

Deep Learning Techniques for Music Generation

2. Introduction to Deep Learning

Jean-Pierre Briot

`Jean-Pierre.Briot@lip6.fr`

Laboratoire d'Informatique de Paris 6 (LIP6)
Sorbonne Université – CNRS



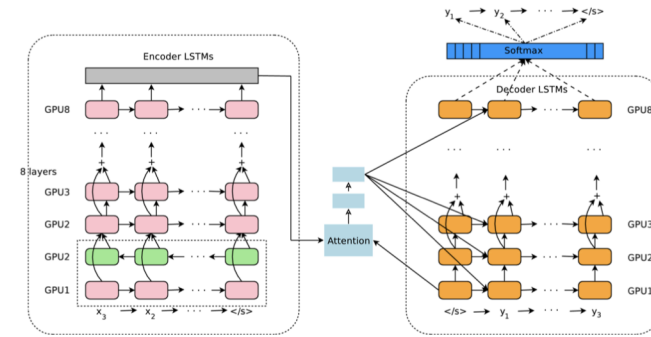
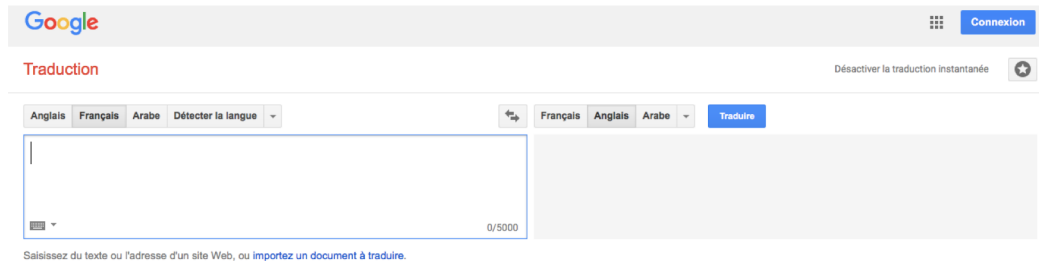
Programa de Pós-Graduação em Informática (PPGI)
UNIRIO



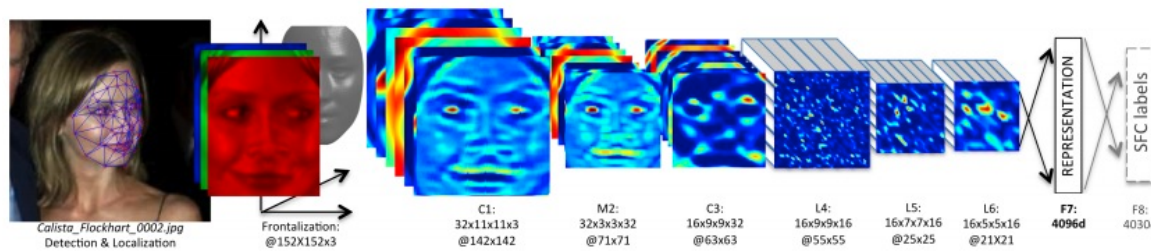
Deep Learning

Deep Learning – Already There

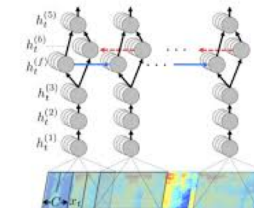
- Translation, ex: Google Translate



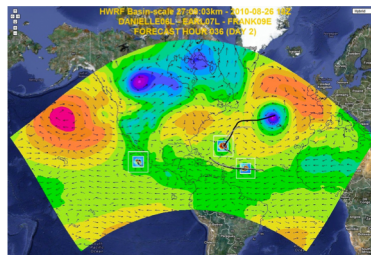
- Image Recognition, ex: Facebook



- Speech Recognition, ex: Apple Siri, Amazon Echo...

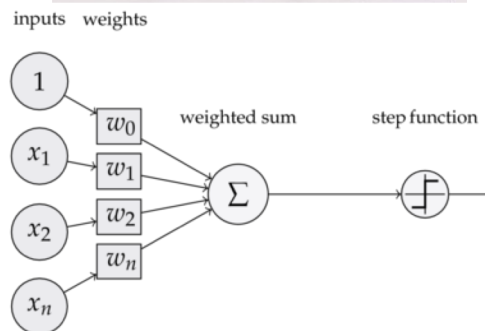
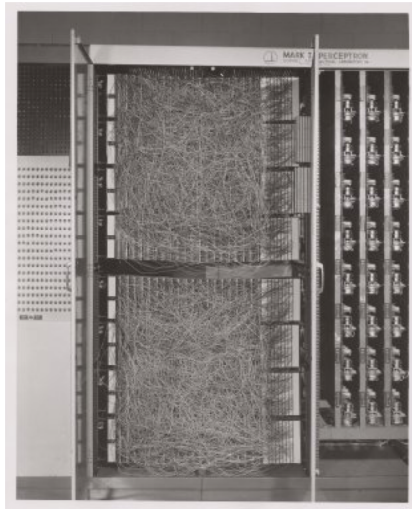


- Weather Prediction



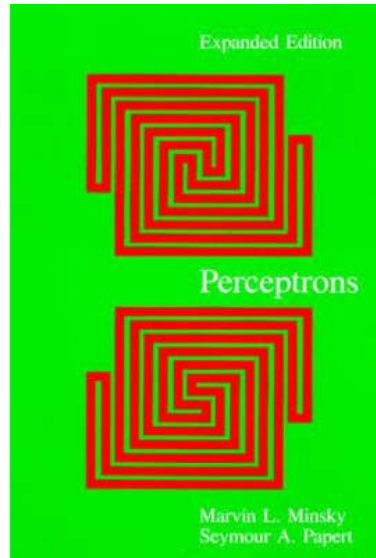
History

History: From Perceptron to Artificial Neural Networks to Deep Learning (1/4)



Perceptron
[Rosenblatt 1957]

Jean-Pierre Briot

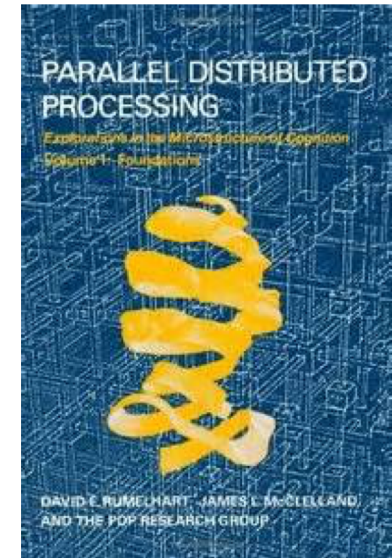


Perceptrons (Book)
[Minsky & Papert 1969]

Linear Separable only
XOR counter example

0	1
1	0

Deep Learning – Music Generation – 2019

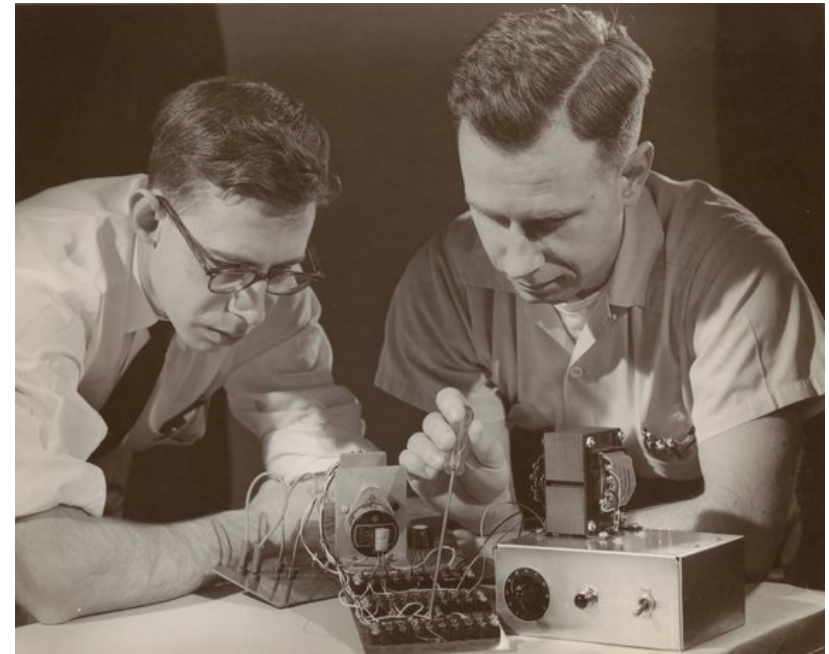


PDP (Books)
[Rumelhart et al. 1986]

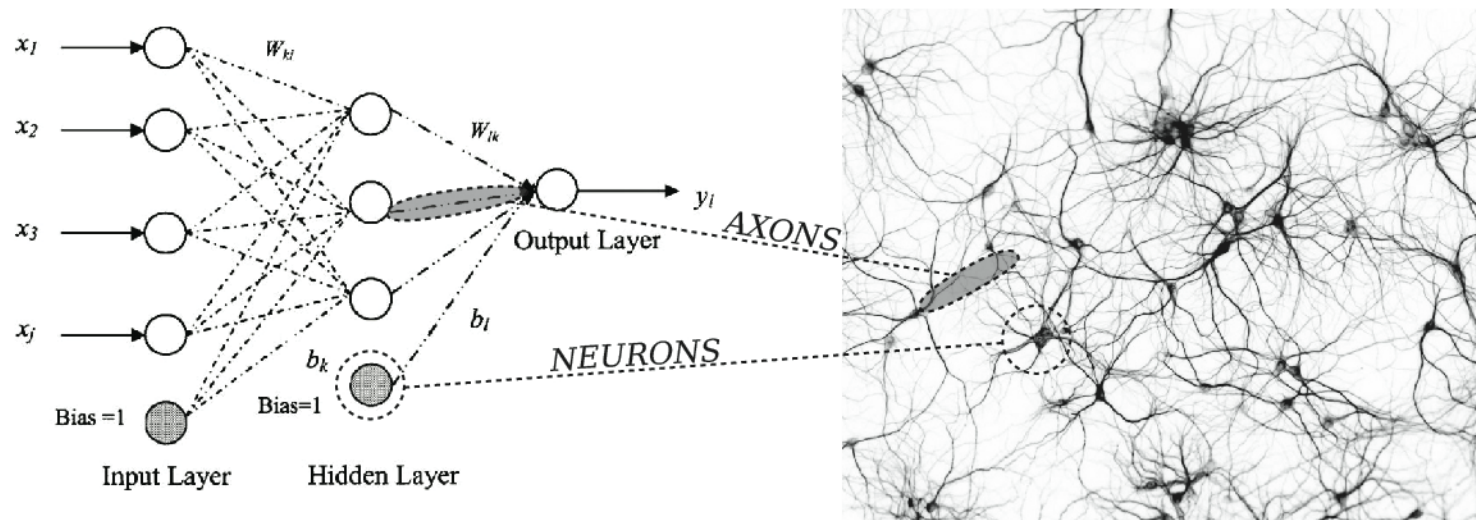
Multi-layer networks
Backpropagation

Deep Learning **Bio-inspired** or/and **Regression-based** ?

