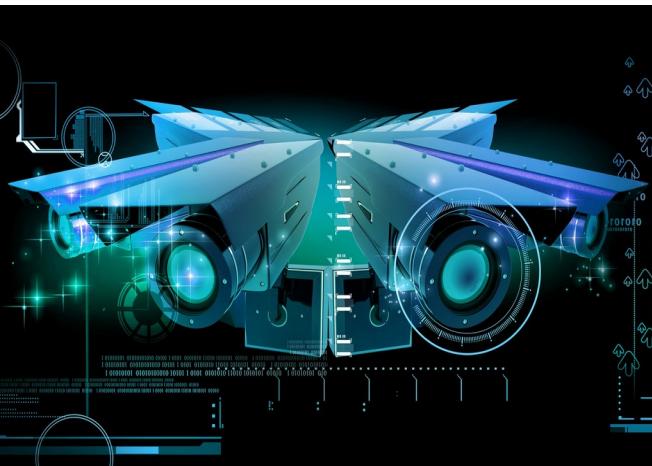
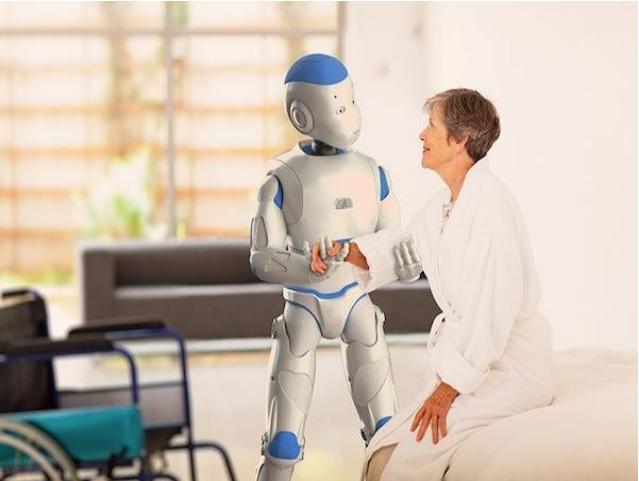
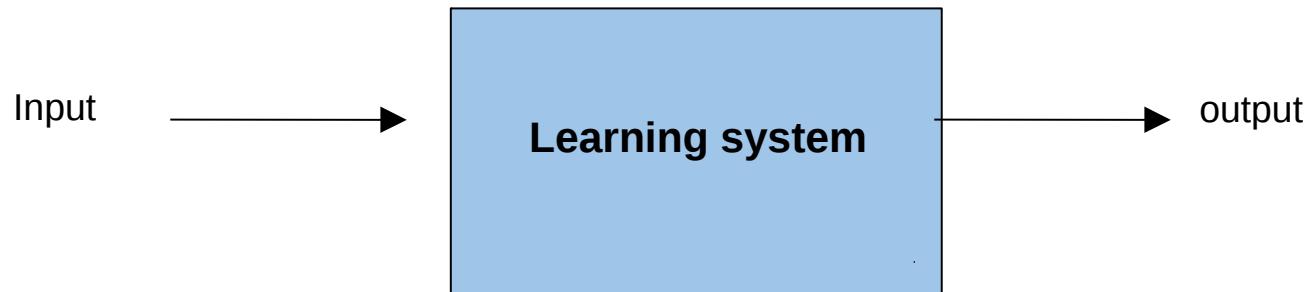
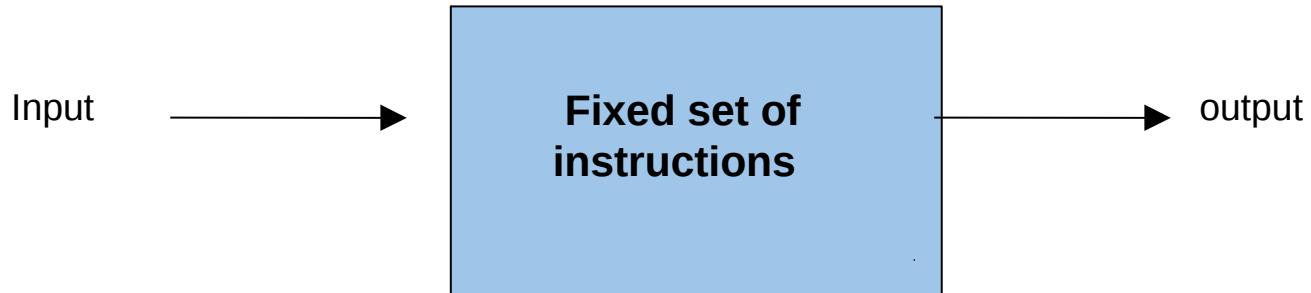


# Automation is future

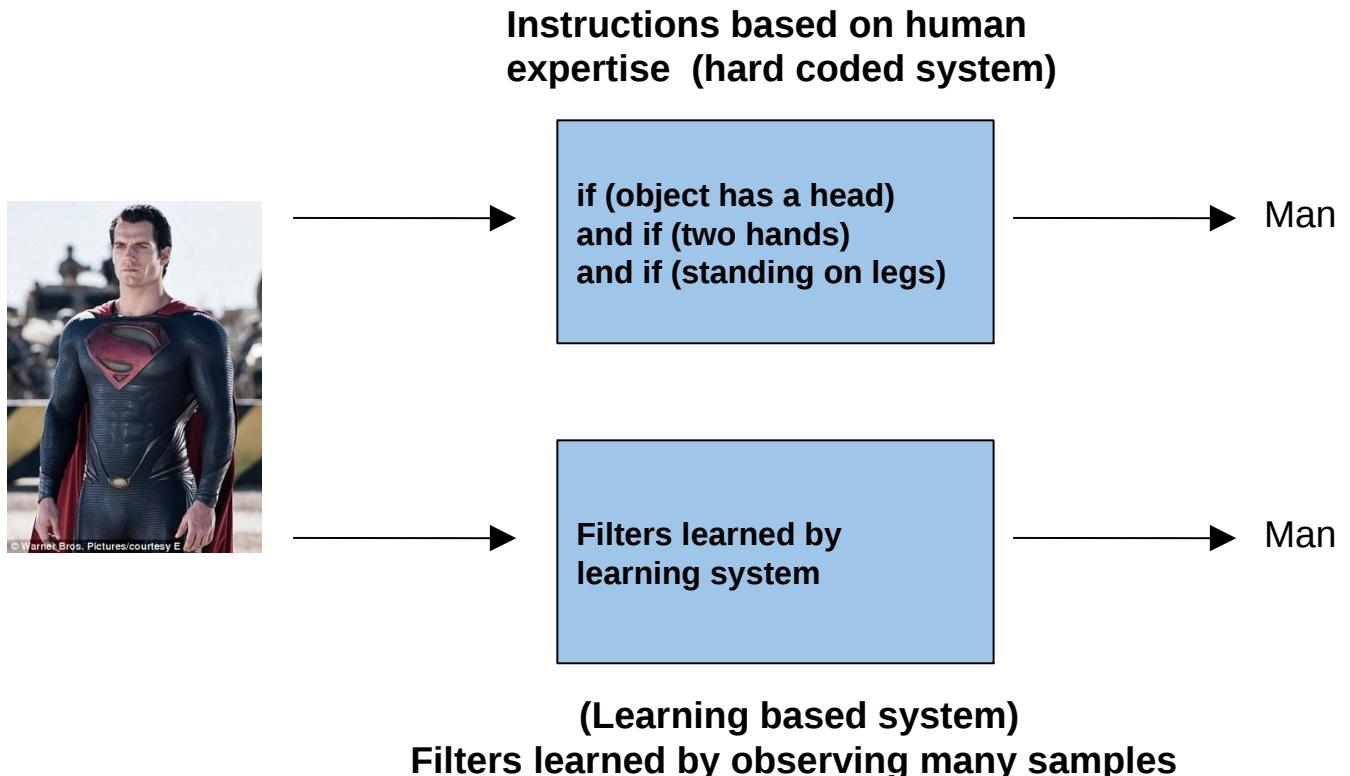




# Two primary ways of doing a task?



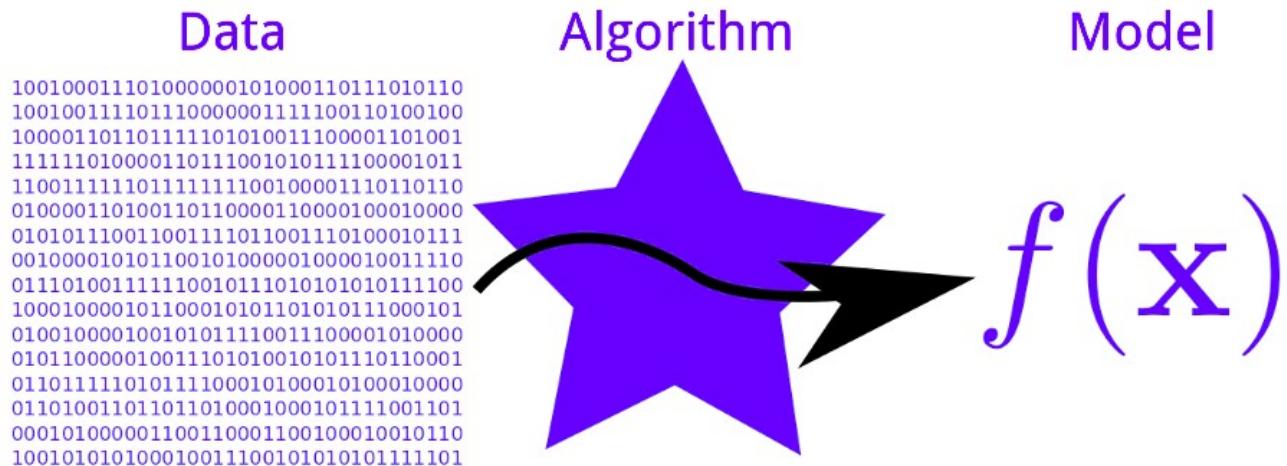
# Two primary ways of doing a task?





# Machine Learning

How can we build computer programs that automatically improve their performance through experience?





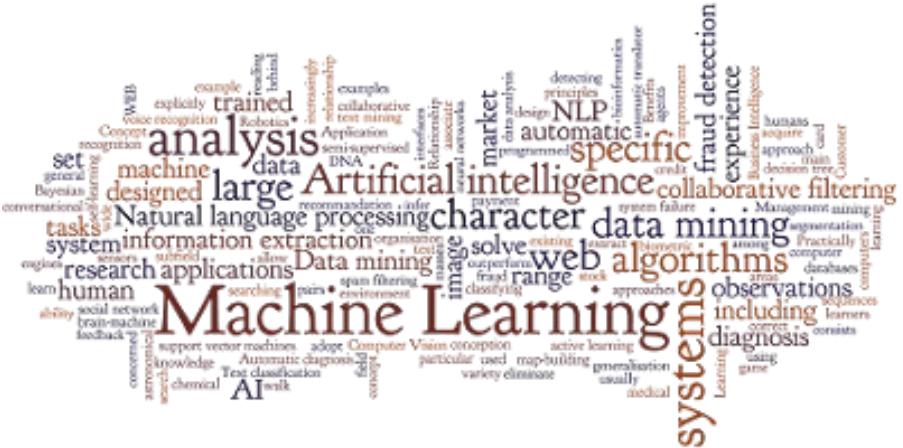
# Machine Learning

## *Approaches:*

- Artificial neural networks
  - Bayesian networks
  - Clustering
  - Representation learning
  - Reinforcement learning
  - ...

## *Applications:*

- Computer vision
  - Natural language processing
  - information retrieval
  - Robotics
  - ...





# Best learning system known to us?

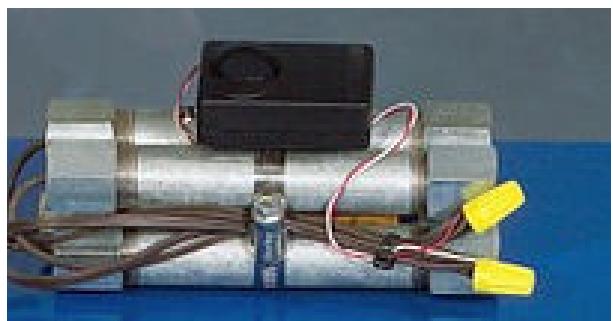
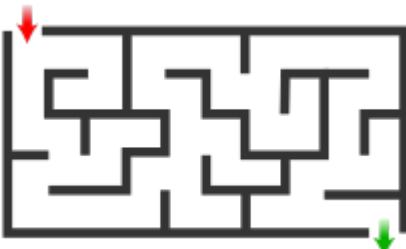
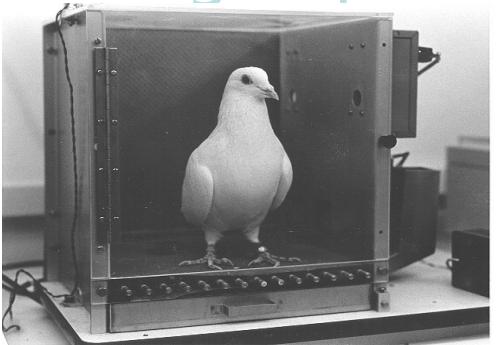


Biological system



Artificial system

# Thinking is possible even with a “small” brain



Pigeons as art experts – 85% (Watanabe et al. 1995)

Source: [https://en.wikipedia.org/wiki/Marc\\_Chagall](https://en.wikipedia.org/wiki/Marc_Chagall)  
[https://en.wikipedia.org/wiki/Vincent\\_van\\_Gogh](https://en.wikipedia.org/wiki/Vincent_van_Gogh)

Mice trained to run mazes[1], detect drugs

[1] <http://animals.pawnation.com/training-mice-run-mazes-11136.html>  
<http://www.ratbehavior.org/RatsAndMazes.htm>

[2]

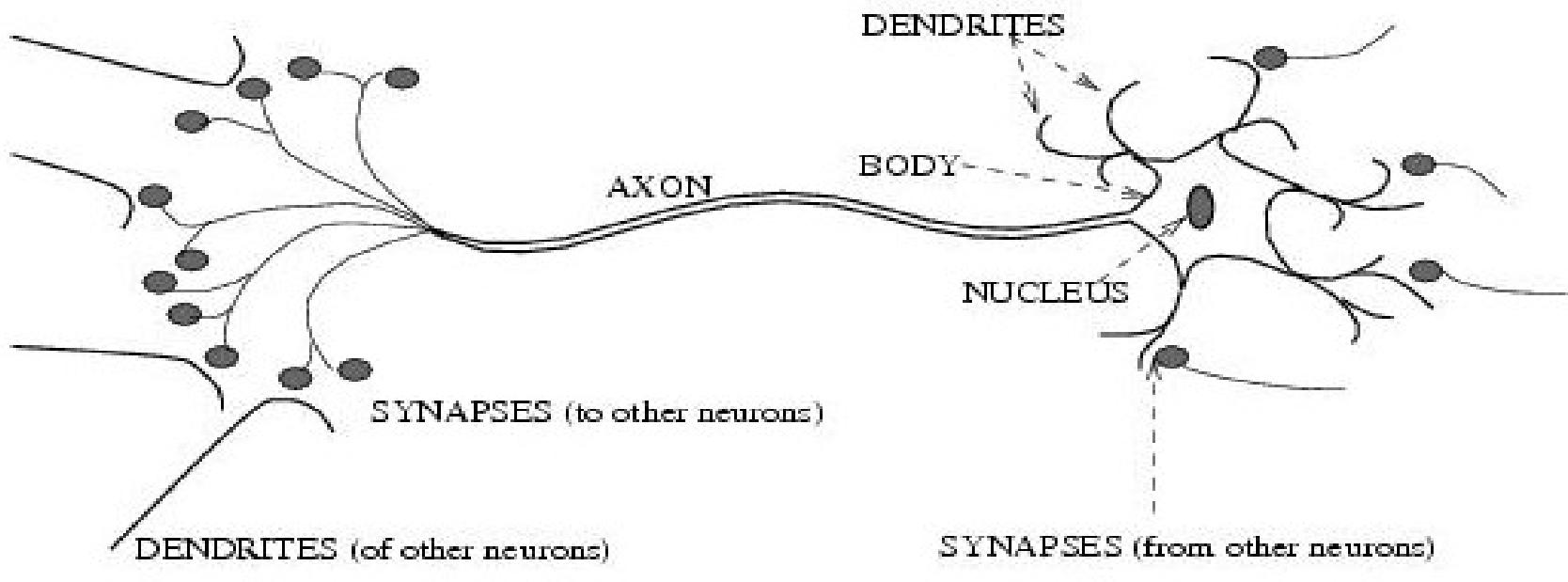
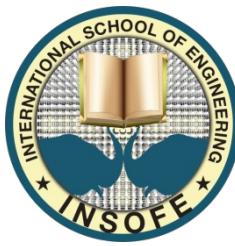
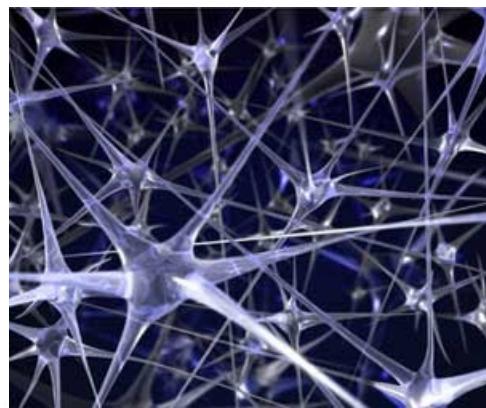
<http://newsfeedtime.com/2012/11/16/israeli-company-trains-mice-to-detect-drugs/>



# The results

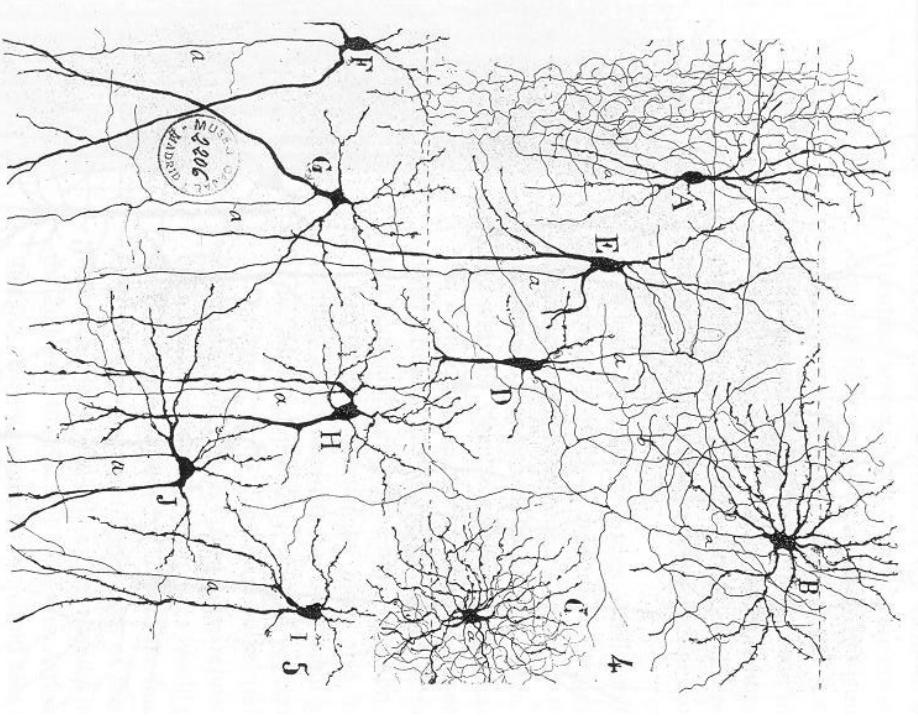
- Pigeons were able to discriminate between Van Gogh and Chagall with 95% accuracy (when presented with pictures they had been trained on)
- Discrimination still 85% successful for previously unseen paintings of the artists
- Mice can memorize mazes, odours of contraband (drugs / chemicals / explosives)

# So, how does the brain work?

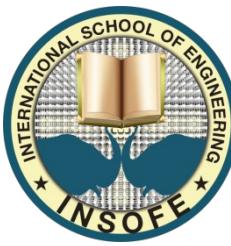


Direction of signal is along the Axon from Nucleus to synapse

# Biological neural networks



- Fundamental units are termed neurons.
- Connections between neurons are synapses.
- Adult human brain consists of 100 billion neurons and 1000 trillion synaptic connections.
- Equivalent to computer with one trillion bit per second processor.

