Idea de los Auto Encoders

Plan de Ataque

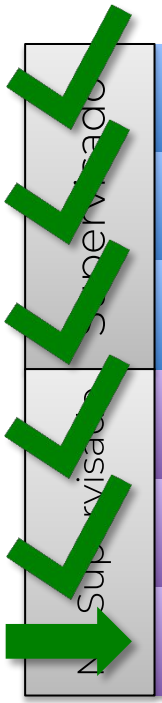
Plan de Ataque

Qué vamos a aprender en esta sección:

- Auto Codificadores (Auto Encoders)
- Cómo entrenar un Auto Encoder
- Sobrecompletar las capas ocultas
    - Sparse Auto Encoders
    - Denoising Autoencoders
    - Contractive Autoencoders
- Stacked Autoencoders
- Deep Autoencoders

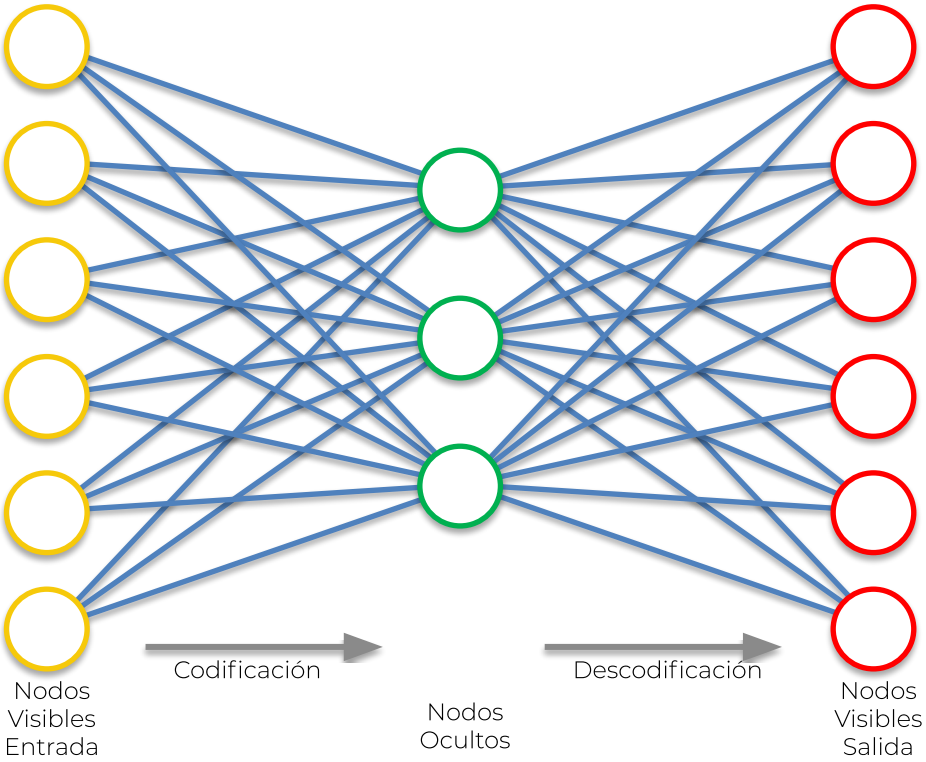
Auto Encoders

Auto Encoders

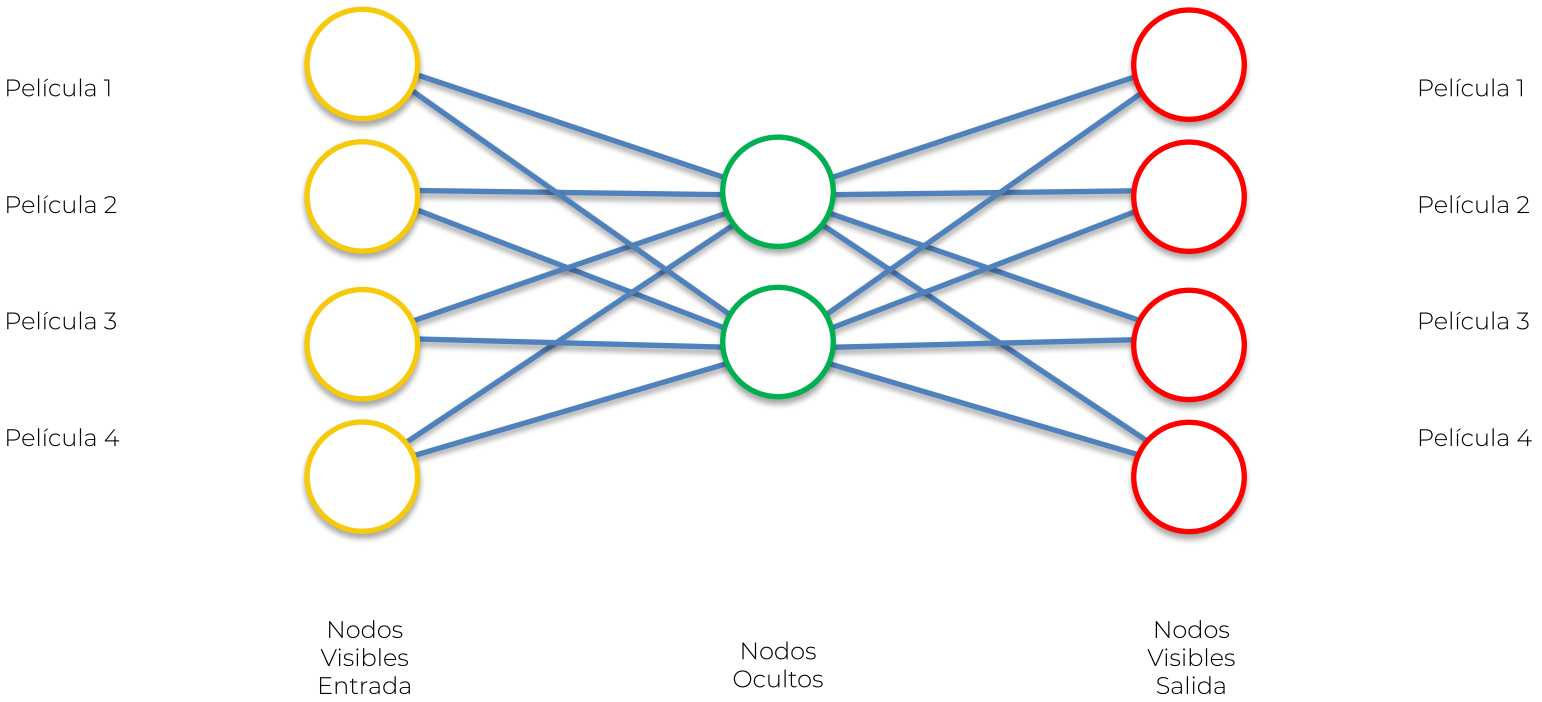


Redes neuronales artificiales	Utilizado para regresión y clasificación
Redes neuronales convolucionales	Utilizado para la visión por ordenador
Redes neuronales recurrentes	Utilizado para el análisis de series temporales
Mapas autoorganizados	Utilizado para la detección de características
Máquinas de Boltzmann Profundas	Utilizado para los sistemas de recomendación
AutoEncoders	Utilizado para los sistemas de recomendación