- Historically Perceptron bioinspired
- Aimed at Character Recognition

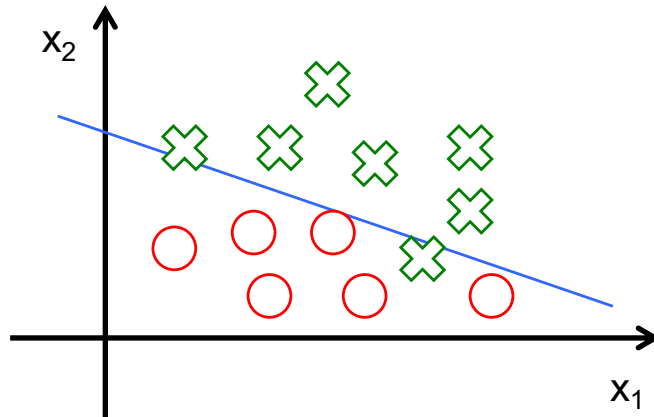


NEURAL NETWORK MAPPING

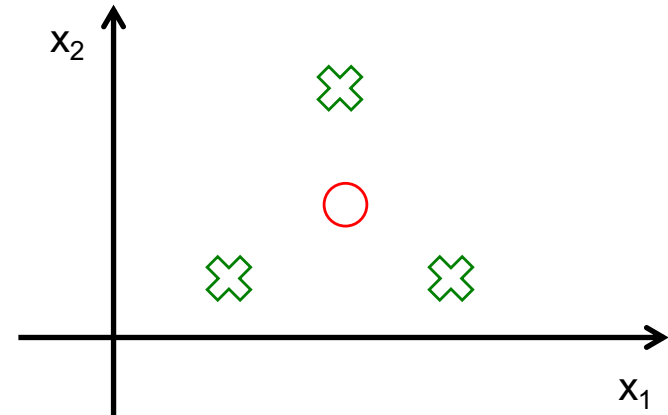
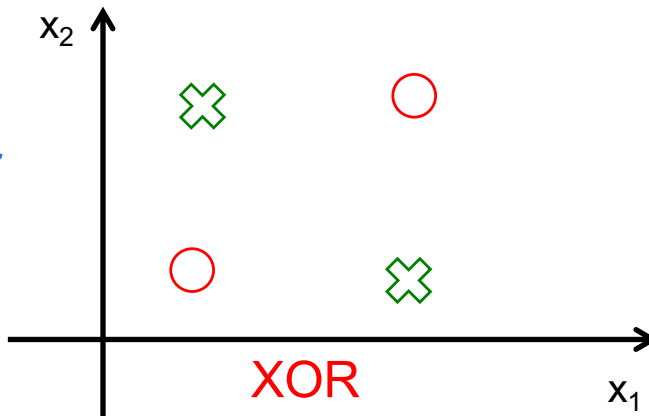


Linear vs Non Linear Decision Boundary

- Linear



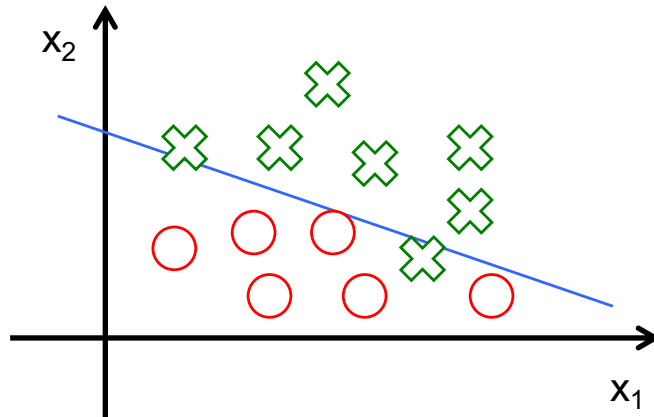
- Non Linear



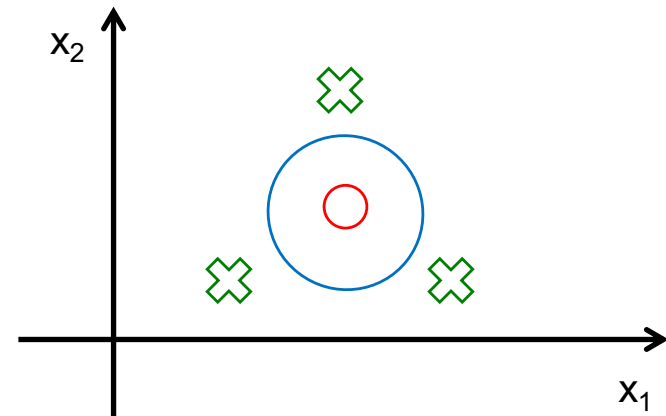
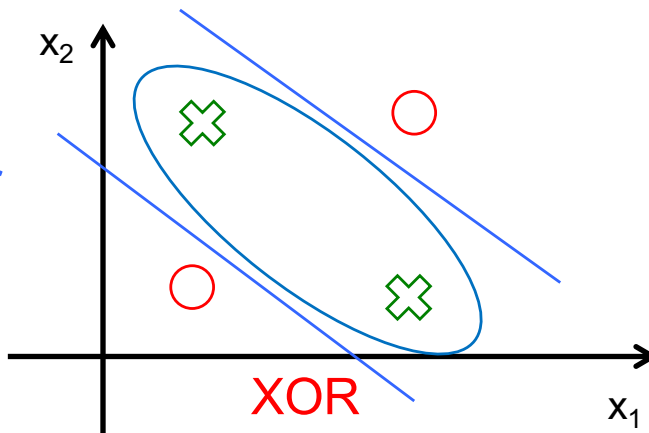
- Argument (XOR) used by [Minsky & Papert 1969] to criticize Perceptrons [Rosenblatt 1957] (and advocate Symbolic Artificial Intelligence)
- This stopped research on Perceptrons/Neural Networks for a long while
 - until Hidden Layers and Backpropagation or/and Kernel Trick (see later)

Linear vs Non Linear Decision Boundary

- Linear

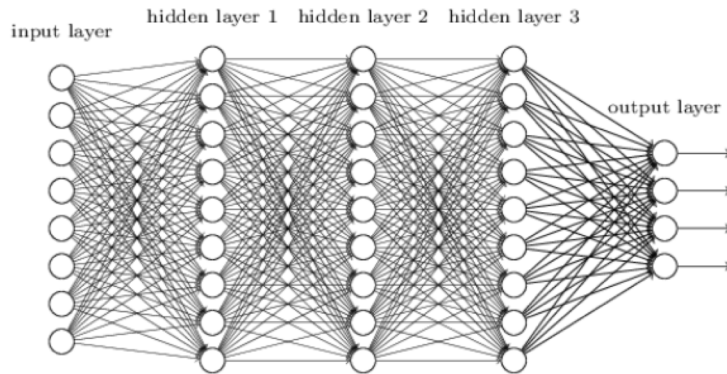


- Non Linear



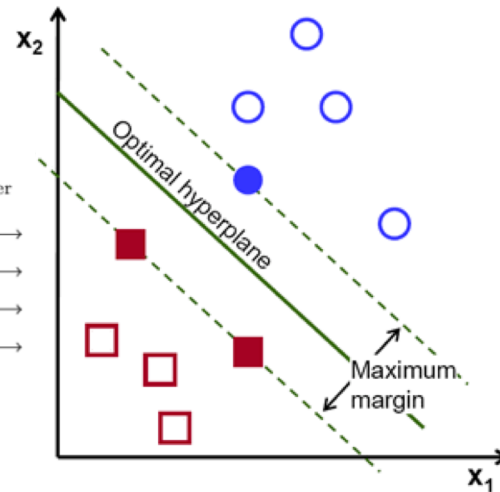
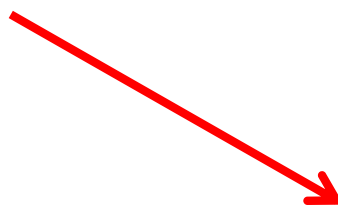
- Argument (XOR) used by [Minsky & Papert 1969] to criticize Perceptrons [Rosenblatt 1957] (and advocate Symbolic Artificial Intelligence)
- This stopped research on Perceptrons/Neural Networks for a long while
 - until Hidden Layers and Backpropagation or/and Kernel Trick (see later)

History: From Perceptron to Artificial Neural Networks to Deep Learning (2/4)



Difficulty to Efficiently Train Networks with many Layers

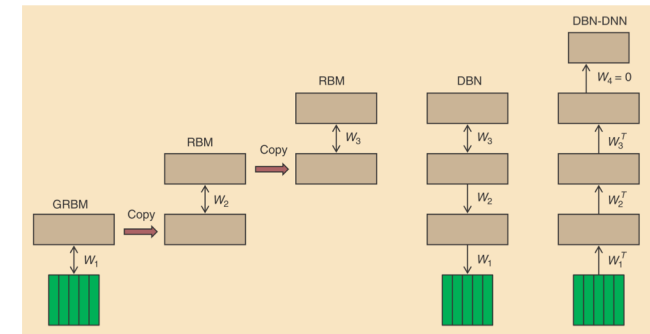
Unstable Gradients



**SVM [Vapnik 1963]
SVM + Kernel Trick [Vapnik et al. 1992]**

Nice Model and Optimized Implementation

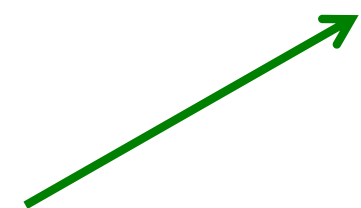
Margin Optimization



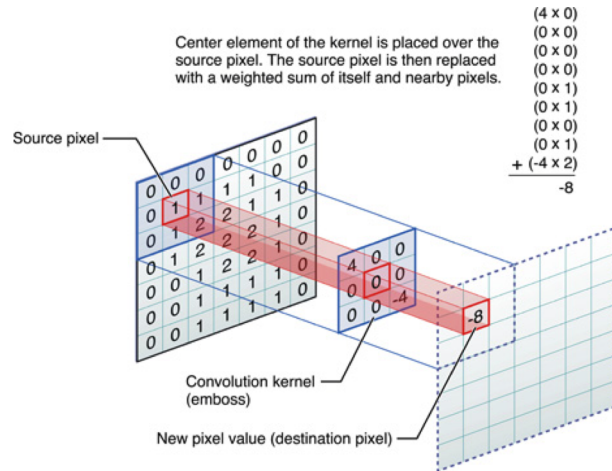
**Pre-Training [Hinton et al. 2006]
Layer-Wise Self-Supervised Training/Initialization**