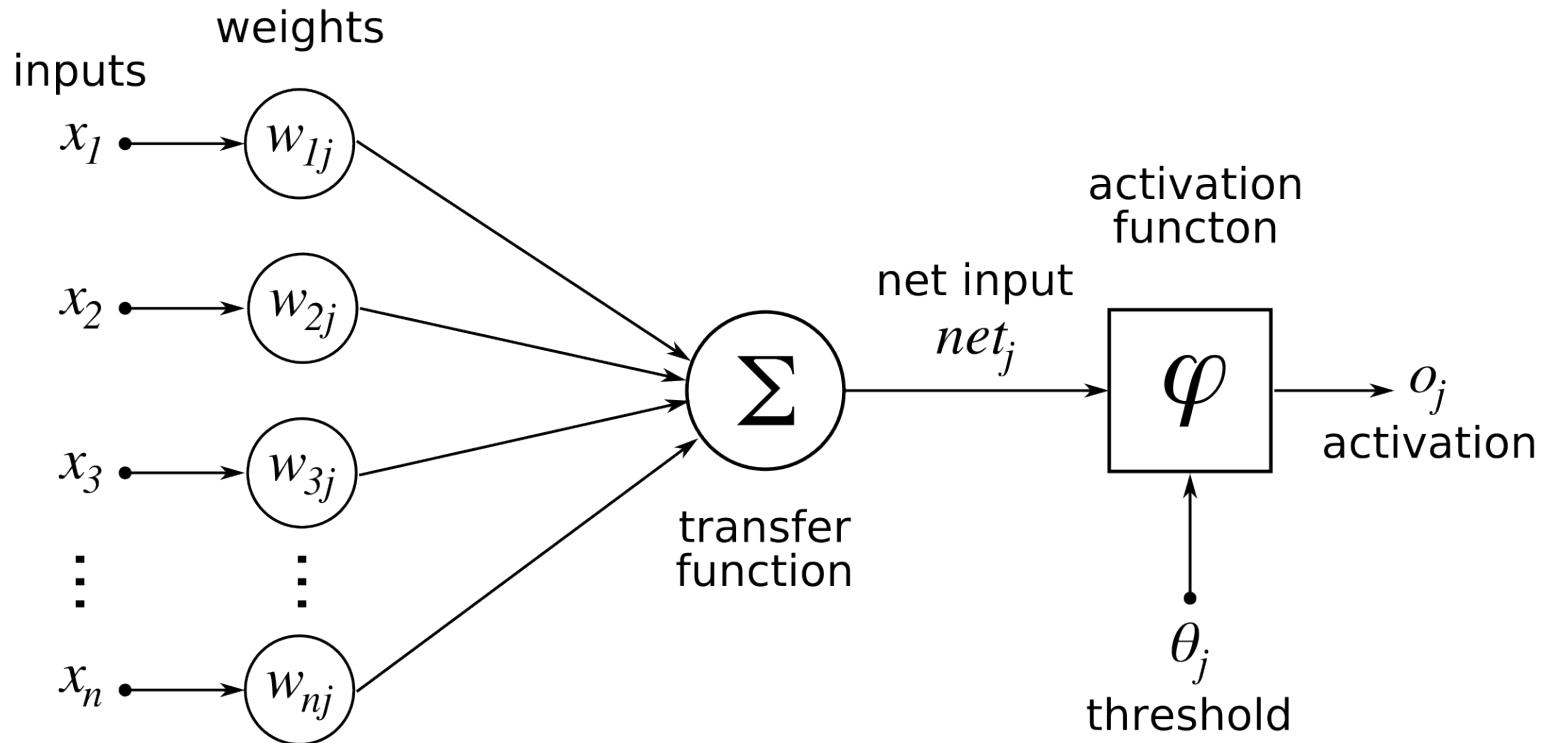


# Biological Neural Networks

Neurons	Example species
302	Nematode worm
$10^3$	Snail
$10^4$	Ant
$10^5$	Fly (Compound eyes, nerves in the legs, vibrissae)
$0.8 * 10^6$ to $10^6$	Honeybee
$4 * 10^6$	Mouse
$1.5 * 10^7$	Frog (Continuous targeting, catching while in 3D motion)
$5 * 10^7$	Bat
$1.6 * 10^8$	Dog
$3 * 10^8$	Cat
$6 * 10^8$	Chimp
$10^{11}$	Human
$>10^{11}$	Whales, Elephants, Dolphins

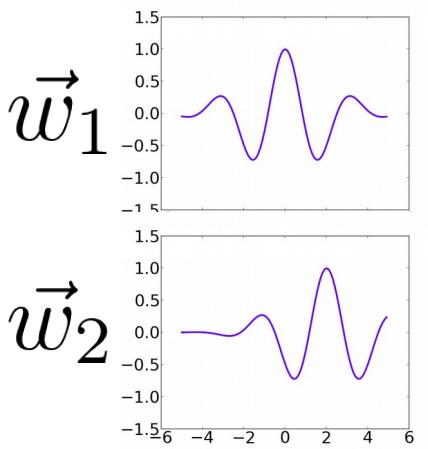
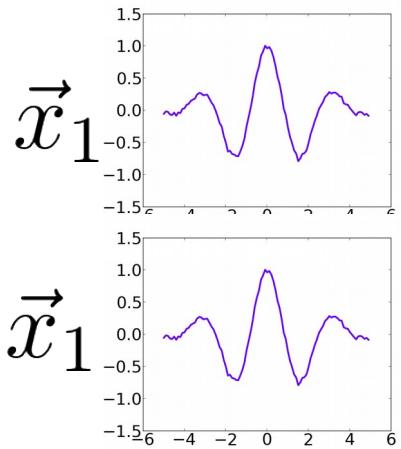
Simulating worm brain in software: <http://tinyurl.com/mx7bdd4>

# Artificial neural model

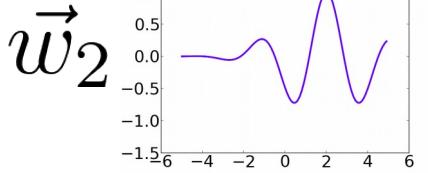
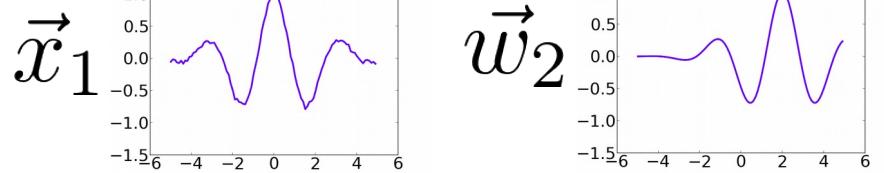




# What are weights or filters or features?



$$\vec{x}_1 \odot \vec{w}_1 = 512.31$$



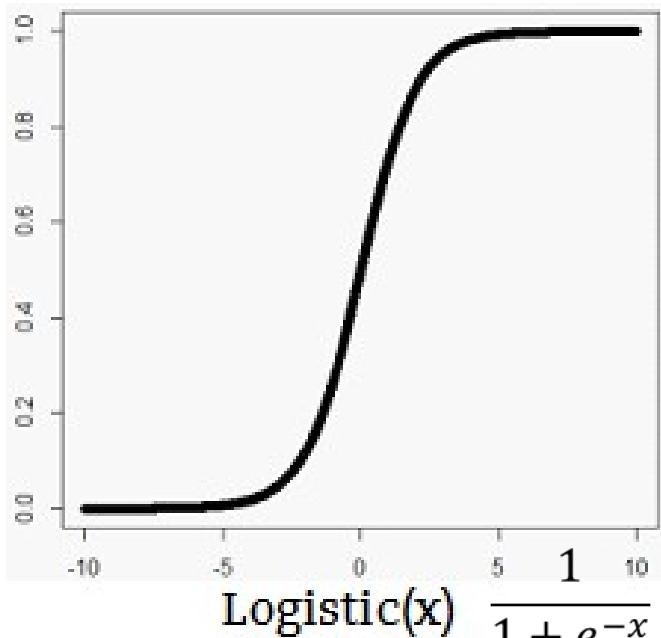
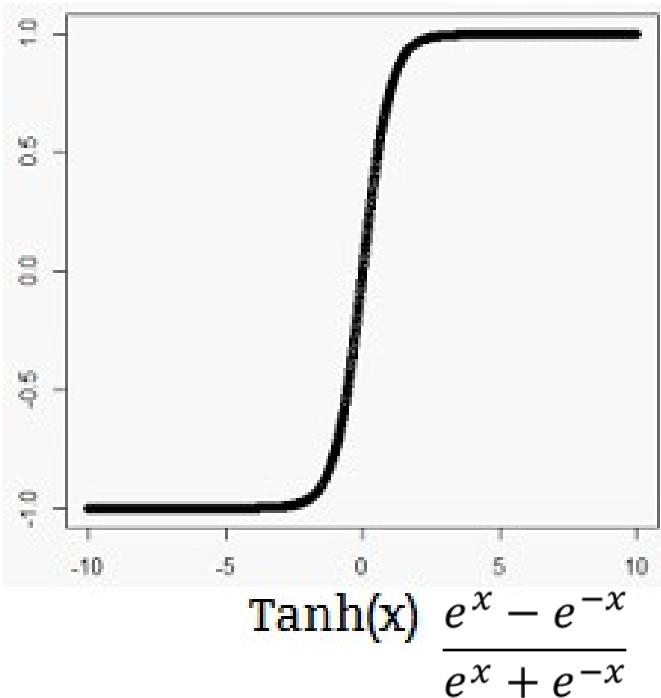
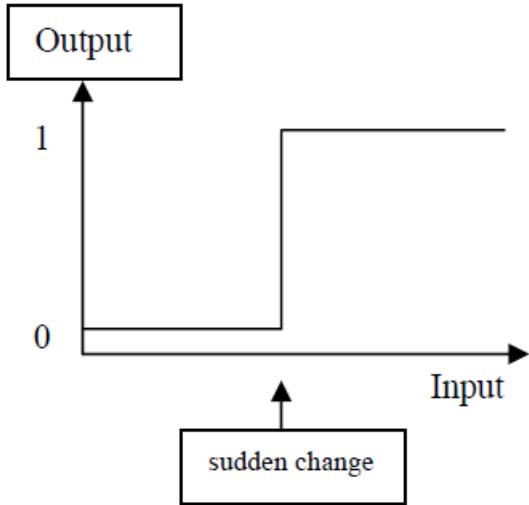
$$\vec{x}_1 \odot \vec{w}_2 = 2.12$$

If  $\vec{x}_1$  and  $\vec{w}_1$  are two vectors with N dimensions each

$x_1^1, x_1^2, x_1^3, \dots, x_1^N$  and  $w_1^1, w_1^2, w_1^3, \dots, w_1^N$  respectively,

$$\text{the dot product } \vec{x}_1 \odot \vec{w}_1 = \sum_{i=1}^N x_1^i w_1^i$$

# Squashing functions



Softmax:  $f = \frac{e^{-x}}{\sum_n e^{-x}}$   
Useful for mult-class problems

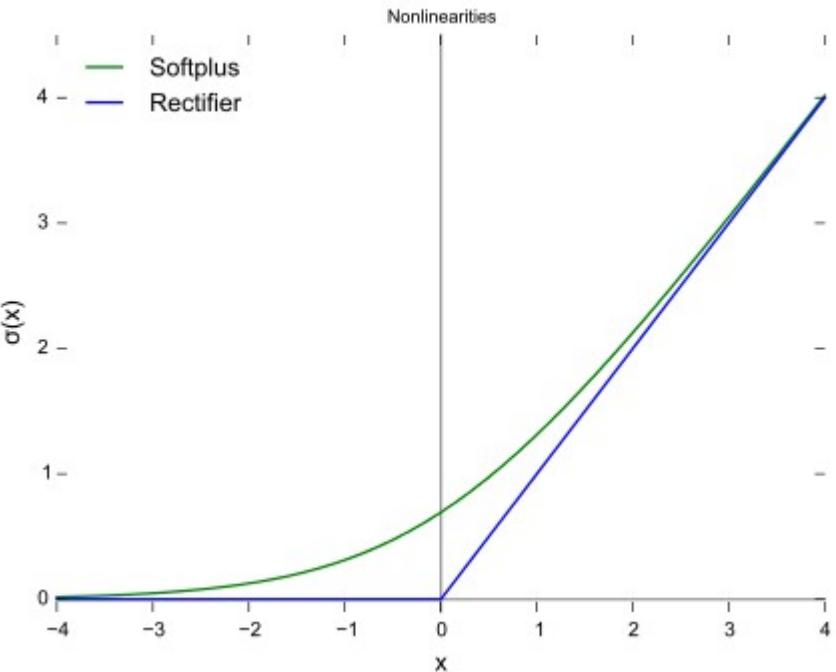
Good for deviations  
(time series, classification)

[http://www.dkriesel.com/\\_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf](http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf)

Good for averages  
(regression)

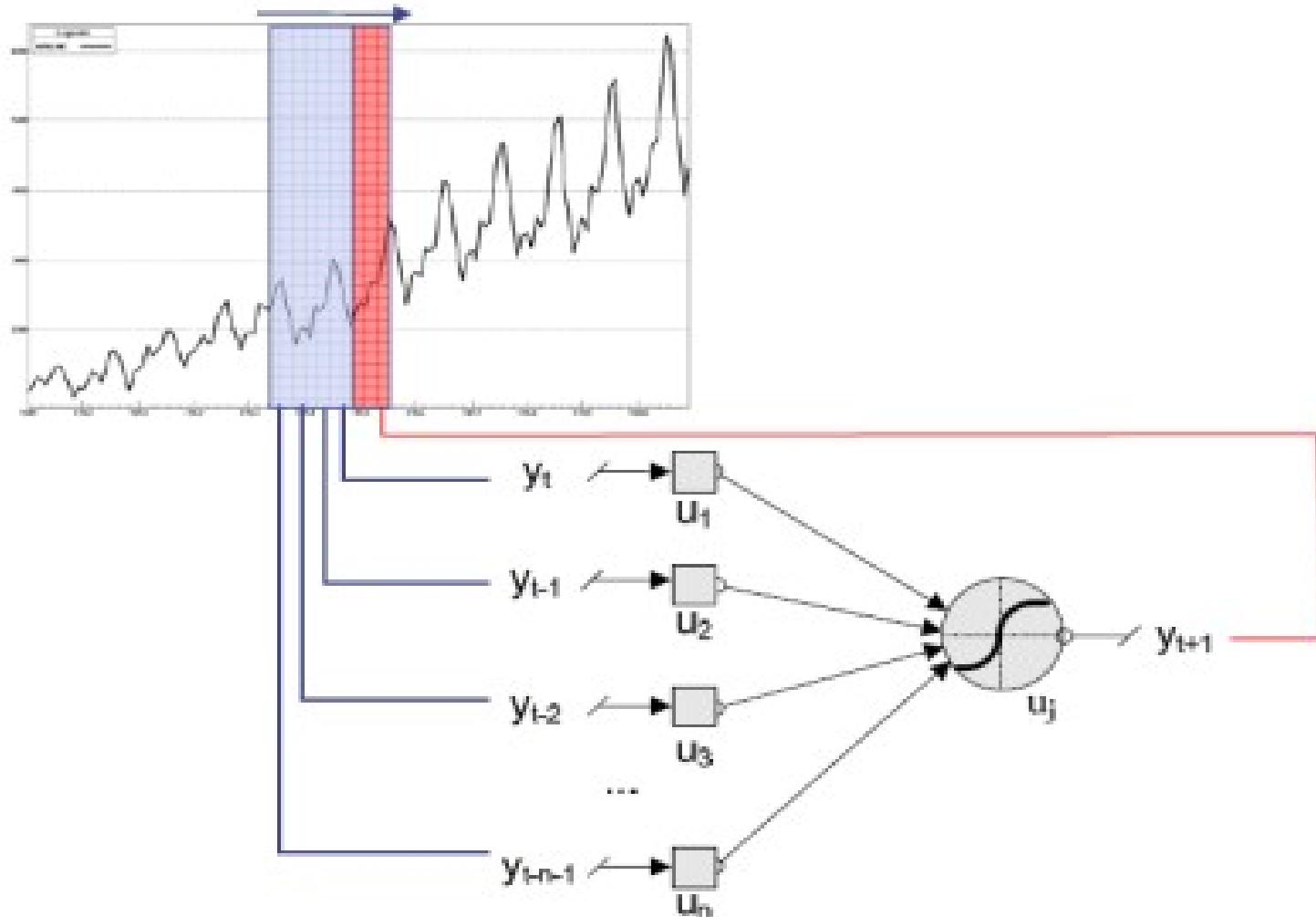
# Rectified linear functions

- More biologically plausible
- Computationally faster
- Most popular activation function for deep neural networks
- Efficient gradient propagation:  
No vanishing gradient problem or exploding effect



[https://en.wikipedia.org/wiki/Rectifier\\_neural\\_networks](https://en.wikipedia.org/wiki/Rectifier_neural_networks)

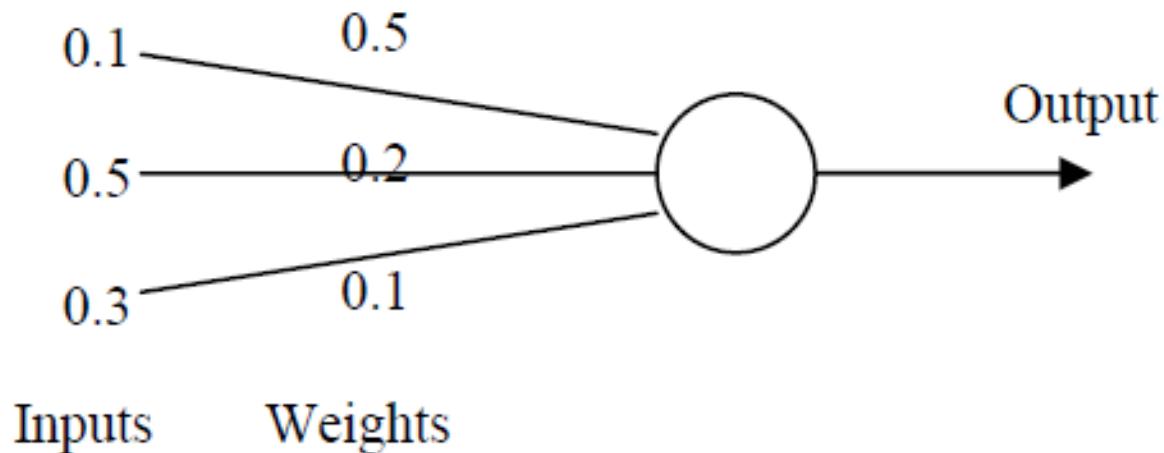
# Perceptron in a time series



**Q**



Calculate the output from the neuron below assuming a threshold of 0.5:



What is the output for a sigmoid function?

$$\frac{1}{1 + e^{-x}}$$

[http://www.dkriesel.com/\\_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf](http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf)



# Question

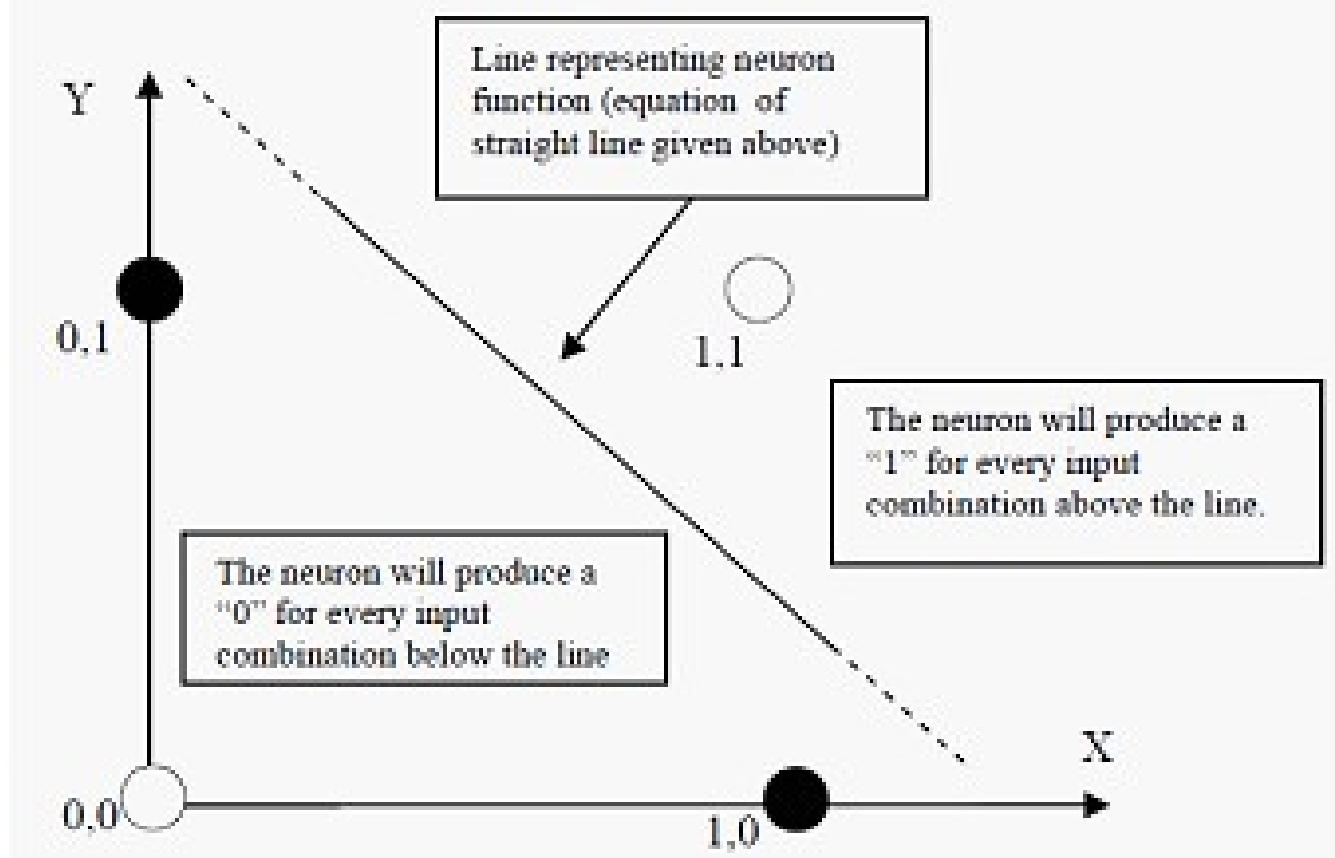
- A time series has a auto correlation with past day, last week same day, last quarter same day and last year same day.
- How many nodes do you have in the input



# LIMITATIONS OF PERCEPTRON AND HOW TO OVERCOME

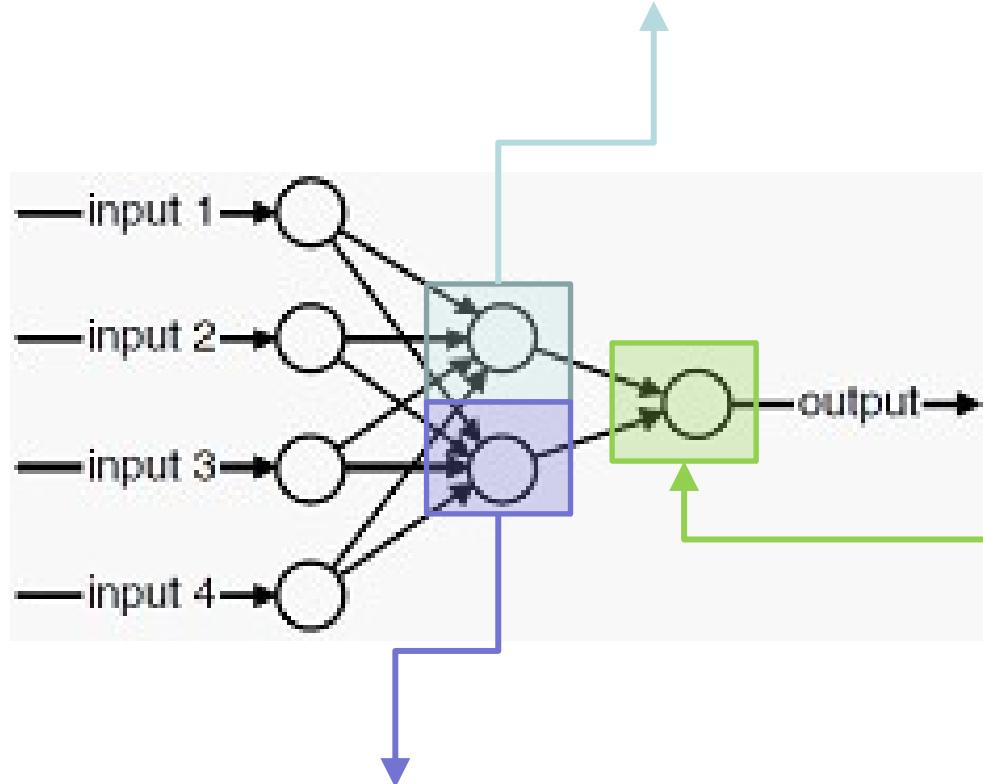
# Will it work here?