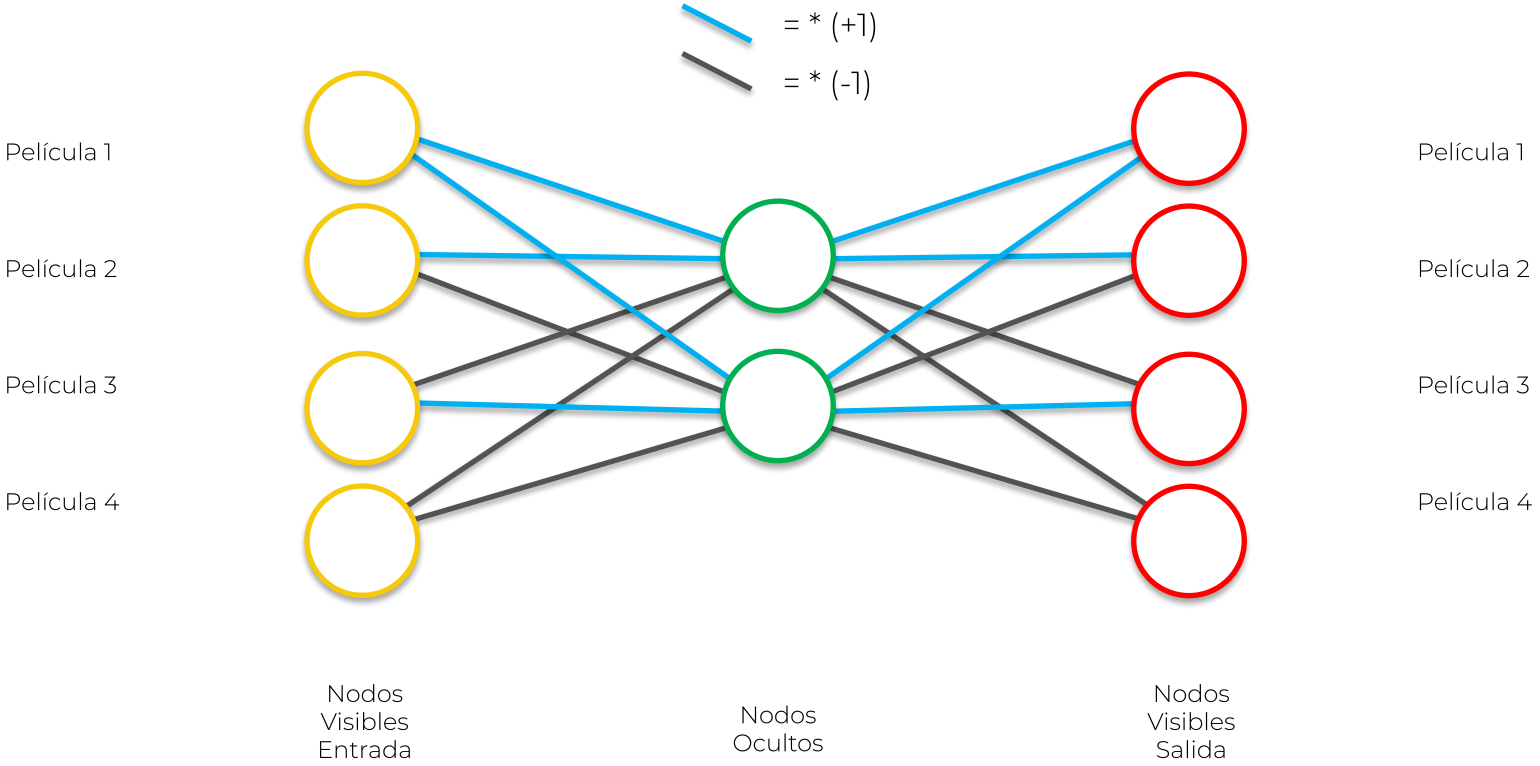
Auto Encoders



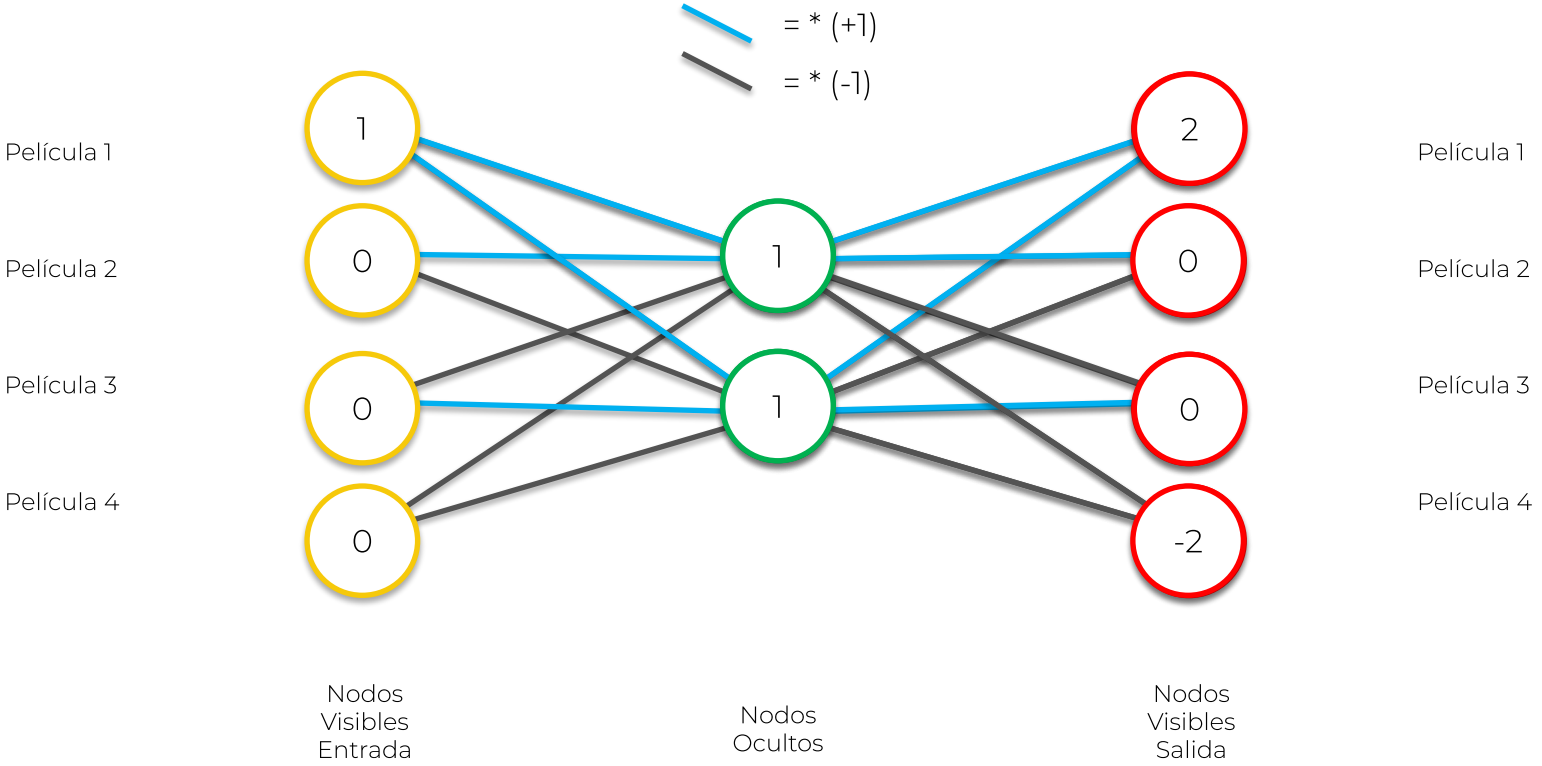
Auto Encoders



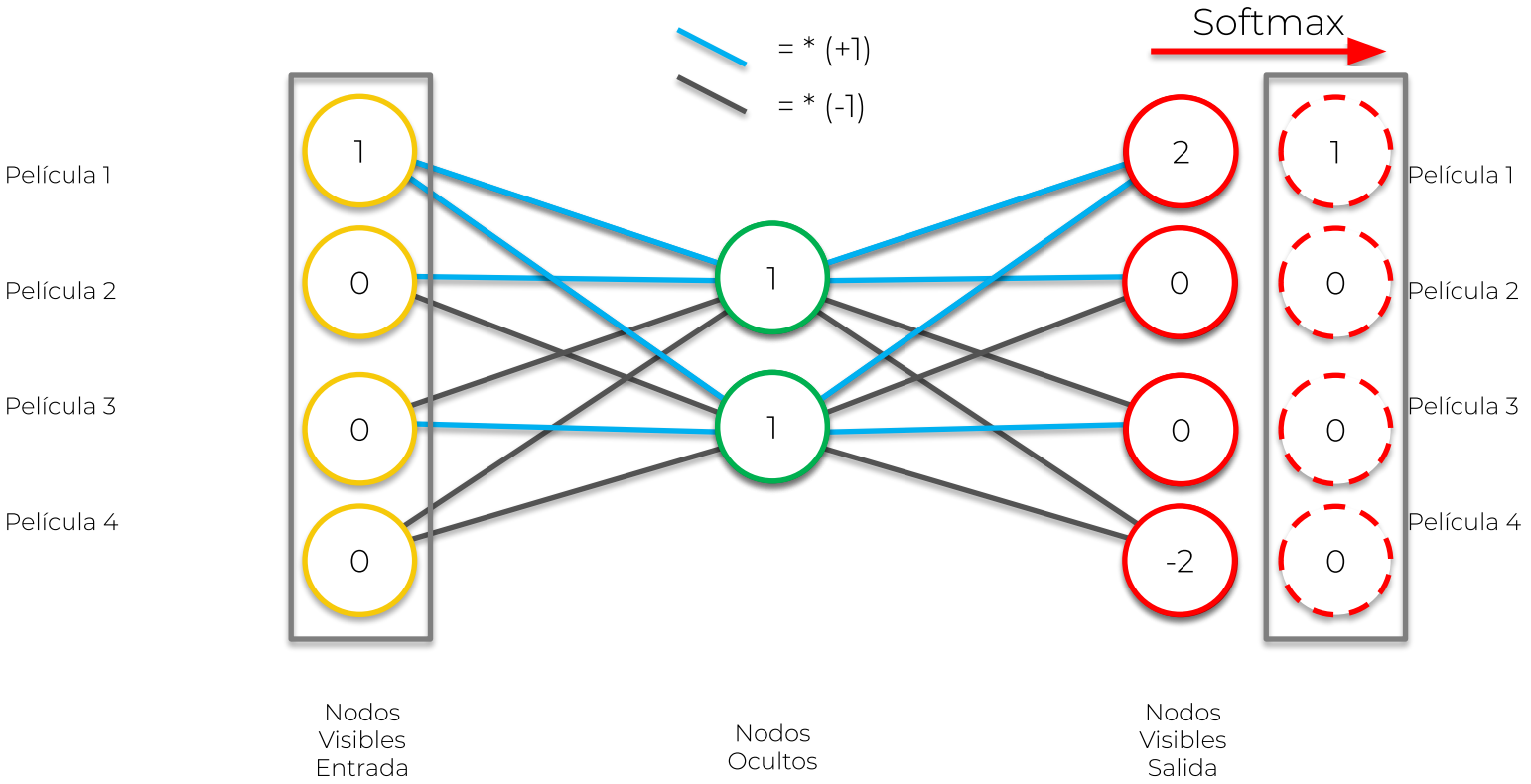
Auto Encoders



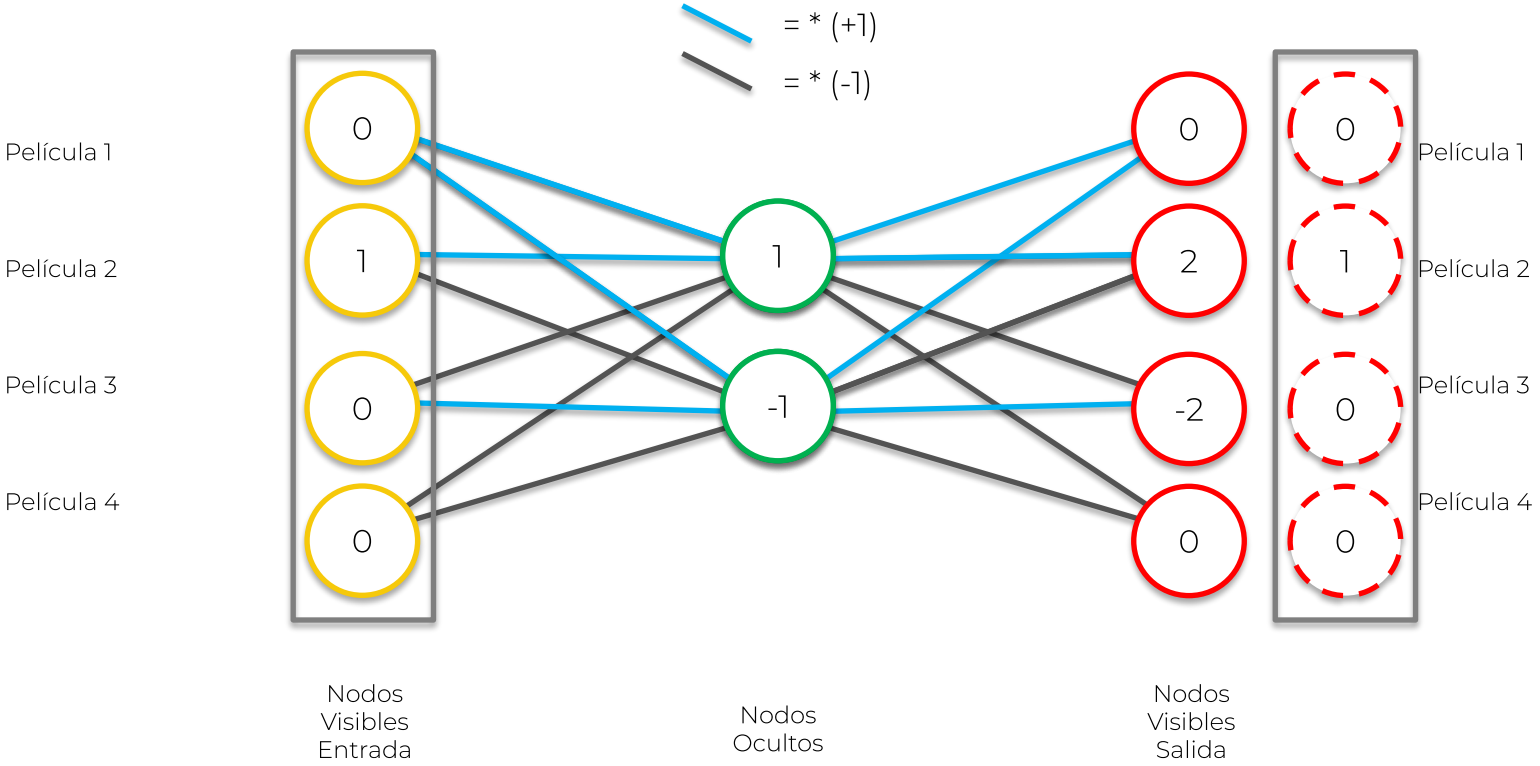
Auto Encoders



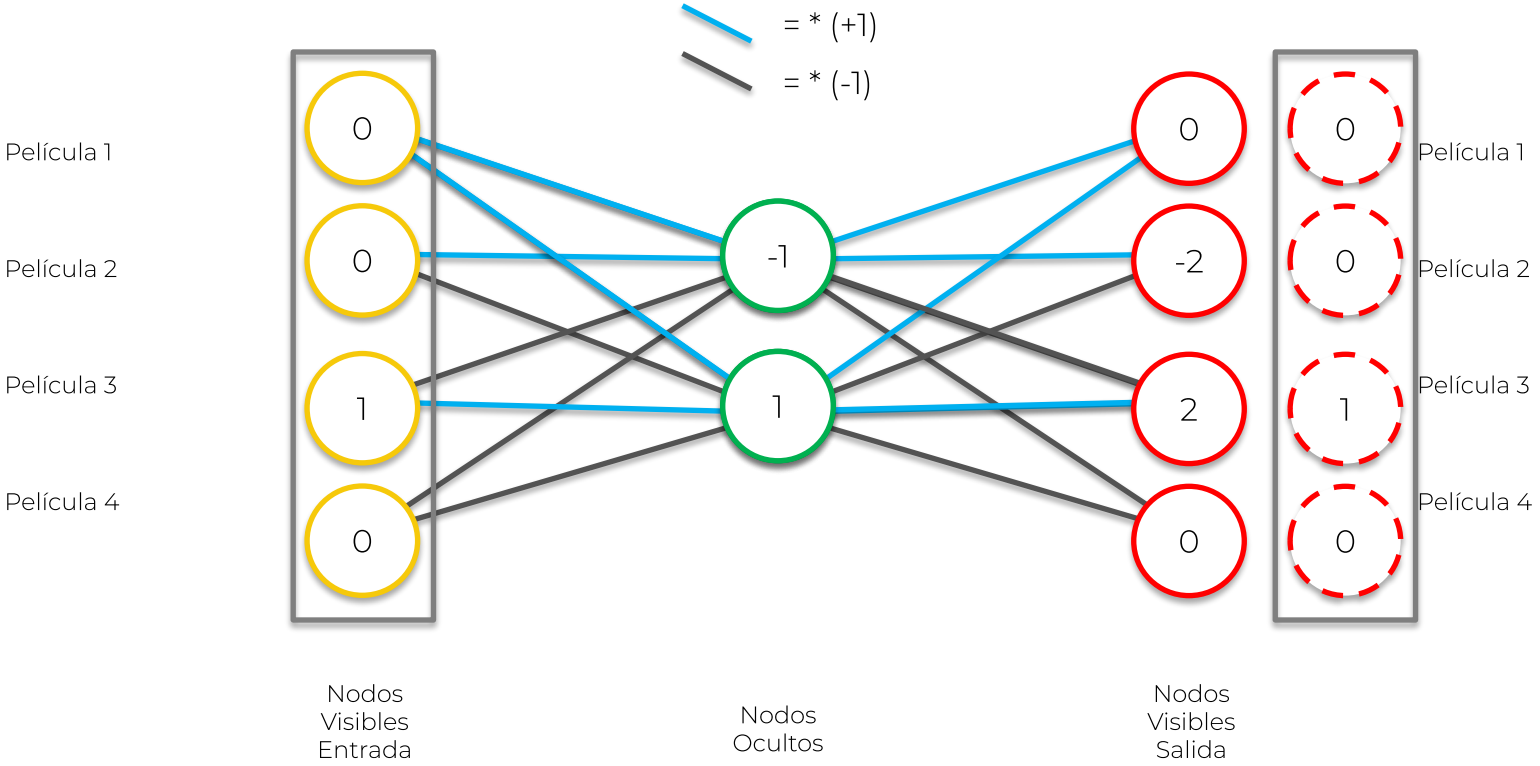
Auto Encoders



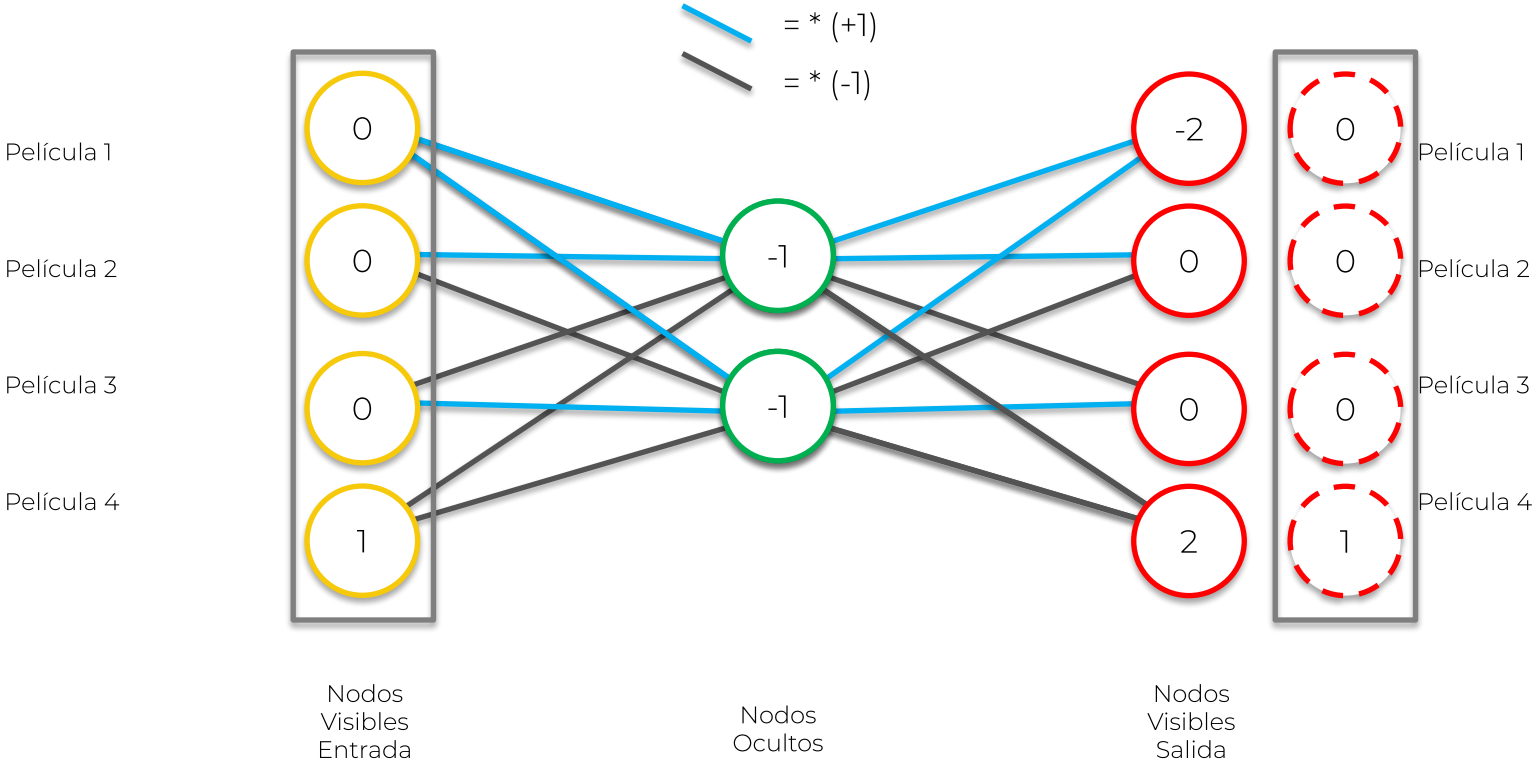
Auto Encoders



Auto Encoders



Auto Encoders

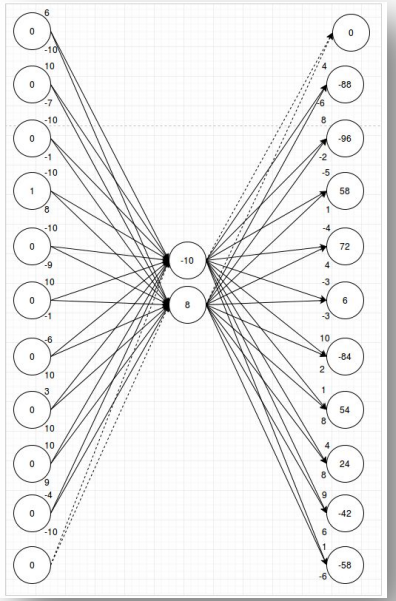


Auto Encoders

Lectura Adicional:

Neural Networks Are Impressively Good At Compression

de Malte Skarupke (2016)