Rank	Name	Error rate	Description
1	U. Toronto	0.15315	Deep learning
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

ImageNet 2012 Image Recognition Challenge Breakthrough

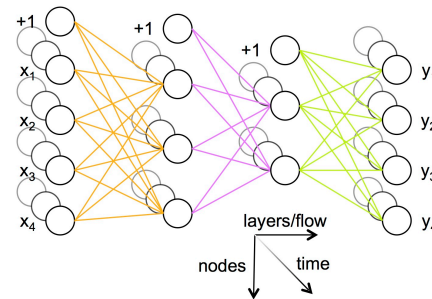


History: From Perceptron to Artificial Neural Networks to Deep Learning (3/4)

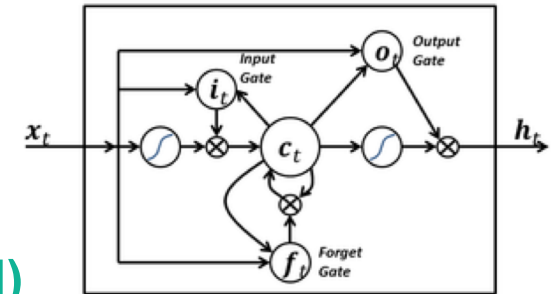


Convolutional Networks
[Le Cun et al. 1998]

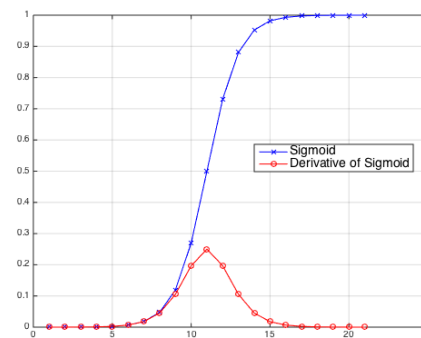
**Equivariance (to translation)
& Invariance (to small transformations)**



Recurrent Neural Networks (RNN)
(1986)
Temporal Invariance



Long Short-Term Memory
(LSTM)
[Hochreiter & Schmidhuber 1997]



**Gradient Vanishing
or Explosion (1991)**

History: From Perceptron to Artificial Neural Networks to Deep Learning (4/4)



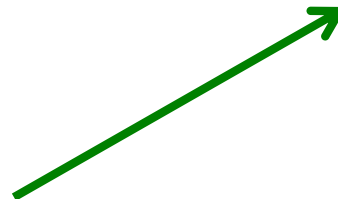
**Massive Data
Available**



**Efficient Implementation
Platforms**



**Affordable Efficient
Parallel Processing
(Graphic Cards)**



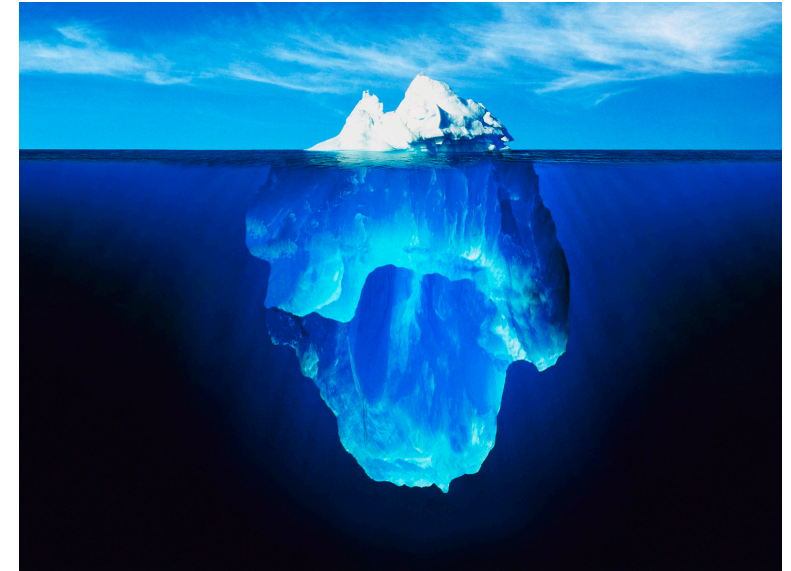
What and Why

Deep Learning

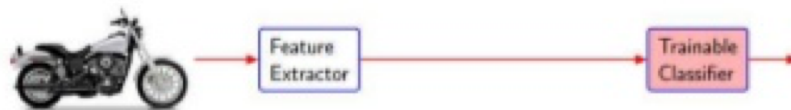
- Many Hidddden Layers
 - Deep Architecture
 - Deep Representation
- Various Types
 - Supervised learning
 - Unsupervised learning
 - Recurrent networks
 - Stochastic Neurons (ex: Restricted Boltzmann Machines)
- Various Objectives
 - *Classification*
 - *Prediction*
 - Feature extraction
 - Generation
 - » Related Content (ex: Melody Accompaniment)
 - » New Content (ex: new Melody)

Why Deep ?

- More Complex Models
- Learns better Complex Functions
- Hierarchical Features/Abstractions
- No Need for Handcrafted Features
 - (Automatically Extracted)



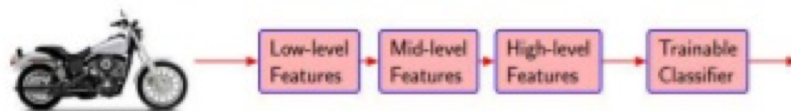
Traditional Pattern Recognition: Fixed/Handcrafted feature extraction



Modern Pattern Recognition: Unsupervised mid-level features



Deep Learning: Train hierarchical representations



Source: Talk: Computer Perception with Deep Learning by Yann LeCun

Distributed Representations

But the choice of the input representation is important

- Ex: for Audio:
 - Waveform or Spectrum
- Ex: For Symbolic representation
 - Additional information, ex: beat, fermata, enharmony (Cb \neq B)

End-to-End Architecture

Ex. of Handcrafted Features – Bag of Features [Brendel, 2019]

- Bag of Features (BoF) Model

- Define key Visual **Features**

– Ex: **Human eye**, **Feather**



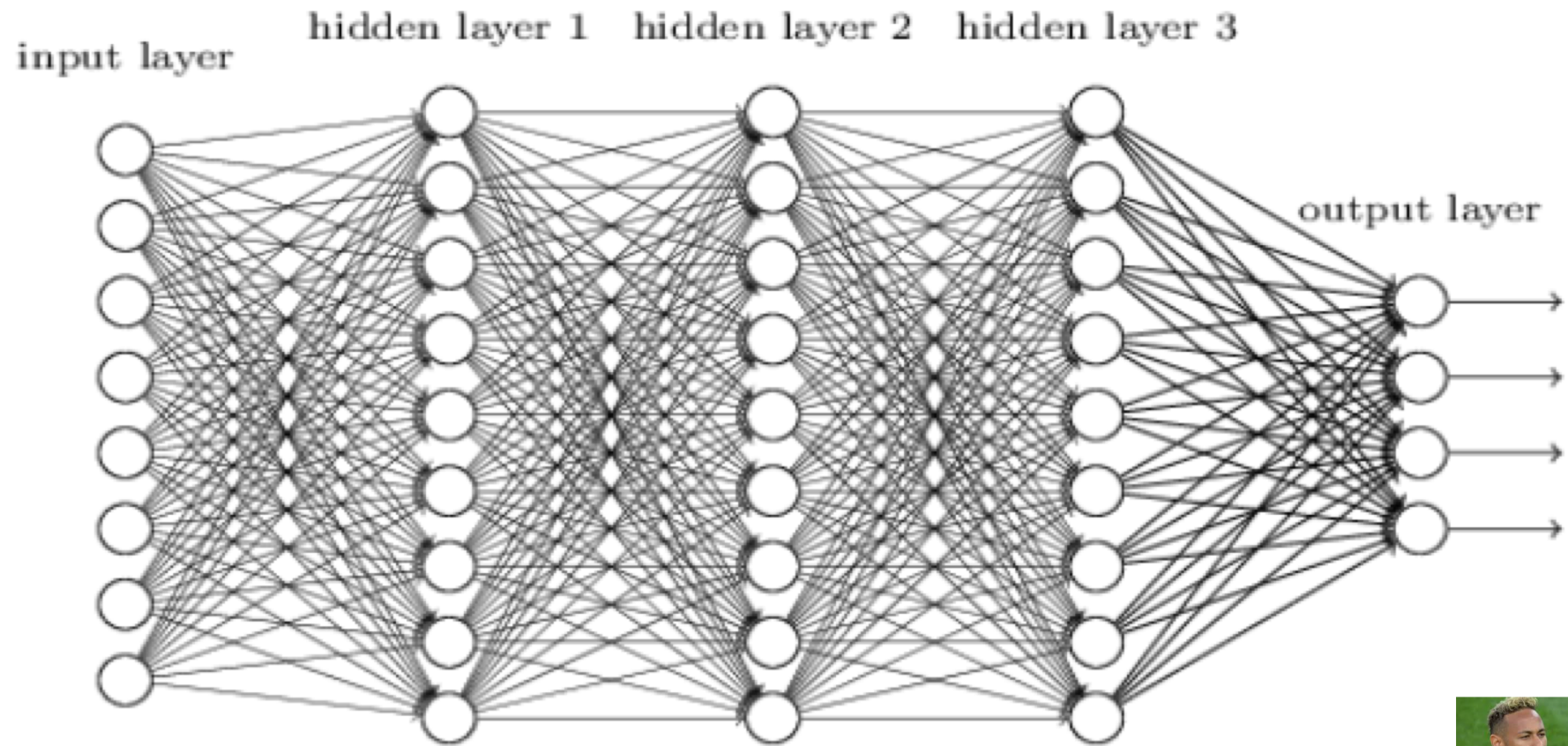
- Count how often each Feature is Present in the Image



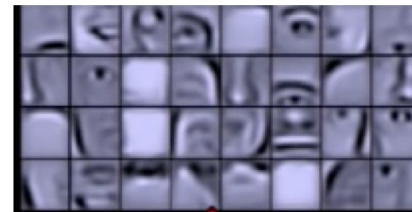
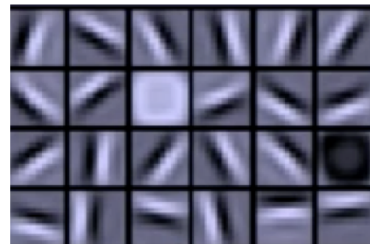
- Classify based on these Numbers