X	Y	OUT
0	0	0
0	1	1
1	0	1
1	1	0

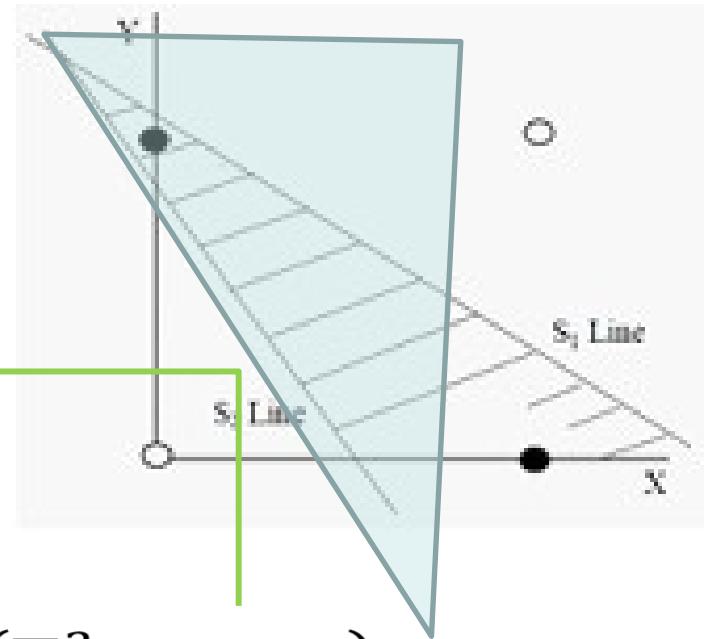
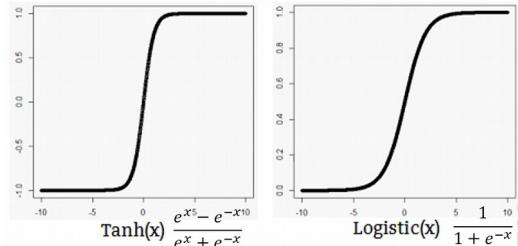


# A 3 layer network can learn any function

$$\phi_1 = f(\sum_{i=1}^4 x_i w_{i1})$$

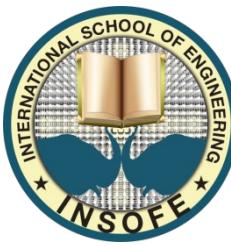


$$\phi_2 = f(\sum_{i=1}^4 x_i w_{i2})$$

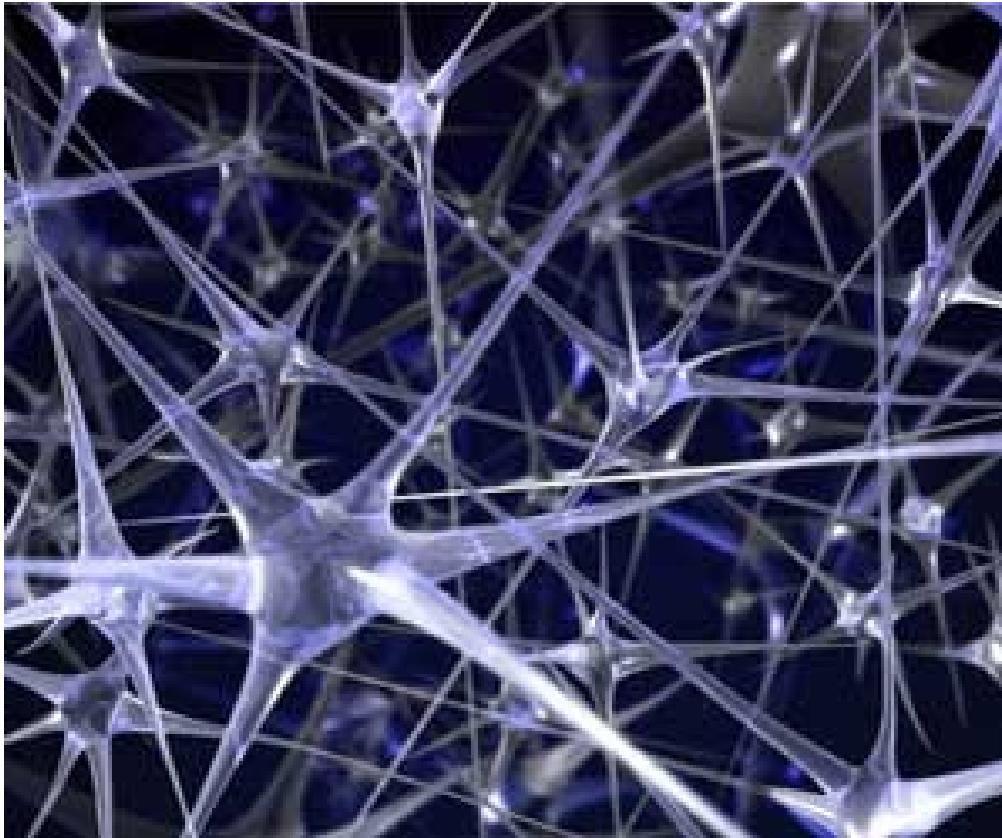


$$Output = f(\sum_{j=1}^2 \phi_j w_j)$$

$$Output = f(\sum_{j=1}^2 f(\sum_{i=1}^4 x_i w_{ij}) w_j)$$

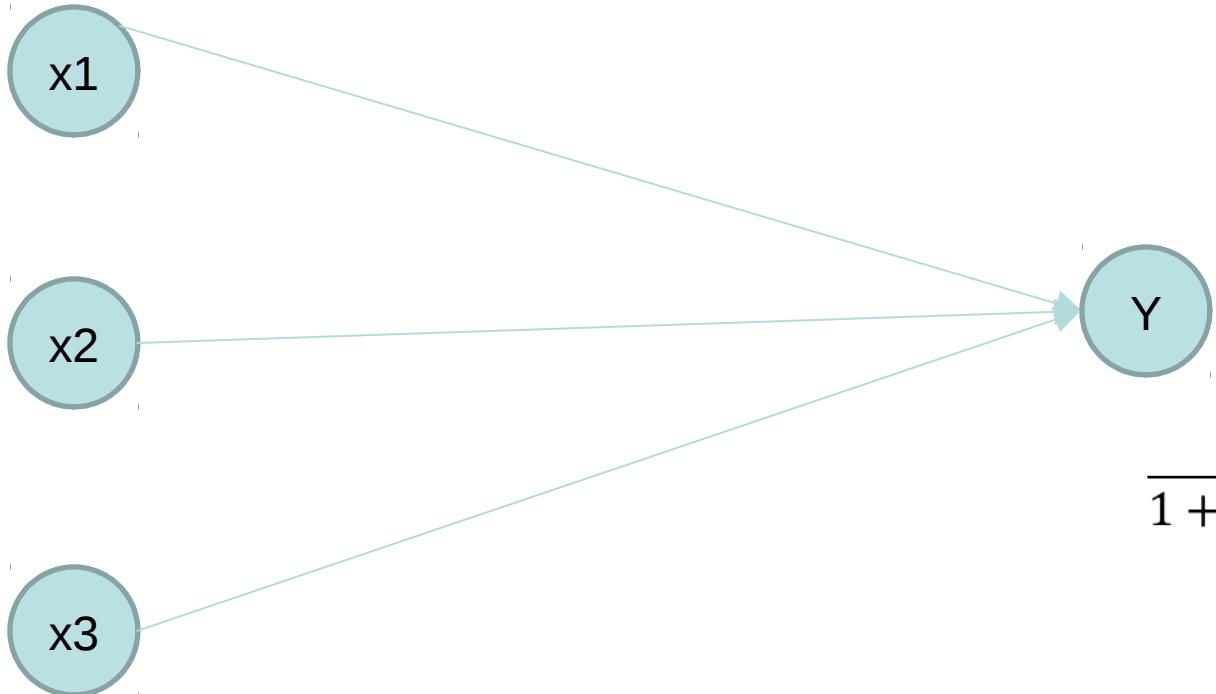


# How does brain do non linearity



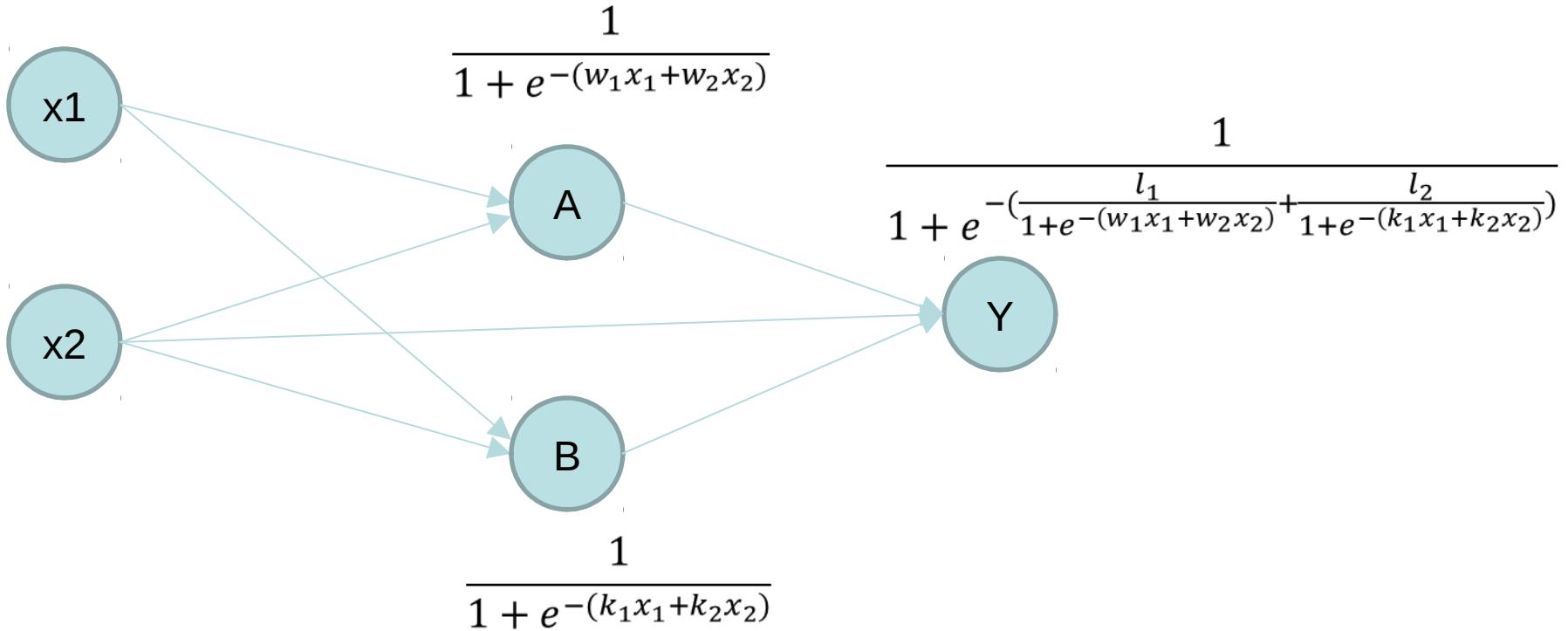
Many neurons connected together

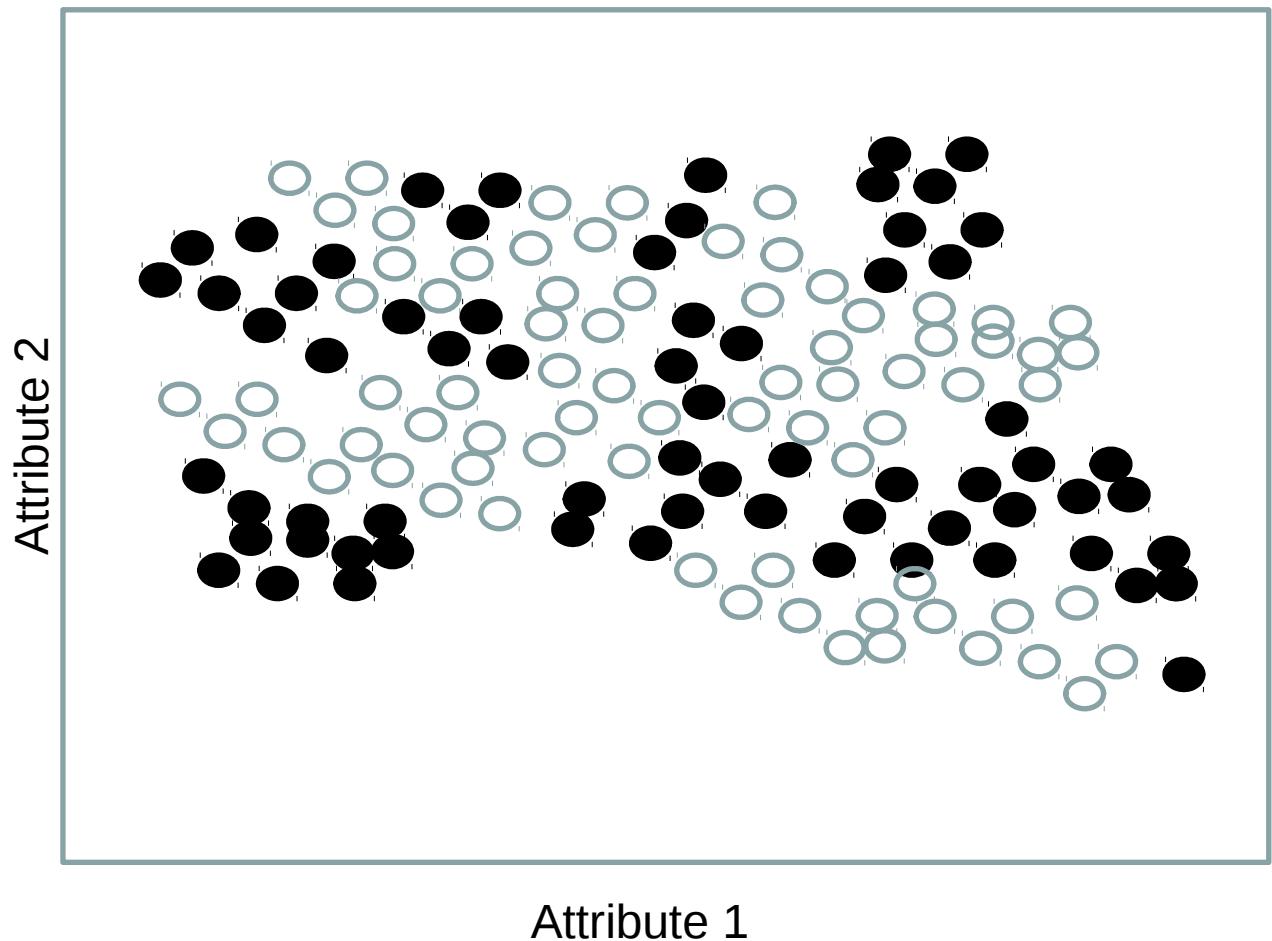
# How to describe non-linearity

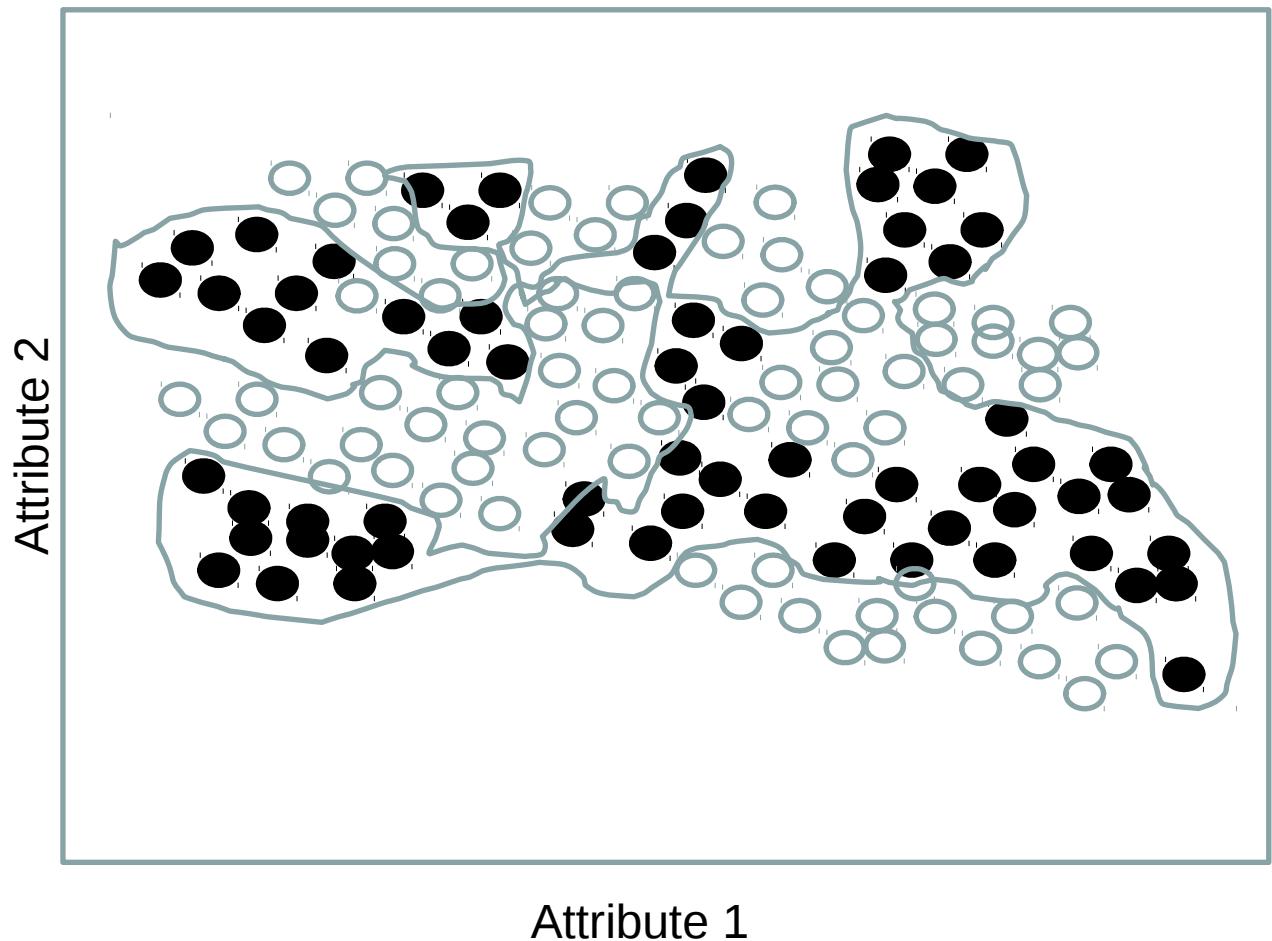


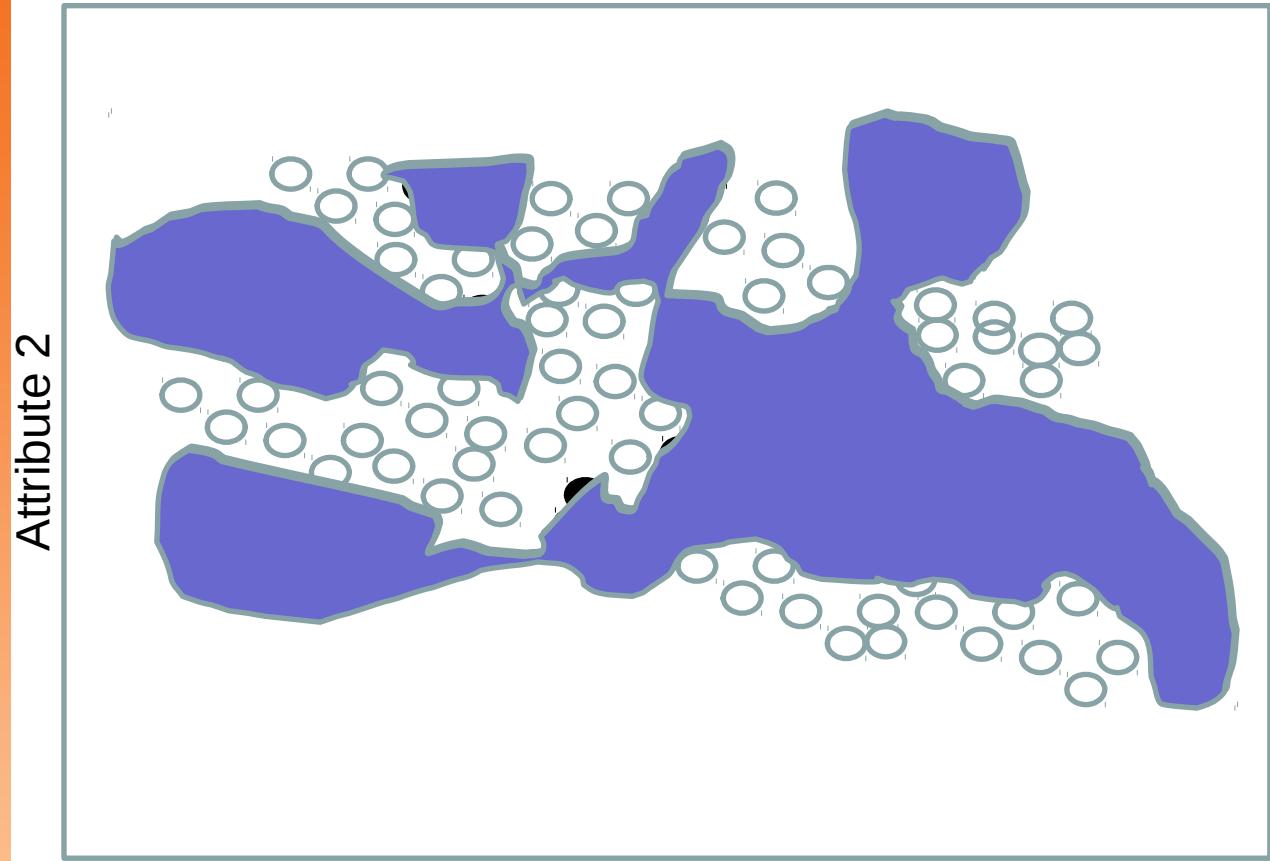
$$\frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + w_3x_3)}}$$