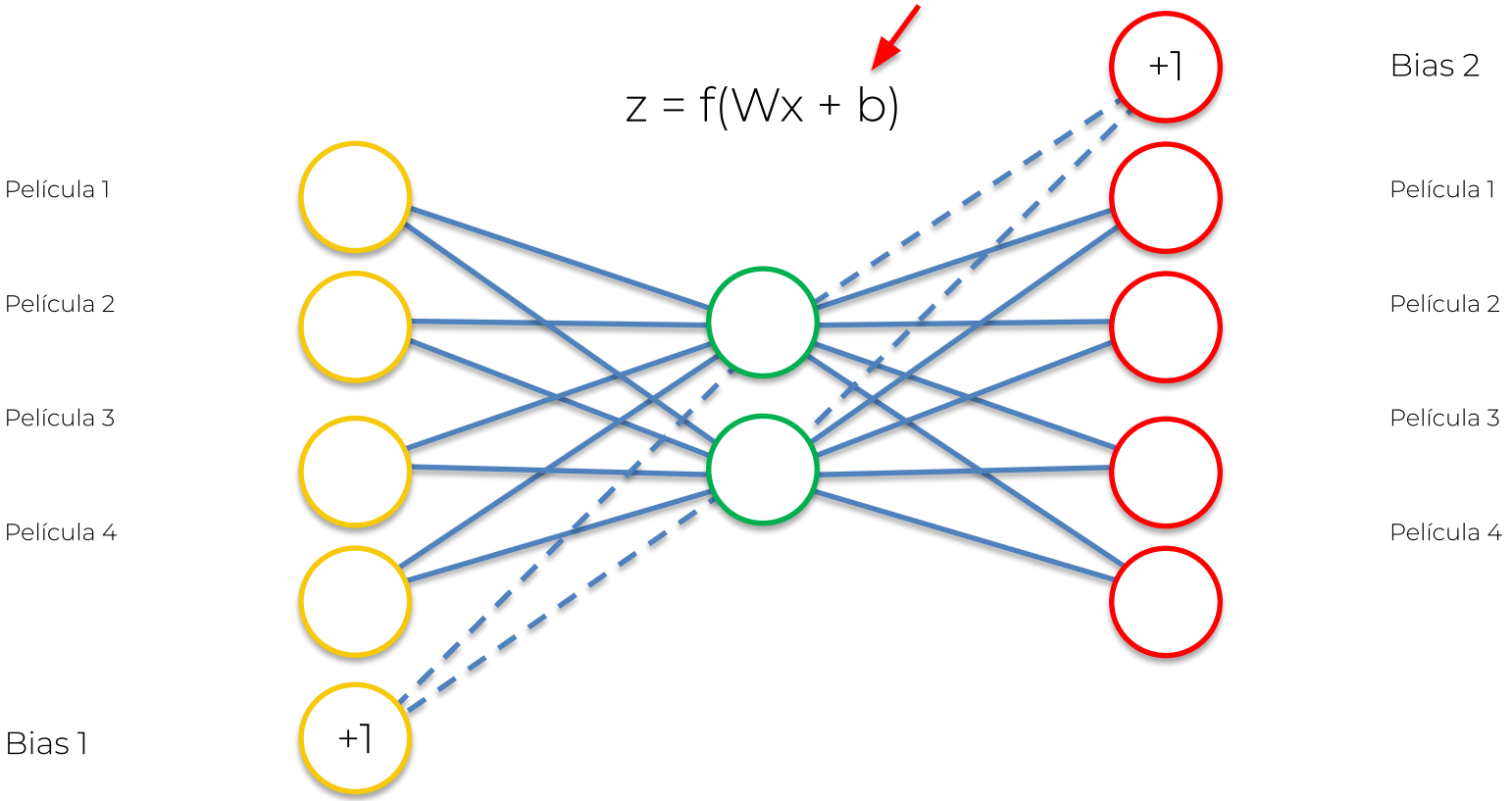


Link:

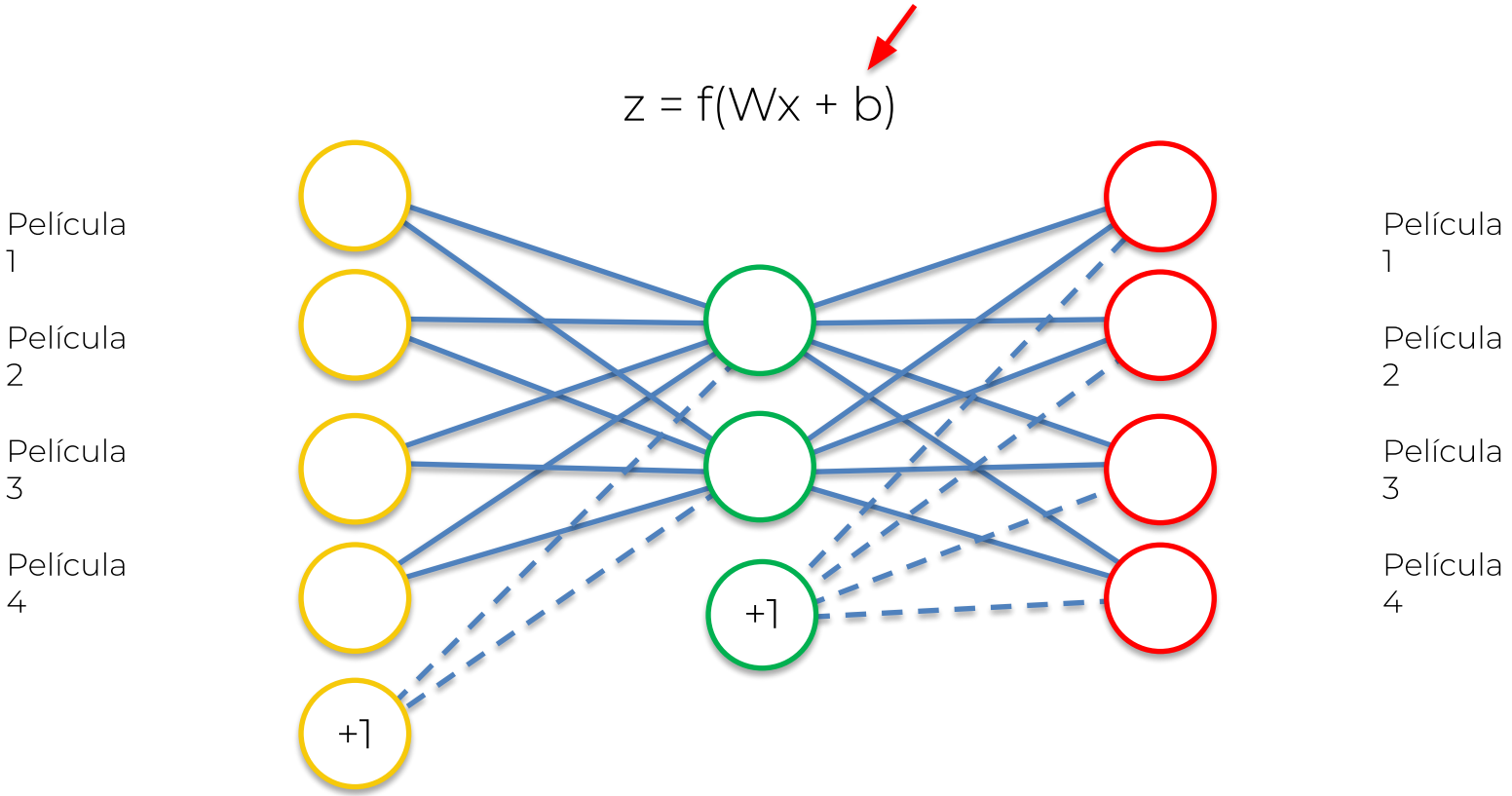
https://probablydance.com/2016/04/30/neural-networks-are-impressively-good-at-compression/

Acerca del término independiente (Bias)

Auto Encoders

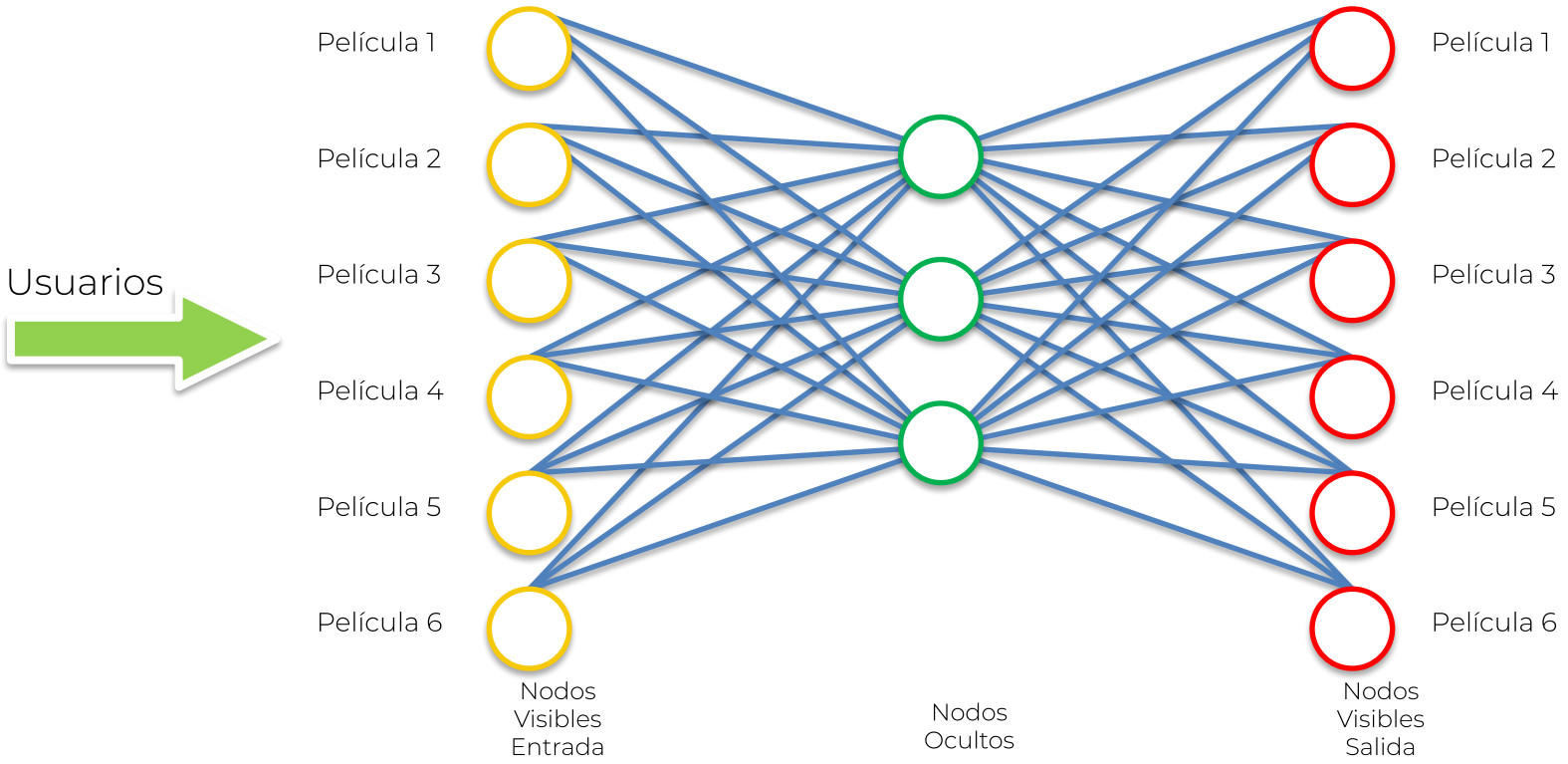


Auto Encoders



Cómo entrenar un Auto Encoder

Cómo entrenar un Auto Encoder



Cómo entrenar un Auto Encoder

PASO 1: Empezamos con un array donde las filas (observaciones) se corresponden a usuarios y las columnas (características) se corresponden a las películas. Cada celda (u, i) contiene la valoración (de 1 a 5, 0 si no hay valoración) de la película i valorada por el usuario u.

PASO 2: El primer usuario entra en la red. El vector de entrada x = (r_1, r_2, ..., r_m) tiene las valoraciones de todas las pelis.

PASO 3: El vector de entrada x se codifica en un vector z de menor dimensión con la función de transformación f (p.e: una función sigmoide):

$z = f(Wx + b)$ donde W es el vector de entrada de pesos y b el bias

PASO 4: z se descodifica en el vector de salida y de la misma dimensión que x para replicar el vector de entrada x.

PASO 5: Se calcula la reconstrucción del error d(x, y) = ||x-y|| con el objetivo de minimizarlo.

PASO 6: Back-Propagation: de derecha a izquierda se propaga el error. Los pesos se actualizan según su responsabilidad en el error. El ratio de error decide cuánto se actualizan dichos pesos.

PASO 7: Se repiten los pasos 1 a 6 y se actualizan los pesos tras cada observación (Reinforcement Learning). O:
Se repite los pasos 1 a 6 pero se actualizan los pesos solo tras un conjunto de observaciones(Batch Learning).

PASO 8: Cuando todo el conjunto de entrenamiento ha pasado por la RNA se completa un epoch. Repetir más veces.