- Human: Eye = 2
- Bird: Feather = 18

Successive Features/Abstractions Constructed



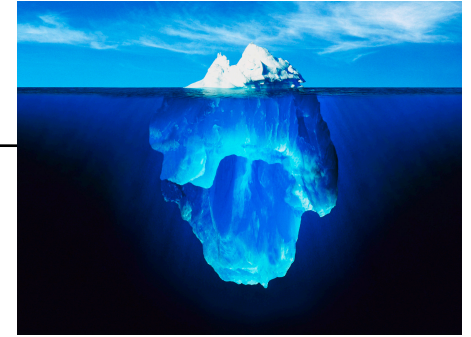
Jean-Pierre Briot



Deep Learning – Music Generation – 2019



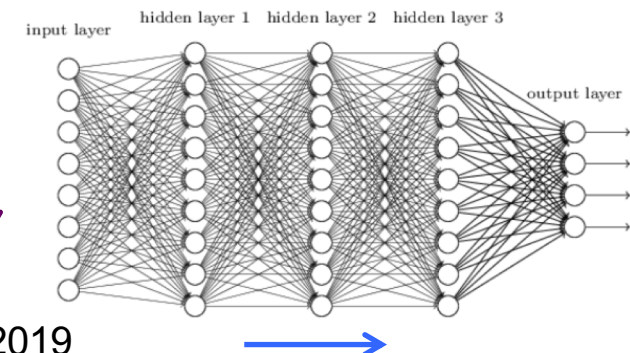
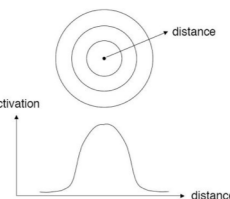
Why Deep ?



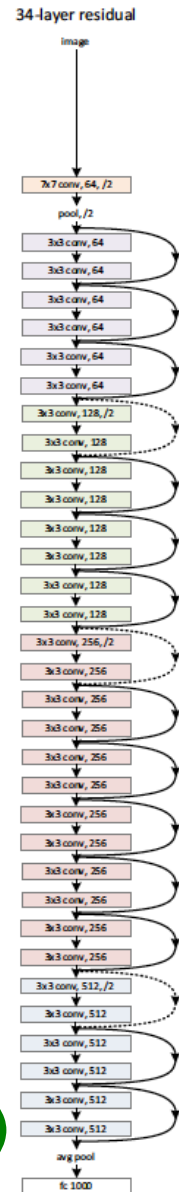
- Need for Deep ?! [Ba & Caruana 2014]
 - Train a Shallow Net to **Mimic** a Deep Net
 - Mimic = Train on the **Logits** (predicted values, **before softmax/probabilities**), **not** on the Original Output Data
 - **Equivalent Accuracy**
 - But for Convolutional Nets, Mimic Net **Does Not Match** the Initial Accuracy ! [Urban et al. & Caruana 2016]
- Counter Argument (Theorem) [Eldan & Shamir 2016]
 - There is a simple radial function on \mathbb{R}^d , expressible by a 3-layer net, but which cannot be approximated by any 2-layer net to more than a constant accuracy unless its width is exponential on the dimension d

– **Depth** → vs/and **Width** ↓

Radial function = Function whose value at each point depends only on distance between point and origin



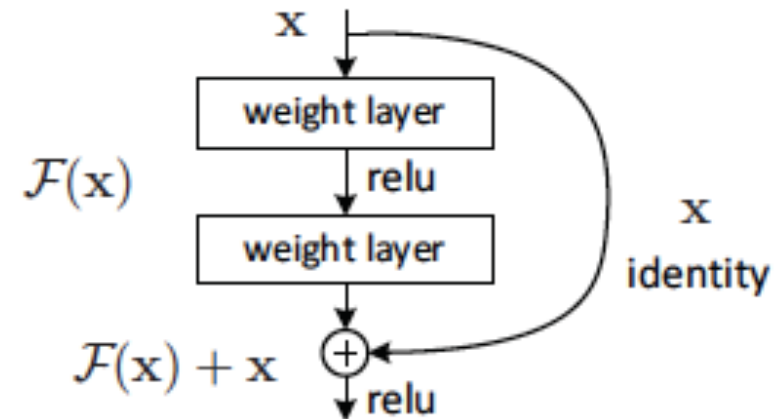
Very Deep Learning



(Very) Deep Networks

Upto 152 Layers !

New Techniques (Tricks ?!) e.g.,
Deep Residual Learning [He et al., 2015]
Replaced Pre-Training (less in vogue)

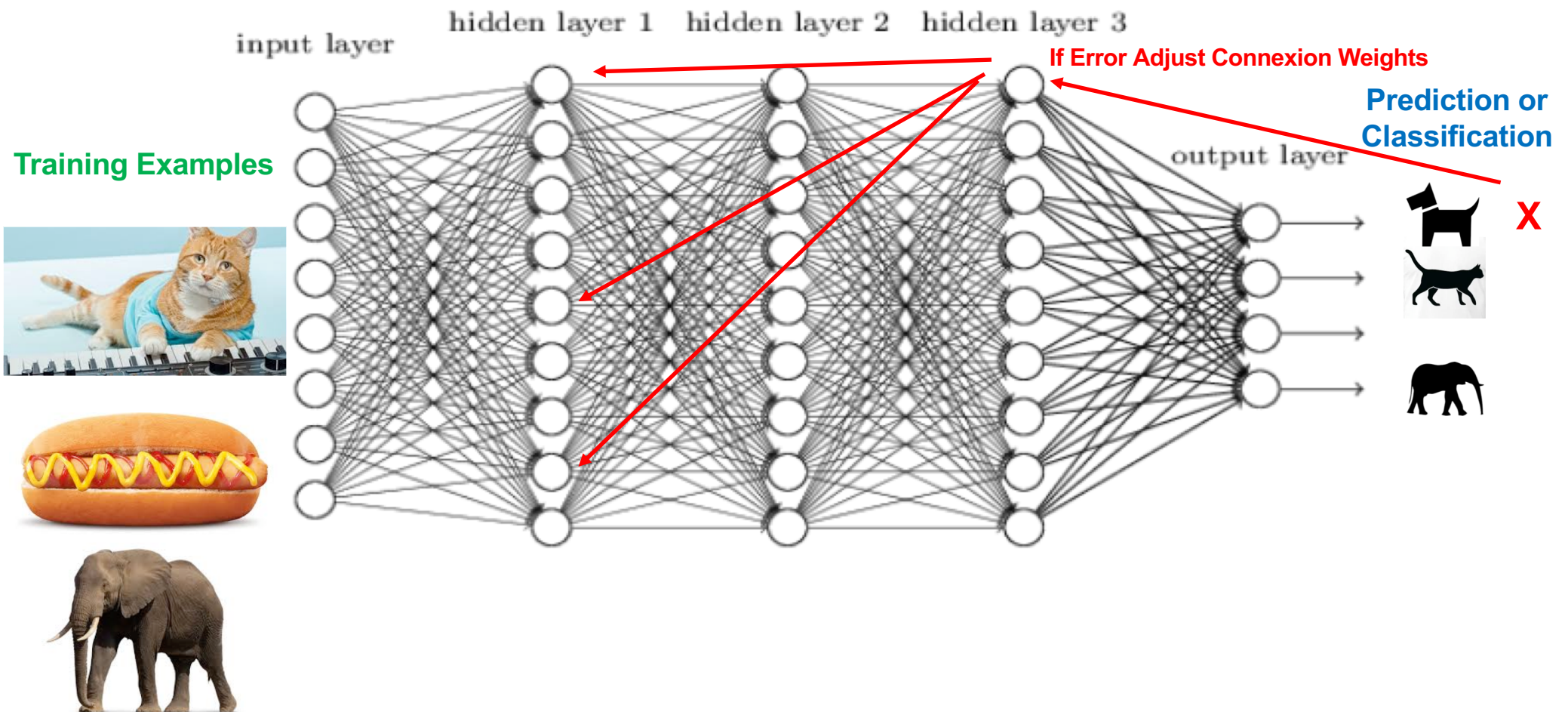


ResNet (2015)