# How to describe non-linearity

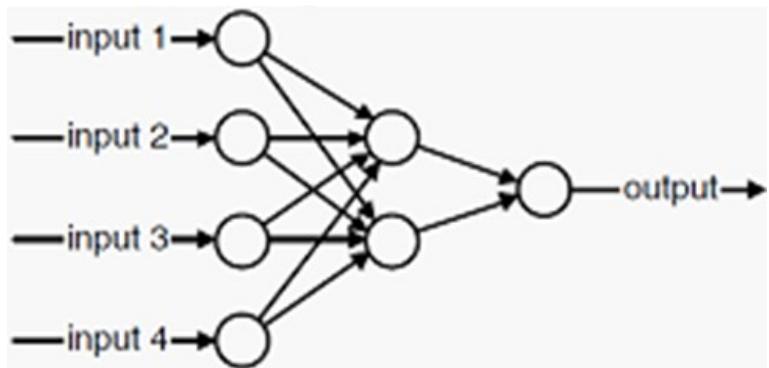




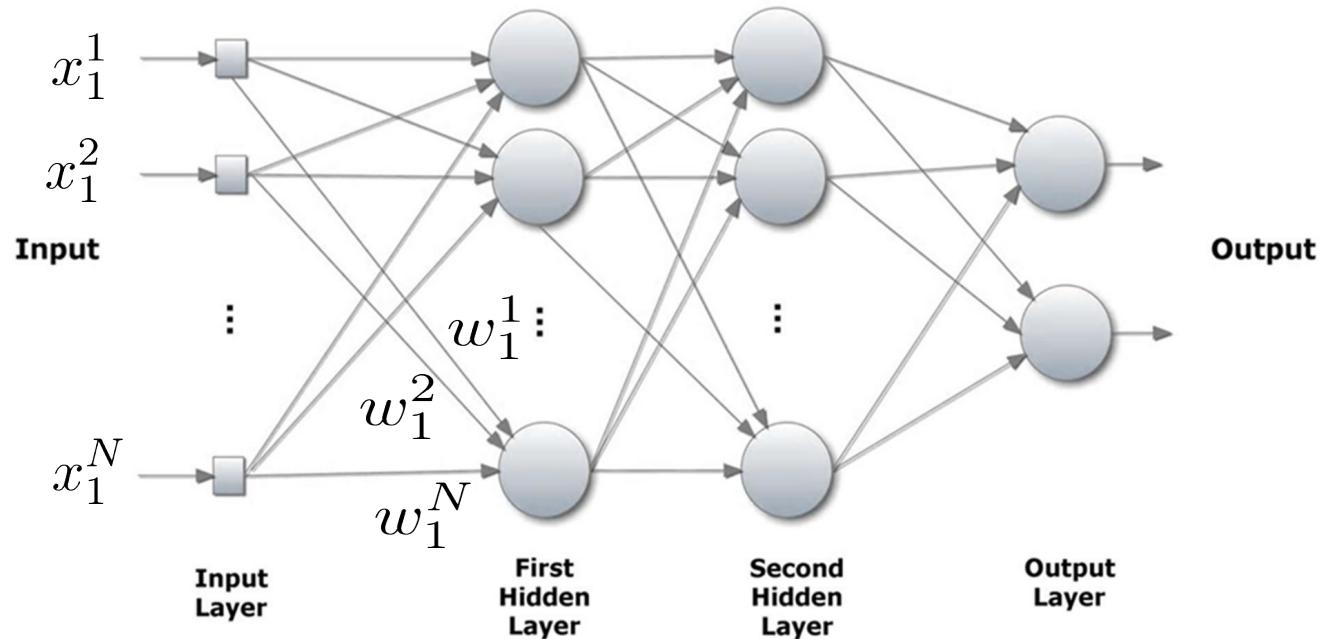




Attribute 1



# Artificial neural networks



A multilayer perceptron (Feed forward network)

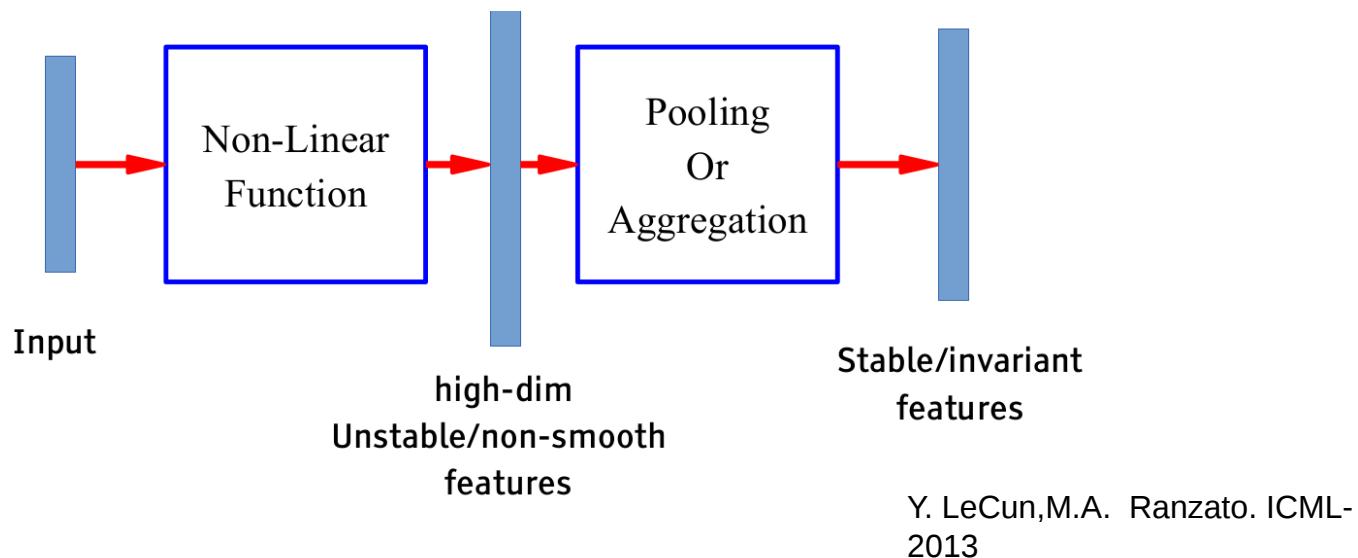
# Basic Idea for Invariant Feature Learning

Embed the input non-linearly into a high(er) dimensional space

In the new space, things that were non separable may become separable

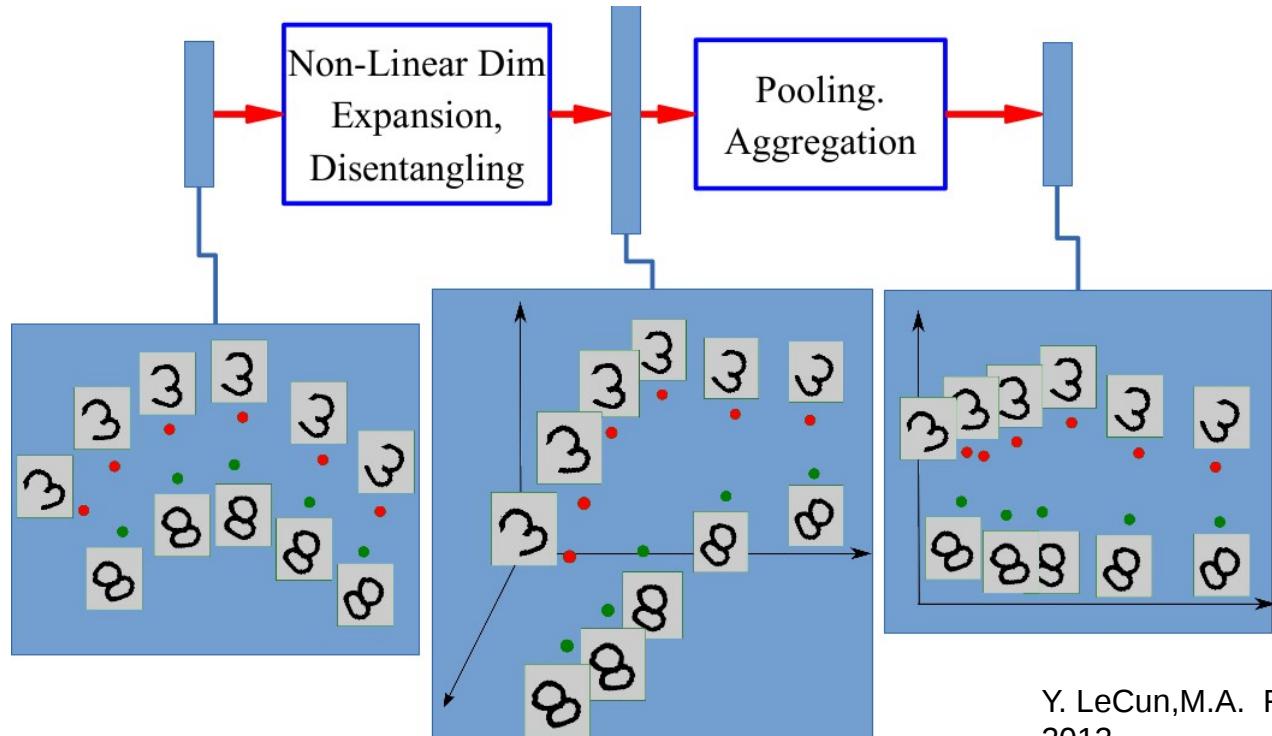
Pool regions of the new space together

Bringing together things that are semantically similar. Like pooling.



# Non-Linear Expansion → Pooling

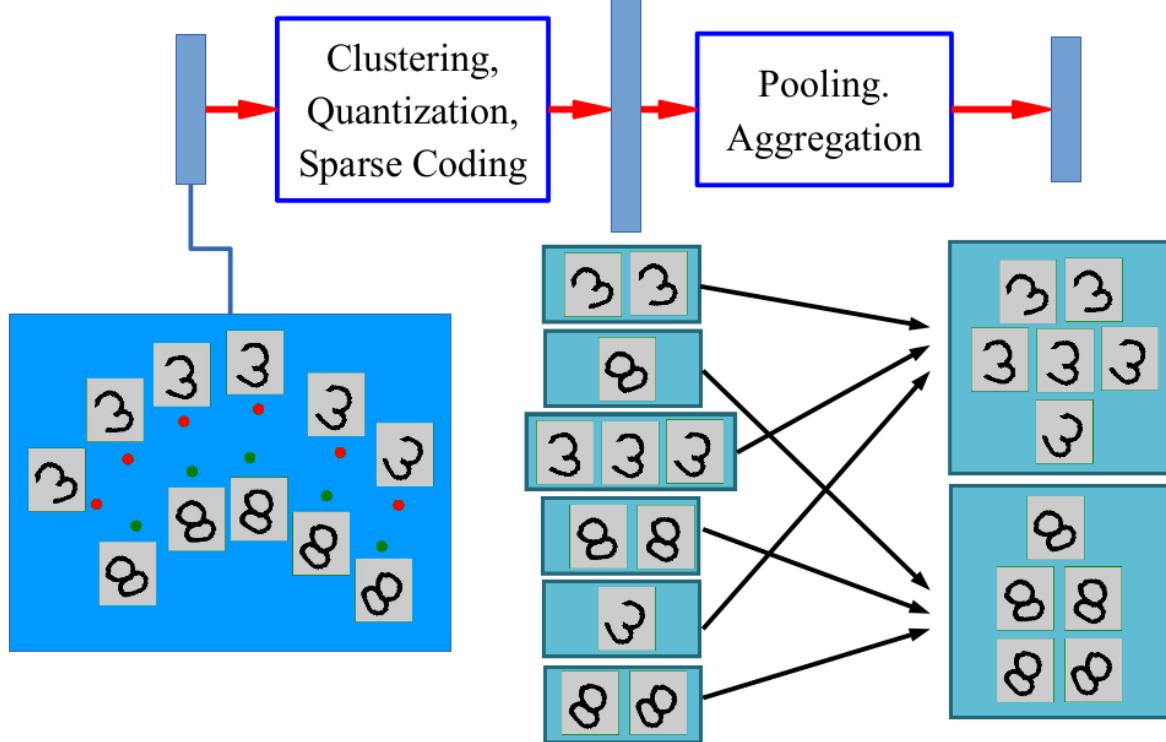
Entangled data manifolds



Y. LeCun, M.A. Ranzato. ICML-2013

# Sparse Non-Linear Expansion → Pooling

Use clustering to break things apart, pool together similar things



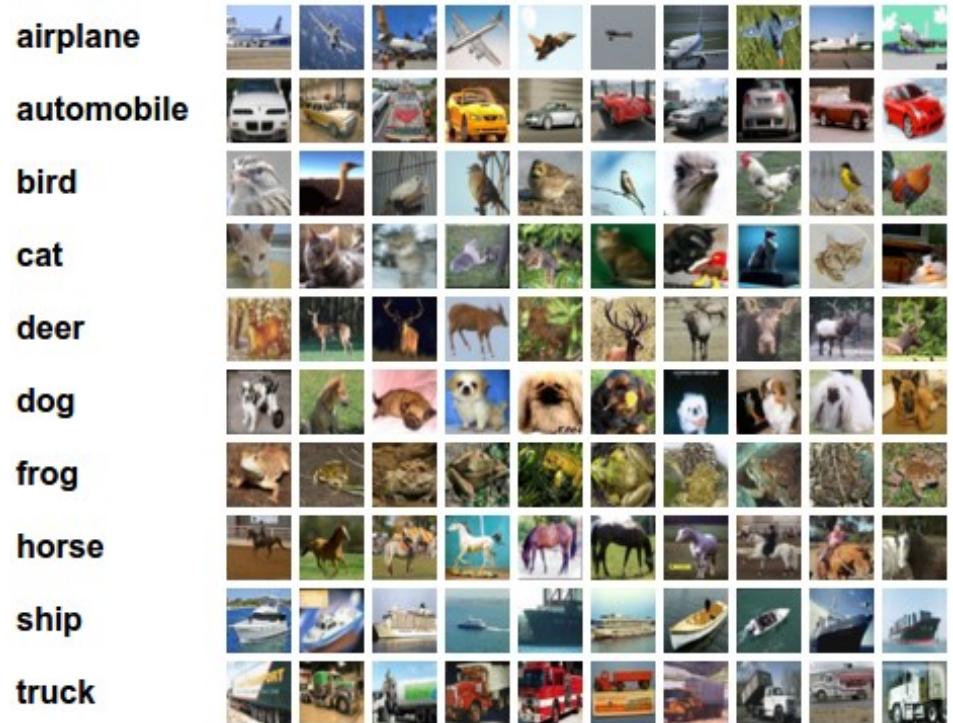
Y. LeCun, M.A. Ranzato. ICML-2013



# Learning of a biological NN

- In 1949 Donald Hebb postulated one way for the network to learn. If a synapse is used more, it gets strengthened – releases more Neurotransmitter. This causes that particular path through the network to get stronger, while others, not used, get weaker.
- You might say that each connection has a *weight* associated with it – larger weights produce more stimulation and smaller weights produce less.

# How to train a network?



*First of all we need a task and corresponding data.*

Task: Image classification

Dataset: CIFAR-10

The dataset consists of

- 50000 train images
- 10000 test images
- 10 different classes



# How to train a network?

- Learning is changing weights
- In the very simple cases
  - **Start random**
  - **If** the output is correct **then** do nothing.
  - **If** the output is too high, decrease the weights attached to high inputs
  - **If** the output is too low, increase the weights attached to high inputs

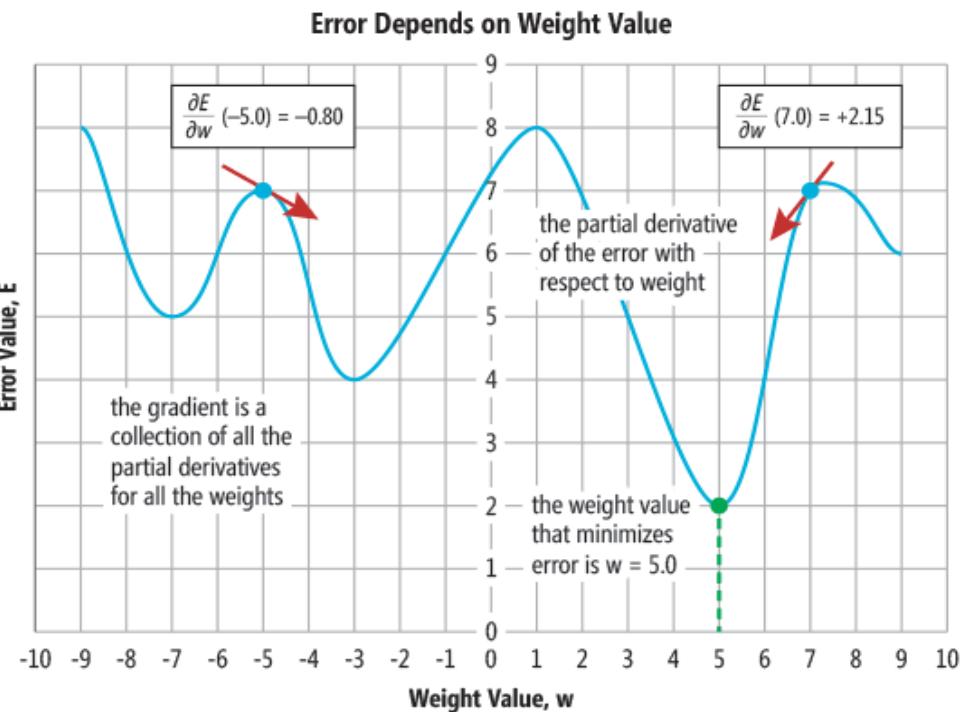


# How to train a network?

- Let be  $(x_i, y_i)$  be input label pairs from the dataset.
- Given an input,  $\tilde{y}_i = f(x_i, W)$  is the network output. Where  $W$  is the set of network parameters.
- The cost function is defined as  $E = \sum_i \|y_i - \tilde{y}_i\|_2^2$

*Our task is to minimize  $E$  by updating  $W$ . How can we do that?*

# Gradient descent?



Given function  $E$  with parameters  $W$

- $E$  can be minimized by moving opposite to gradient of  $E$  w.r.t  $W$ .

$$\Delta W = -\alpha \nabla E$$



# Back propagation

- Calculate the error at the output

$$w_i \leftarrow w_i + \Delta w_i$$

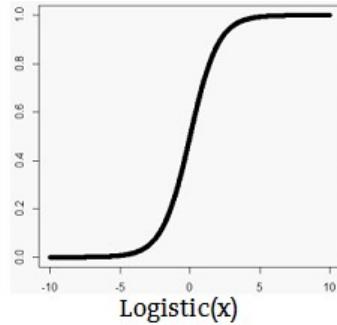
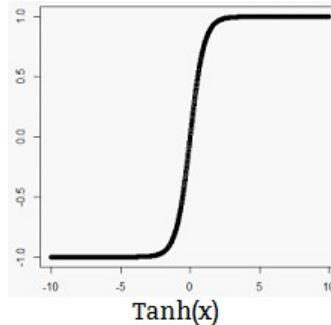
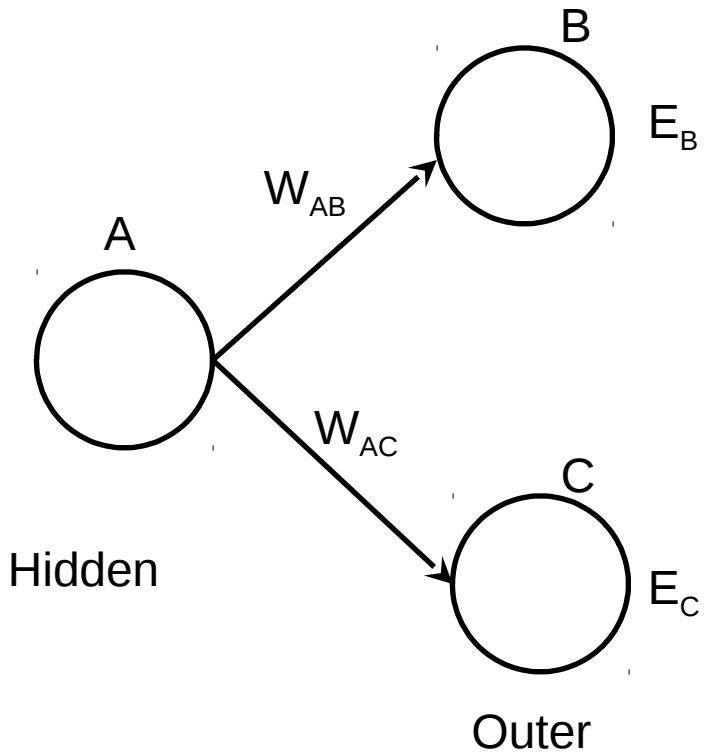
$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$x_{i,j}$  is the output from i to j

$$\Delta w_{i,j} = \eta o_k (1 - o_k) (t_k - o_k) x_{i,j}$$

The “*Output(1-Output)*” term is necessary because of the Sigmoid. If we were only using a threshold neuron it would just be (*Target – Output*).

