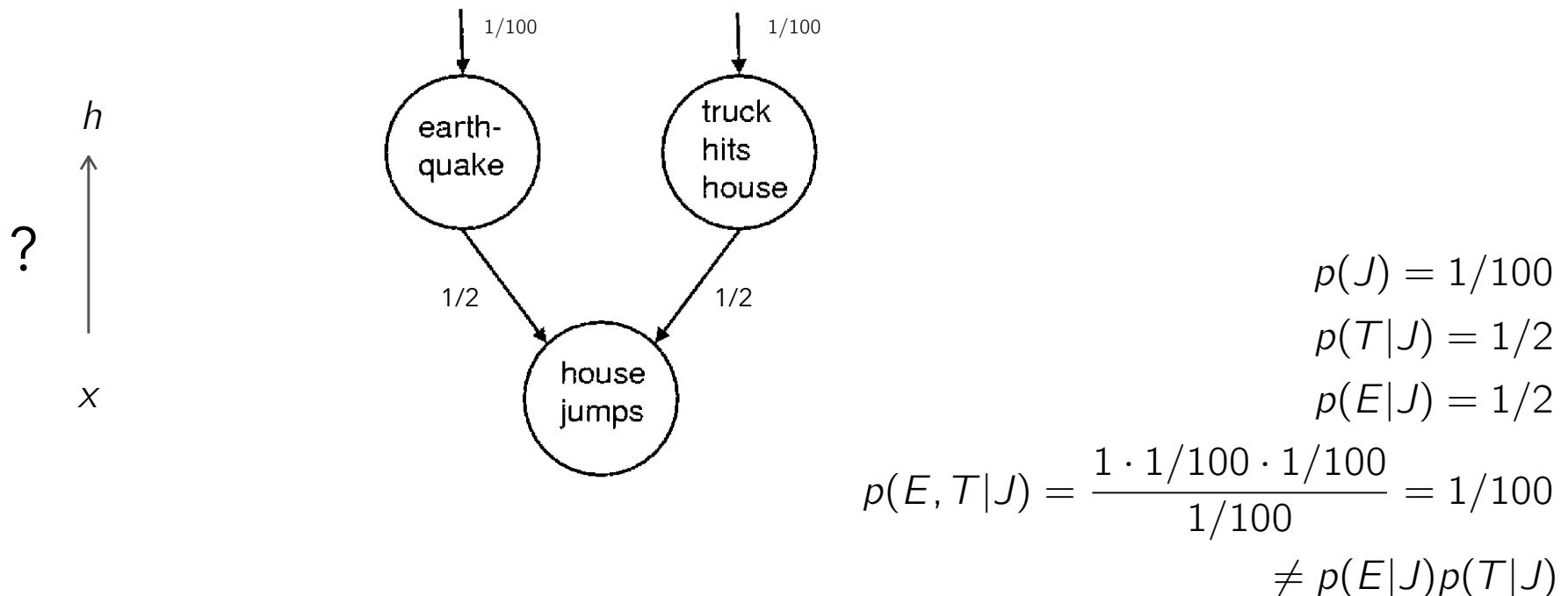
- La reversibilidad de la factorización (con todas sus consecuencias) es una gran ventaja y se obtiene por haber decidido utilizar arcos bi-direccionales. En modelos probabilísticos dirigidos, generalmente la reversibilidad no se da.



Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural Computation* 18.7 (2006): 1527-1554.

# Explaining Away

- En un modelo dirigido puede ocurrir que  $p(x|h)$  se pueda calcular eficientemente, pero que  $p(h|x)$  se vuelve intratable porque no factoriza.



Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural Computation* 18.7 (2006): 1527-1554.

# RBM & Neuronas MP

- ¿Es una RBM una red neuronal? Es fácil ver que se trata de una red no sólo en el sentido de un grafo de cómputo, sino que **los nodos del grafo son neuronas de McCulloch & Pitts!** En efecto en el caso binario:

$$p(h_j|x) = \frac{\exp(h_j W_{j\cdot}x + b_j h_j)}{\sum_{h_j} (h_j W_{j\cdot}x + b_j h_j)}$$

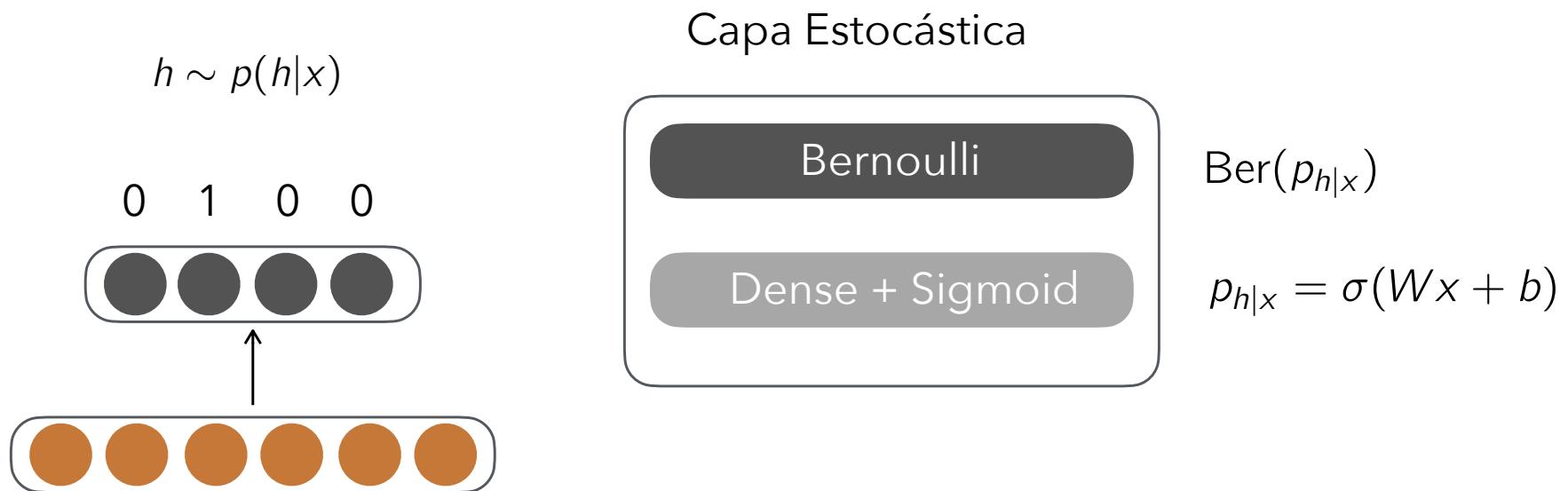
$$\Rightarrow p(h_j = 1|x) = \frac{\exp(W_{j\cdot}x + b_j)}{1 + \exp(W_{j\cdot}x + b_j)} = \sigma(W_{j\cdot}x + b_j)$$

$$\Rightarrow p(h = 1|x) = \frac{\exp(Wx + b)}{1 + \exp(Wx + b)} = \sigma(Wx + b)$$