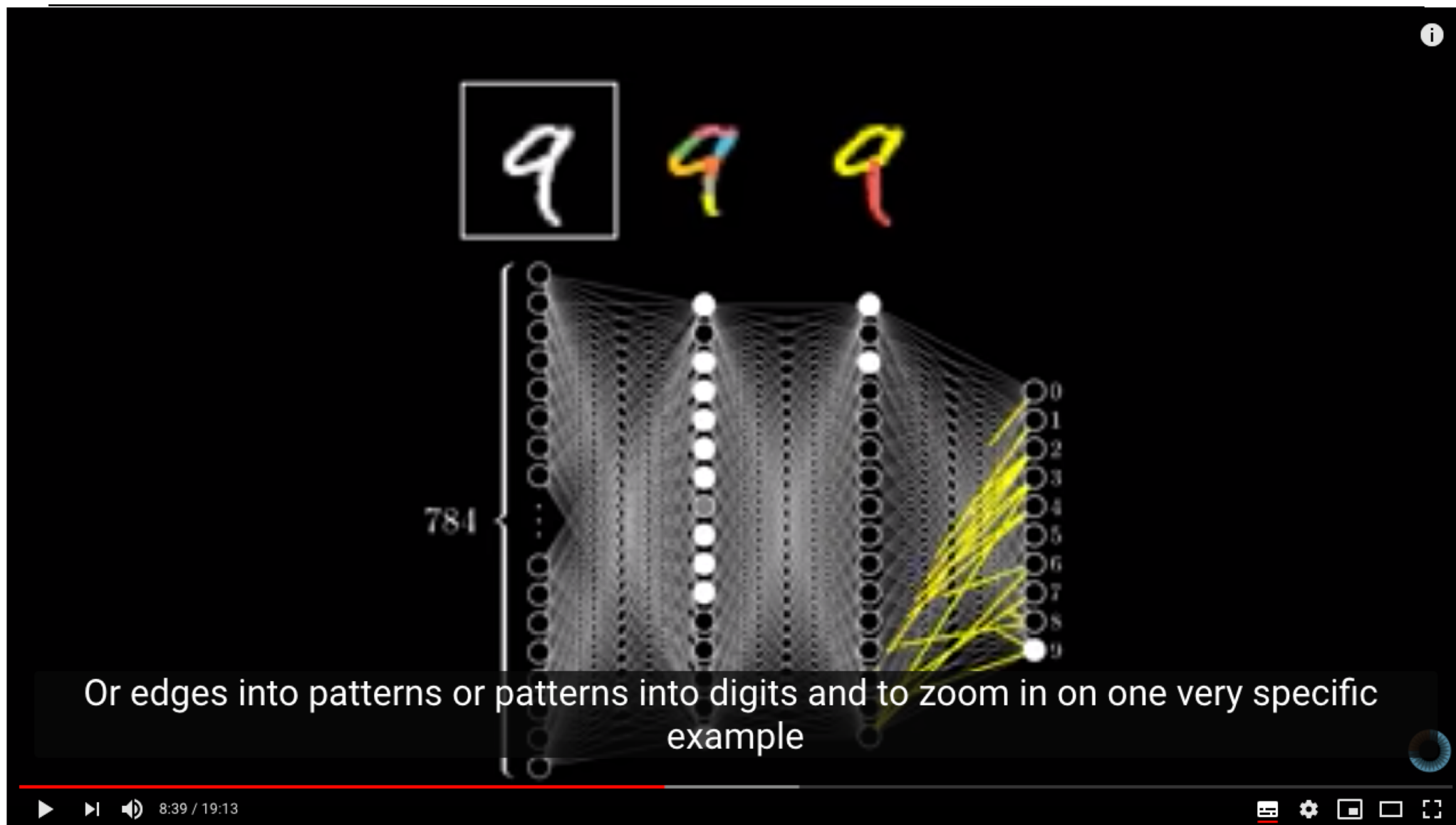
Linear Regression and Neural Networks

Neural Networks in One Slide

Principle – Error Prediction/Classification Feedback



Nice Animation [3Blue1Brown]



<https://www.youtube.com/watch?v=aircAruvnKk>

Machine Learning

- Why learning ?
- What is machine learning ?
- A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its **performance** at **tasks in T**, **as measured by P**, **improves with experience E** [Mitchell, 1997]

Machine Learning Typology

- **Learning** = Using Experience/Memory to Infer Information/Behavior and to Improve Decision/Action
- **Dynamicity**
 - **Off-Line** (**Training Phase** vs **Using/Production Phase**)
 - » Ex: **Supervised Learning**
 - But also Incremental versions
 - **On-Line/Incremental**
 - » Ex: **Reinforcement Learning**
 - But also Off-line versions (ex: Deep Reinforcement Learning Replay Mechanism)
- **Provided Information**
 - **Supervised** (Correct Output)
 - **Unsupervised** (\emptyset)
 - **Reward-Based** (Quality assessment)

Machine Learning Typology (2)

- Data/Statistics-oriented
 - Supervised and Off-Line (mostly)
 - » Neural Networks
 - » Deep Networks
 - » Support Vector Machines
 - » Bayesian Networks
 - Unsupervised
 - » Clustering (ex: k-Means)
 - » Principal Component Analysis (PCA)
 - » Autoencoders
 - » Deep Networks
- Decision/Action-oriented
 - Fitness/Reward-based
 - » Evolutionary Algorithms
 - » Reinforcement Learning
- Knowledge/Rules-oriented
 - » Memory-based Reasoning and Case-Based Reasoning
 - » Inductive Logic Programming (ILP)

Machine Learning and Artificial Intelligence

- Machine Learning is Part of Artificial Intelligence Techniques

But also:

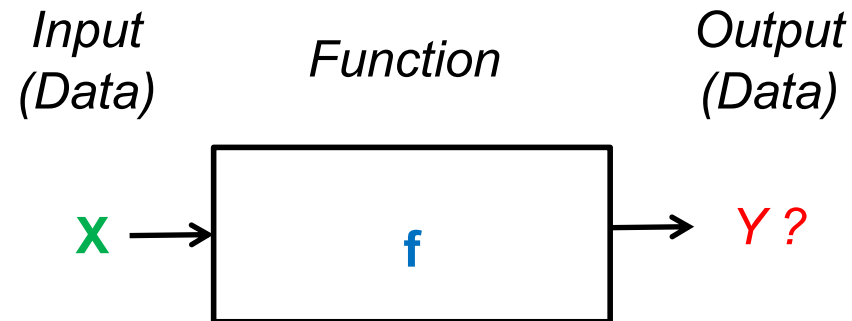
- Reasoning
- Planning
- Knowledge Representation
- User Modeling and Interaction
- Collaboration (Multi-Agent Systems)

- Natural Language Processing
- Dialogue
- Speech Processing

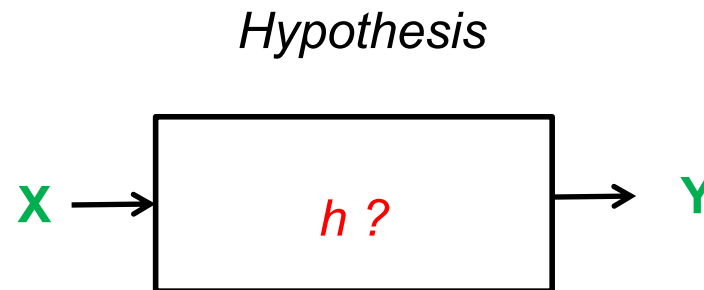
- Decision
- Game Theory
- Optimization
- Robotics

Machine Learning (Data Oriented/Supervised Learning)

- Traditional Programming



- Machine Learning (Supervised Learning)



Function/Module Configuration

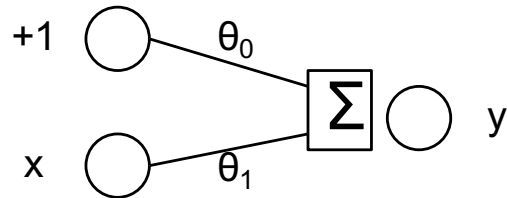
(*Pre-Existing Representation Model*)

Machine Learning (Data Oriented)

- 3 Components [Domingos 2015]:
- Representation
 - To Model Predictors/Classifiers
 - e.g., Decision Tree, Rule, Neural Network, Graphical Model...
- Evaluation
 - of Predictors/Classifiers (Accuracy)
 - Cost, e.g., Squared Error, cross-entropy...
 - Precision, Recall
- Optimization
 - To Search among Predictors/Classifiers,
 - e.g., Batch Gradient Descent, Greedy Search, Stochastic Gradient...

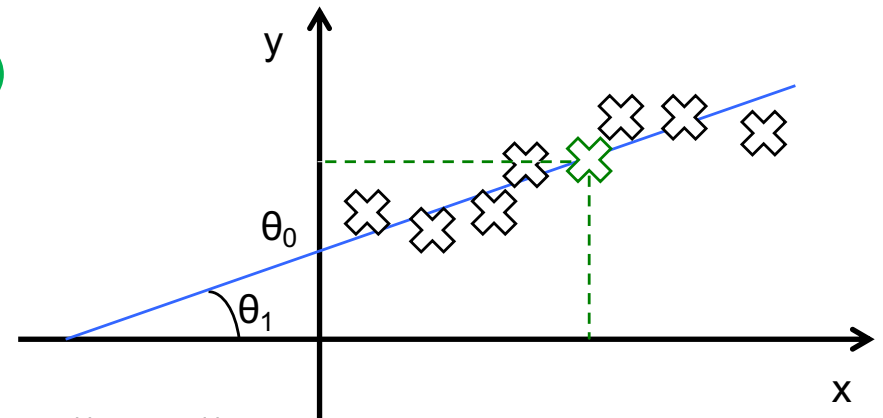
(Simple) Linear Regression (Prediction)

- Representation



Model (Hypothesis)

$$h(x) = \theta_0 + \theta_1 x$$



- Evaluation

Cost function : $J(\theta_0, \theta_1) = J_\theta = 1/2 * m \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$

Distance between predicted and real

- Optimization (Training)

Objective: Find out "best" values of θ_0 and θ_1 to minimize Cost function J_θ

with:

Example⁽¹⁾

Example⁽²⁾

Example^(m)

Input Dataset : X :

$x^{(1)}$

$x^{(2)}$

...

$x^{(m)}$

and Output Dataset : y :

$y^{(1)}$

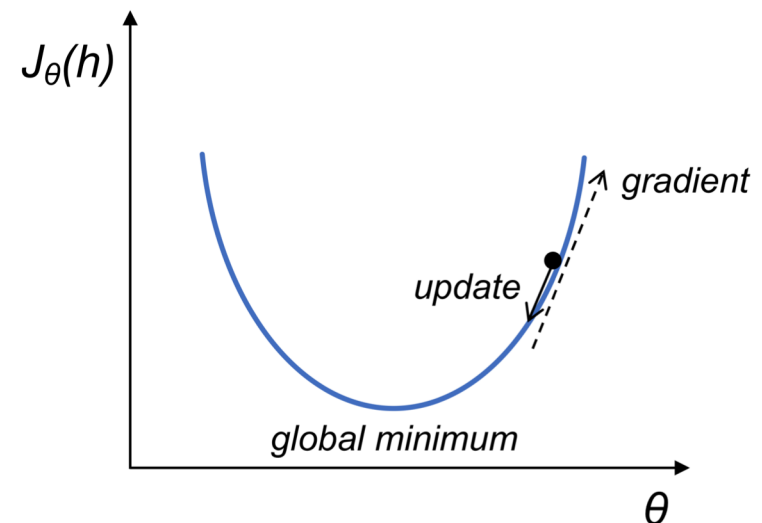
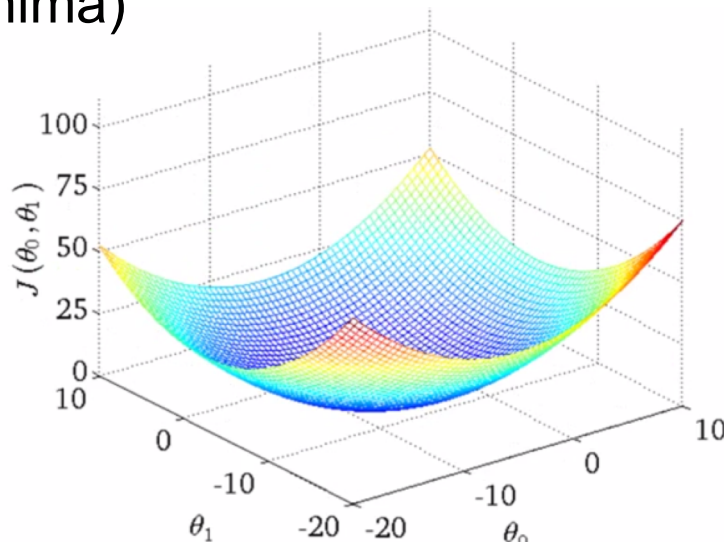
$y^{(2)}$

...