# Back propagation: Send the error back



$$\text{Error}_A = \text{Output}_A (1 - \text{Output}_A)(\text{Error}_B W_{AB} + \text{Error}_C W_{AC})$$

The “*Output(1-Output)*” term is necessary because of the Sigmoid (This is derivative of sigmoid). If we were only using a threshold neuron it would just be (*Target – Output*).

# NN Training

- Initialize all weights
  - +ve and -ve random values
- Forward phase:

$$h_j = f \left( \sum_{i=0}^I x_i v_{ij} \right)$$

$$\hat{h}_j = \sum_{i=0}^I x_i v_{ij}$$

$$o_k = f \left( \sum_{j=0}^J h_j w_{jk} \right)$$

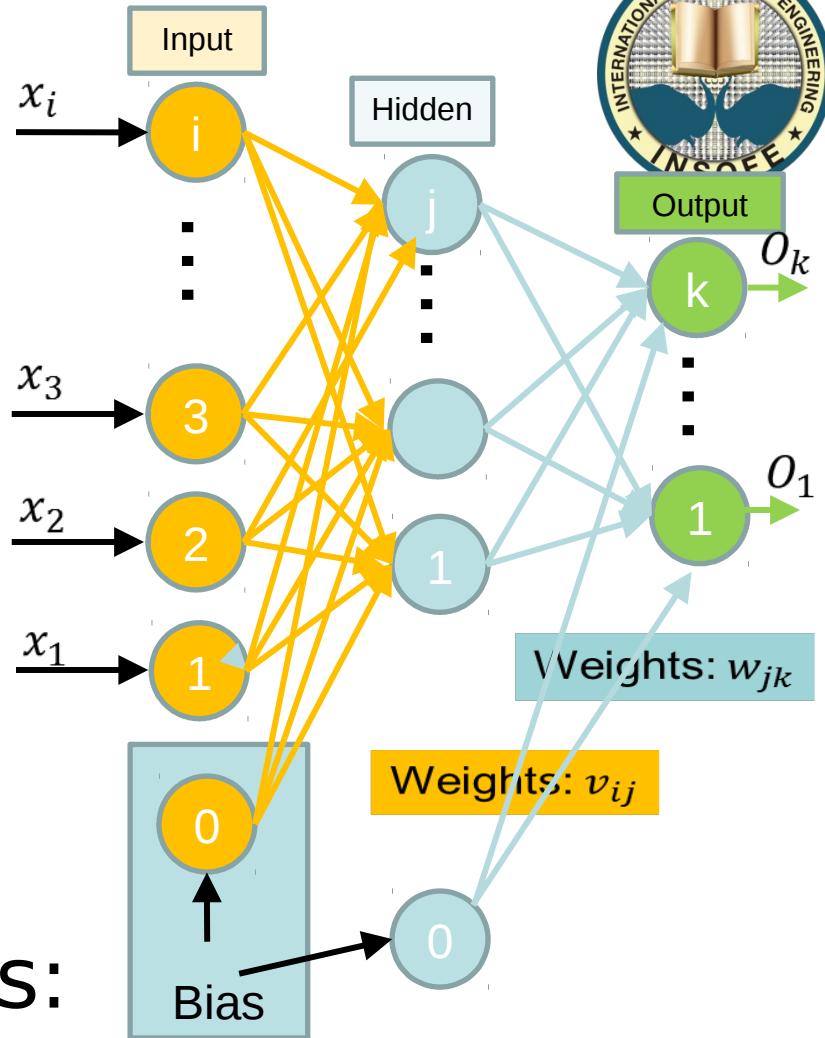
$$\hat{o}_k = \sum_{j=0}^J h_j w_{jk}$$

- Use error to update weights:

$$Error, E = \frac{1}{2} \sum_{k=1 \text{ to } K} (t_k - o_k)^2$$

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

$$\Delta w_{jk} = -\eta * \frac{dE}{dw_{jk}}$$



<http://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

# NN Training

- Hidden:

$$h_j = f\left(\sum_{i=0}^I x_i v_{ij}\right) \quad \hat{h}_j = \sum_{i=0}^I x_i v_{ij}$$

$$o_k = f\left(\sum_{j=0}^J h_j w_{jk}\right)$$

$$\text{Error, } E = \frac{1}{2} \sum_{k=1 \text{ to } K} (t_k - o_k)^2$$

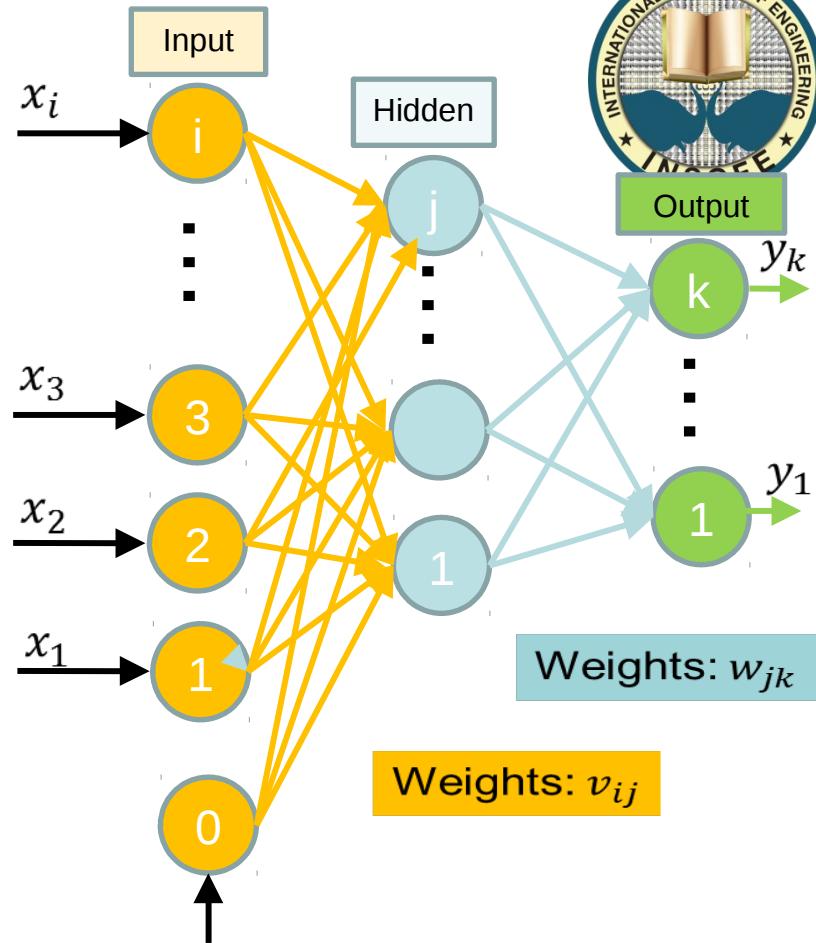
$$\frac{dE}{dv_{ij}} = \frac{dE}{dh_j} \frac{dh_j}{d\hat{h}_j} \frac{d\hat{h}_j}{dv_{ij}}$$

$$\frac{dE}{dh_j} = \underbrace{\frac{dE}{do_k} \frac{do_k}{d\hat{o}_k} \frac{d\hat{o}_k}{dh_j}}_{\delta_{o_{jk}}} \quad [\hat{h}_j * (1 - \hat{h}_j)]$$

$$x_i$$

$$[t_k - o_k][\hat{o}_k (1 - \hat{o}_k)]$$

$$\frac{d \sum_{j=0}^J h_j w_{jk}}{dh_j} \quad \left[ \sum_{k=1}^K w_{jk} * \delta_{o_{jk}} \right]$$



<http://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

# NN Training

- Forward Phase:
- Backward phase:

$$\delta_{output_{jk}} = [t_k - y_k][y_k(1 - y_k)] h_j$$

$$\delta_{hidden_{ij}} = \left[ \sum_{k=1}^K w_{jk} * \delta_{output_{jk}} \right] [h_j * (1 - h_j)] x_i$$

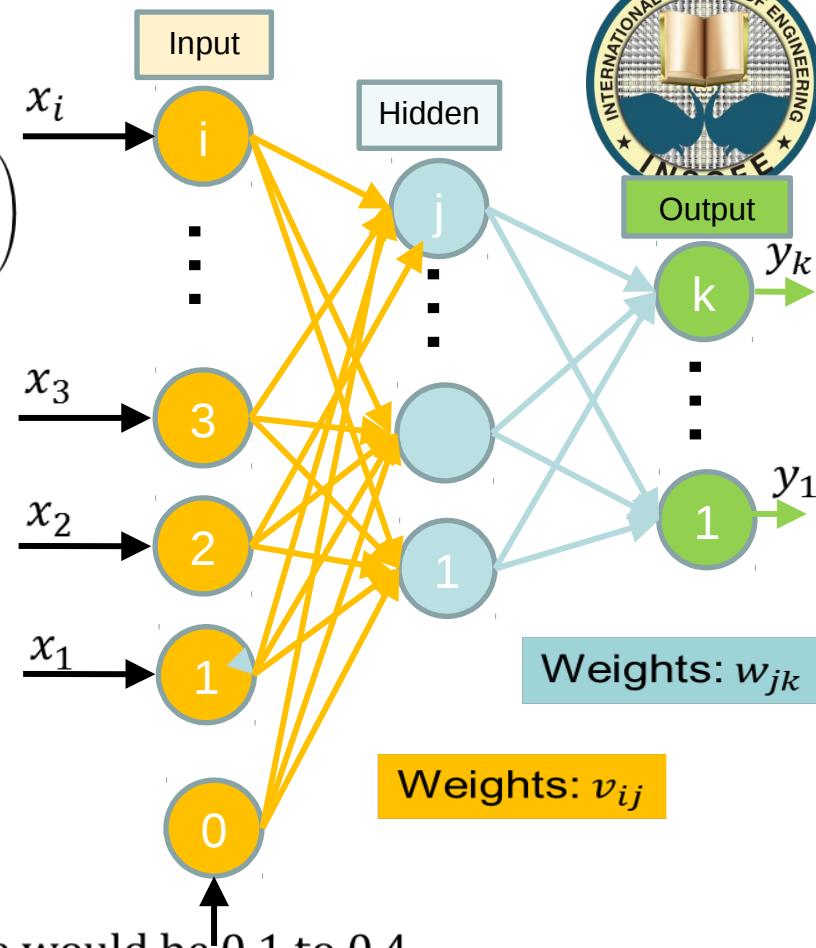
- Update weights:

$$w_{jk} \leftarrow w_{jk} + \eta * \delta_{output_{jk}}$$

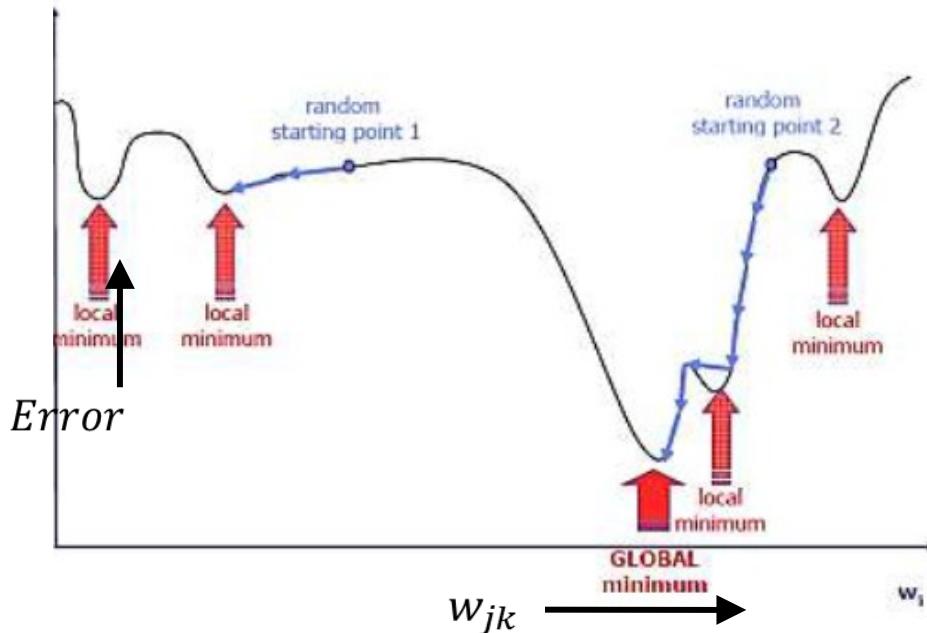
$$v_{ij} \leftarrow v_{ij} + \eta * \delta_{hidden_{ij}}, \text{ where } \eta \text{ is the learning rate}$$

$0 < \eta < 1$ , typical value would be 0.1 to 0.4

- When do you stop?
- Is training data always in same sequence?



# NN Training: Other Methods



- Gradient descent (Back propagation, conjugate gradient)
- Evolutionary techniques (Genetic algorithms and simulated annealing)
- Better error measures like cross entropy

$$-\frac{1}{n} \sum_{\text{input samples}} [t \ln(y) + (1 - t)\ln(1 - y)]$$

<http://neuralnetworksanddeeplearning.com/chap3.html>



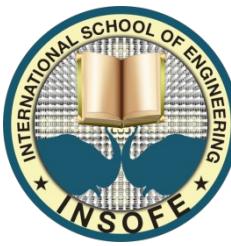
# NN Training: Additional Considerations

- Which is correct
  - Show an input...adjust weights, show another and adjust weights...,once the input is all over, start with row 1 if needed
  - Show an input train until convergence, show second one, train until convergence...



# Local versus global

- Gradient descent is greedy
- Multiple initiations and selection of the best is the option



# NN Training: Additional Considerations

- Momentum: Resistance to directional change

$$w_{jk} \leftarrow w_{jk} + \eta * \delta_{output_{jk}} + \alpha * \Delta w_{jk}^{t-1}$$

$\alpha$  is the momentum,  
where  $0 < \alpha < 1$

$$v_{ij} \leftarrow v_{ij} + \eta * \delta_{hidden_{ij}} + \alpha * \Delta v_{ij}^{t-1}$$

- Learning rate: Resistance to fast learning
  - The learning rate ( $\eta$ ), can be set to a high value  
 $0 < \eta < 1$ , typical value would be 0.1 to 0.4
  - Gradually reduced with subsequent epochs



# NN Training: Additional Considerations

- Jittering: Adding small amounts of noise to the input data
- Is training data always in the same sequence
- Weight decay: Weights that do not change are multiplied with  $0 < e < 1$
- Early stopping: Stop training when the change in error is very low



# NN Training: Additional Considerations

- Variable selection
  - Input layer will need to have as many nodes as there are inputs
  - Dependence, correlation, dimensionality reduction etc.
  - Use decision trees or random forests to first identify the relevant inputs
- Data pre-processing
- Data separation into training, validation and test models



# NN Training: Additional Considerations

- Variable selection example
  - Original time series data is  $X_1, X_2, X_3, X_4..X_n$
  - Auto Correlation Function (ACF) showed three lags
  - What will be your input and output?

Input			
Column 1	Column 2	Column 3	Output
$X_1$	$X_2$	$X_3$	$X_4$
$X_2$	$X_3$	$X_4$	$X_5$
$X_4$	$X_5$	$X_6$	$X_7$
...	...	...	...
$X_{n-2}$	$X_{n-1}$	$X_n$	$X_{n+1}$



# Question

- A time series has a auto correlation with past day, last week same day, last quarter same day and last year same day.
- How many nodes do you have in the input



# NN Training: Additional Considerations

- Sensitivity: Which input is most influential?
- Find the average value for each input. We can think of this average value as the center of the test set.
- Measure the output of the network when all inputs are at their average value.
- Measure the output of the network when each input is modified, one at a time, to be at its minimum and maximum values (usually -1 and 1, respectively).



- There are two inputs A, B. The output at average values of A and B is 5
- When A is at minimum, the output is 0 and when B is at minimum, the output is 15.
- Which node influences the network more



# Topology

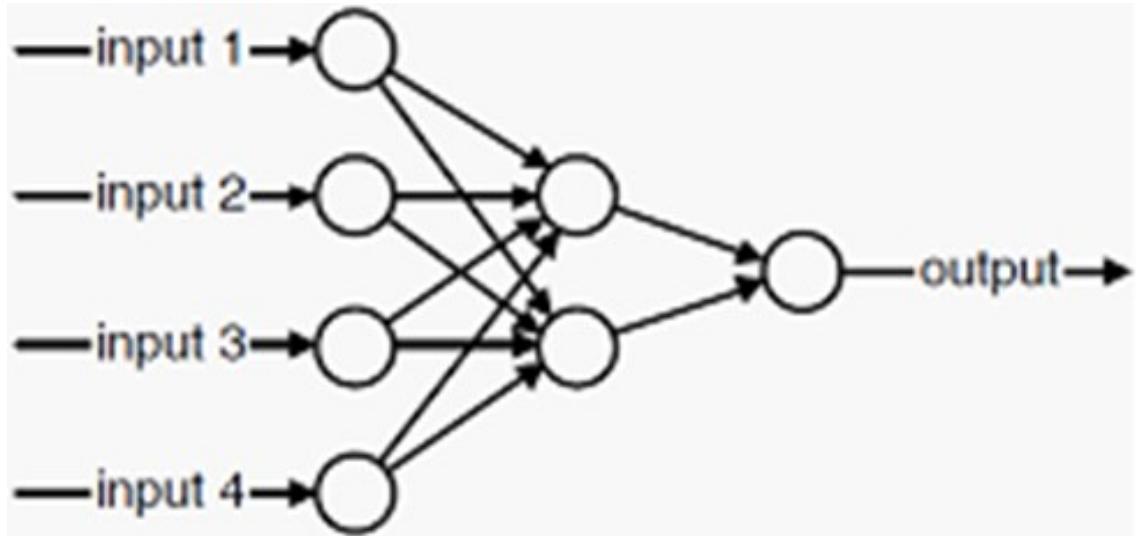
- 90%: One hidden layer
- 10%: Two hidden layers
- Never more than that! It might lead to overfitting



# Topology: Number of nodes

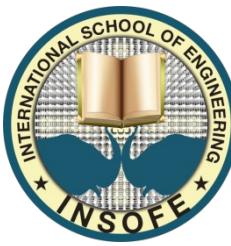
- One hidden node for each class
- 0.5 to 3 times the input neurons
- Geometric pyramid rule: Where input has m nodes and output has n nodes, the hidden layer should have  $\sqrt{m \times n}$

# Topology: Nodes and Data



$$H * (I + O) + H + O$$

5 input features and 10 units in the hidden network and 1 output, then there are 71 weights in the network.



# Topology: Other Considerations

- The weights should be  $1/100^{\text{th}}$  of the amount of training data set
  - A NN with 4 input, 2 hidden and 2 outputs require how much of data?
    - $H * (I + O) + H + O$
    - $= 2*(4+2) + 2 + 2$
    - $= 16$
- Approx 1600 samples

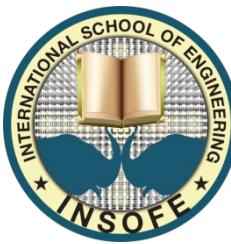


# Topology: Baum-Haussler

- Baum-Haussler rule states that

$$N_{\text{hidden}} \leq \frac{N_{\text{train}} E_{\text{tolerance}}}{N_{\text{input}} N_{\text{output}}}$$

$N_{\text{hidden}}$  is the number of hidden nodes,  
 $N_{\text{train}}$  is the number of training patterns,  
 $E_{\text{tolerance}}$  is the error,  
 $N_{\text{input}}$  and  $N_{\text{output}}$  are the number of input and output nodes respectively.



- For 9 inputs, 4 outputs, if I have 10000 test data and allow only 0.01 error, how many nodes will I need

$$N_{\text{hidden}} \leq \frac{N_{\text{train}} E_{\text{tolerance}}}{N_{\text{input}} N_{\text{output}}}$$
$$\frac{10000 * 0.01}{9 * 4} = 2.78 \geq 2$$

# Do we always have a teacher?

For training we need  $(x,y)$  pairs.

$x$  being input,  $y$  corresponding label,  
hence termed supervised learning.

What if there is no  $y$  but only  $x$  ?



# Types of learning

- Supervised learning: Training data (input, label).
- Unsupervised learning: Only input without labels
- Semi-supervised learning: More inputs without labels and few with labels.