# RBM & Neuronas MP

- Los nodos (neuronas) del grafo computan la **probabilidad de activarse** en vez de la activación en si misma. Luego, se activan con probabilidad



- El entrenamiento de capas estocásticas vía BackPropagation es un logro de los últimos 2-3 años.

# RBM & Neuronas MP

- Lo mismo vale para la capa visible

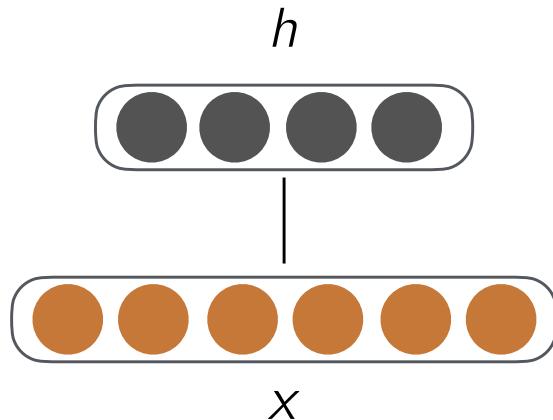
$$p(x = 1|h) = \frac{\exp(W^T h + c)}{1 + \exp(W^T h + c)} = \sigma(W^T h + c)$$



- Noten como los pesos se trasponen para operar al revés.

# Entrenamiento de RBMs

- **Cómo entrenamos el modelo?**



$$p(x) = \sum_h p(x, h) \quad p(x, h) = \frac{\exp(-E(x, h))}{Z}$$

$$\begin{aligned} E(x, h) &= -H(x, h) = -\sum_{ij} h_j W_{ji} x_i - \sum_j b_j h_j - \sum_i c_i x_i \\ &= -(h^T W x + b^T h + c^T x) \end{aligned}$$

- Nuestros datos son ahora, una colección no etiquetada de datos  $S = \{x^{(\ell)}\}_{\ell=1}^n$
- La gran mayoría de los modelos probabilísticos se entrena maximizando la verosimilitud. **Funciona para la RBM?**

$$I(W) = \sum_{\ell} \ln p(x^{(\ell)}|W) = \sum_{\ell} \left( c^T x^{(\ell)} + \sum_j \psi(W_{j\cdot} x^{(\ell)} + b_j) \right) - \ln Z$$

# RBMs como Producto de Expertos

## Teorema

$$p(x) = \frac{\exp(-F(x))}{Z} = \frac{\exp(c^T x))}{Z} \prod_j \exp(\psi_j(x))$$

$$\text{con } F(x) = -(c^T x + \sum_j \psi(W_j \cdot x + b_j)) ,$$

$$\text{y } \psi(x) = \ln(1 + \exp(x))$$

$$\psi_j(x) = \psi(W_j \cdot x + b_j)$$

$F(x)$  se denomina "la energía libre".

Hinton, G. E. "Products of Experts." 1999 Ninth International Conference on Artificial Neural Networks ICANN 1999.



# RBMs como Producto de Expertos

## Teorema

$$p(x) = \frac{\exp(-F(x))}{Z} = \frac{\exp(c^T x)}{Z} \prod_j \exp(\psi_j(x))$$

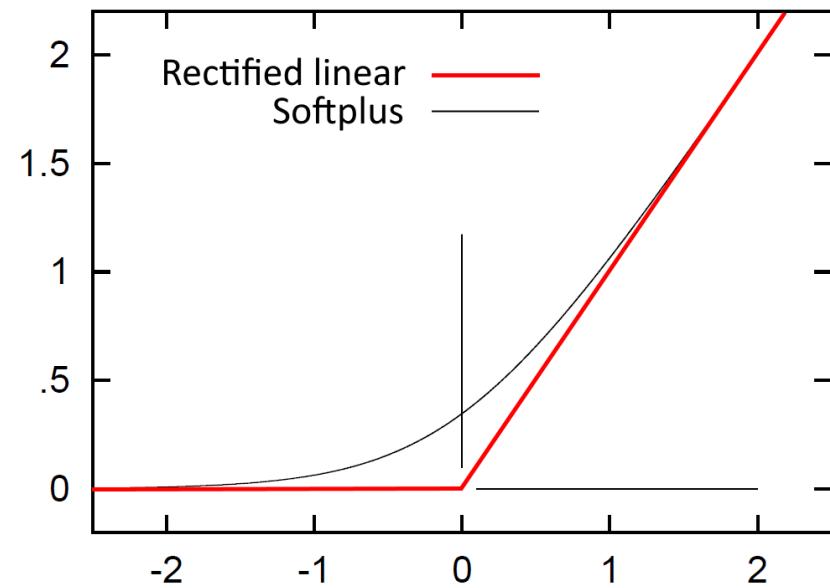
$$\psi_j(x) = \psi(W_{j \cdot} x + b_j)$$

$$\psi(x) = \ln(1 + \exp(x))$$

Notemos que  $W_{j \cdot}$  es el vector de pesos de conexión de  $h_j$  hacia  $x$ , por lo que el experto  $\psi_j$  representa simplemente una parte de la distribución  $p(x)$  que captura la variable latente  $h_j$ .

$F(x)$  se denomina "la energía libre".

Hinton, G. E. "Products of Experts." 1999 Ninth International Conference on Artificial Neural Networks ICANN 1999.