$y^{(m)}$

Optimization

- Objective: **Minimization of Cost function** J_{θ}
- Usual/simple algorithm: **Gradient descent**
- Therefore, we need to compute the Cost, but also the **Gradients** (partial derivatives respective to each θ (weight)) in order to guide gradient descent
- As this Cost function is **convex**, there is a global minimum (and no local minima)

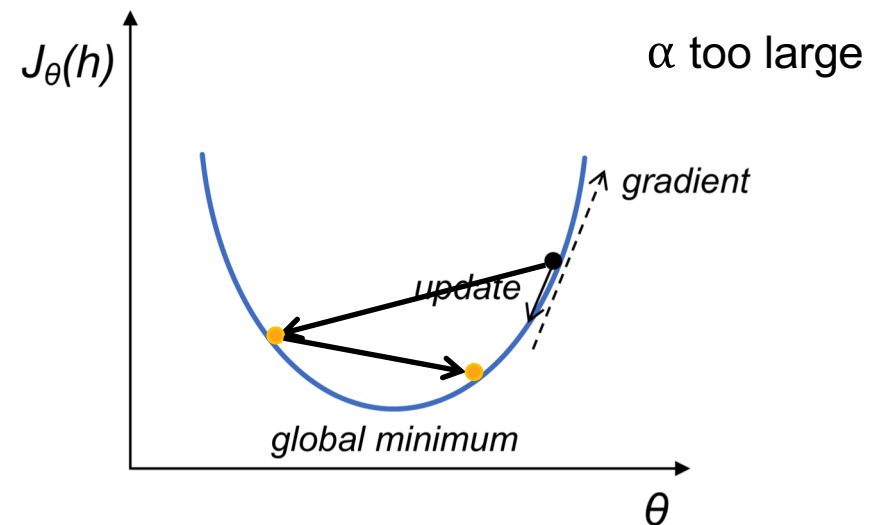
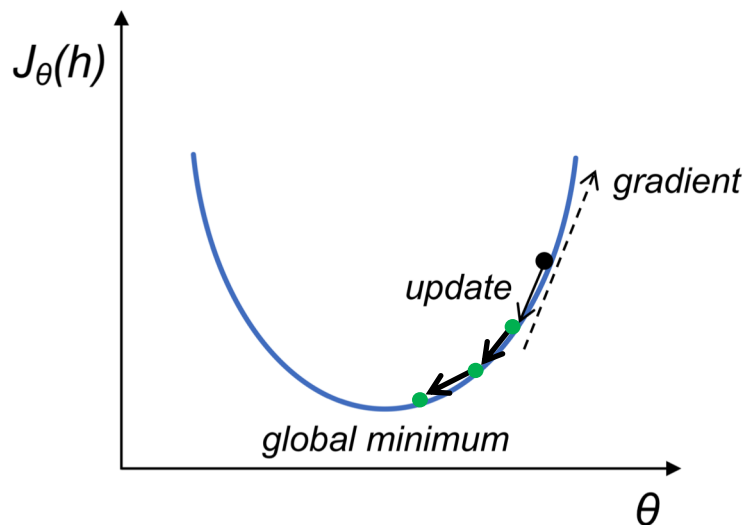


Learning rate and Stochastic Gradient Descent (SGD)

Update Rule

$$\theta_0 := \theta_0 - \alpha \frac{dJ(\theta_0, \theta_1)}{d\theta_0}$$
$$\theta_1 := \theta_1 - \alpha \frac{dJ(\theta_0, \theta_1)}{d\theta_1}$$

α : Learning rate

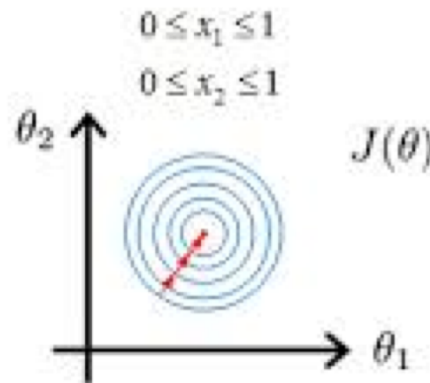
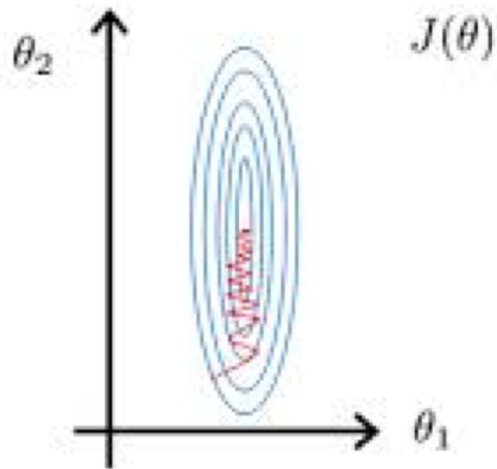


Compute Cost and Gradients

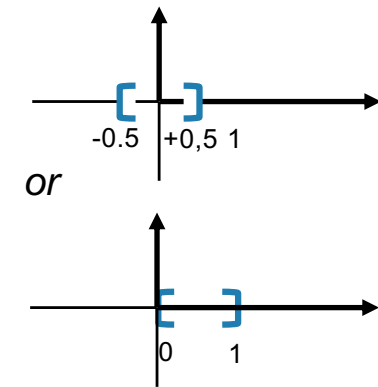
- Gradient Descent (SGD): for ALL examples
- Stochastic Gradient Descent (SGD): for ONE example (randomly chosen)
- MiniBatch: for a Batch (subset) randomly chosen

Scaling and Mean Normalization

- To improve learning convergence (faster), better to have features ($x_1, x_2 \dots$) on a similar scale



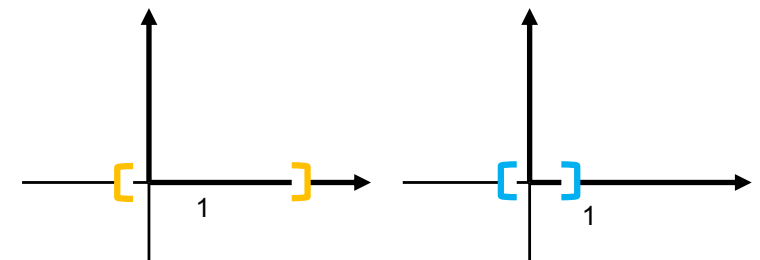
scaled and normalized



- Feature Scaling

$$x := x / \text{range}(x)$$

-> Range of 1

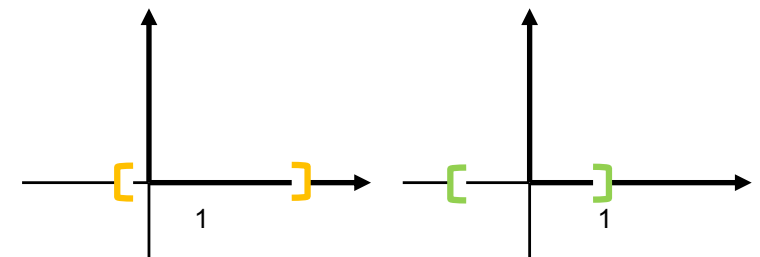


(Could also apply)

- Mean Normalization

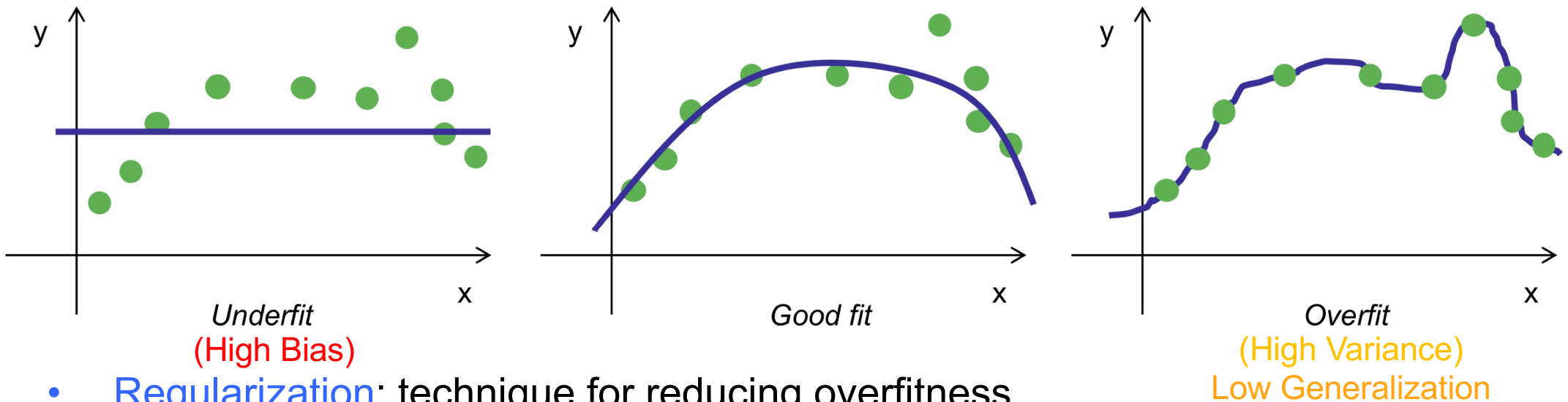
$$x := x - \text{mean}(x)$$

-> Centered



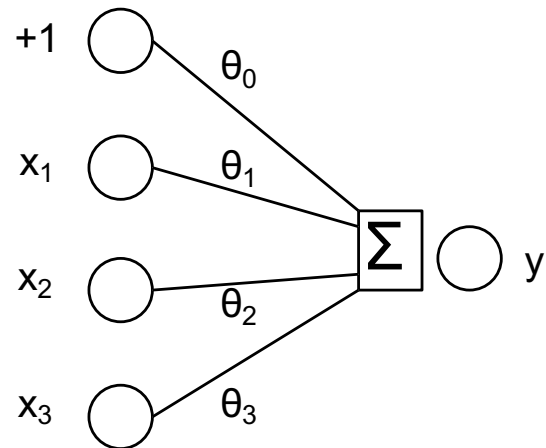
Best Hypothesis

- Objective: Find out "best" hypothesis (values of θ_0 and θ_1)
- Best hypothesis means best fit to data
- And **not just best fit only** to **Train data**
- Otherwise the best strategy would be to **exactly memorize every entry**
- Best hypothesis means best fit for future/yet unknown data: **Test data**
- i.e., with good capacity for **Generalization**