Cómo entrenar un Auto Encoder

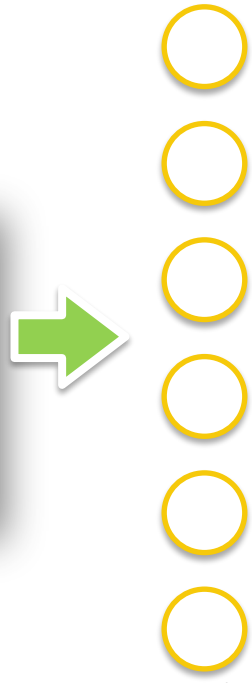
PASO 1

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

Cómo entrenar un Auto Encoder

PASO 2

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

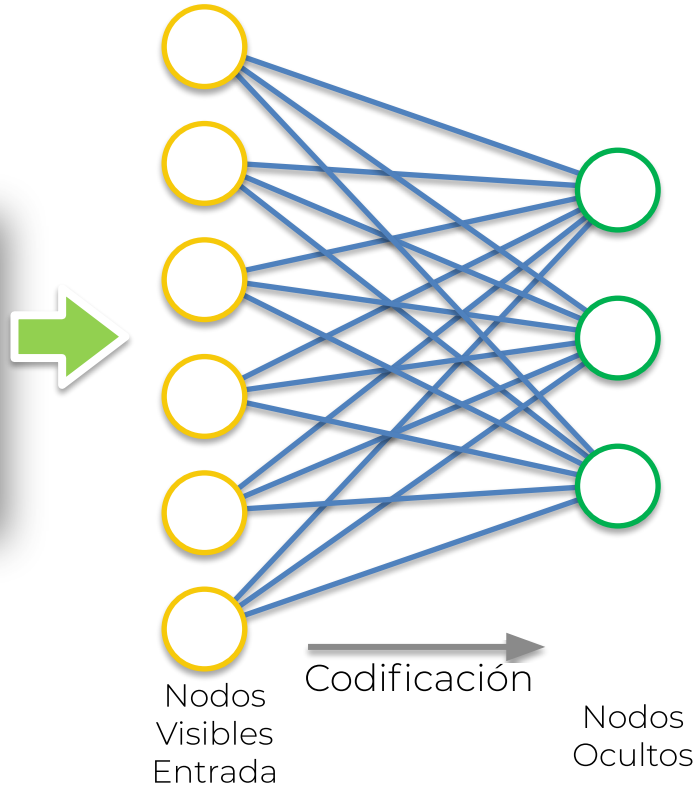


Nodos
Visibles
Entrada

Cómo entrenar un Auto Encoder

PASO 3

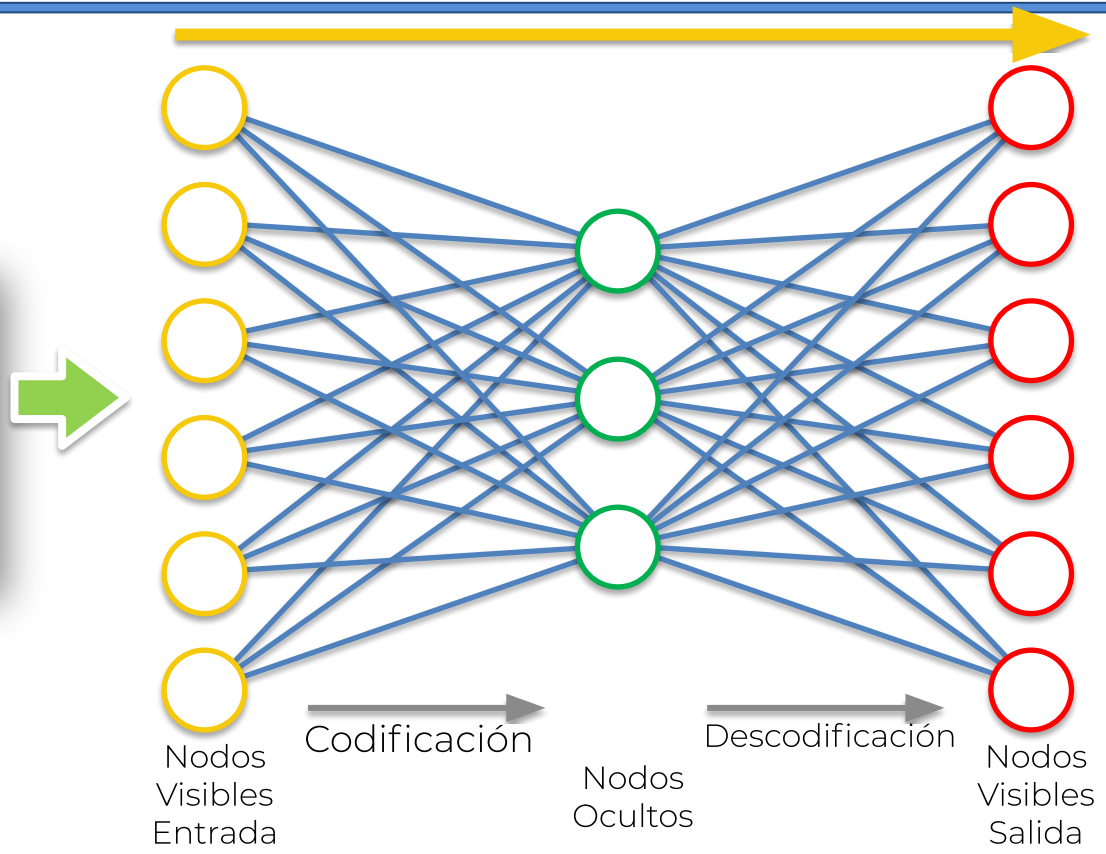
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



Cómo entrenar un Auto Encoder

PASO 4

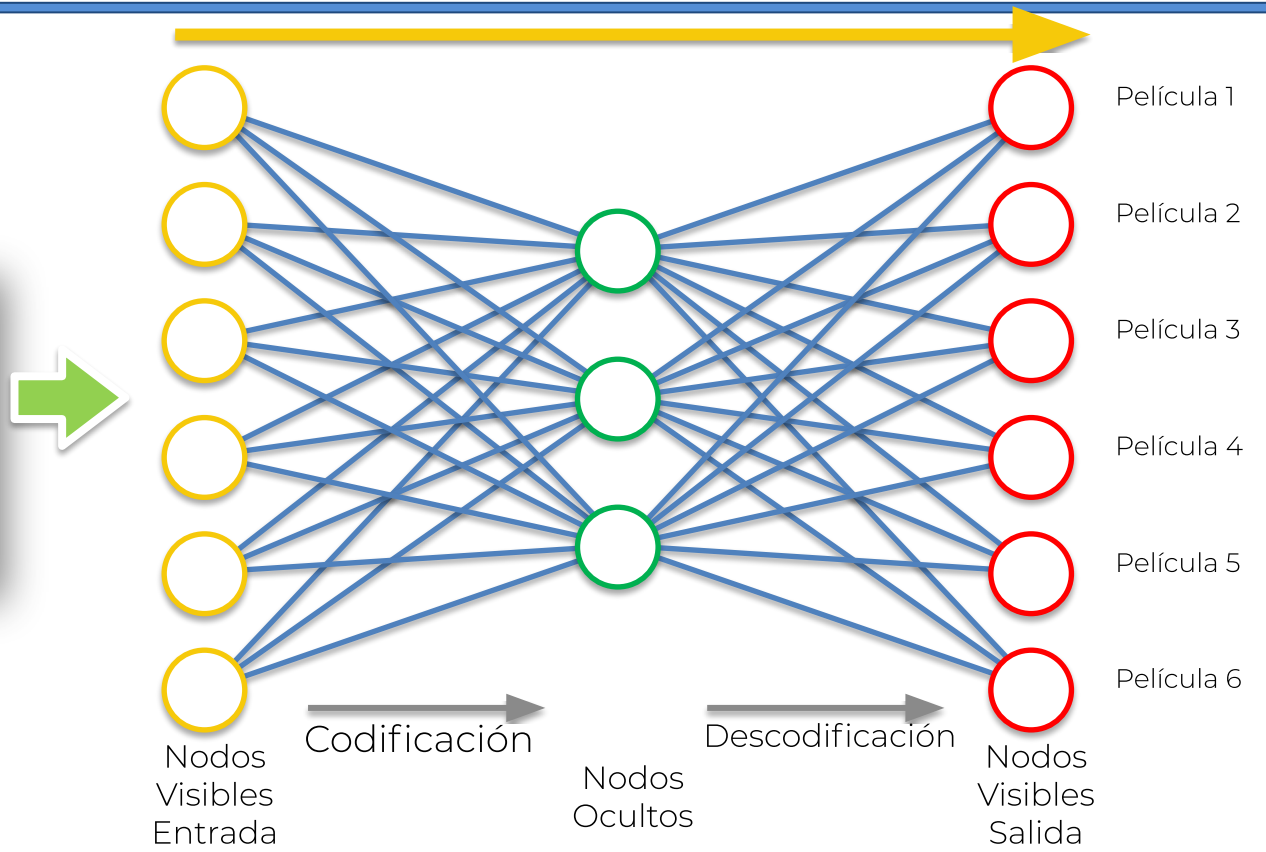
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



Cómo entrenar un Auto Encoder

PASO 5

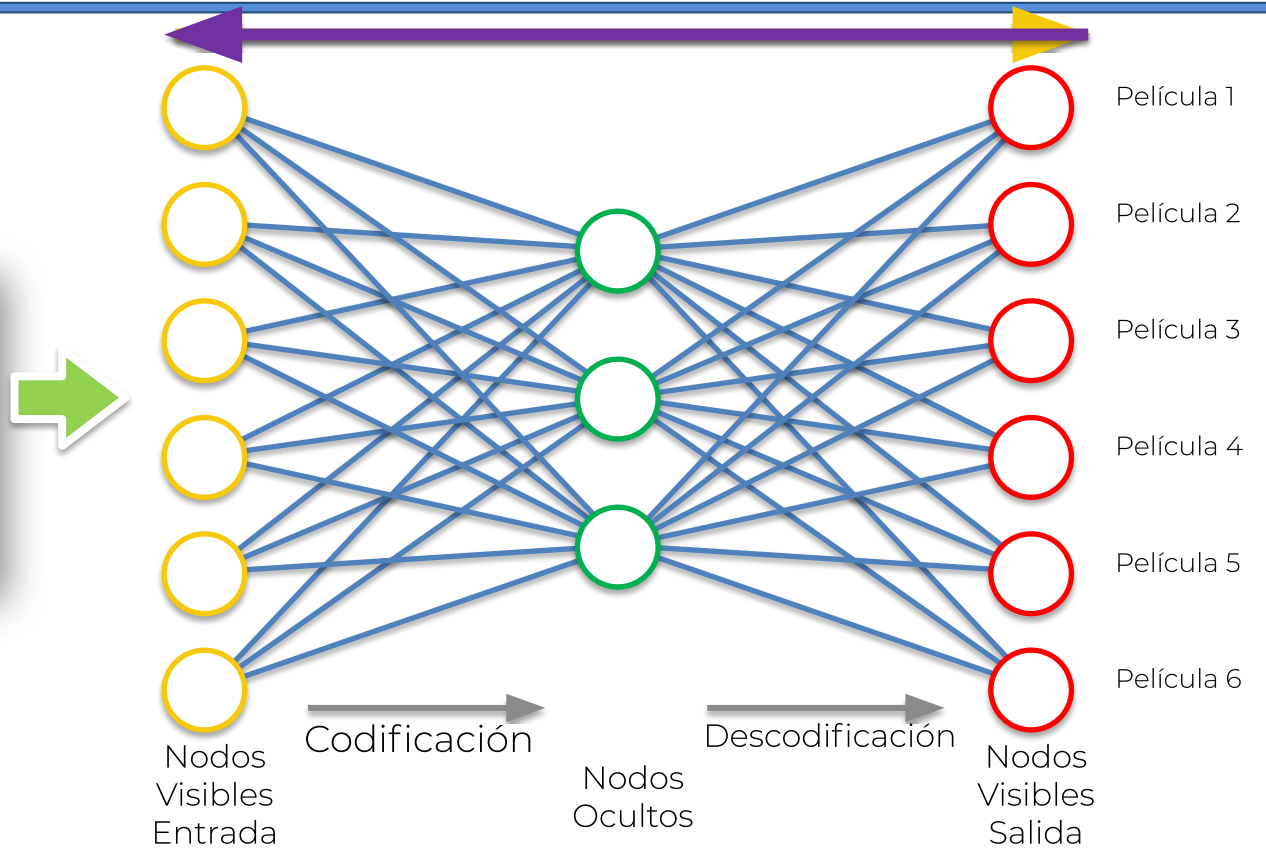
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



Cómo entrenar un Auto Encoder

PASO 6

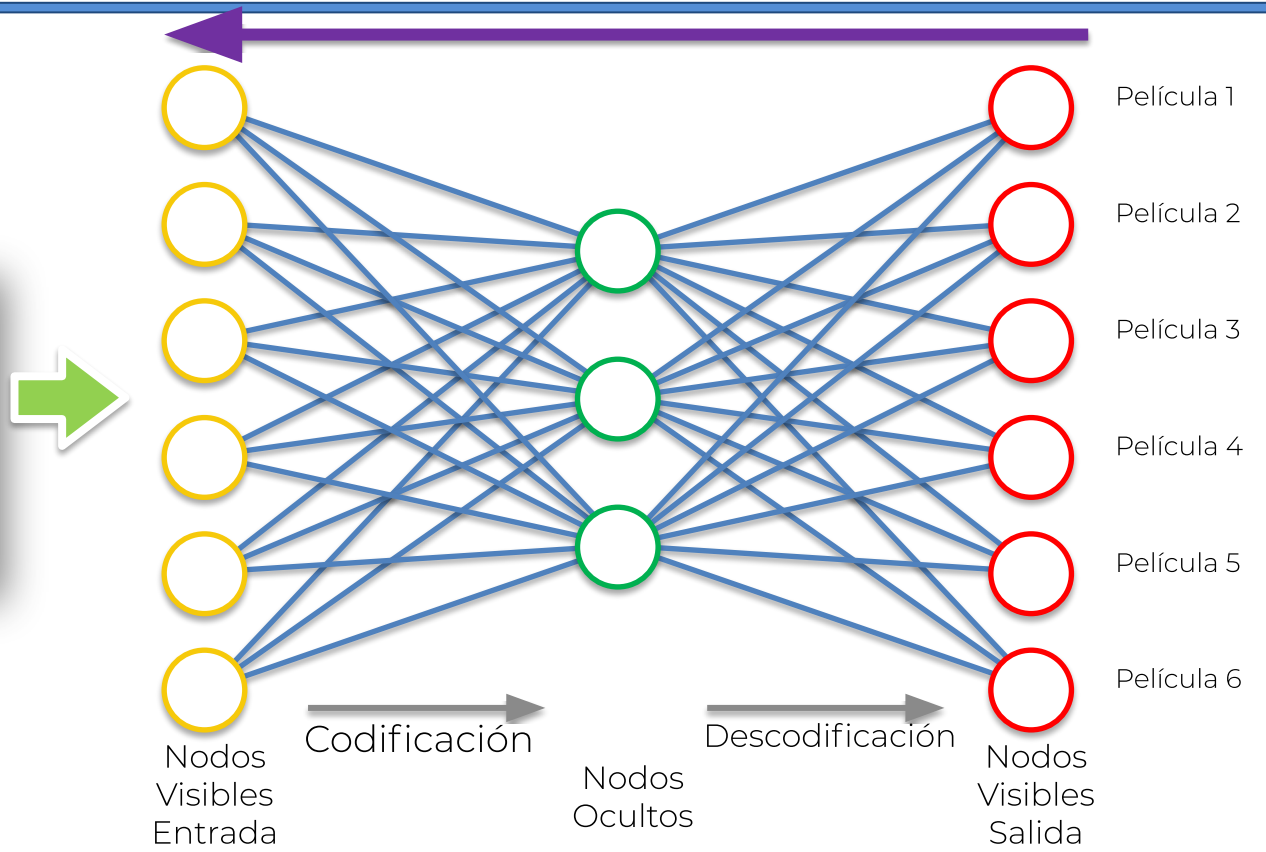
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