# RBMs como Producto de Expertos

Demostración

$$\begin{aligned}\sum_h \exp(-E(x, h)) &= \sum_h \exp(h^T W x + b^T h + c^T x) \\&= \exp(c^T x) \sum_h \exp\left(\sum_j (h_j W_{j \cdot} x + b_j h_j)\right) \\&= \exp(c^T x) \sum_{h_1} \sum_{h_2} \cdots \sum_{h_k} \prod_j \exp(h_j W_{j \cdot} x + b_j h_j) \\&= \exp(c^T x) \sum_{h_1} \sum_{h_2} \cdots \sum_{h_k} \prod_{j \neq k} \exp(h_j W_{j \cdot} x + b_j h_j) \exp(h_k W_{k \cdot} x + b_k h_k) \\&= \exp(c^T x) \sum_{h_1} \sum_{h_2} \cdots \prod_{j \neq k} \exp(h_j W_{j \cdot} x + b_j h_j) \sum_{h_k} \exp(h_k W_{k \cdot} x + b_k h_k) \\&= \exp(c^T x) \left( \sum_{h_k} \exp(h_k W_{k \cdot} x + b_k h_k) \right) \sum_{h_1} \sum_{h_2} \cdots \prod_{j \neq k} \exp(h_j W_{j \cdot} x + b_j h_j)\end{aligned}$$



# RBMs como Producto de Expertos

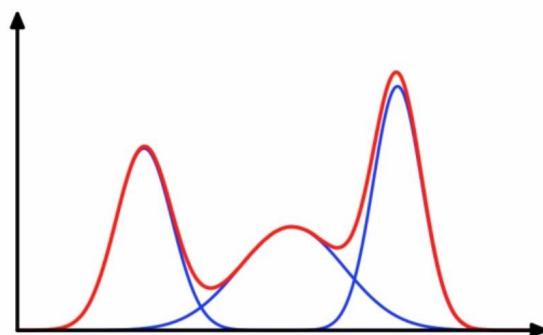
Desde acá basta reordenar un poco,

$$\begin{aligned}\sum_h \exp(-E(x, h)) &= \exp(c^T x) \prod_j \left( \sum_{h_j} \exp(h_j W_{j \cdot} x + b_j h_j) \right) \\ &= \exp(c^T x) \prod_j (1 + \exp(W_{j \cdot} x + b_j)) \\ &= \exp(c^T x) \exp \ln \prod_j (1 + \exp(W_{j \cdot} x + b_j)) \\ &= \exp(c^T x) \exp \sum_j \ln (1 + \exp(W_{j \cdot} x + b_j)) \\ &= \exp(c^T x) \exp \sum_j \psi(W_{j \cdot} x + b_j) \\ &= \exp(c^T x) \prod_j \exp(\psi(W_{j \cdot} x + b_j))\end{aligned}$$



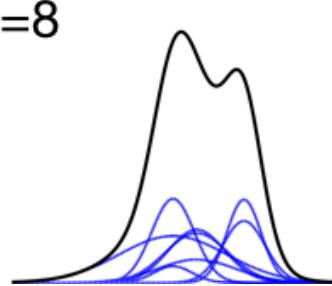
# Productos versus Sumas de Expertos

Los modelos de mezcla (suma de expertos) representan uno de los modelos más conocidos para aproximar o aprender distribuciones.