- **Regularization**: technique for reducing overfitness
 - Weights decay, to penalize over-preponderant weights
 - Also: Dropout, Early stopping, Dataset augmentation

Multiple Linear Regression – More than 1 Variable



$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

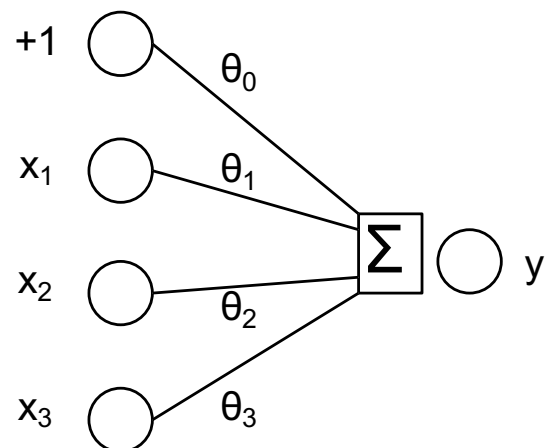
Input Dataset : X :

$x_1^{(1)}$	$x_1^{(2)}$	\dots	$x_1^{(m)}$
$x_2^{(1)}$	$x_2^{(2)}$		$x_2^{(m)}$
$x_3^{(1)}$	$x_3^{(2)}$		$x_3^{(m)}$

Output Dataset : y :

$y^{(1)}$	$y^{(2)}$	\dots	$y^{(m)}$
-----------	-----------	---------	-----------

Multiple Linear Regression Vectorization



$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$= \sum_{i=0}^n \theta_i x_i \quad (\text{with } x_0 = 1)$$

$$= [\theta_0 \ \theta_1 \ \theta_2 \ \theta_3] * \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\text{scalar}(1 \times 1) = \text{vector}(1 \times 4) * \text{vector}(4 \times 1)$$

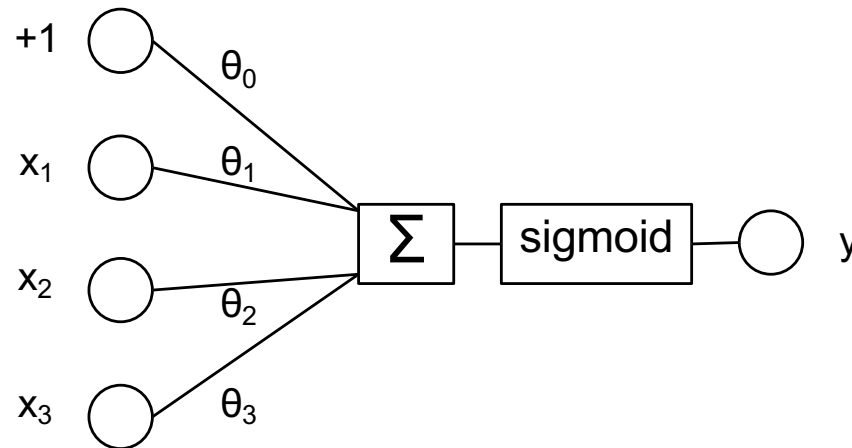
$$= \theta * x$$

Computing predictions for a set of examples X :

$$\theta * X = [\theta_0 \ \theta_1 \ \theta_2 \ \theta_3] * \begin{bmatrix} \begin{bmatrix} 1 \\ x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} & \dots & \begin{bmatrix} 1 \\ x_1^{(m)} \\ x_2^{(m)} \\ x_3^{(m)} \end{bmatrix} \end{bmatrix} = [h^{(1)} \ \dots \ h^{(m)}]$$

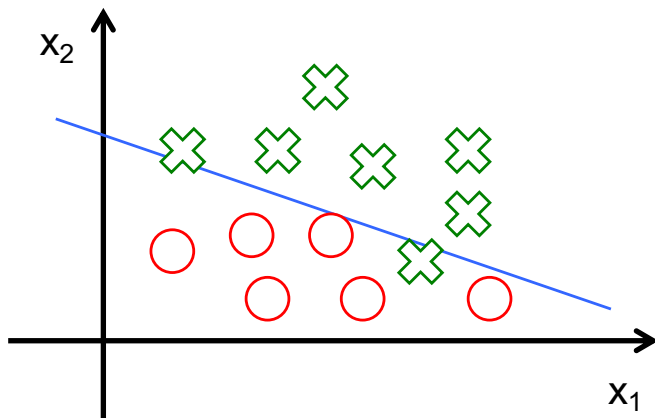
$$\text{vector}(1 \times 4) * \text{matrix}(4 \times m) = \text{vector}(1 \times m)$$

Multiple Logistic Regression (Classification)

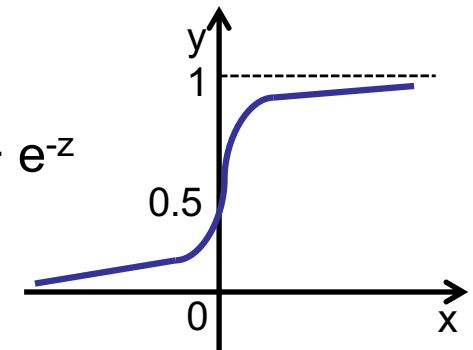


$$h(x) \in [0 \ 1]$$

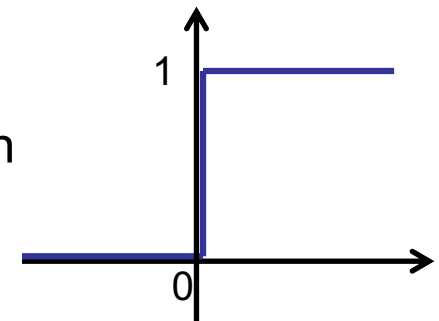
$$h(x) = \text{sigmoid}(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3) \\ = \text{sigmoid}(\theta * x)$$



$$\text{sigmoid}(z) = \sigma(z) = 1 / 1 + e^{-z}$$



Perceptron [Rosenblatt 1957] used unit step function



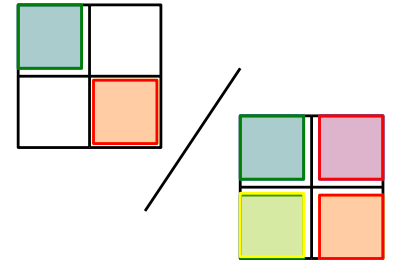
Current Neural Networks use Softmax (transforms into probabilities)

Classifier Evaluation Metrics

y \ h	+	-
+	True Positive (Hit)	False Negative
-	False Positive (False Alarm)	True Negative

Table of Confusion
(Confusion Matrix)

Accuracy = $\frac{TP + TN}{\forall}$
% correctly classified



Classifier Evaluation Metrics

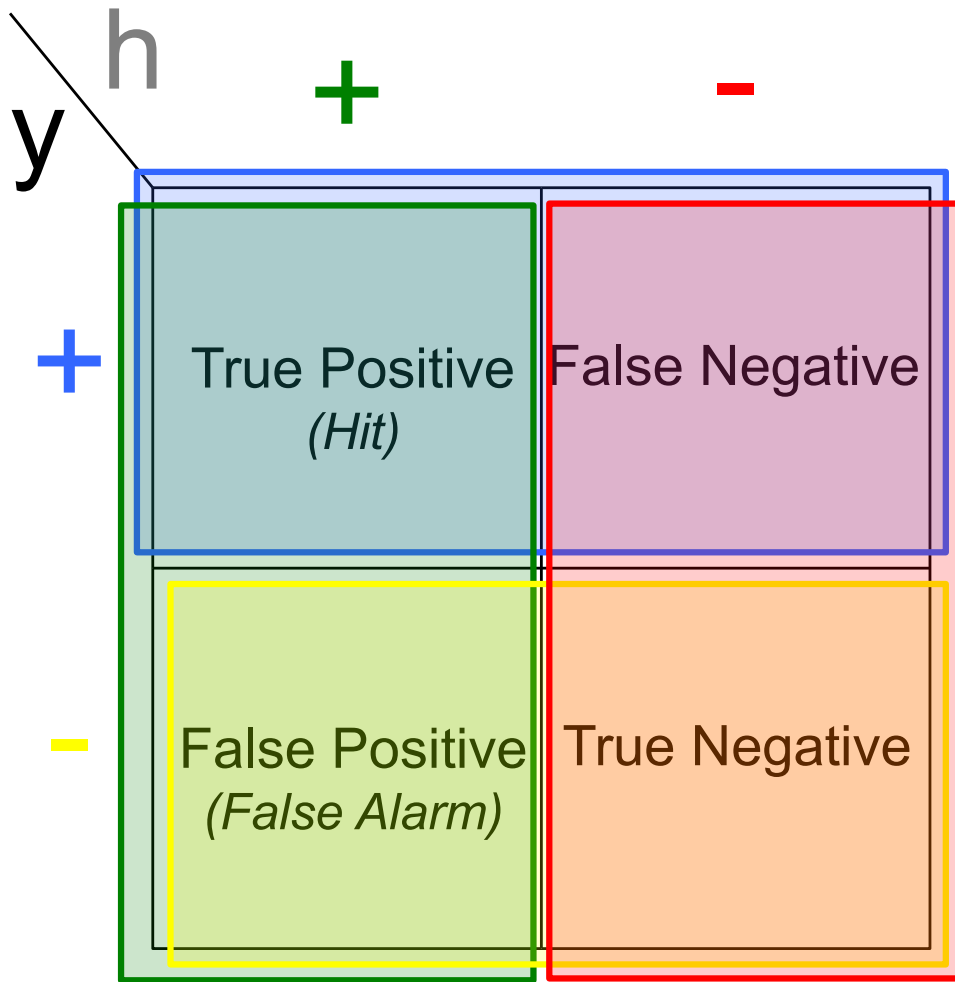
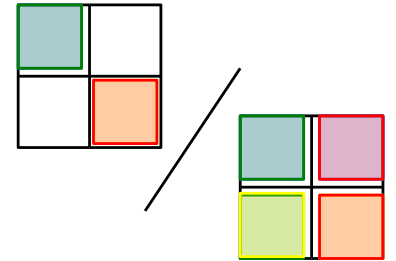
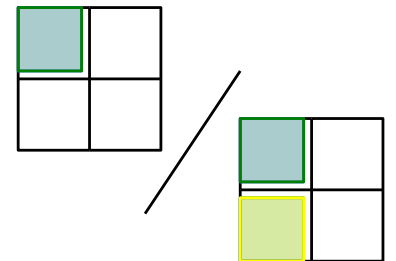


Table of Confusion
(Confusion Matrix)