Cómo entrenar un Auto Encoder

PASO 7

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

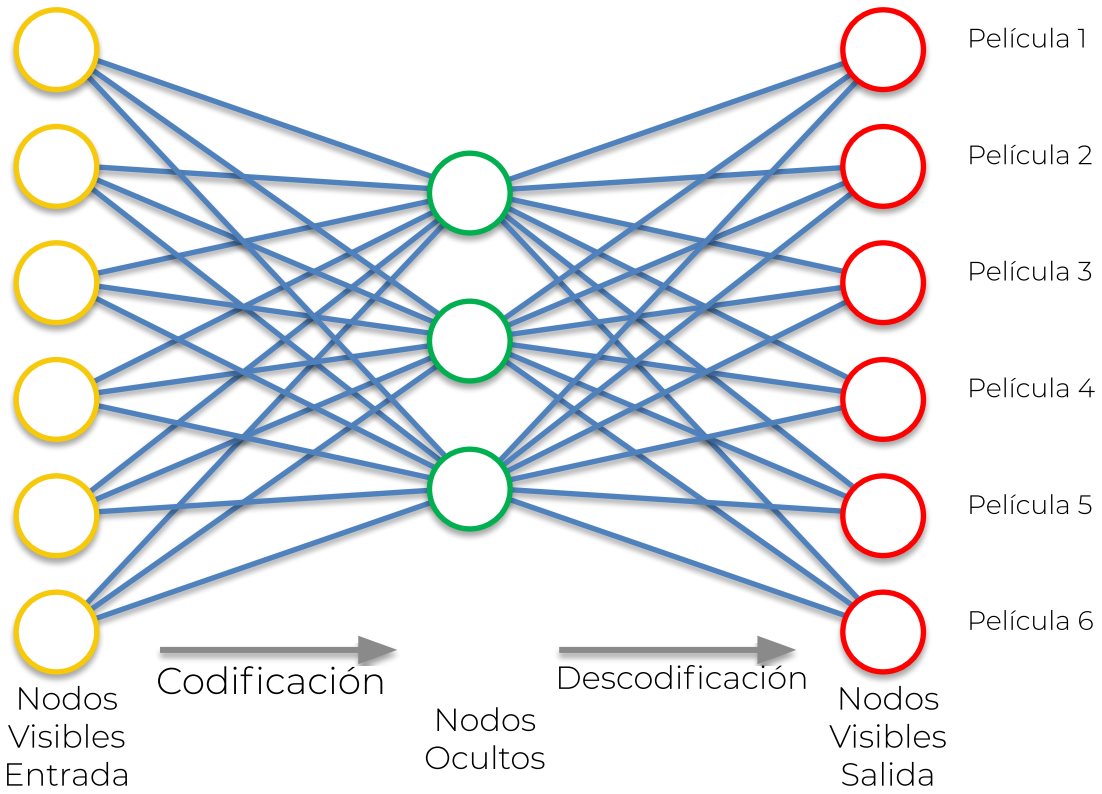


Cómo entrenar un Auto Encoder

PASO 8

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

Repetimos

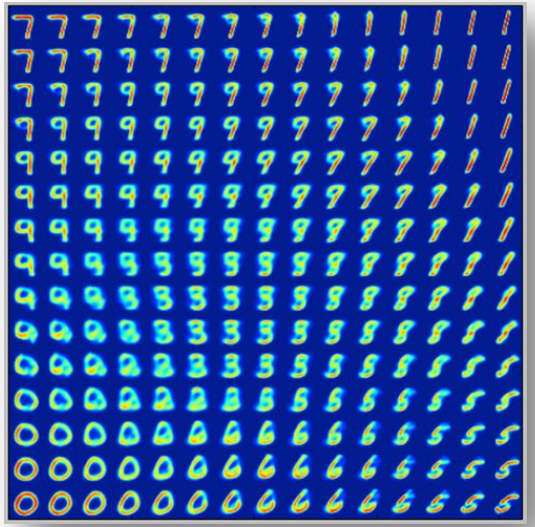


Cómo entrenar un Auto Encoder

Lectura adicional

Building Autoencoders in Keras

de Francois Chollet (2016)

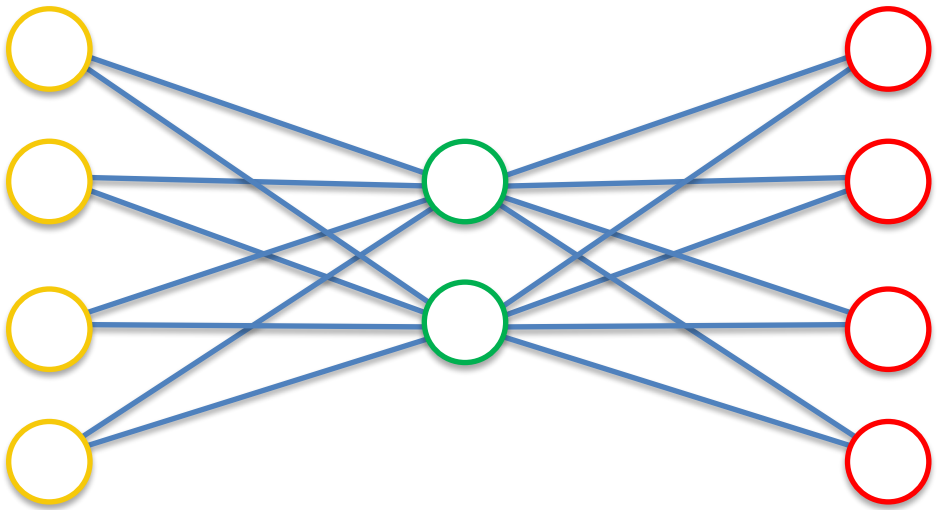


Link:

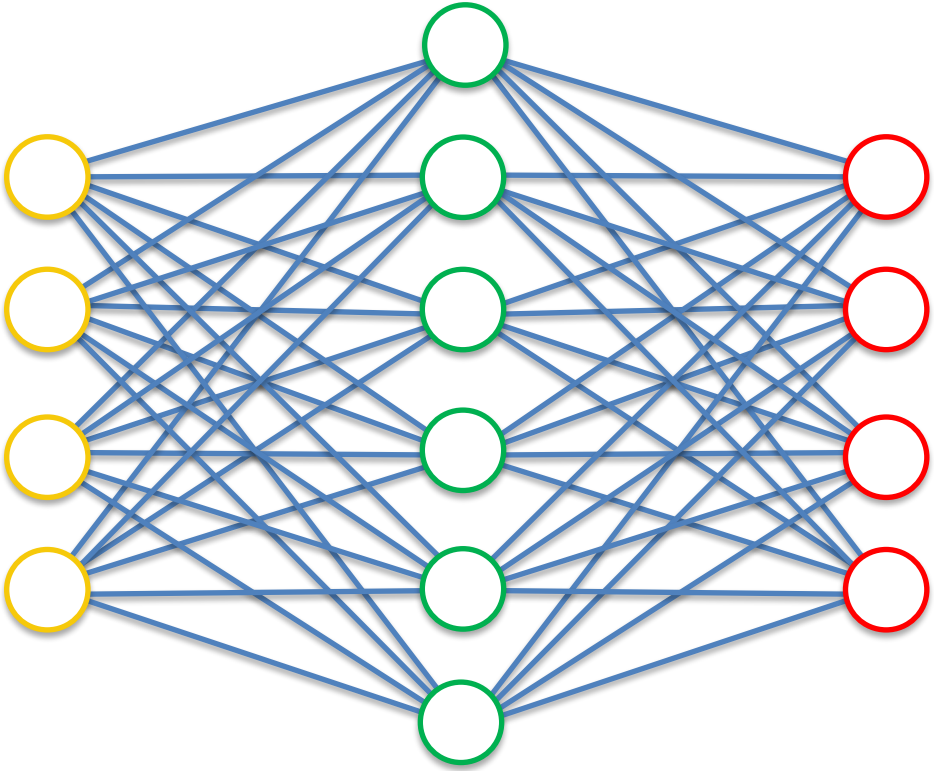
https://blog.keras.io/building-autoencoders-in-keras.html

Sobrecompletar Capas Ocultas

Sobrecompletar Capas Ocultas

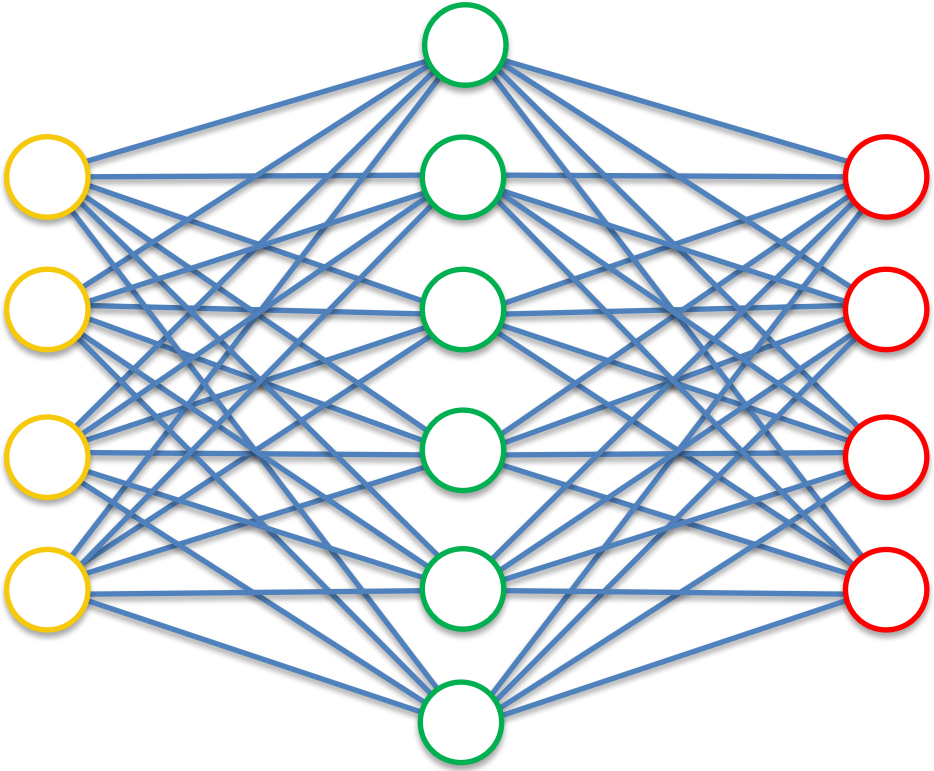


Sobrecompletar Capas Ocultas

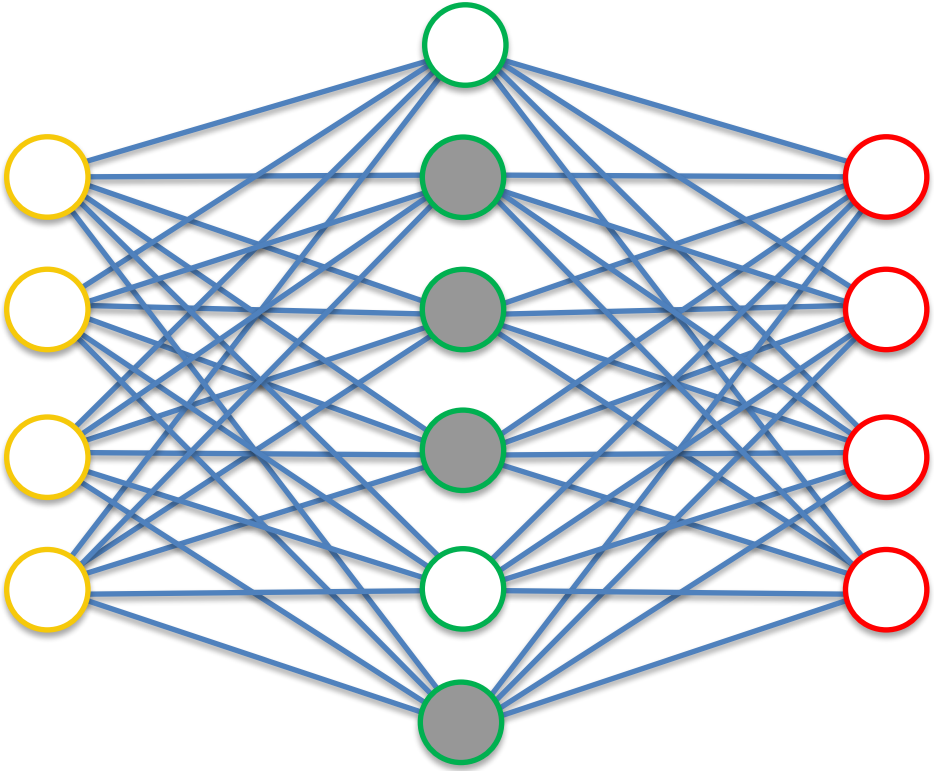


Sparse Autoencoders

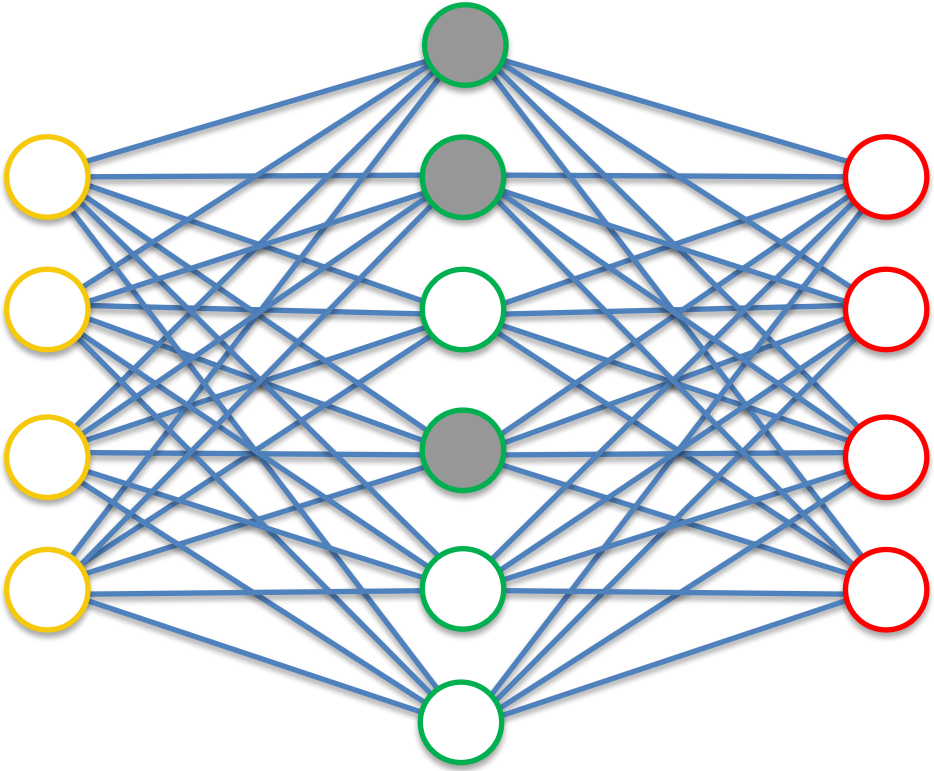
Sparse Autoencoders



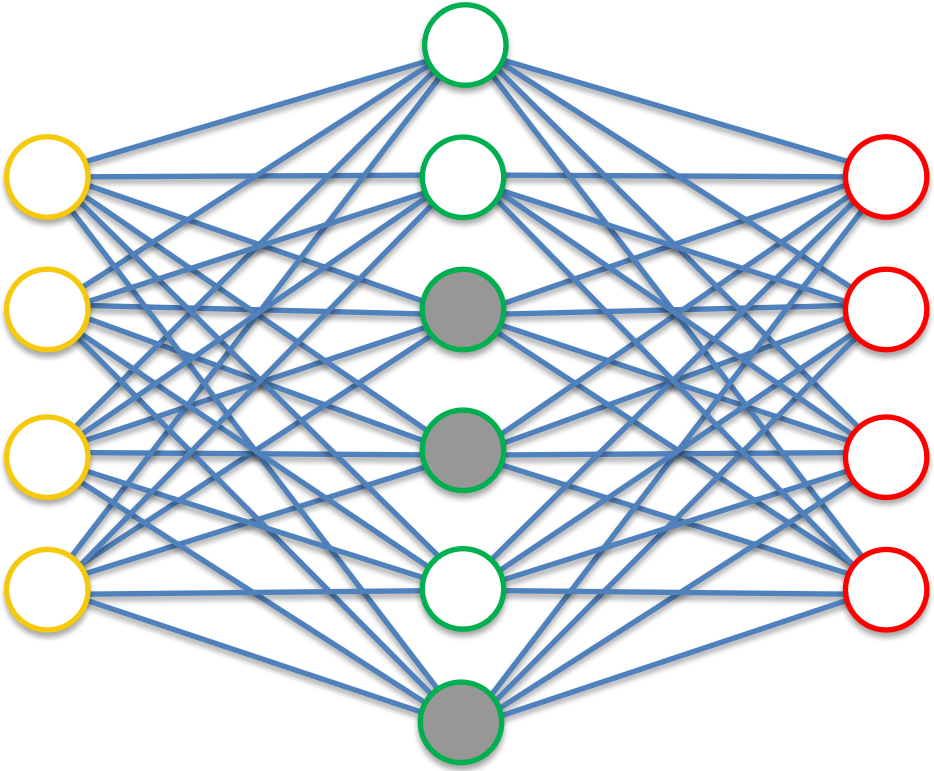
Sparse Autoencoders



Sparse Autoencoders



Sparse Autoencoders

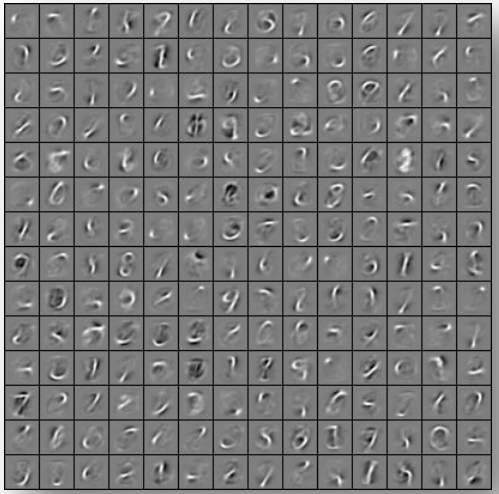


Sparse Autoencoders

Lectura adicional

Deep Learning Tutorial - Sparse Autoencoder

de Chris McCormick (2014)