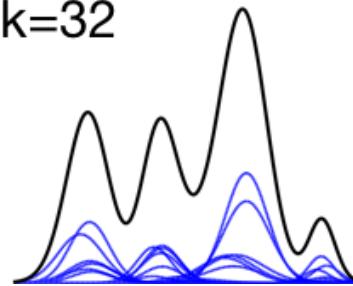


$$p(x) = \sum_j \alpha_j \psi_j(x)$$

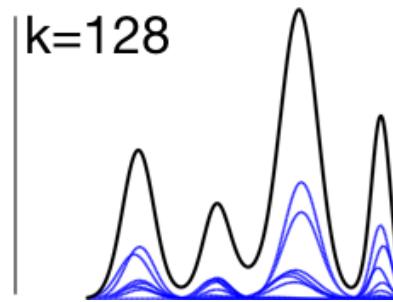
$k=8$



$k=32$

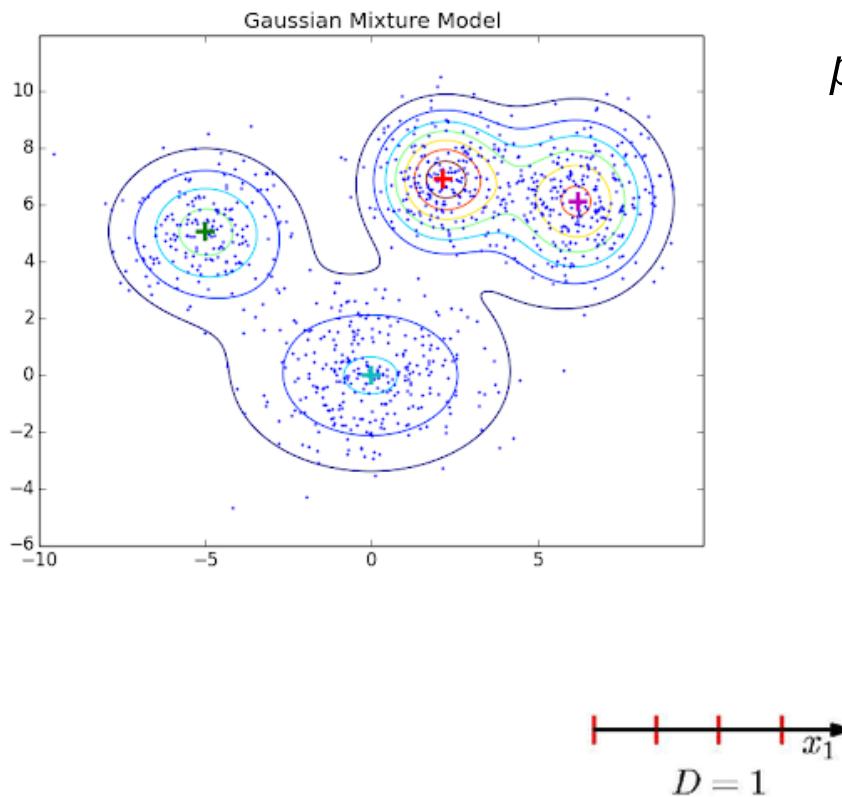


$k=128$

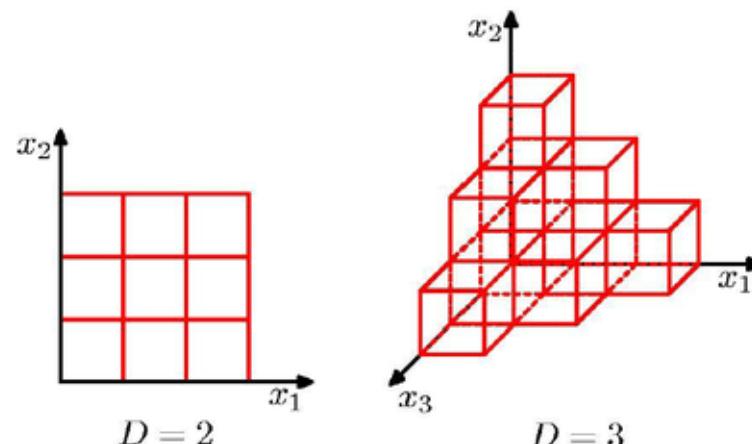


# Productos versus Sumas de Expertos

Una de las desventajas más relevantes de una suma de expertos es que sufren severamente el problema de la maldición de la dimensionalidad

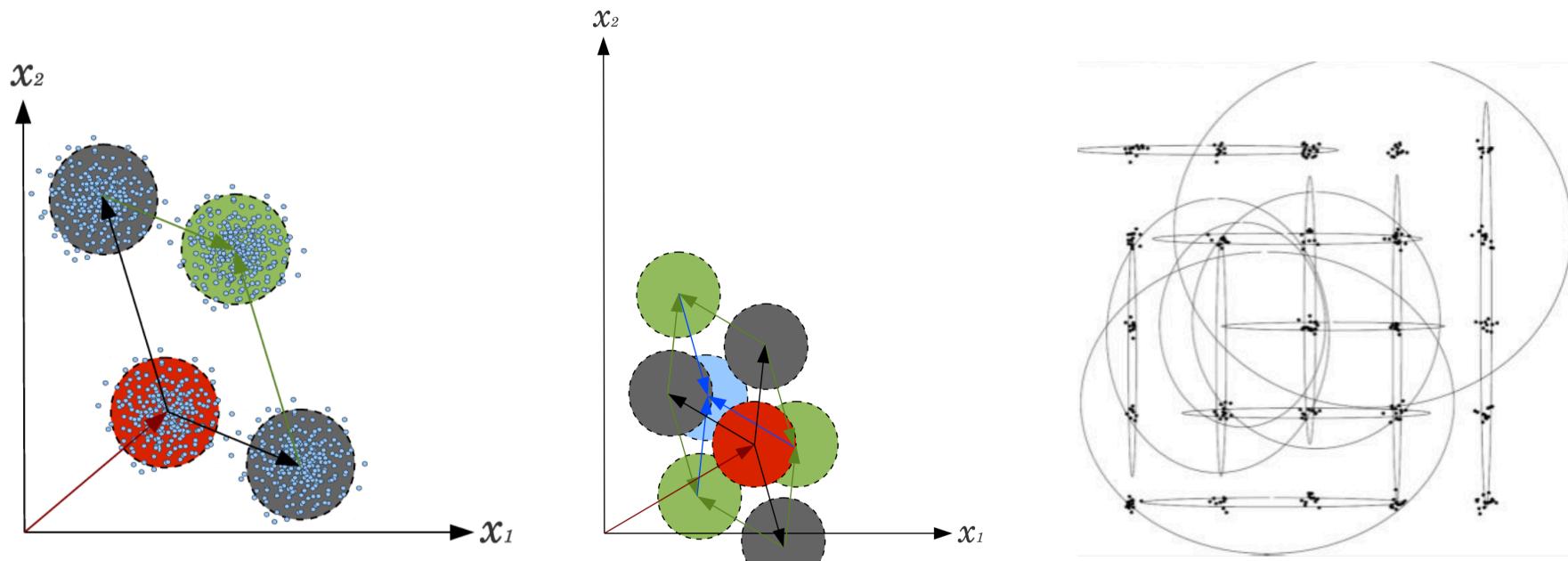


$$p(x) = \sum_j \alpha_j \psi_j(x)$$



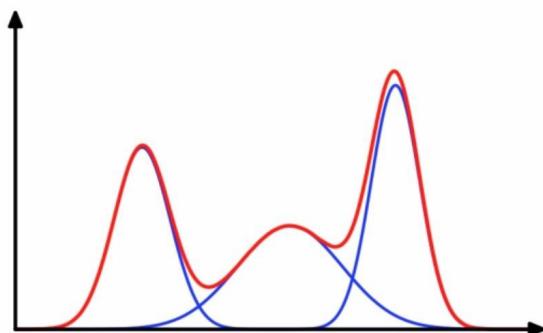
# Productos versus Sumas de Expertos

Un producto de expertos combate naturalmente este problema de dos formas: (i) primero proyecta los datos en un sub-espacio de menor dimensionalidad (capa oculta) y (ii) las regiones de alta probabilidad se construyen multiplicando regiones de alta probabilidad en cada dimensión:  $K$  regiones en cada dimensión producen  $K^d$  regiones en el espacio.