# Unsupervised learning

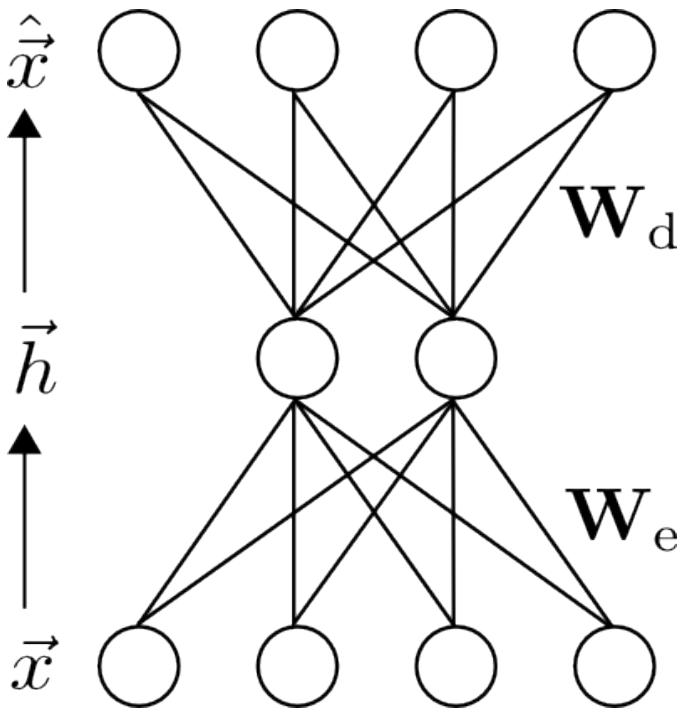
- Learning by reconstruction.
- Compression followed by reconstruction.

$$\vec{h} = W_e \vec{x}$$

$$\hat{\vec{x}} = W_d \vec{h}$$

- Error or cost function now is

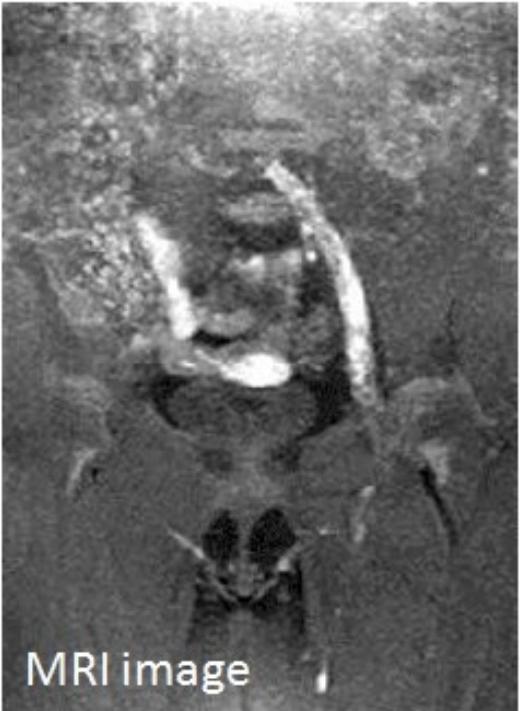
$$E = ||\vec{x} - \hat{\vec{x}}||_2^2$$



Autoencoder



# Autoencoder applications



MRI image

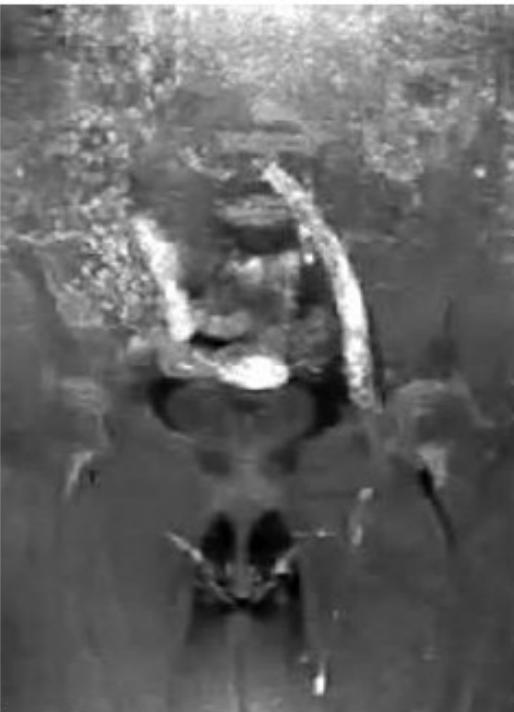


Image denoising

# Applications of CNNs: Image annotation

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
			
A person riding a motorcycle on a dirt road.	Two dogs play in the grass.	A skateboarder does a trick on a ramp.	A dog is jumping to catch a frisbee.
			
A group of young people playing a game of frisbee.	Two hockey players are fighting over the puck.	A little girl in a pink hat is blowing bubbles.	A refrigerator filled with lots of food and drinks.
			
A herd of elephants walking across a dry grass field.	A close up of a cat laying on a couch.	A red motorcycle parked on the side of the road.	A yellow school bus parked in a parking lot.

--<http://googleresearch.blogspot.in>



# Applications of CNNs: Video classification



*Large-scale Video Classification with Convolutional Neural Networks,*  
--Andrej Karpathy  
(Google/Stanford)

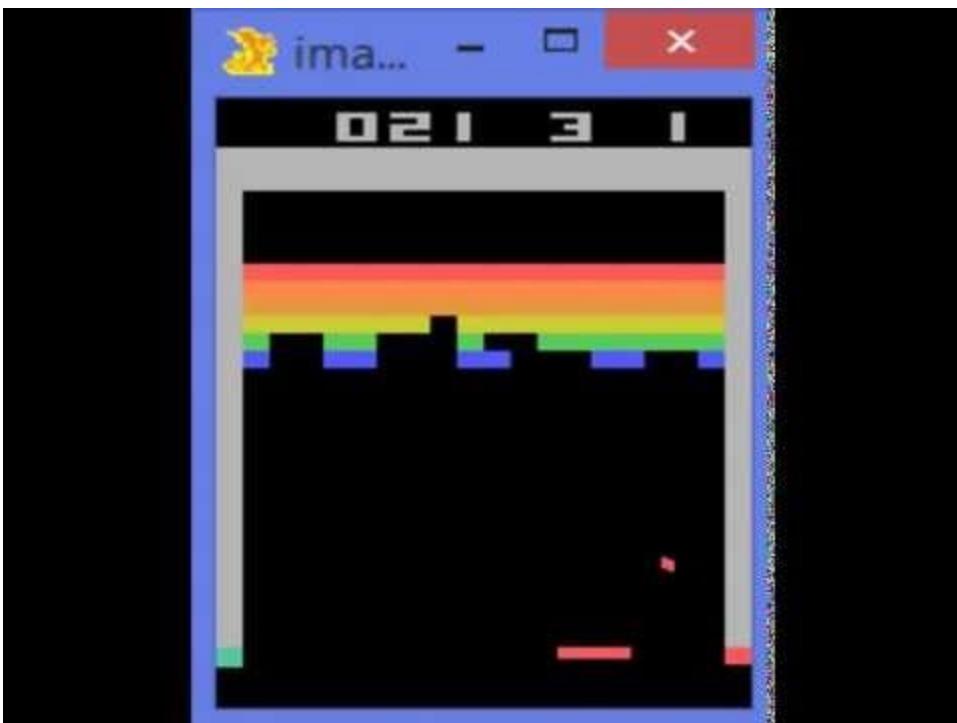


# **Applications of CNNs: Natural language processing (NLP)**

A word cloud centered around the term "language". The words are in various sizes and shades of gray. Key words include "language", "machine", "speech", "words", "evaluation", "systems", "NLP", "natural", "processing", "recognition", "tasks", and "understanding". Other visible words include "algorithms", "data", "common", "example", "computer", "determine", "automatic", "complexity", "analysis", "developed", "English", "enables", "difficult", "compared", "considered", "human", "input", "possible", "especially", "hand-written", "morphology", "models", "information", "names", "number", "linguistics", "problems", "output", "statistical", "sentence", "segmentation", "standard", "sentences", "written", "task", "set", "translation", "project", "word", "training", "research", "semantics", and "text".

- semantic parsing
  - search query retrieval
  - sentence modeling
  - classification
  - prediction

# Applications of CNNs: Atari games



Human-level control through deep reinforcement learning, Google DeepMind

# Applications of CNN: Autonomous driving

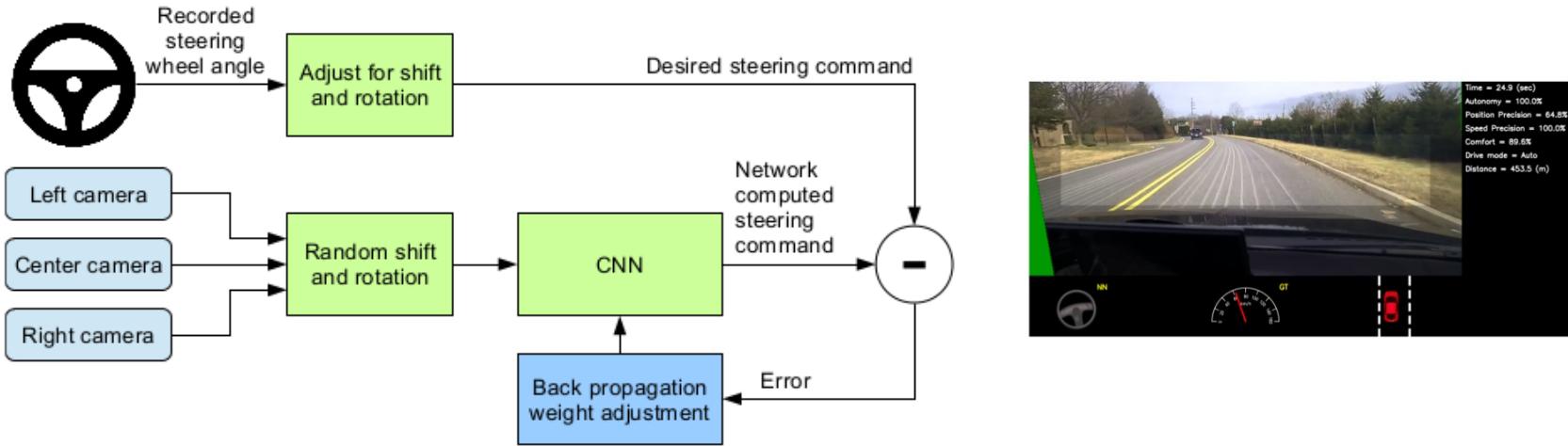


Figure 2: Training the neural network.

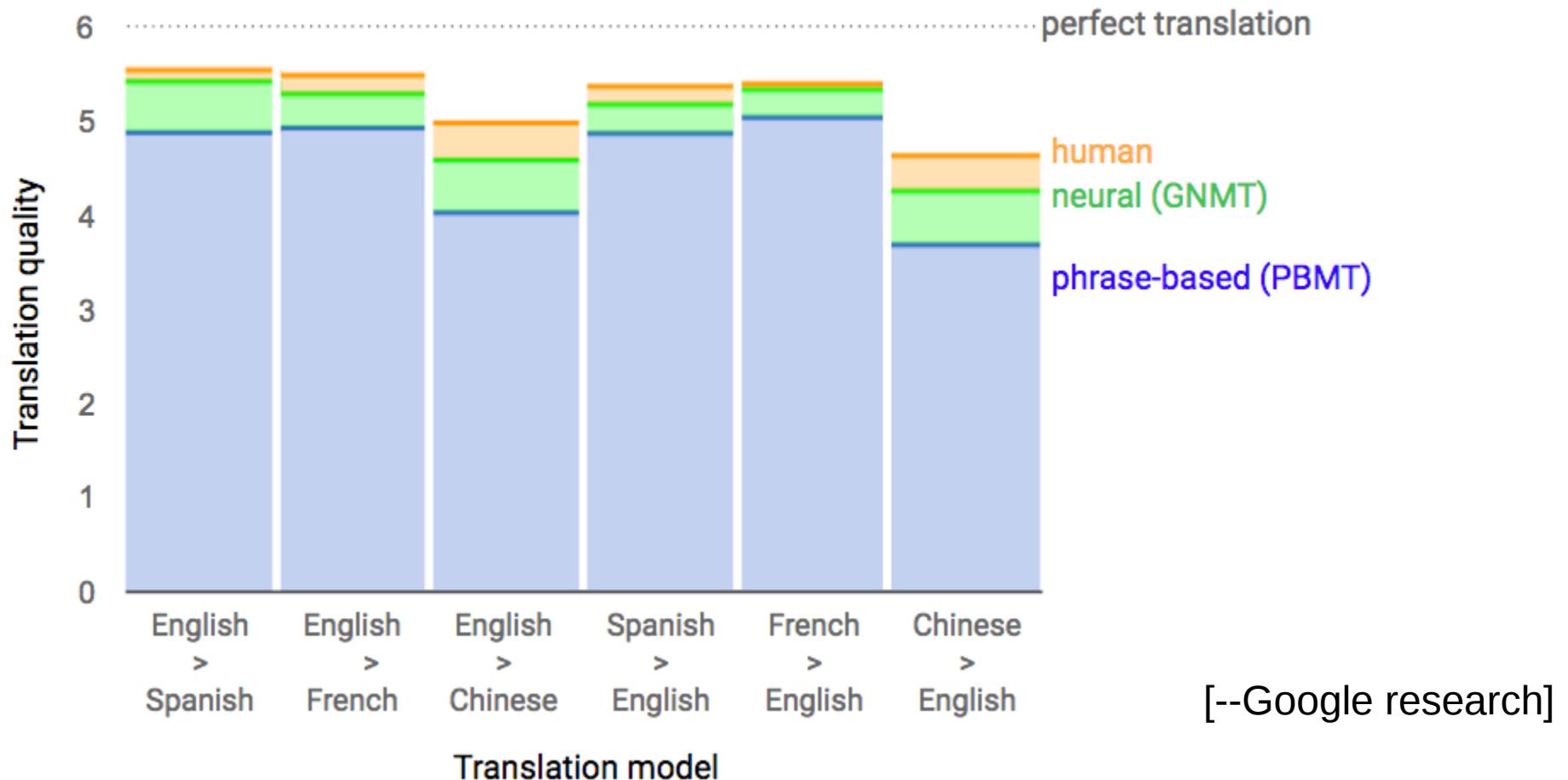
Once trained, the network can generate steering from the video images of a single center camera. This configuration is shown in Figure 3.

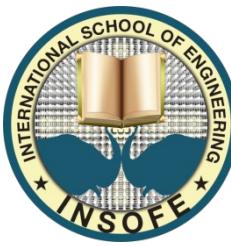
[NVIDIA Corporation]





# Neural machine translation (NMT)





# History of Artificial Neural Networks

- A computational model for biological neural networks was created in 1943 by Warren McCulloch and Walter Pitts.
- One of the key breakthroughs was the backpropagation algorithm by Werbos in 1975.
- Parallel distributed computing, introduced by David E. Rumelhart and James McClelland.
- The neocognitron is a hierarchical, multilayered artificial neural network proposed by Kunihiko Fukushima in the 1980. (Basis of CNNs)
- Neural networks were overshadowed by the popularity of other ML methods in the 1990s.
- Some people like Yann LeCun, Geoffrey Hinton, Yoshua Bengio and others continued believing in artificial neural networks.



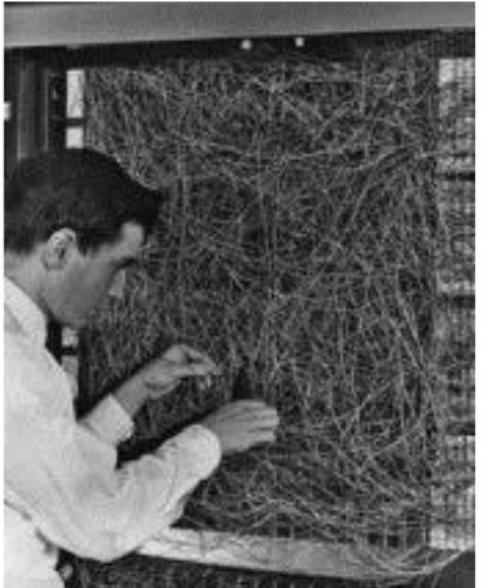
# Revival of Artificial Neural Networks

Three main reasons:

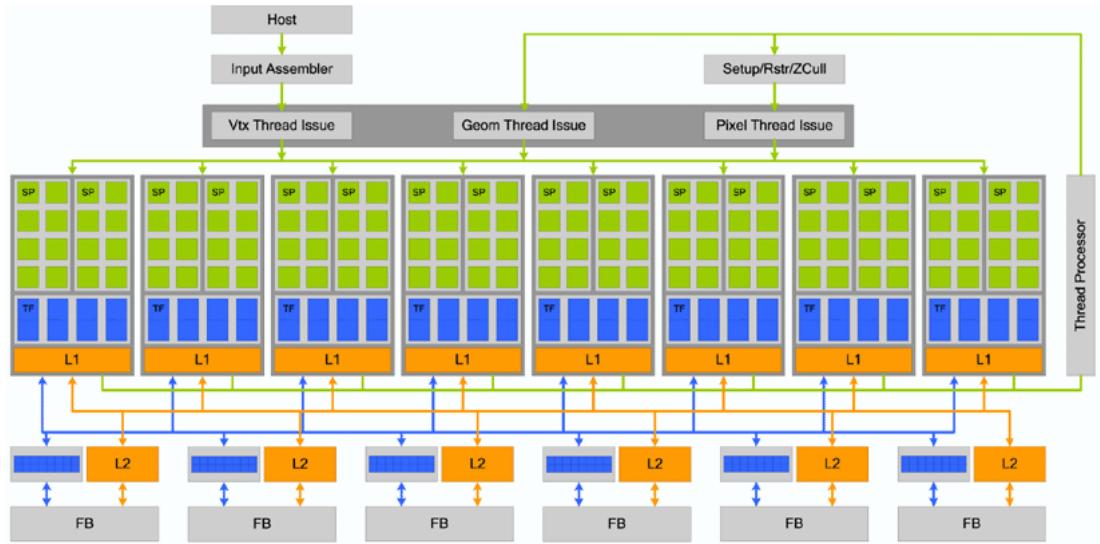
- Better training approaches,  
thanks to **scientists**.
- Increased availability of training data,  
thanks to **digital media and internet**.
- computational power  
thanks to revolution in computing hardware: **Graphical processing units (GPU)**.



## Hardware: Early perceptron



# Hardware: Modern GPUs



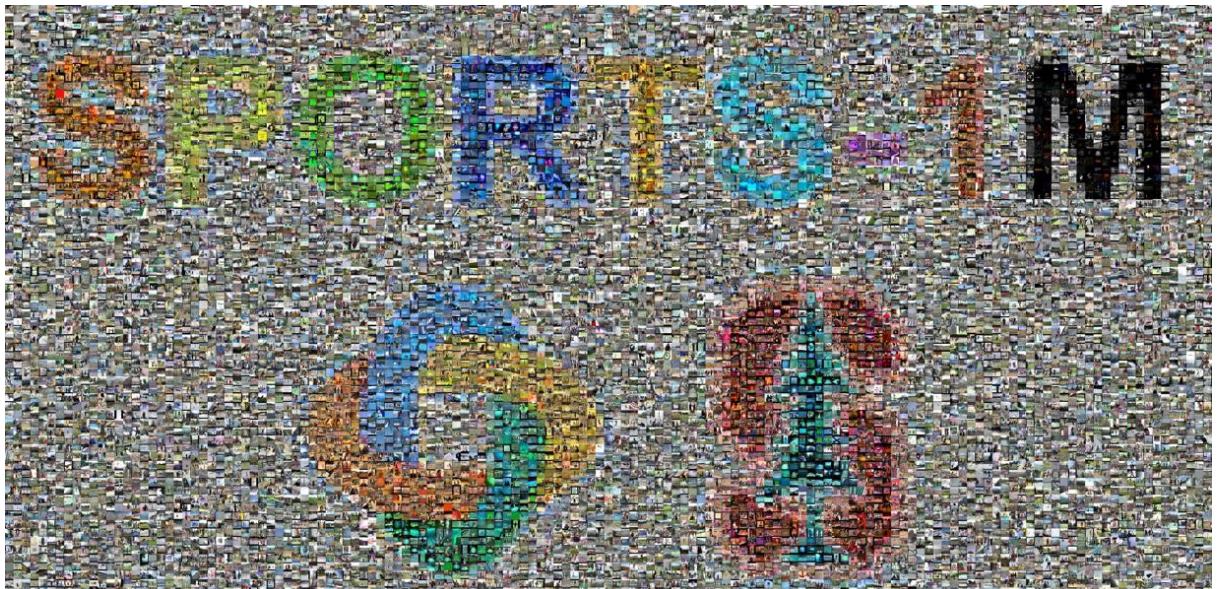
GPU computing is very efficient for matrix algebra:  
 Most mathematical computations in Artificial Neural Networks are matrix algebra.