Link:

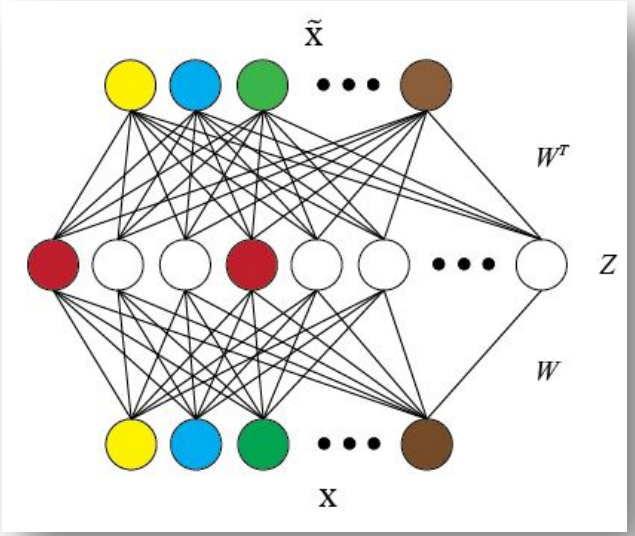
http://mccormickml.com/2014/05/30/deep-learning-tutorial-sparse-autoencoder/

Sparse Autoencoders

Lectura adicional

Deep Learning: Sparse Autoencoders

de Eric Wilkinson (2014)



Link:

http://www.ericlwilkinson.com/blog/2014/11/19/deep-learning-sparse-autoencoders

Sparse Autoencoders

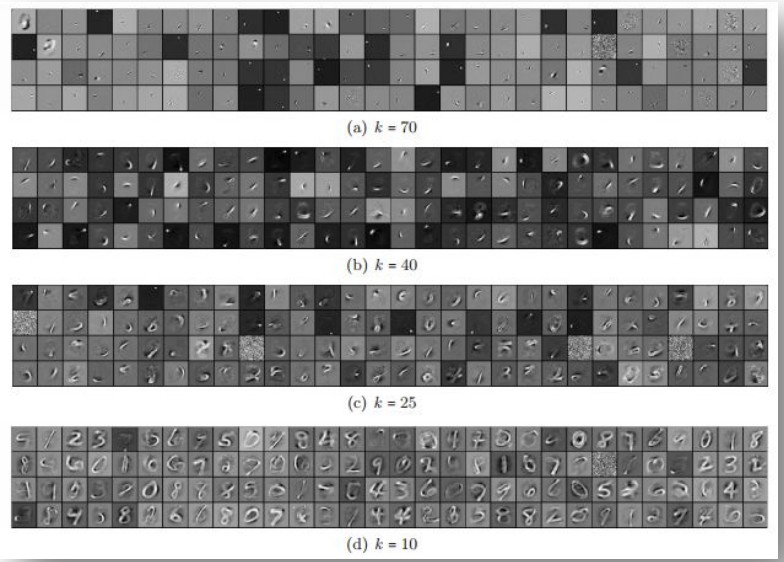
Lectura adicional

k-Sparse Autoencoders

de Alireza Makhzani et al. (2014)

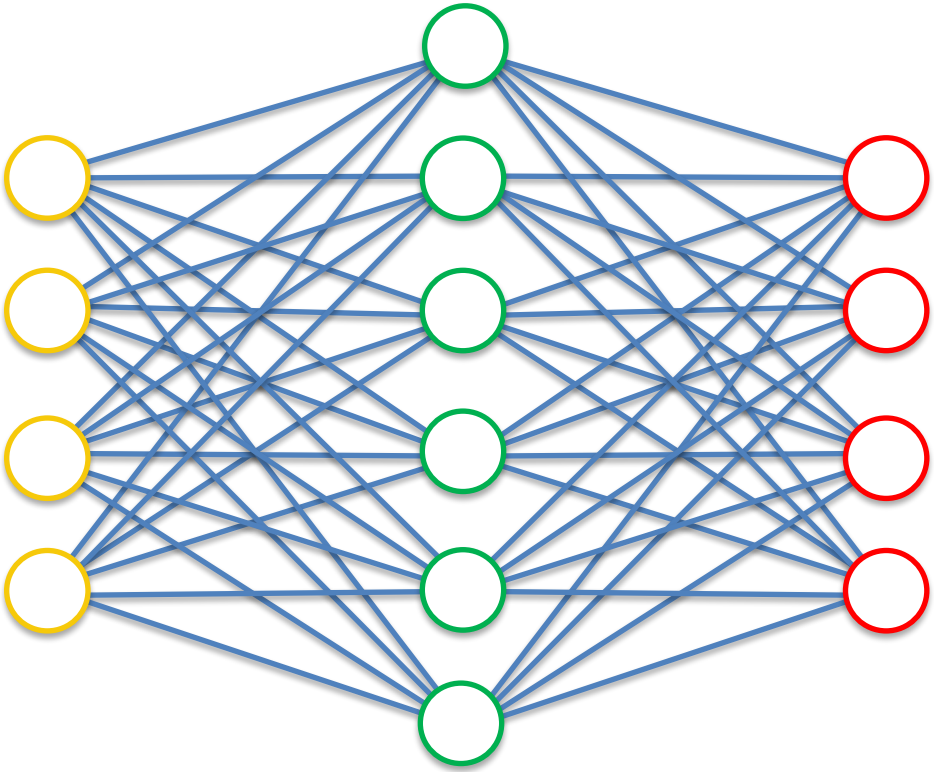
Link:

https://arxiv.org/pdf/1312.5663.pdf

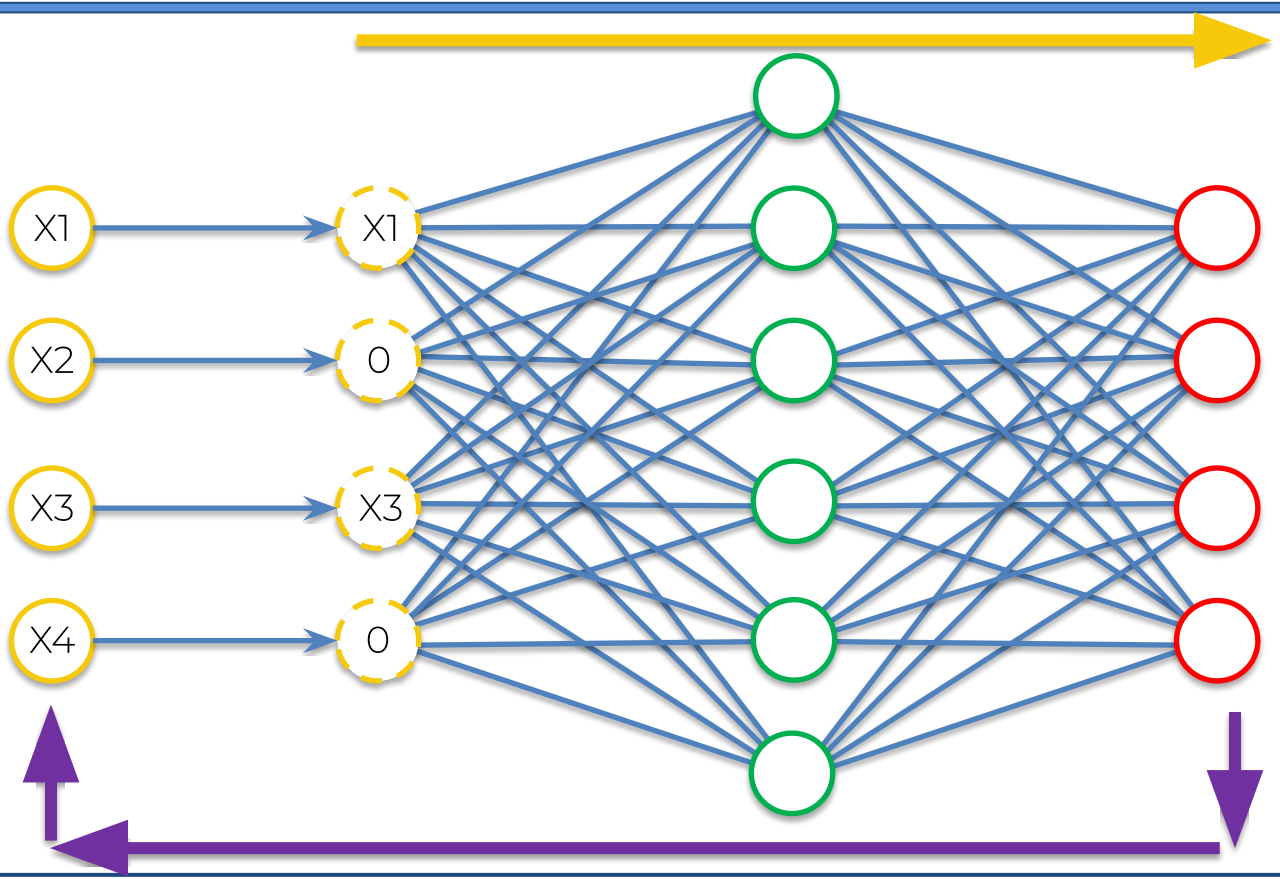


Denoising Autoencoders

Denoising Autoencoders



Denoising Autoencoders

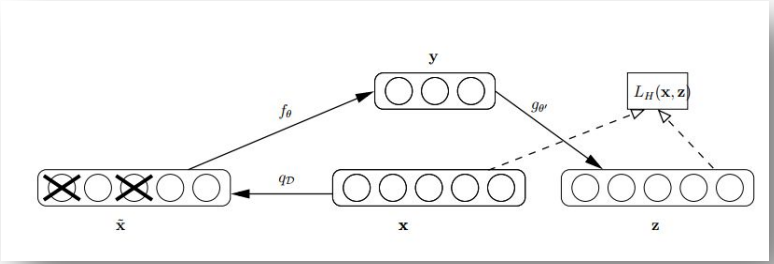


Denoising Autoencoders

Lectura adicional

Extracting and Composing Robust Features with Denoising Autoencoders

de Pascal Vincent et al. (2008)

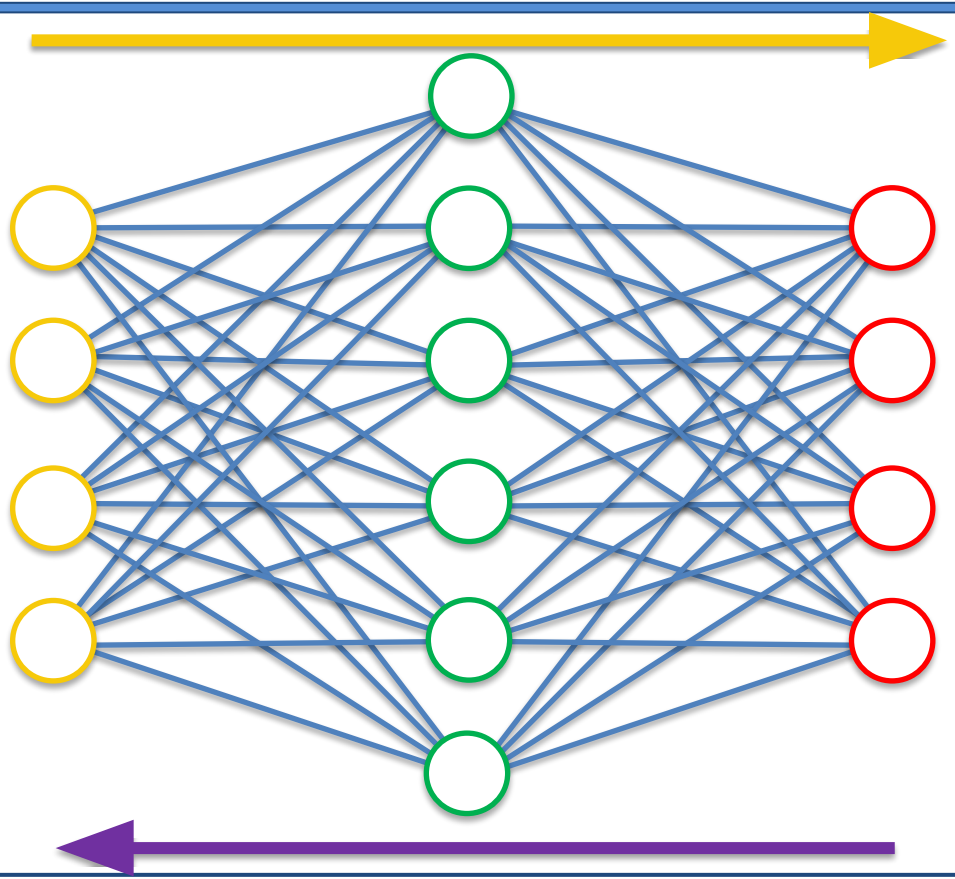


Link:

http://www.cs.toronto.edu/~larocheh/publications/icml-2008-denoising-autoencoders.pdf

Contractive Autoencoders

Contractive Autoencoders

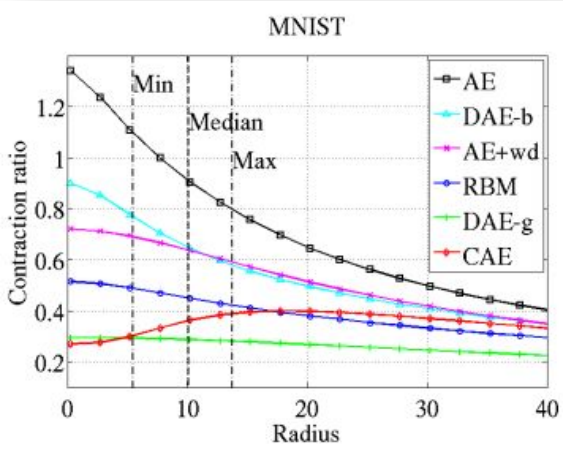


Contractive Autoencoders

Lectura adicional

Contractive Auto-Encoders: Explicit Invariance During Feature Extraction

de Salah Rifai et al. (2011)