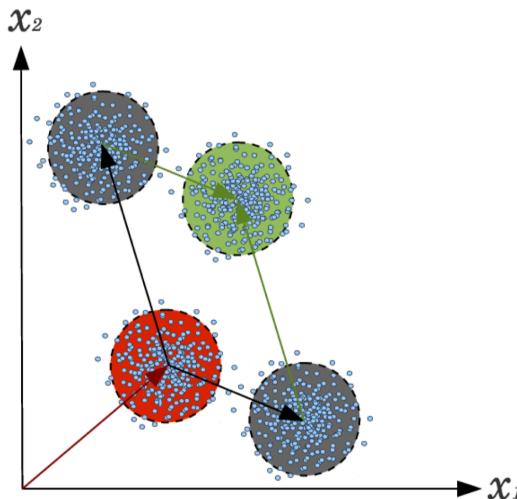


# Productos versus Sumas de Expertos

Otra diferencia fundamental es que en un MoE, cada componente es un experto local y basta que 1 experto esté "seguro" acerca de un dato, para que modelo asigne alta probabilidad a ese dato. En un PoE, los expertos no son locales (e.g. por el embedding) y se requiere que todos o muchos estén de acuerdo para que un dato obtenga alta probabilidad.



$$p(x) = \sum_j \alpha_j \psi_j(x)$$



$$p(x) = \prod_j \psi_j(x)$$

# Productos versus Sumas de Expertos

Otra diferencia relevante es lo que sucede al considerar la log-verosimilitud de un dato (usada típicamente para aprender los parámetros del modelo).

**PoE**

$$\begin{aligned}-\log p(x) &= -\log \prod_j \psi_j(x) \\ &= -\sum_j \log \psi_j(x)\end{aligned}$$

**MoE**

$$-\log p(x) = -\log \sum_j \psi_j(x)$$

**RBM**

$$p(x) = \frac{\exp(-F(x))}{Z}$$

$$\text{con } F(x) = -(c^T x + \sum_j \psi(W_j \cdot x + b_j))$$



# Productos versus Sumas de Expertos

La ventaja anterior viene con un desventaja: normalizar una suma de expertos es sencillo (normalizar cada componente), pero normalizar un producto de expertos es muy caro en el caso altamente dimensional.

**PoE**

$$\sum_x p(x) = \sum_x \underbrace{\prod_j \psi_j(x)}_{2^m \text{ sumas}}$$

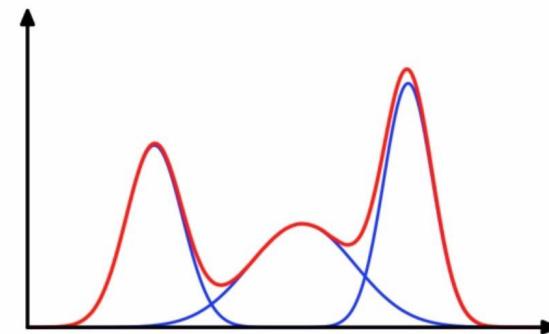
**MoE**

$$\sum_x p(x) = \sum_j \alpha_j \underbrace{\sum_x \psi_j(x)}_{m \text{ sumas}}$$

**RBM**

$$p(x) = \frac{\exp(-F(x))}{Z}$$

$$Z = \sum_{x,h} p(x, h) = \sum_{x,h} \exp(-E(x, h))$$



# Aproximación Universal

## Teorema

Una RBM binaria (Bernoulli-Bernoulli) con  $n$  unidades visibles y  $m$  nodos ocultos puede aproximar arbitrariamente bien cualquier distribución de probabilidad sobre  $\{0,1\}^n$ .

Montufar, Guido, and Nihat Ay. "Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines." *Neural Computation* 23.5 (2011): 1306-1319.



# Aproximación Universal

## Teorema

Una RBM binaria (Bernoulli-Bernoulli) con  $n$  unidades visibles y  $m$  nodos ocultos puede aproximar arbitrariamente bien cualquier distribución de probabilidad sobre  $\{0,1\}^n$ .

... siempre y cuando  $m \geq 2^{n-1} - 1$

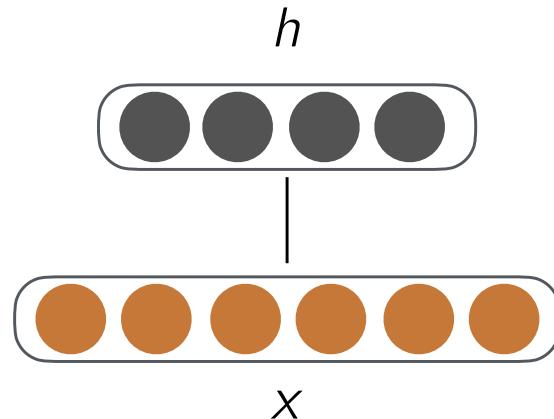
(esto implica que el número de parámetros requeridos no es peor que en un modelo de mezcla)

Montufar, Guido, and Nihat Ay. "Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines." *Neural Computation* 23.5 (2011): 1306-1319.



# Entrenamiento de RBMs

- **Cómo entrenamos el modelo?**



$$p(x) = \sum_h p(x, h) \quad p(x, h) = \frac{\exp(-E(x, h))}{Z}$$

$$\begin{aligned} E(x, h) &= -H(x, h) = -\sum_{ij} h_j W_{ji} x_i - \sum_j b_j h_j - \sum_i c_i x_i \\ &= -(h^T W x + b^T h + c^T x) \end{aligned}$$

- La gran mayoría de los modelos probabilísticos se entrena maximizando la verosimilitud. **Funciona para la RBM?**

$$I(W) = \sum_\ell \ln p(x^{(\ell)} | W) = \sum_\ell \left( c^T x^{(\ell)} + \sum_j \psi(W_{j\cdot} x^{(\ell)} + b_j) \right) - \ln Z$$

# Gradientes

$$I(W) = \sum_{\ell} \ln p(x^{(\ell)}|W) = \sum_{\ell} \left( c^T x^{(\ell)} + \sum_j \psi(W_j \cdot x^{(\ell)} + b_j) \right) - \ln Z$$

**RBM**

$$-\log p(x) = F(x) \quad \text{con} \quad F(x) = -(c^T x + \sum_j \psi(W_j \cdot x + b_j))$$

$$I(W) = \sum_{\ell} \ln p(x^{(\ell)}|W) = \sum_{\ell} \left( c^T x^{(\ell)} + \sum_j \psi(W_j \cdot x^{(\ell)} + b_j) \right) - \ln Z$$

$$I(W) = \sum_{x \in S} \left( c^T x + \sum_j \psi(W_j \cdot x + b_j) \right) - \ln Z$$