## Big data and internet



1 million labeled images

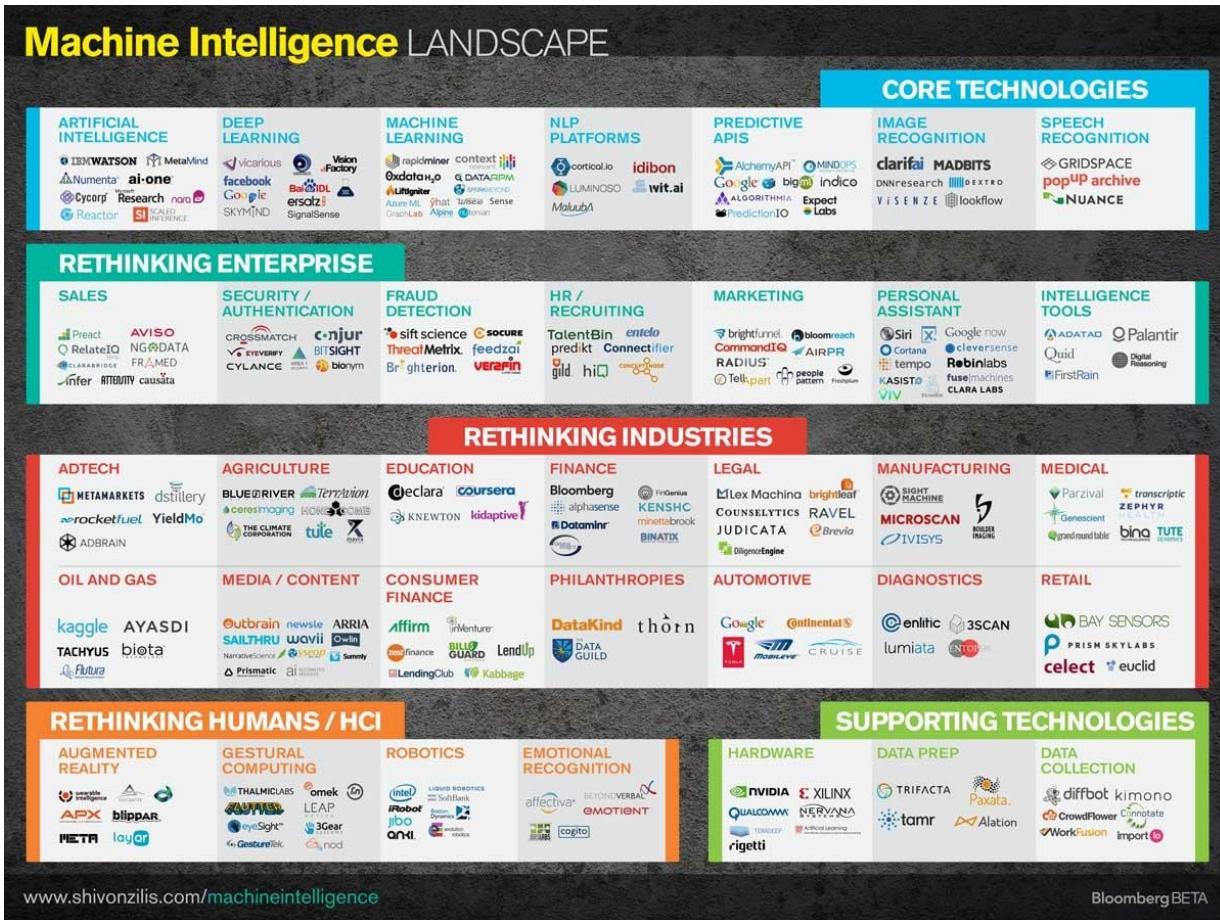


The Sports-1M dataset

# Industries and Machine learning

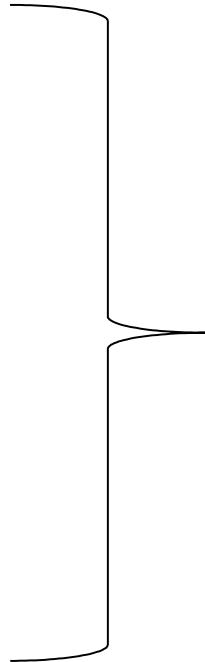


# Machine Intelligence LANDSCAPE



# How can you contribute to the society?

- Bioinformatics
- Brain-machine interfaces
- Cheminformatics
- Classifying DNA sequences
- Detecting credit card fraud
- Internet fraud detection
- Medical diagnosis
- Structural health monitoring
- Security and surveillance
- Weather prediction
- Agricultural productivity analysis



ML expert



# **FEED FORWARD IS JUST ONE ARCHITECTURE**

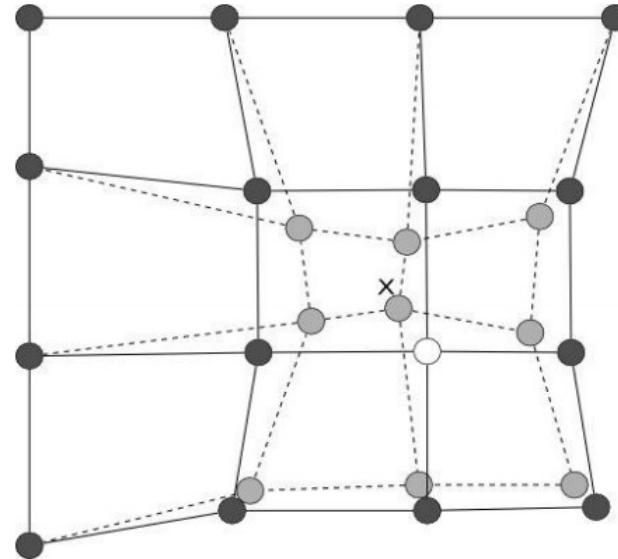
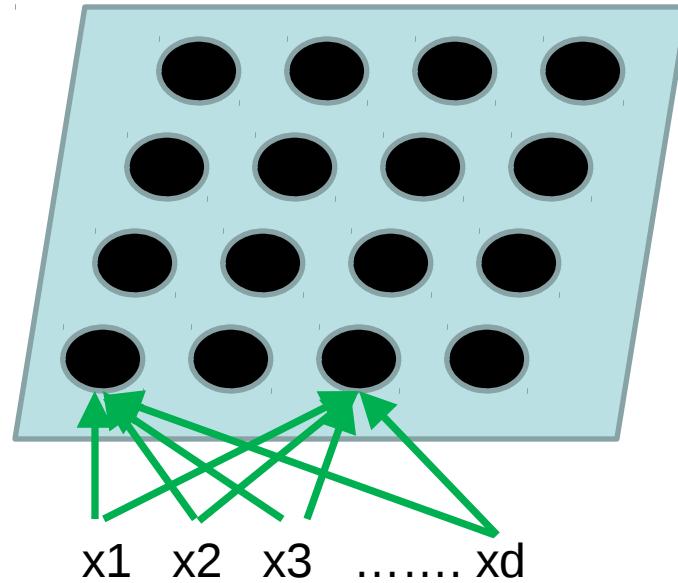


# Self organizing maps

- Invented by Teuvo Kohonen, a professor of the Academy of Finland.
- Kohonen described SOM as a visualization and analysis tool kit for high dimensional data.
- SOMS are however used for clustering, dimensionality reduction and classification.

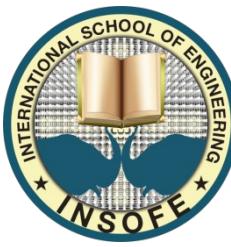
# The Basics

<http://ssdi.di.fct.unl.pt/aadm/aadm1011/slides/LectureSOM.pdf>



- Sample
- Best matching unit
- Original nodes
- Nodes in weight space

- Several nodes arranged in a grid at uniform distance.
- Weight from each input (1...d) to node (1...n)
- Training modifies weights:
  - Similar inputs activate neighboring nodes
  - After training, distance between weights indicates how close the nodes are



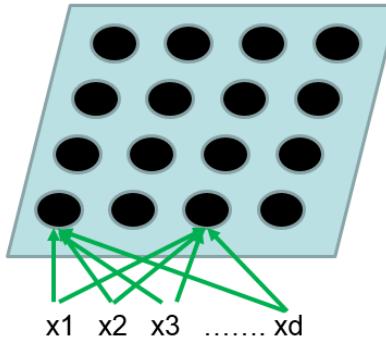
- Which of the two nodes 1:(0.1, 0.2, 0.3) and 2: (0.9, 0.1, 0.1) are closer to the input vector  $I=(1, 0, 0)$ .
- $Distance1 = \sqrt{(1 - 0.1)^2 + (0 - 0.2)^2 + (0 - 0.3)^2} = 0.96$

$Distance2 = 0.17$

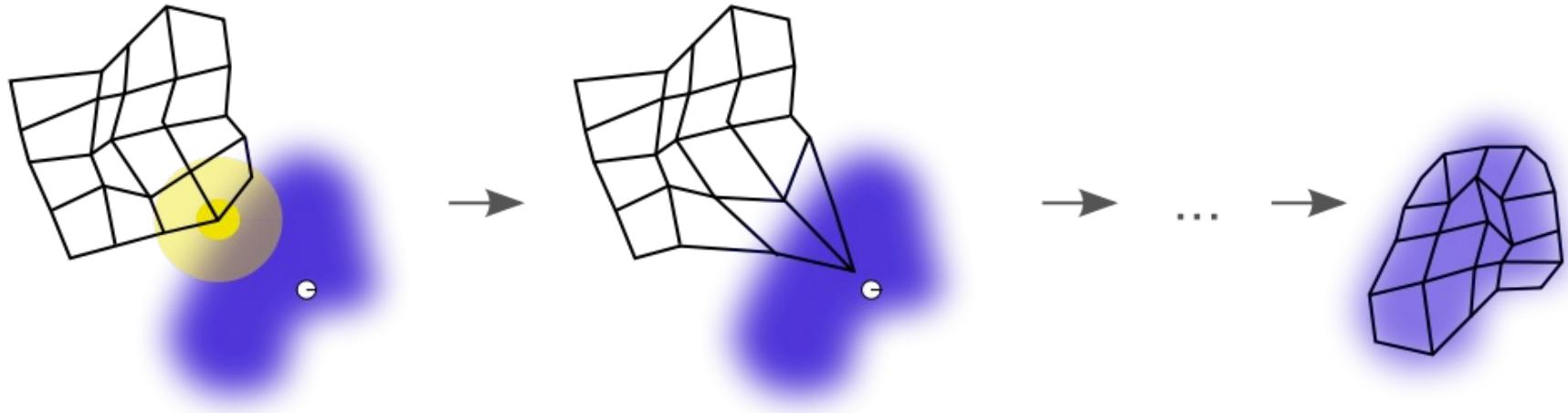
Hence, BMU is 2

# Training?

- Start with random weights
- For each input sample ( $X$ ):
  - Find  $d_j(X) = \sum_{i=1}^D (x_i - w_{ij})^2$   $j$  is from 1 to  $N$ ,
  - B=Best Match Unit/Node which gives least  $d_j(X)$
  - Transfer factor from B to node j:  $T_{B,j} = \exp(-S_{B,j}^2/2\sigma^2)$ 
    - $S_{k,j}$  is dist from node k to j in the grid space
  - Update each node weight:  $\Delta w_{ij} = \eta * T_{B,j} * (x_i - w_{ij})$
- Shuffle the training set, repeat till weights stop changing



# Training?

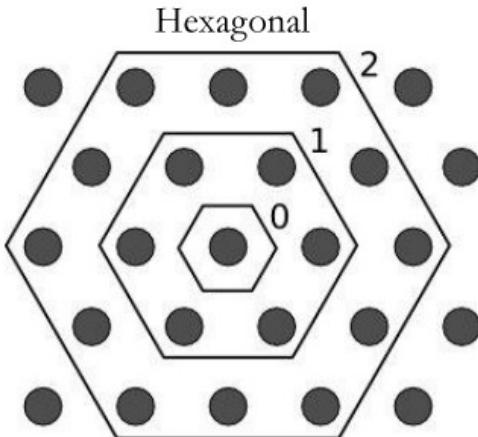
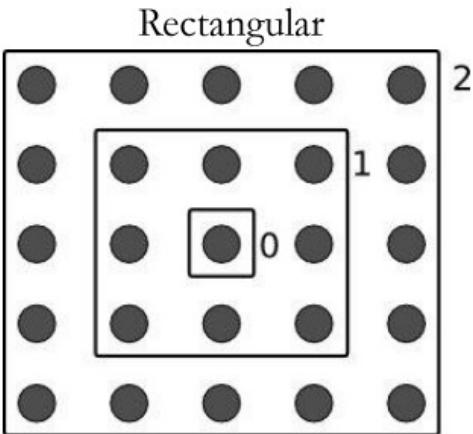


An illustration of the training of a self-organizing map. The blue blob is the distribution of the training data, and the small white disc is the current training datum drawn from that distribution. At first (left) the SOM nodes are arbitrarily positioned in the data space. The node (highlighted in yellow) which is nearest to the training datum is selected. It is moved towards the training datum, as (to a lesser extent) are its neighbors on the grid. After many iterations the grid tends to approximate the data distribution (right)

[https://en.wikipedia.org/wiki/Self-organizing\\_map](https://en.wikipedia.org/wiki/Self-organizing_map)

# SOM Parameters

<http://ssdi.di.fct.unl.pt/aadm/aadm1011/slides/LectureSOM.pdf>

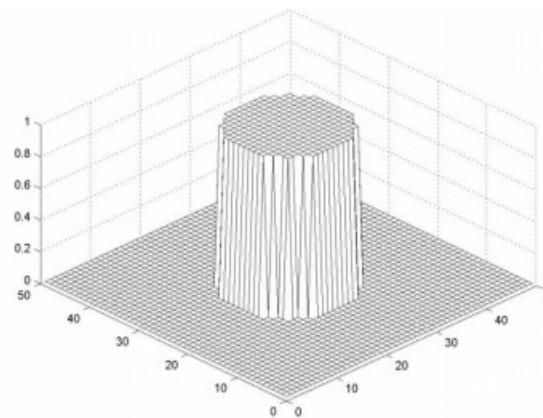
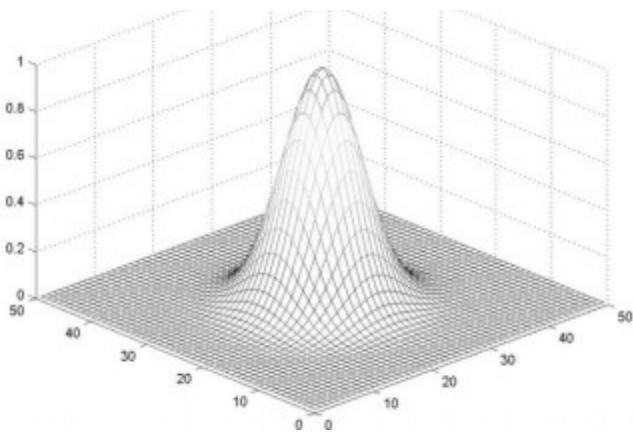


Classical vs Batch

Weights updated at each sample

Weights updated at each epoch

Node distance:  $S_{j,k}$  can be Manhattan or Euclidian  
Rectangular/Hexagonal



Radius set to entire SOM,  
reduced over time

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_\sigma}\right)$$

Reducing influence:  $T_{j,B} = \exp(-S_{j,B}^2 / 2\sigma^2)$   
 $\sigma^2$  may be reduced with epochs

Constant influence:

$$T_{j,B} = 1 \text{ if } \|S_{j,k}\|^2 < \delta(t) \text{ else } 0$$

$$\tau_\sigma = \frac{N}{\log(\sigma_0)}$$



# How to use SOM?

Pre-process data:  
Standardization  
Remove invalids  
Remove outliers

Build Self  
Organizing Maps

Generate  
heatmaps and  
clusters to  
interpret data

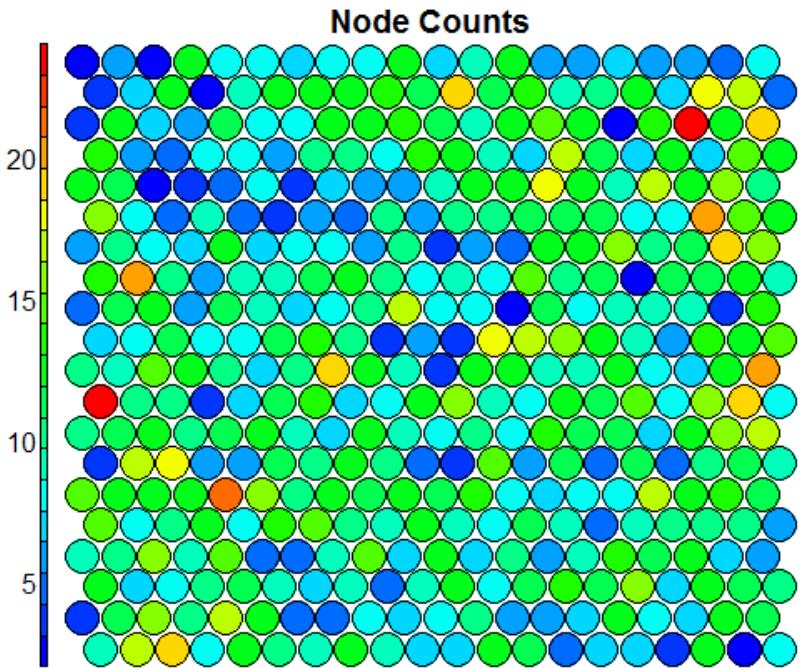
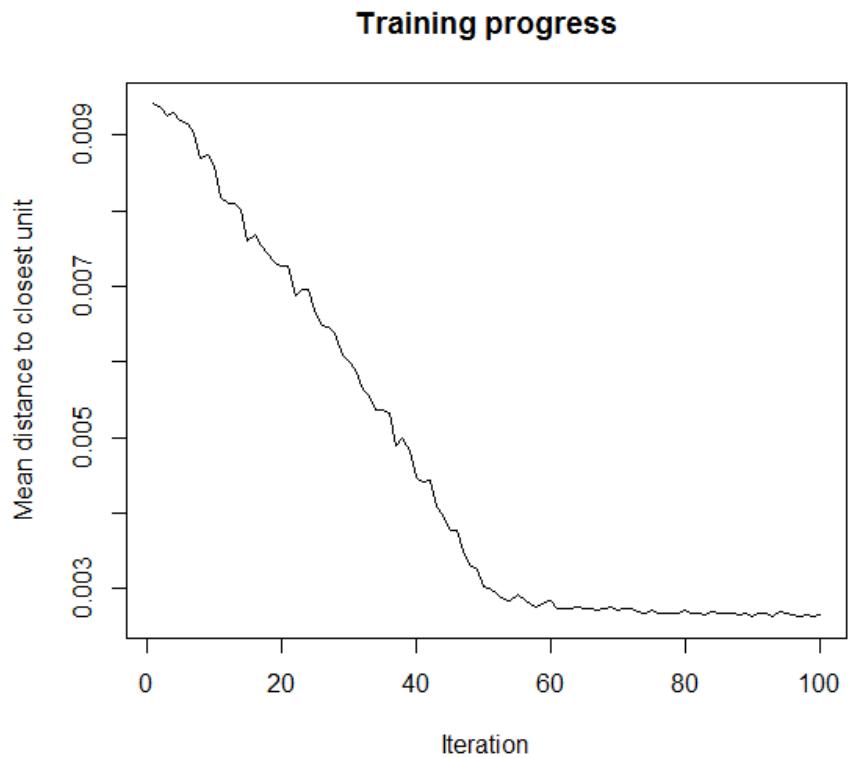
- In a random class, if students with similar features sit as close as possible to each other
  - Age, grades, hobbies etc
- If you examine their grades with respect to new positions, this will be a SOM



# SOMs in Practice

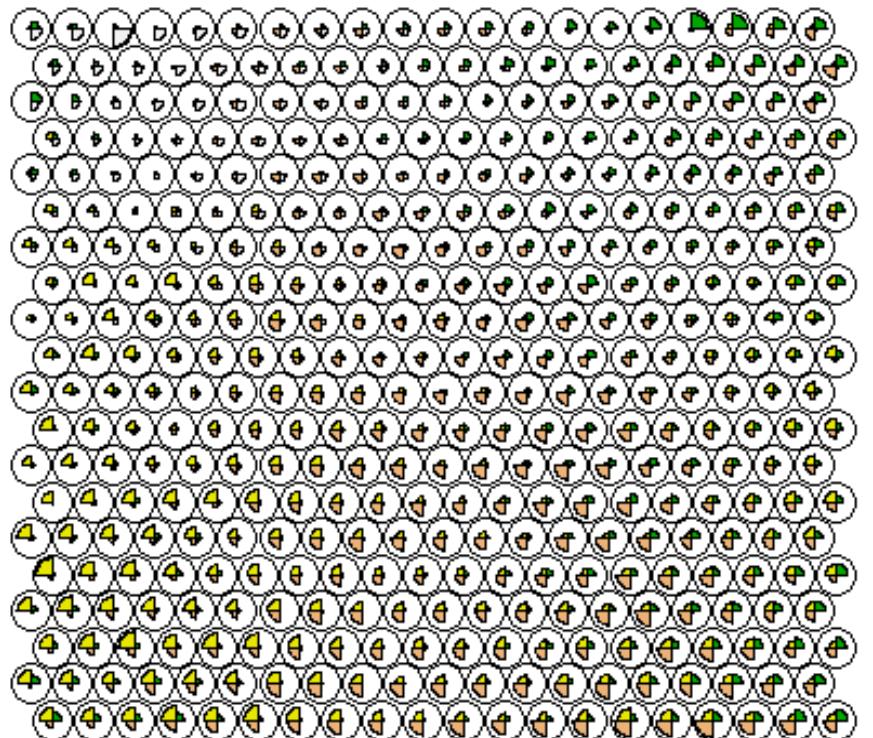
- Example from a tutorial by Shane Lynn
  - Other examples:
    - <http://www.r-bloggers.com/self-organising-maps-for-customer-segmentation-using-r/>
    - [http://manuals.bioinformatics.ucr.edu/home/R\\_BioCondManual#clustering\\_primer](http://manuals.bioinformatics.ucr.edu/home/R_BioCondManual#clustering_primer)
- Census data from Dublin:
  - ~4000 localities
  - Average values for: age, household size, education, car ownership,
  - Percentage values for: health, rent, employment, internet, marital status

# SOM on Census Data

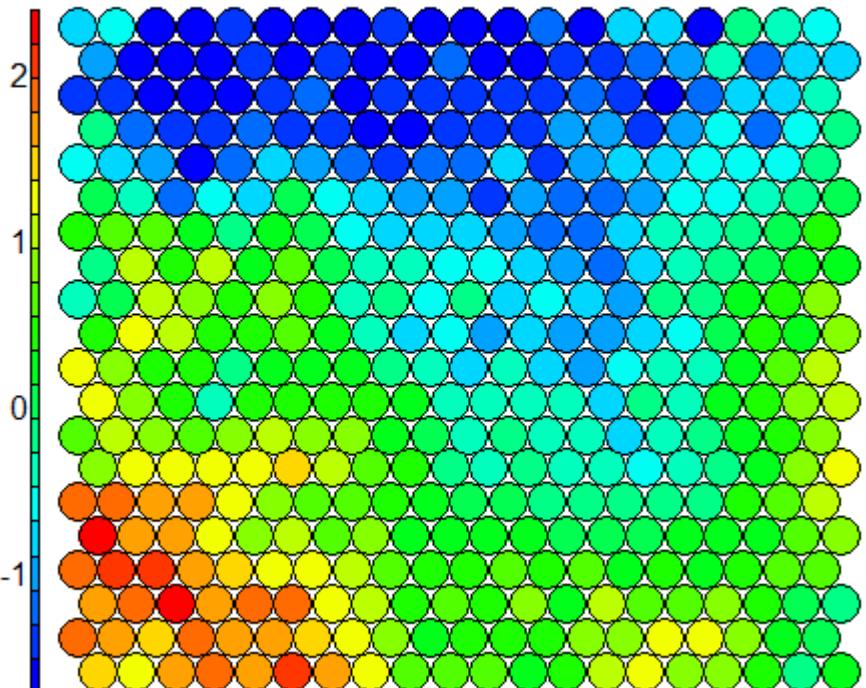


Number of data points mapping to each node.  
Many blues/red indicate too large/small model