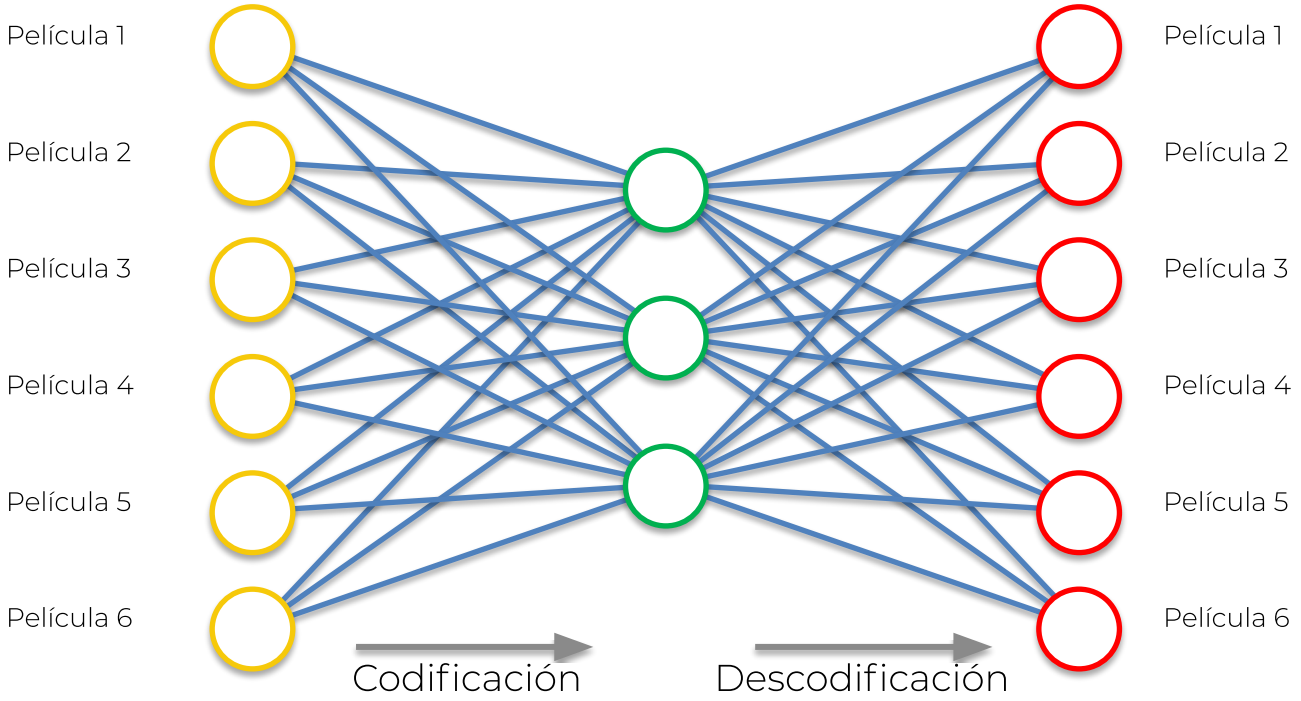


Link:

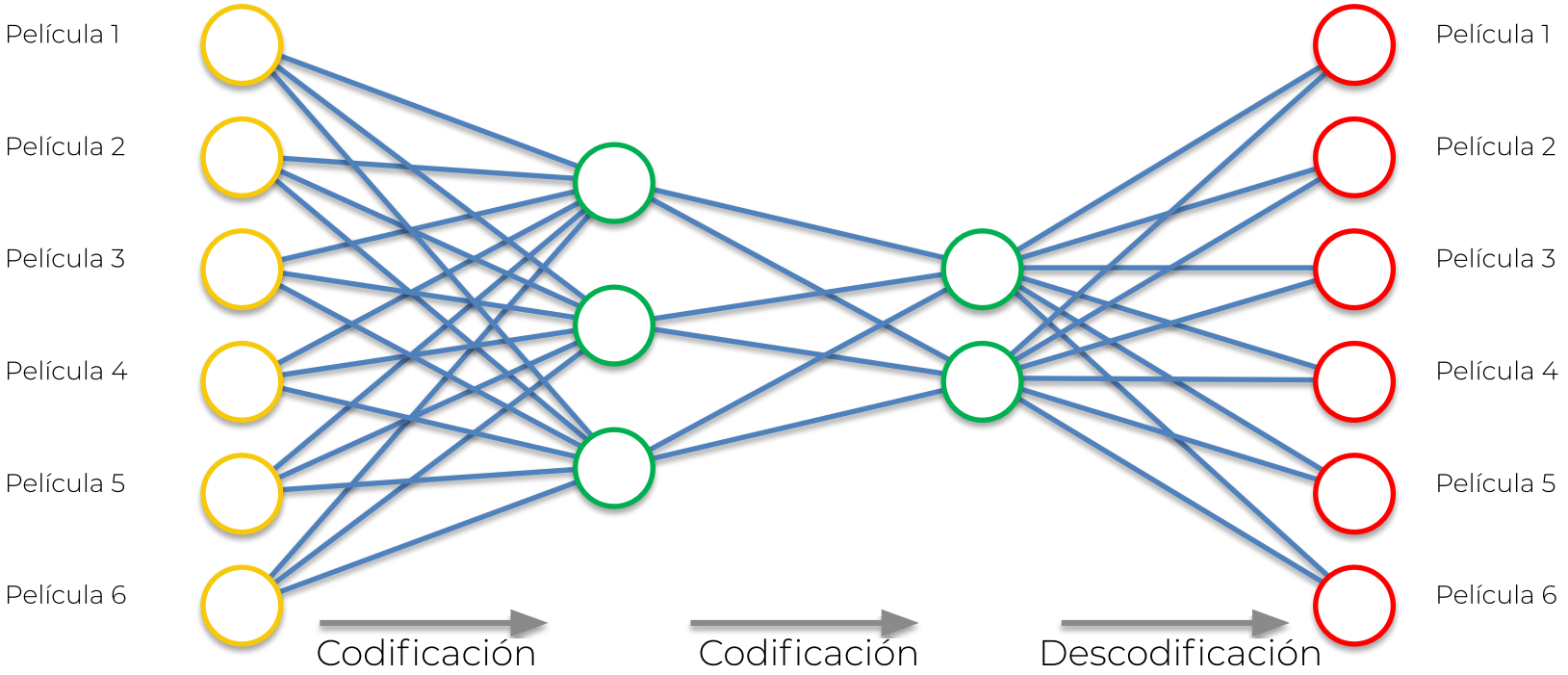
http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Rifai_455.pdf

Stacked Autoencoders

Stacked Autoencoders



Stacked Autoencoders

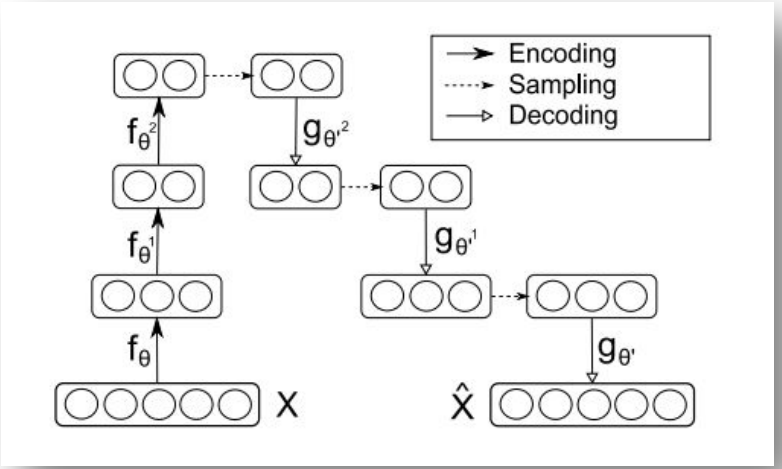


Stacked Autoencoders

Lectura adicional

Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion

de Pascal Vincent et al. (2010)



Link:

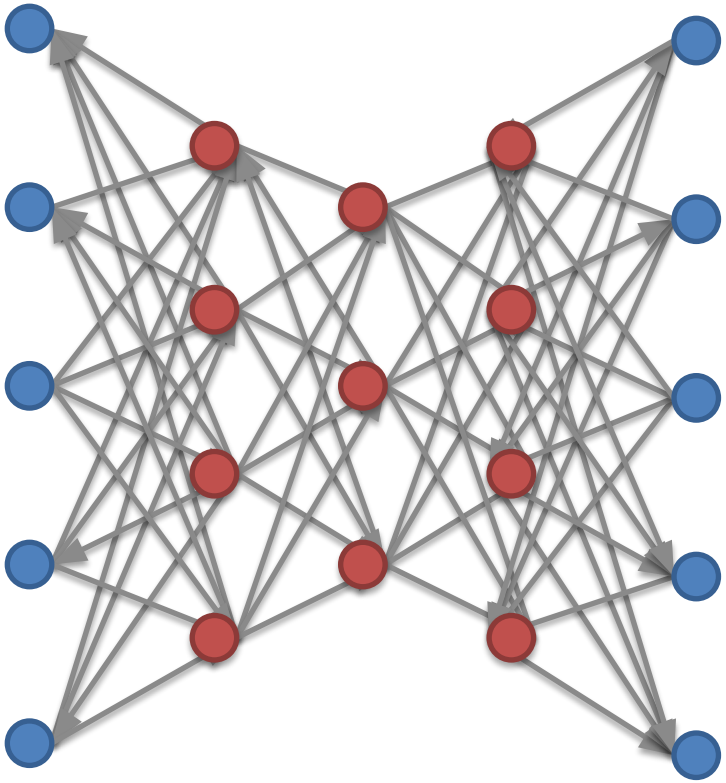
http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf

Deep Autoencoders

Deep Autoencoders

Stacked AE <> Deep AE

Deep Autoencoders

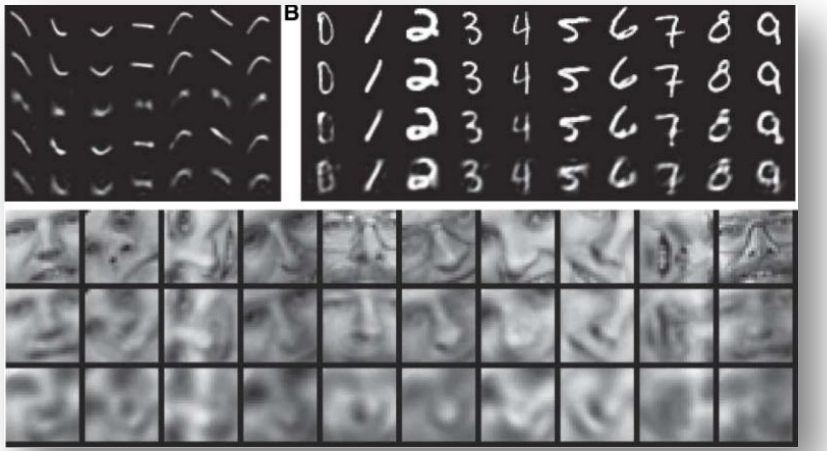


Deep Autoencoders

Lectura adicional

Reducing the Dimensionality of Data with Neural Networks

de Geoffrey Hinton et al. (2006)



Link:

https://www.cs.toronto.edu/~hinton/science.pdf

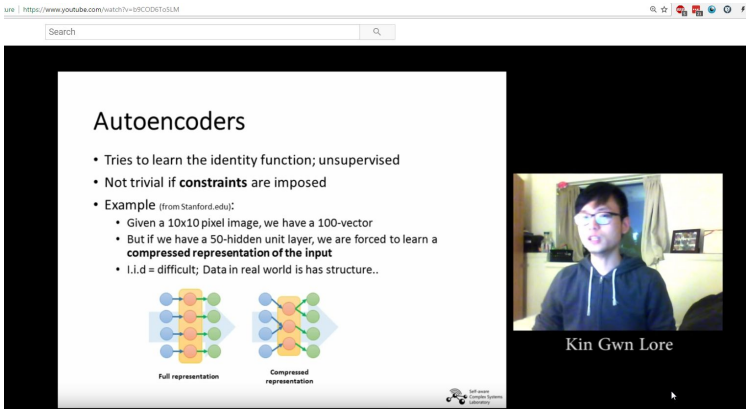
EXTRA STUFF

********** EXTRA STUFF

https://www.youtube.com/watch?v=b9COD6To5LM

Unsupervised -> only the training data is available without any labels

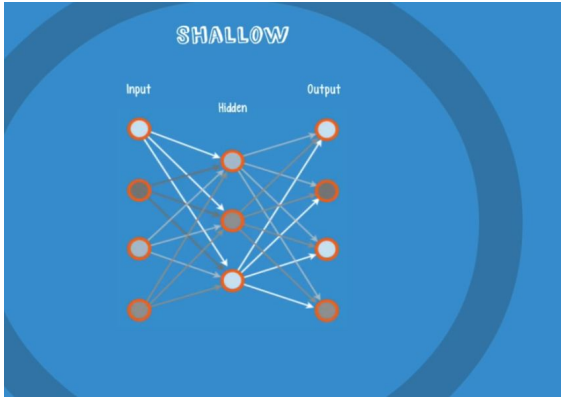
“By restricting the number of hidden units we are forcing the network to learn a compressed representation of the input”



********** EXTRA STUFF

https://www.youtube.com/watch?v=s96mYcicbpE

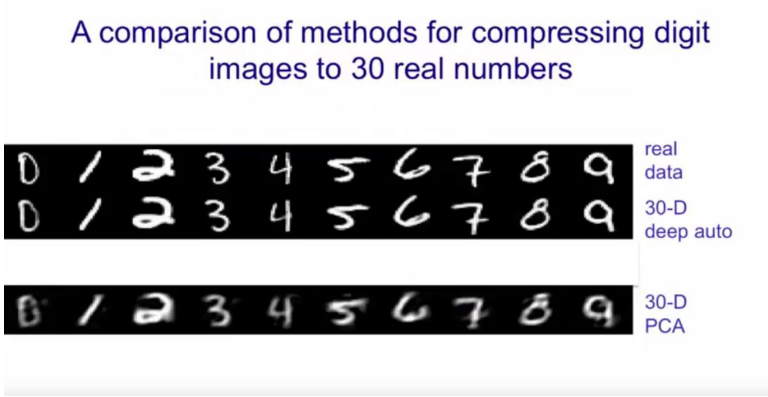
~1m 10s – Wooow! “an rbm is an example of an autoencoder with only 2 layers”
 Debatable…



********** EXTRA STUFF

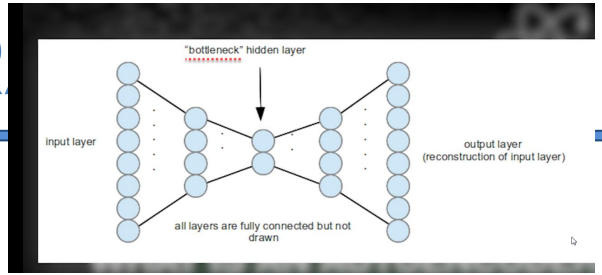
https://www.youtube.com/watch?v=_Ex1Ur85AVs

~3:30 DAE vs PCA



********** EXTR

https://www.youtube.com/watch?v=Rdpbnd0pCiI



********** EXTRA STUFF

https://blog.keras.io/building-autoencoders-in-keras.html

The Keras Blog

Keras is a Deep Learning library for Python, that is simple, modular, and extensible.

Archives Github Documentation Google Group

Building Autoencoders in Keras

In this tutorial, we will answer some common questions about autoencoders, and we will cover code examples of the following models:

- a simple autoencoder based on a fully-connected layer
- a sparse autoencoder
- a deep fully-connected autoencoder
- a deep convolutional autoencoder
- an image denoising model
- a sequence-to-sequence autoencoder
- a variational autoencoder

Sat 14 May 2016
By François Chollet
In Tutorials.