La derivada de la primera parte queda

$$\frac{\partial (c^T x + \sum_j \psi(W_j \cdot x + b_j))}{\partial W_{ji}} = \psi'(W_j \cdot x + b_j) x_i$$



# Gradientes

Interesantemente, la derivada de a softplus es la sigmoidal!!

$$\psi'(\xi) = \frac{\exp(\xi)}{1 + \exp(\xi)} = \sigma(\xi)$$

Obtenemos ...

$$\begin{aligned} \frac{\partial (c^T x + \sum_j \psi(W_{j\cdot}x + b_j))}{\partial W_{ji}} &= \sigma(W_{j\cdot}x + b_j)x_i^\ell = p(h_j = 1|x)x_i \\ &= \sum_{h_j} p(h_j|x)h_jx_i \\ &= \mathbb{E}_{h_j|x} h_jx_i \\ &= \mathbb{E}_{h|x} h_jx_i \end{aligned}$$



# Gradientes

Para el segundo miembro

$$\begin{aligned}\frac{\partial \ln Z}{\partial W_{ji}} &= \frac{1}{Z} \frac{\partial Z}{\partial W_{ji}} = - \sum_{\tilde{x}, h} \frac{\exp(-E(x, h))}{Z} \frac{\partial E(\tilde{x}, h)}{\partial W_{ji}} \\ &= - \sum_{\tilde{x}, h} p(\tilde{x}, h) \frac{\partial E(\tilde{x}, h)}{\partial W_{ji}}\end{aligned}$$

Pero

$$\frac{\partial E(x, h)}{\partial W_{ji}} = - \frac{\partial(h^T W x + b^T h + c^T x)}{\partial W_{ji}} = -h_j x_i$$



# Gradientes

Obtenemos

$$\frac{\partial \ln Z}{\partial W_{ji}} = \sum_{\tilde{x}, h} p(x, h) h_j x_i = \mathbb{E}_{\tilde{x}, h} h_j \tilde{x}_i$$

Juntando todo para un ejemplo específico

$$\frac{\partial I}{\partial W} = \mathbb{E}_{h|x} h x^T - \mathbb{E}_{\tilde{x}, h} h x^T$$

Para los bases obtenemos

$$\frac{\partial I}{\partial c} = x - \mathbb{E}_{\tilde{x}} \tilde{x} \quad \frac{\partial I}{\partial b} = \mathbb{E}_{h|x} h - \mathbb{E}_h h$$



# Contrastive Divergence

Si podemos calcular el gradiente anterior, la aplicación del GD al entrenamiento de la red (a partir de un conjunto de datos  $S$ ) quedaría como sigue:

---

**Algorithm 1:** GD for the RBM

---

- 1 Inicializar los parámetros de la máquina.
  - 2 **for**  $T = 1, \dots, \text{do}$
  - 3    $W \leftarrow W + \eta_t \sum_{x \in S} (\mathbb{E}_{h|x} h x^T - \mathbb{E}_{\tilde{x}, h} h x^T)$ .
  - 4    $b \leftarrow b + \eta_t \sum_{x \in S} (\mathbb{E}_{h|x} h - \mathbb{E}_h h)$ .
  - 5    $c \leftarrow c + \eta_t \sum_{x \in S} (x - \mathbb{E}_{\tilde{x}} \tilde{x})$ .
- 

Cómo computamos gradiente?

$$\frac{\partial I}{\partial W} = \mathbb{E}_{h|x} h x^T - \mathbb{E}_{\tilde{x}, h} h x^T$$



# Contrastive Divergence

El primer miembro es fácil porque se trata de un valor esperado condicional a los datos de entrenamiento:

$$\frac{\partial I}{\partial W} = \mathbb{E}_{h|x} hx^T - \mathbb{E}_{\tilde{x}, h} hx^T$$

$$\mathbb{E}_{h|x} hx^T = p(h=1|x)x^T = \sigma(Wx + b_j)x^T$$

El segundo término en cambio representa un valor esperado marginal, respecto de la distribución de probabilidad implementada por la máquina en ese momento.

$$\mathbb{E}_{\tilde{x}, h} hx^T = \underbrace{\sum_{\tilde{x}, h} p(\tilde{x}, h)hx^T}_{2^{m+n} \text{ sumas}}$$



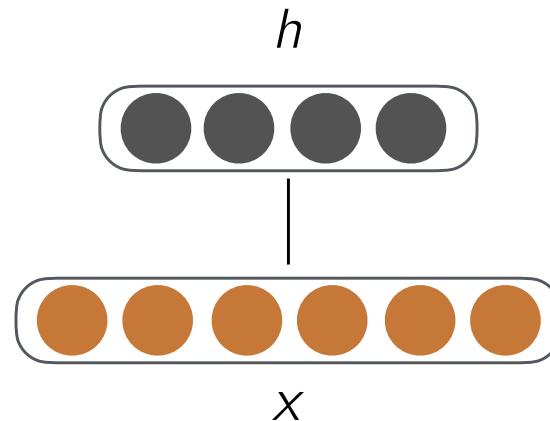
# Contrastive Divergence

Contrastive Divergence (CD) es un método para aproximar ese cálculo.