# SOM on Census Data

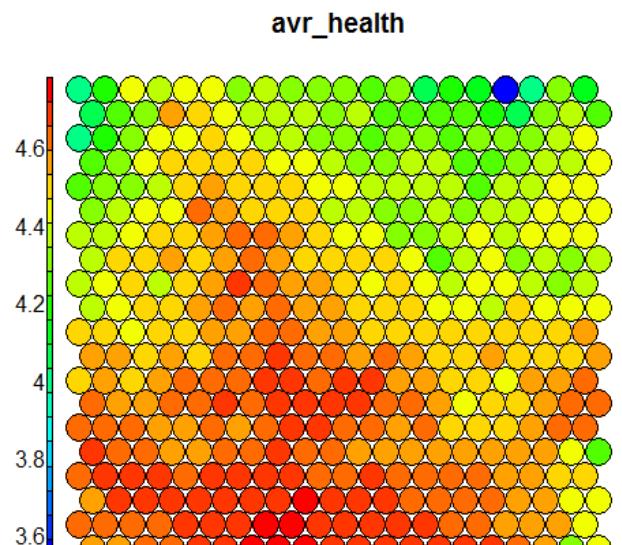


avr\_education\_level

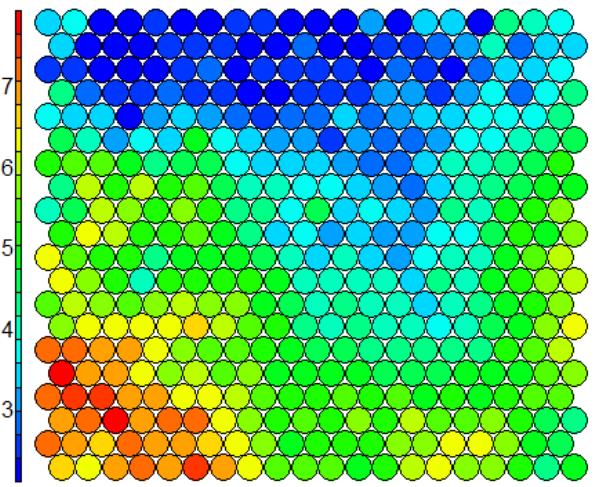


Heat map of average education level

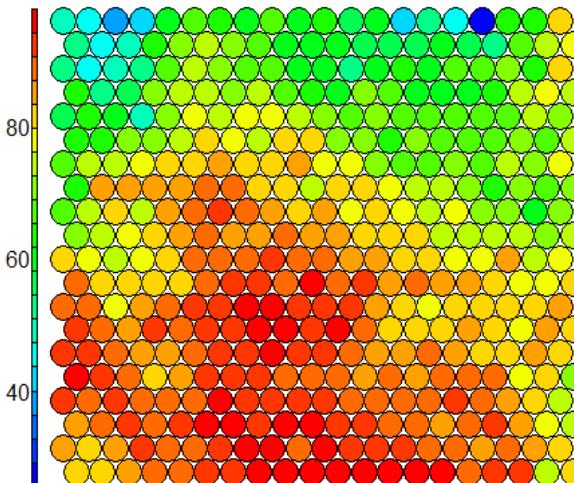
# SOM on Census Data



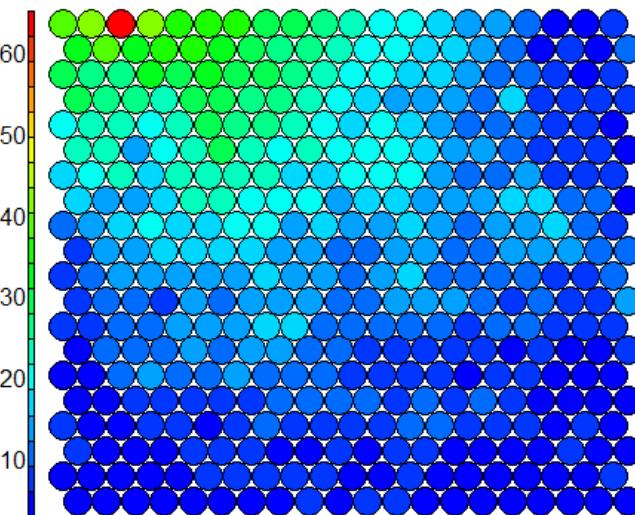
avr\_education\_level



internet\_percent



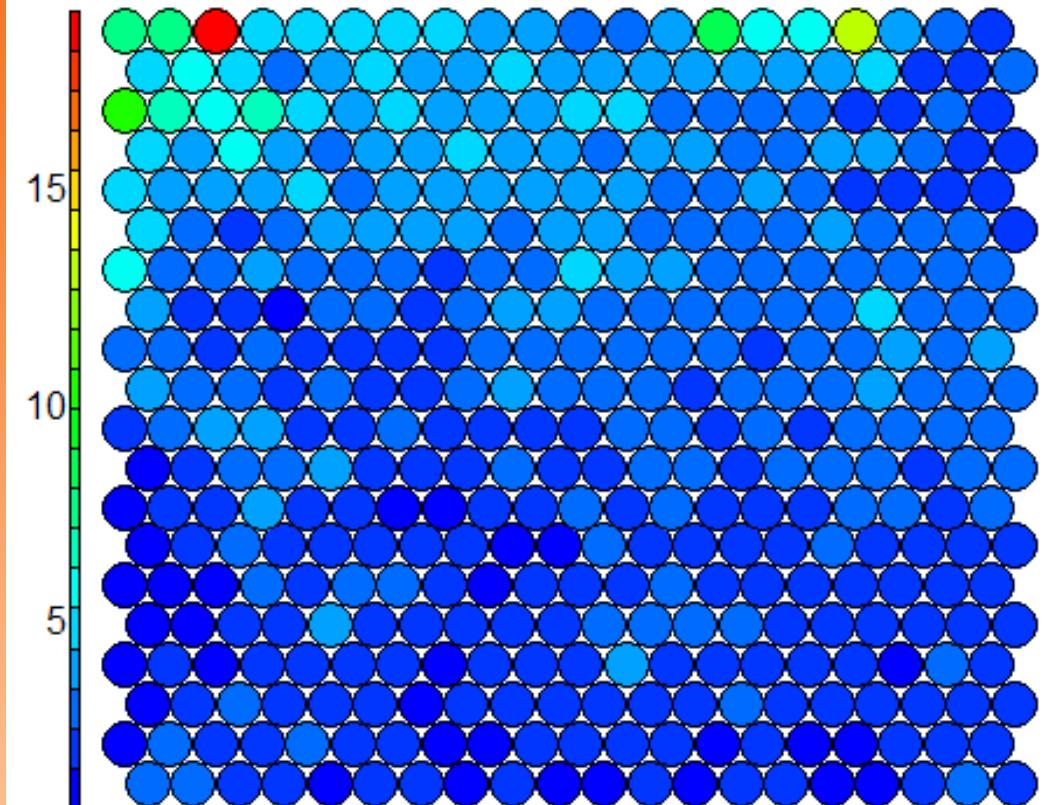
unemployment\_percent



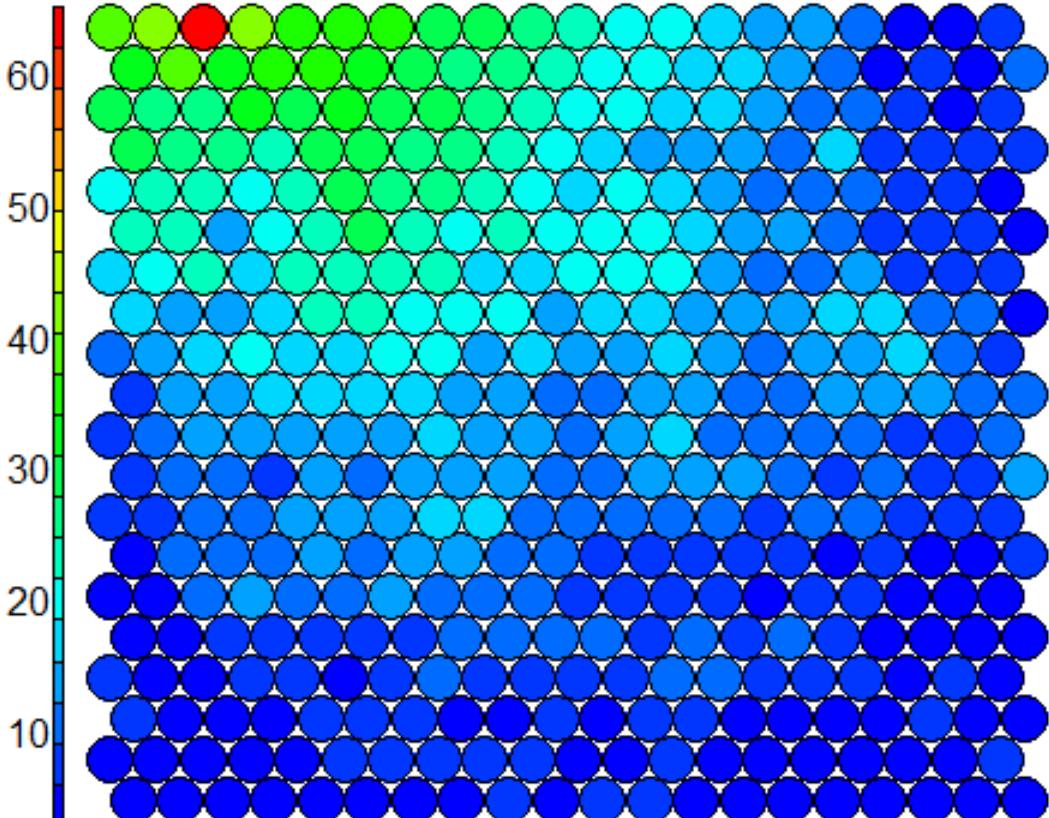


# SOM on Census Data

separated\_percent

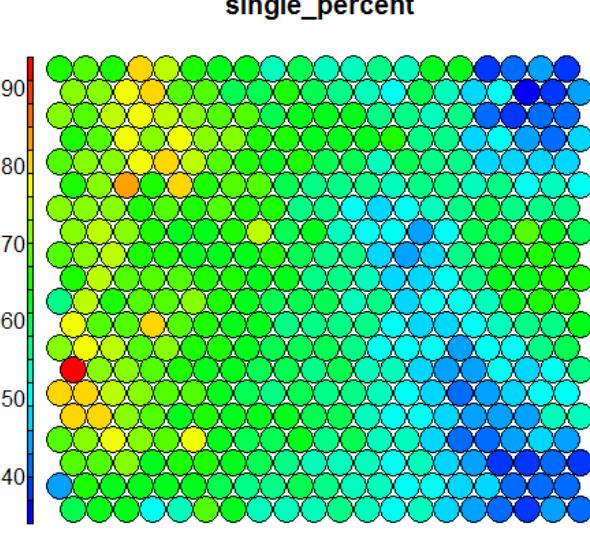
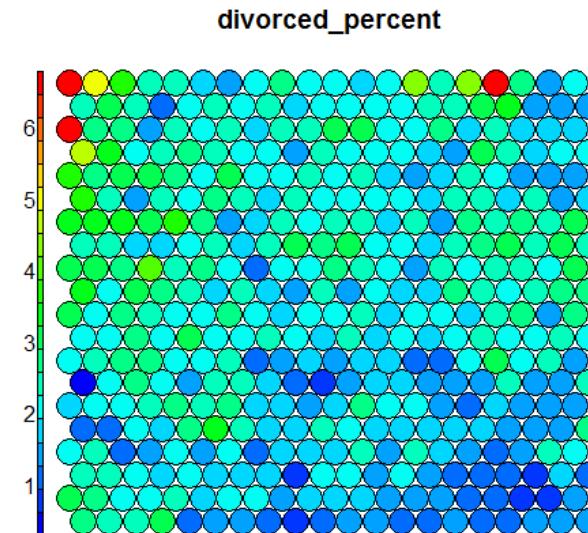
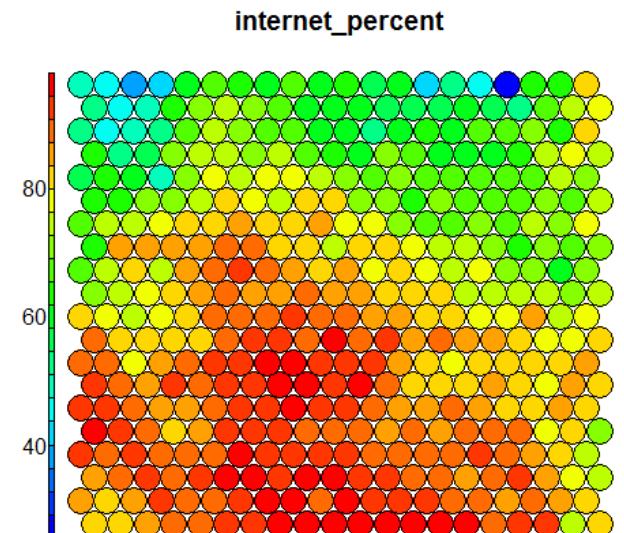
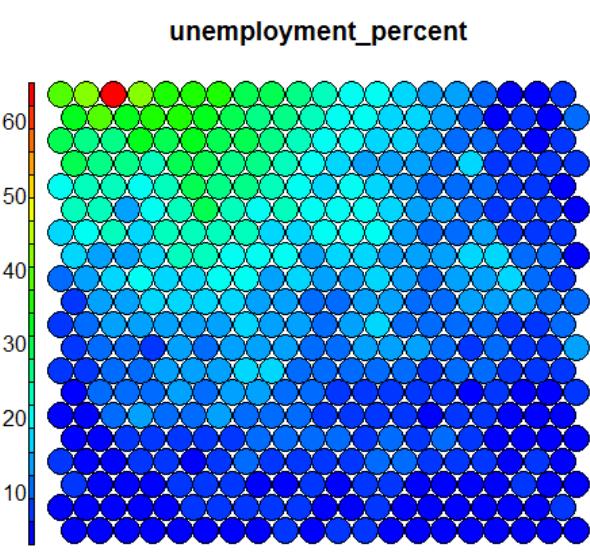
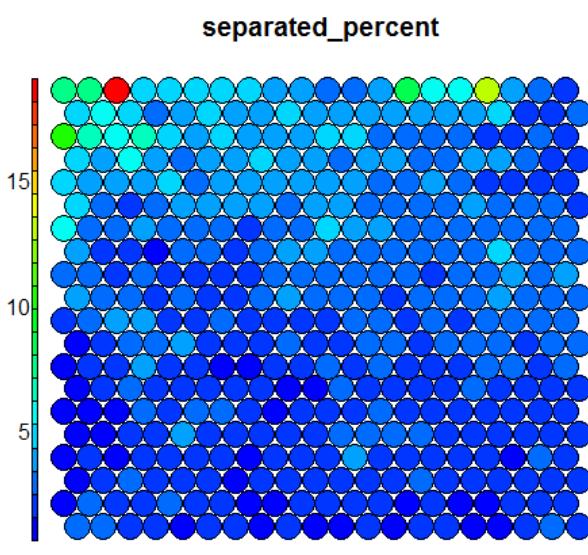
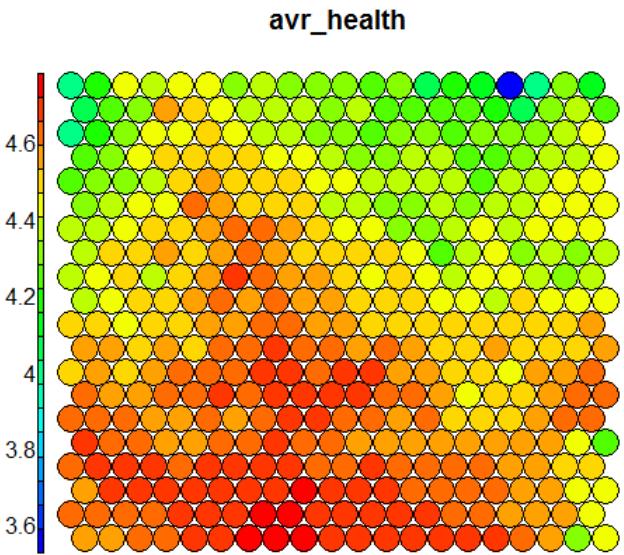


unemployment\_percent



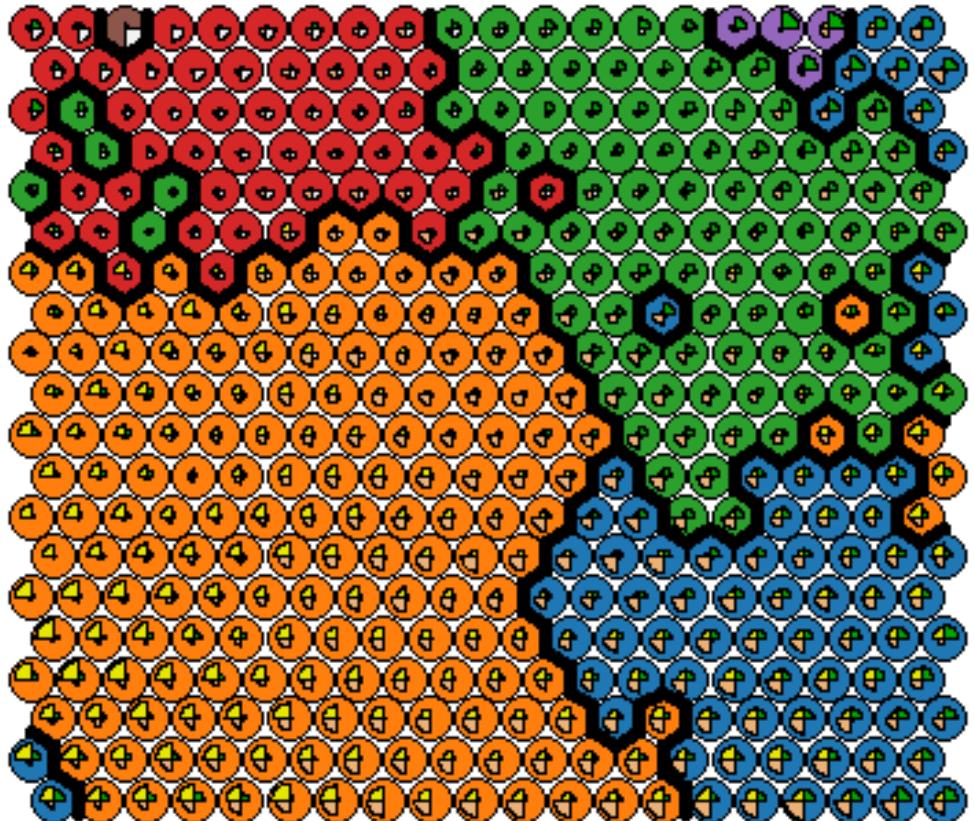


# SOM on Census Data



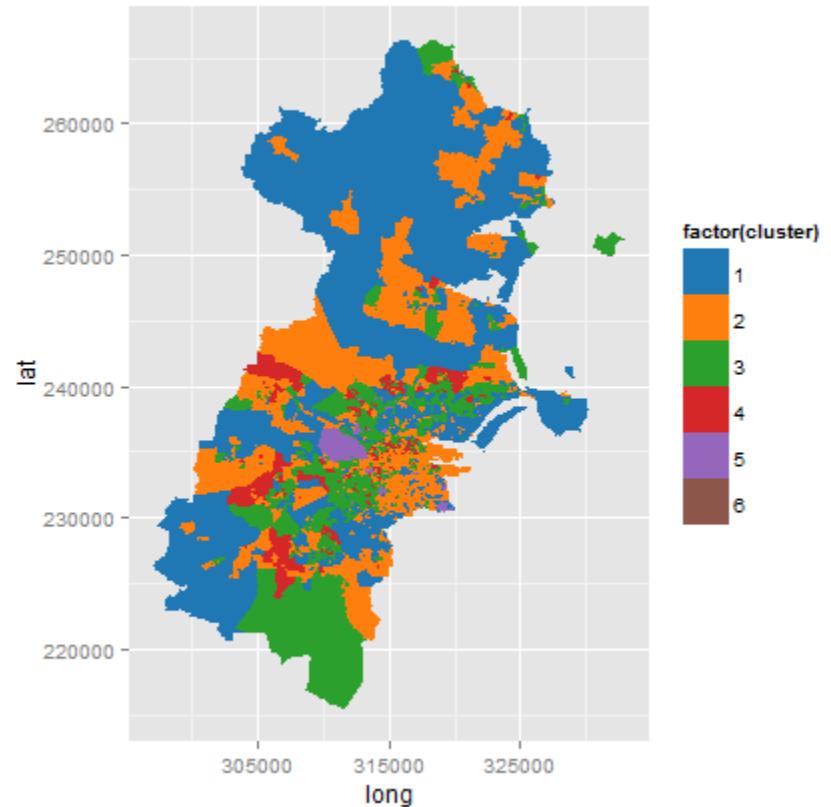
# SOM on Census Data

Clusters



- |                                       |                     |                                       |                      |
|---------------------------------------|---------------------|---------------------------------------|----------------------|
| <span style="color: green;">■</span>  | avr_age             | <span style="color: orange;">■</span> | avr_num_cars         |
| <span style="color: yellow;">■</span> | avr_education_level | <span style="color: white;">□</span>  | unemployment_percent |

Cluster Nodes using Hierarchical clustering



Clusters on the map of Dublin



# References for Kohonen SOMs

- Online tutorials:

- <https://www.youtube.com/watch?v=LjJeT7rwvF4>
- <http://www.cs.bham.ac.uk/~jxb/INC/I16.pdf>
- <http://www.r-bloggers.com/self-organising-maps-for-customer-segmentation-using-r/>
- <http://ssdi.di.fct.unl.pt/aadm/aadm1011/slides/LectureSOM.pdf>
- [http://www.academia.edu/11322466/Using\\_R\\_to\\_Map\\_Crime\\_Density\\_and\\_Demographics\\_in\\_Boston\\_from\\_2012-2014](http://www.academia.edu/11322466/Using_R_to_Map_Crime_Density_and_Demographics_in_Boston_from_2012-2014)