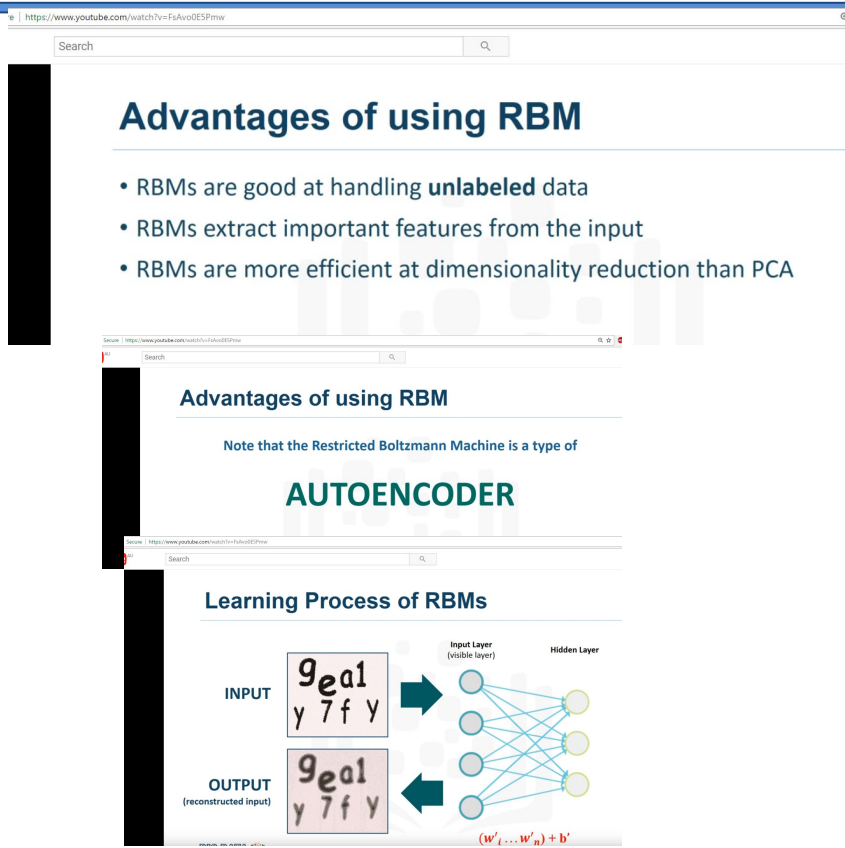
Note: all code examples have been updated to the Keras 2.0 API on March 14, 2017. You will need Keras version 2.0.0 or higher to run them.

What are autoencoders?

"Autoencoding" is a data compression algorithm where the compression and decompression functions are 1) data-specific, 2) lossy, and 3) *learned automatically from examples* rather than engineered by a human. Additionally, in almost all contexts where the term "autoencoder" is used, the compression and decompression functions are implemented with neural networks.

********** EXTRA STUFF

https://www.youtube.com/watch?v=FsAvo0E5Pmw

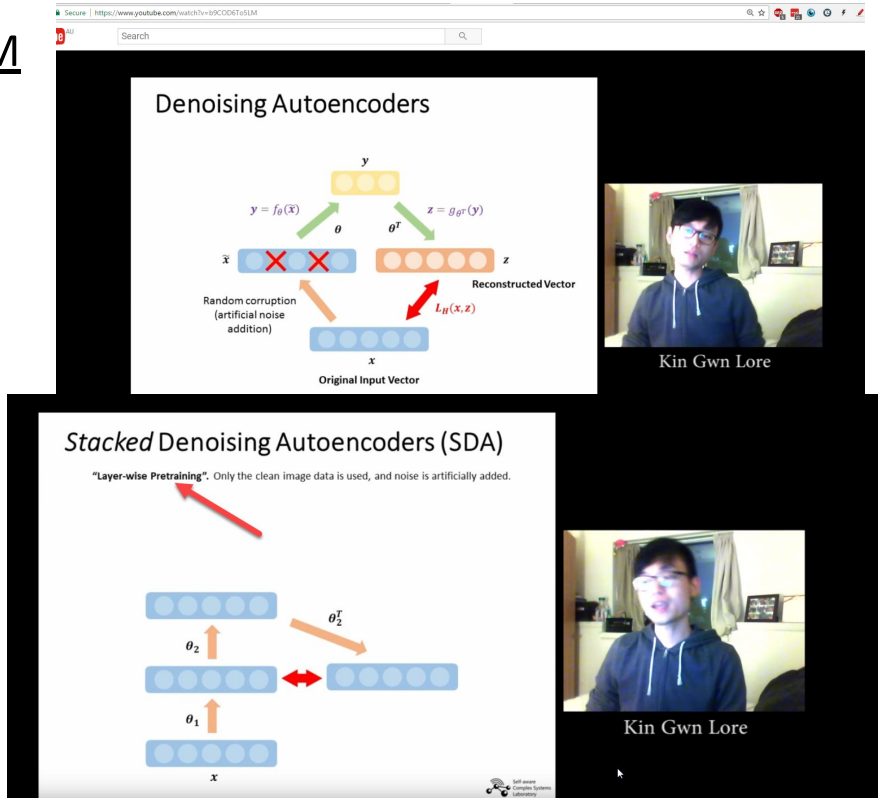


********** EXTRA STUFF

https://www.youtube.com/watch?v=b9COD6To5LM

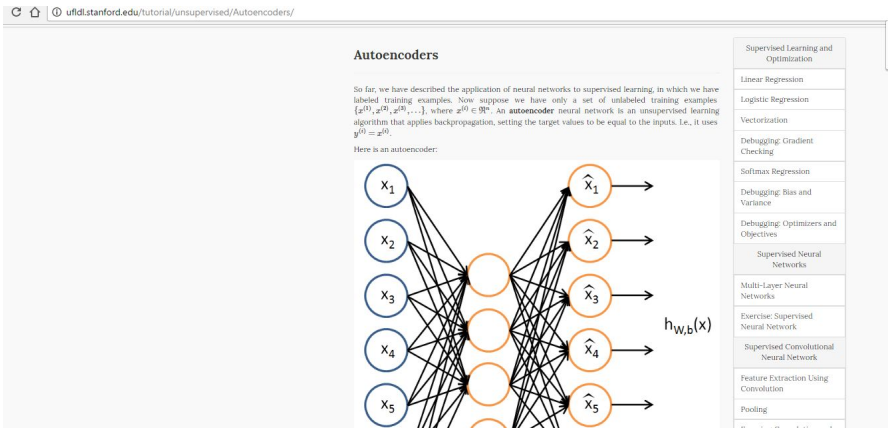
Denoising autoencoders ~2:30
“Randomly corrupt the input data”

Layer-wise pre-training ~3:40



********** EXTRA STUFF

http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

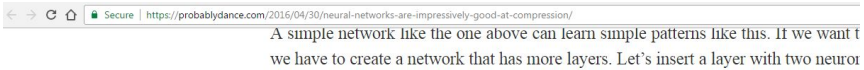


If the input were completely random—say, each xixi comes from an IID Gaussian independent of the other features—then this compression task would be very difficult. But if there is structure in the data, for example, if some of the input features are correlated, then this algorithm will be able to discover some of those correlations.

BTW: If you are using a tanh activation function, then we think of a neuron as being inactive when it outputs values close to -1.

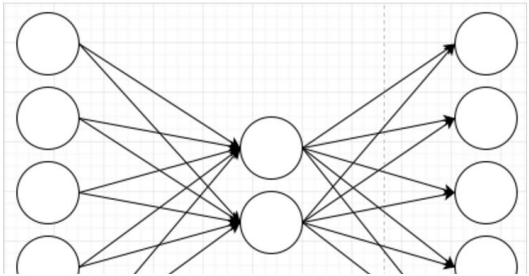
Sparsity: Our argument above relied on the number of hidden units s2s2 being small. But even when the number of hidden units is large (perhaps even greater than the number of input pixels), we can still discover interesting structure, by imposing other constraints on the network. In particular, if we impose a "'sparsity"' constraint on the hidden units, then the autoencoder will still discover interesting structure in the data, even if the number of hidden units is large. <<! Find out more info about this.

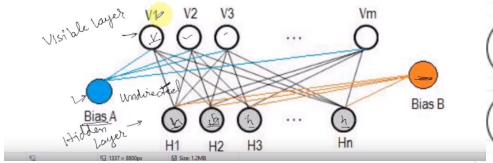
********** EXTRA STUFF

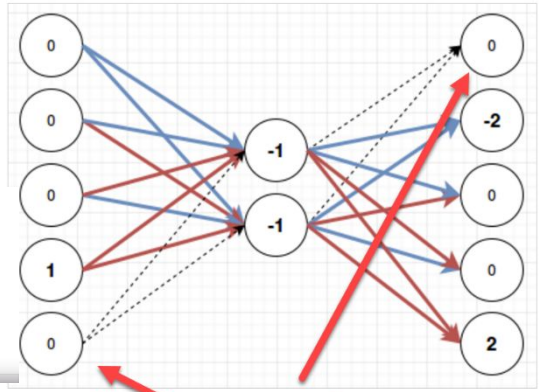


https://probablydance.com/2016/04/30/neural-networks-are-impressively-good-at-compression/

(not obvious, start with this): If you count the number of connections on that last picture you will notice that there are fewer than there were in the first network. There were 5*5=25 at first, now there are 5*2*2=20. That reduction would be larger if I had more input and output nodes. For 10 nodes that reduction would be from 100 connections down to 40 when inserting two nodes in the middle.







********** EXTRA STUFF

https://probablydance.com/2016/04/30/neural-networks-are-impressively-good-at-compression/
Interesting:

1. Explanation why for Tanh(): -1 =non-active and 1= active: because after that a softmax function is used and it's e to the power of something. "Also by using an exponent I get no negative numbers. 2^-96 is just a number that's very close to zero. " -> we need to use softmax

2. Explanation why we need Tanh(): "In fact since I just used linear math in my explanation above and didn't need to run tanh() on anything, couldn't softmax have learned those weighs even without using a tanh layer? In theory it could, but in practice it will keep on bumping up the numbers to get small improvements and you will very quickly overflow floating point numbers. The tanh layer in the middle is then just responsible for keeping the numbers small: it clamps its outputs to the range -1 to 1, and the inputs won't grow past a certain point either because at some point a bigger number just means that the output goes from 0.995 to 0.997. But since they can usually still get a small improvement you don't lose that nice property of softmax where neurons can keep on edging out small improvements over each other." -> we need Tanh()

Summary: "So if you just use softmax your weights will explode and if you just use tanh your weights will stagnate too soon before a good solution emerges. If you use both you get nice greedy behavior where the connections keep on looking for better solutions, but you also keep the weights from exploding." -> we need both!

********** EXTRA STUFF

https://probablydance.com/2016/04/30/neural-networks-are-impressively-good-at-compression/
Interesting:

Now all I've shown is that my network can learn the rule that when input n fires, it needs to fire output n+1. It should be easy to see that just by creating a different permutation of the weights of the connections I can teach my network that for any input neuron x, it should fire any other output neuron y. The impressive part is that without the middle layer I would have needed 11*11 = 121 connections to have a network that can learn any combination, and with that middle layer I can do it in only 11*2*2 = 44 connections.

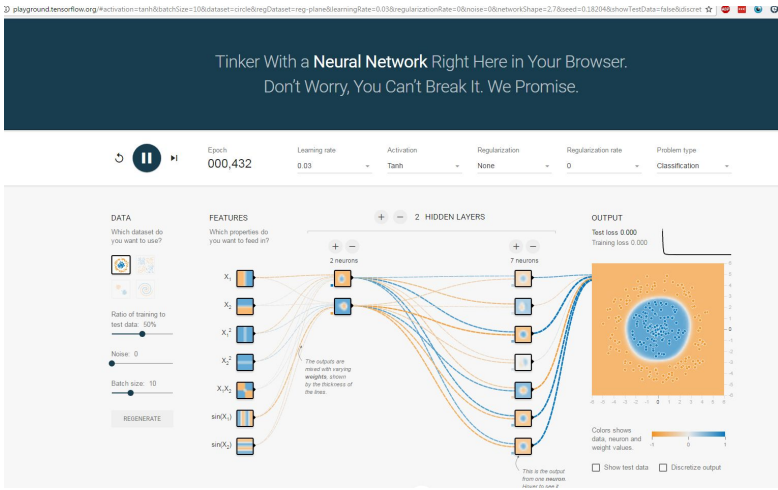
Eleven inputs and outputs is close to the limit for what you can do with two nodes in the middle. If you ask it to do much more the weights will fluctuate and they will keep on PASOping on each others toes. But with three nodes in the middle I can learn these patterns for something like thirty inputs and outputs, and with four nodes, I can learn direct connections for more than a hundred inputs and outputs. It doesn't grow linearly because the number of combinations doesn't grow linearly.

But backprop DOES scale with the #of connections. So we win! "Luckily the back propagation algorithm scales with the number of connections, not with the number of combinations. So with a linear increase in computation cost I get a super-linear increase in the amount of stuff I can learn."

********** EXTRA STUFF

Understand how to use this

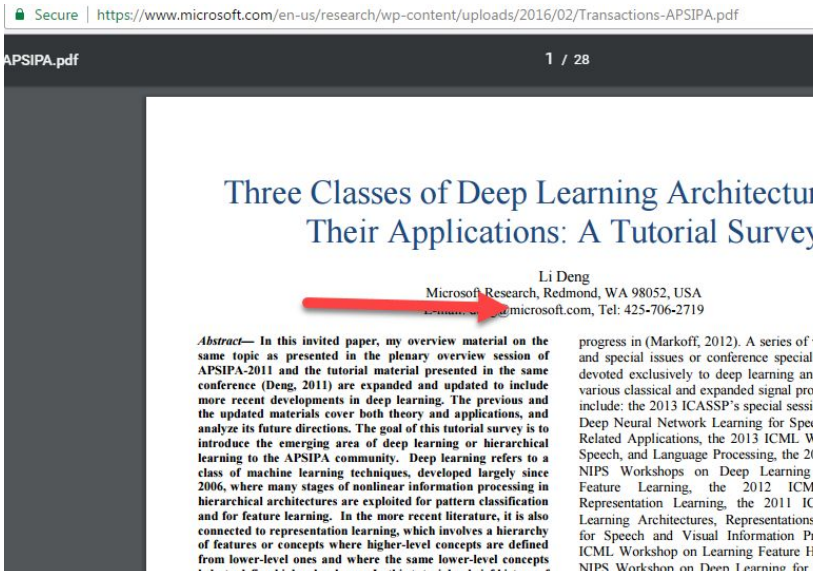
http://playground.tensorflow.org



********** EXTRA STUFF

https://www.microsoft.com/en-us/research/publication/three-classes-of-deep-learning-architectures-and-their-applications-a-tutorial-survey/?from=http%3A%2F%2Fresearch.microsoft.com%2Fpubs%2F192937%2Ftransactions-apsipa.pdf

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Transactions-APSIPA.pdf



********** EXTRA STUFF

AE vs RBM

https://www.quora.com/What-is-the-difference-between-Deep-Boltzmann-Machine-Deep-Belief-Networks-and-Deep-Auto-Encoders-Is-there-is-tutorial-which-explains-the-difference-between-procedures-for-pre-training-and-fine-tuning-the-above-networks-pseudo-code-would-be-of-additional-help