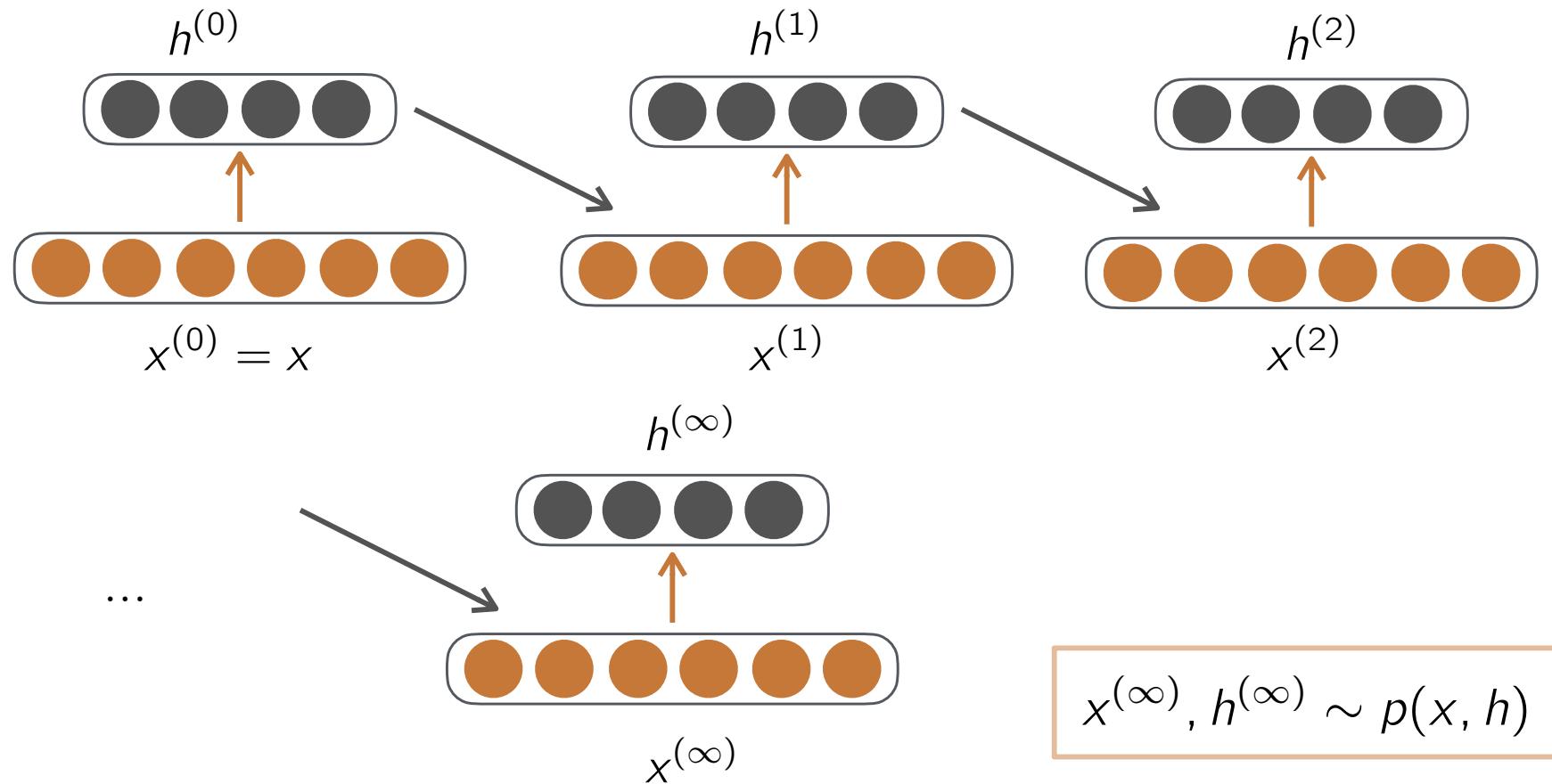
Por la ley de los grandes números, es claro que, si pudiésemos obtener  $n^*$  muestras  $(x^*, h^*)$  de la máquina, una aproximación MC básica es:

$$\mathbb{E}_{\tilde{x}, h} h x^T \approx \frac{1}{n^*} \sum_{(x^*, h^*)} h^* x^{*T}$$

Cómo podemos obtener muestras que sigan la distribución de probabilidad de la máquina?



# Contrastive Divergence



# Contrastive Divergence

Este método es una especialización del método de Gibbs para simular/muestrear un modelo de probabilidad.

Si tenemos que muestrear  $p(x_1, x_2, \dots, x_n)$  y de alguna forma podemos calcular rápidamente  $p(x_i | x_{-i}) = p(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ , el método itera los siguientes pasos:

---

**Algorithm 1:** Gibbs Sampling

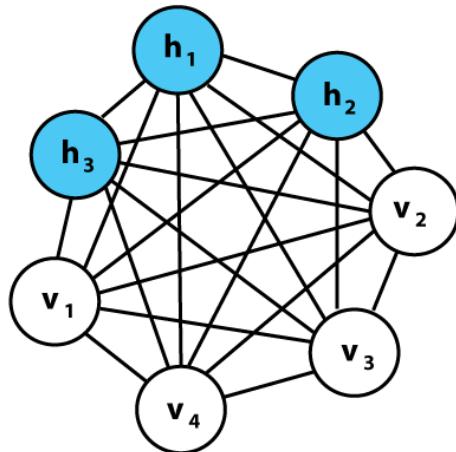
---

- 1 Inicializar  $x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}$ .
  - 2 **for**  $t = 1, \dots, \text{do}$ 
    - 3      $x_1^{(t)} \sim p(x_1 | x_{-1} = x_2^{(t-1)}, \dots, x_n^{(t-1)})$ .
    - 4      $x_2^{(t)} \sim p(x_2 | x_{-2} = x_1^{(t)}, x_3^{(t-1)}, \dots, x_n^{(t-1)})$ .
    - 5      $x_3^{(t)} \sim p(x_3 | x_{-3} = x_1^{(t)}, x_2^{(t)}, x_4^{(t-1)}, \dots, x_n^{(t-1)})$ .
    - 6      $\dots$
    - 7      $x_n^{(t)} \sim p(x_n | x_{-n} = x_1^{(t)}, x_2^{(t)}, x_3^{(t)}, \dots, x_{n-1}^{(t)}))$ .
- 



# Porqué RBM?

- Una RBM puede verse como un caso particular de Red de Boltzmann. Este tipo de red, están permitidas las conexiones entre neuronas de la misma capa. Esto hace aparecer términos de segundo orden la función de energía que hacen mucho más compleja la operación y entrenamiento del modelo.

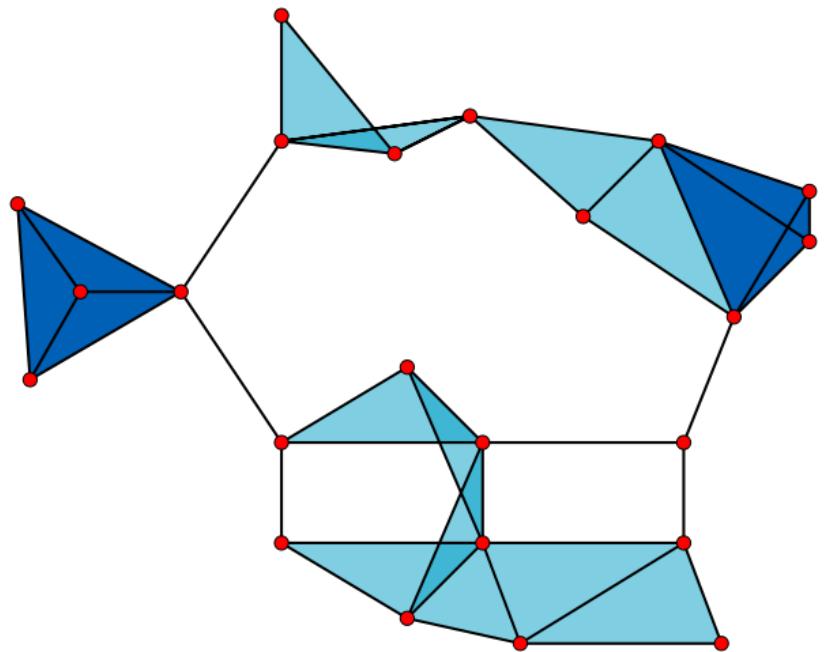


$$p(x) = \sum_h p(x, h)$$
$$p(x, h) = \frac{\exp(-E(x, h))}{Z}$$

$$\begin{aligned} E(x, h) &= -H(x, h) = -\sum_{ij} x_j W_{ji}^x x_i - \sum_{ij} h_j W_{ji}^h h_i - \sum_{ij} h_j W_{ji} x_i - \sum_j b_j h_j - \sum_i c_i x_i \\ &= -(x^T W^x x + h^T W^h h + h^T W x + b^T h + c^T x) \end{aligned}$$

# Porqué RBM?

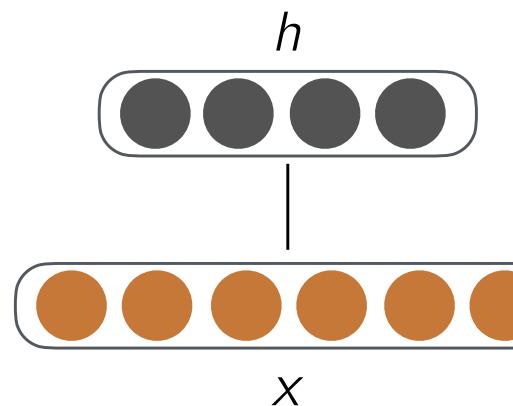
- Algunas de las propiedades de la RBM se extienden al caso general usando el concepto de clique (maximal) de un grafo. Por ejemplo, es posible mostrar que  $p(x, h)$  factoriza como producto de "mini-distribuciones" sobre el conjunto de cliques maximales del grafo.



$$p(x, h) = \prod_{c \in C} \phi_c(x_c, h_c)$$

# RBM Bernoulli-Gaussiana

- ¿Cómo manejar el caso de data continua? Una extensión matemáticamente sencilla consiste en sustituir las neuronas visibles por neuronas Gausianas. Más específicamente, la función de energía se modifica como sigue:



$$p(x) = \sum_h p(x, h)$$
$$p(x, h) = \frac{\exp(-E(x, h))}{Z}$$

$$\begin{aligned} E(x, h) &= -H(x, h) = -\sum_{ij} \frac{h_j W_{ji} x_i}{\sigma^2} - \sum_j b_j h_j + \sum_i \frac{(x_i - \mu_i)^2}{\sigma^2} \\ &= -\left( \frac{h^T W x}{\sigma^2} + b^T h - \frac{\|x - \mu\|^2}{\sigma^2} \right) \end{aligned}$$

# RBM Bernoulli-Gaussiana

- Es posible demostrar que

$$p(h_j = 1|x) = \frac{1}{1 + \exp\left(-b_j - \frac{W_{j \cdot} x}{\sigma^2}\right)} = \text{sigmoid}\left(\frac{W_{j \cdot}}{\sigma^2}x + b_j\right)$$

$$p(x_i|h) = \mathcal{N}(W_{i \cdot}^T h + c_i; \sigma^2)$$

- Se verifica también que el modelo marginal  $p(x)$  es un producto de expertos donde cada experto es la suma de dos gausianas.
- Se verifica también que el modelo es una suma de expertos con un número exponencialmente grande de expertos ( $2^n$ ).



Melchior, Jan, Nan Wang, and Laurenz Wiskott. "Gaussian-binary restricted Boltzmann machines for modeling natural image statistics." *PLoS one* 12.2 (2017): e0171015.

# RBM Bernoulli-Gaussiana

KyungHyun Cho

## Improved Learning Algorithms for Restricted Boltzmann Machines

Master's thesis

Espoo, 14th March 2011

Improved Learning of Gaussian-Bernoulli Restricted Boltzmann Machines. International Conference on Artificial Neural Networks, 2011.



# Demo

- La RBMs se popularizaron sólo hacia 2006-2007 después que fuesen empleadas como pieza clave en la solución ganadora del Netflix Prize. Veamos por lo tanto cómo entrenar una RBM para hacer recomendaciones.



A screenshot of a Jupyter Notebook interface. The title bar says "INF395-RecRBM.ipynb". The left sidebar has a tree view with nodes like "índice", "RBM para Recomendaciones" (which is expanded), "Créditos", "Datos", "Preparación", "Modelo", "Entrenamiento", "Evaluación", and "+ Sección". The main area contains a section titled "RBM para Recomendaciones" with the sub-instruction "Construiremos un recomendador muy sencillo usando un RBM binaria." Below this, there are three rows of movie thumbnails with titles: "Emmy-winning US TV Shows" (Rick and Morty, Family Guy, Arrested Development, House of Cards, Orange Is the New Black, The Good Wife), "Police Detective TV Dramas" (Peaky Blinders, The Walking Dead, Dark, The Method, Altered Carbon, Broadchurch), and "Critically Acclaimed Witty TV Shows" (The Good Place, My Next Guest Needs No Introduction with David Letterman, BoJack Horseman, The IT Crowd, Grace and Frankie, Big Mouth).