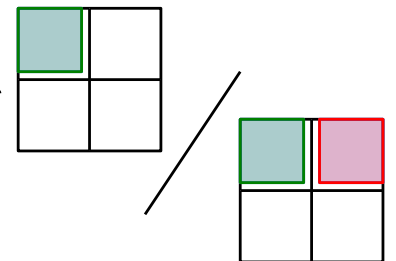
Accuracy = $\frac{TP + TN}{N}$
% correctly classified



Precision = $\frac{TP}{P}$ ($TP + FP$)
true / classified positive
How useful the results



Recall = $\frac{TP}{P}$ ($TP + FN$)
true / real positive
How complete the results



Skewed Classes

		h	
		0	100
y	2	0 True Positive (Hit)	2 False Negative
	98	0 False Positive (False Alarm)	98 True Negative

Accuracy = $\frac{TP + TN}{\forall} = 0.98$
 % correctly classified

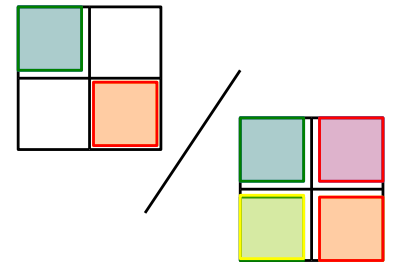


Table of Confusion
 (Confusion Matrix)

Ex: Maligne Cancer Prediction: 2% Rate

Optimist Hypothese: Always (100%) False

Skewed Classes

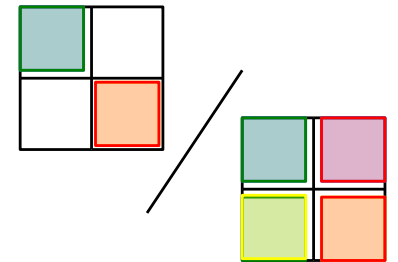
		h	
		0	100
y	2	<div>0</div> <div>True Positive (Hit)</div>	<div>2</div> <div>False Negative</div>
	98	<div>0</div> <div>False Positive (False Alarm)</div>	<div>98</div> <div>True Negative</div>

Table of Confusion
(Confusion Matrix)

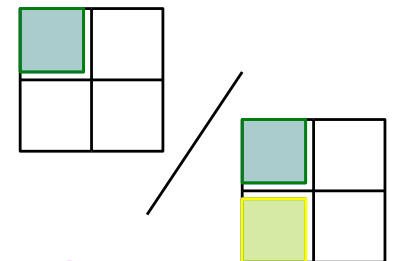
Ex: Maligne Cancer Prediction: 2% Rate

Optimist Hypothese: Always (100%) False

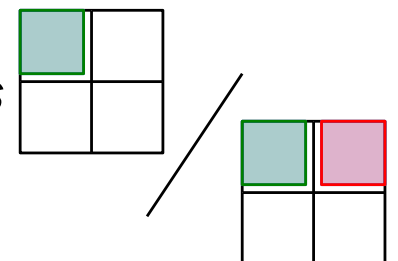
Accuracy = $\frac{TP + TN}{\forall} = 0.98$
% correctly classified



Precision = $\frac{TP}{P} = \frac{TP}{TP + FP} = 0$
true / classified positive
How useful the results



Recall = $\frac{TP}{P} = \frac{TP}{TP + FN} = 0$
true / real positive
How complete the results



Skewed Classes

		h	
		0	100
y	2	1 True Positive (Hit)	1 False Negative
	98	2 False Positive (False Alarm)	96 True Negative

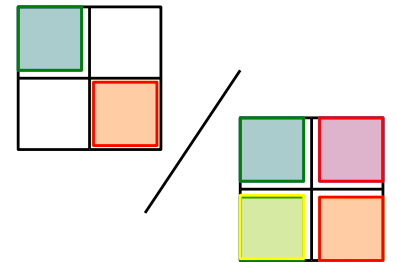
Table of Confusion
(Confusion Matrix)

Ex: Maligne Cancer Prediction: 2% Rate

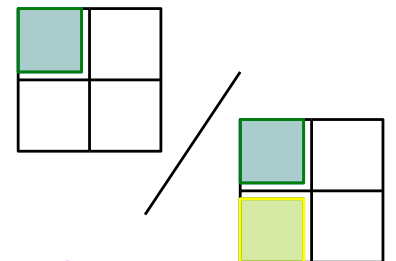
#1 Test

Jean-Pierre Briot

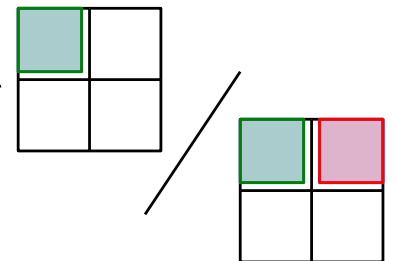
Accuracy = $\frac{TP + TN}{\forall} = 0.98$
% correctly classified



Precision = $\frac{TP}{P} (TP + FP) = 0.33$
true / classified positive
How useful the results



Recall = $\frac{TP}{P} (TP + FN) = 0.5$
true / real positive
How complete the results



Skewed Classes

		h	
		0	100
y	2	1 True Positive (Hit)	0 False Negative
	98	2 False Positive (False Alarm)	97 True Negative

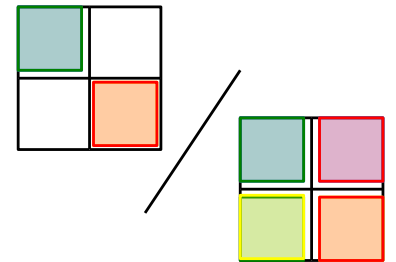
Table of Confusion
(Confusion Matrix)

Ex: Maligne Cancer Prediction: 2% Rate

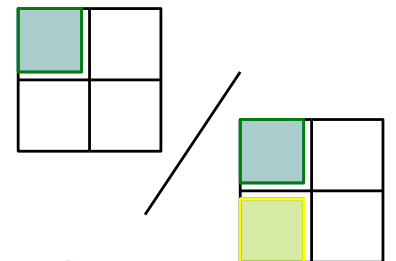
#2 Test

Jean-Pierre Briot

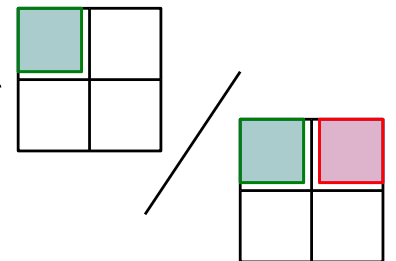
Accuracy = $\frac{TP + TN}{\forall} = 0.98$
% correctly classified



Precision = $\frac{TP}{P} (TP + FP) = 0.33$
true / classified positive
How useful the results



Recall = $\frac{TP}{P} (TP + FN) = 1$
true / real positive
How complete the results



Multilabel/class Confusion Matrix

y \ h				
	A	B	C	
A	30	50	20	Total A 100
B	20	60	20	Total B 100
C	10	10	80	Total C 100
Total Predicted A 60		Total Predicted B 120		Total Predicted C 120

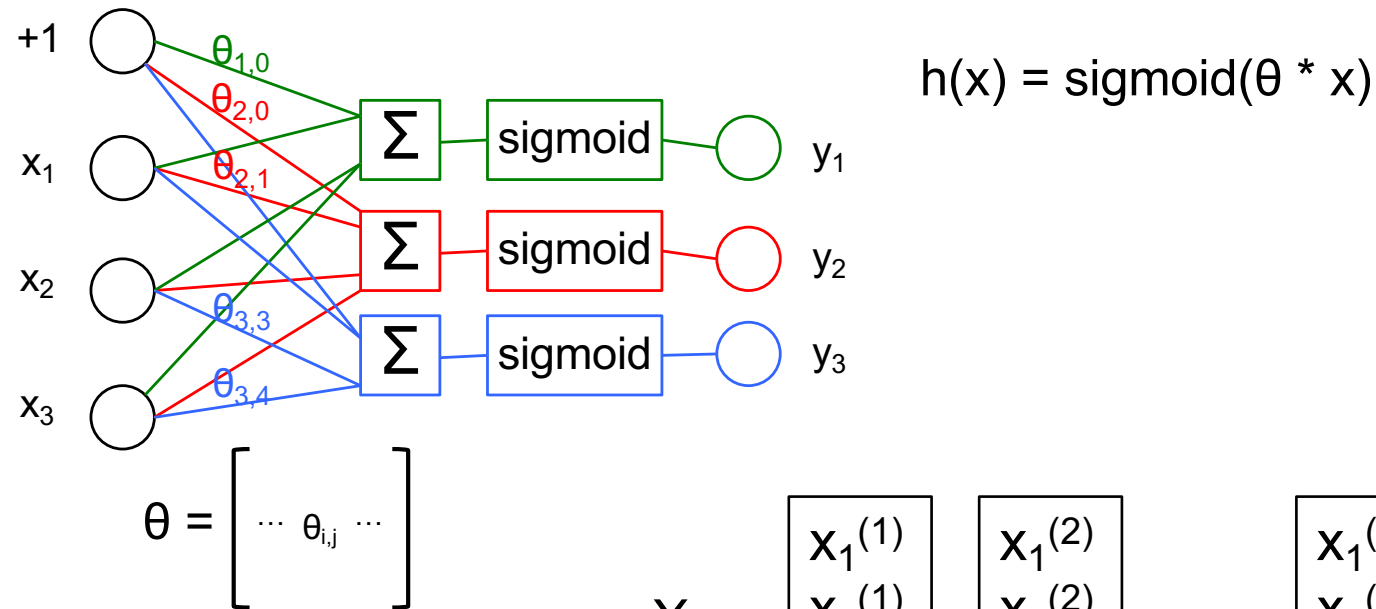
Precision(label X)
= True Positive(X) / Total Predicted(X)

Ex: Precision(A)
= True Positive(A) / Total Predicted(A)
= 30/60 = 0.5

Recall(label X)
= True Positive(X) / Total Label(X)

Ex: Recall(A)
= True Positive(A) / Total Label(A)
= 30/100 = 0.3

Multivariate Multiple Logistic Regression (Classification)



$$X : \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} \quad \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} \quad \dots \quad \begin{bmatrix} x_1^{(m)} \\ x_2^{(m)} \\ x_3^{(m)} \end{bmatrix}$$

$$y : \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_3^{(1)} \end{bmatrix} \quad \begin{bmatrix} y_1^{(2)} \\ y_2^{(2)} \\ y_3^{(2)} \end{bmatrix} \quad \dots \quad \begin{bmatrix} y_1^{(m)} \\ y_2^{(m)} \\ y_3^{(m)} \end{bmatrix}$$

Current Neural Networks use **Softmax**

Multivariate Multiple Classification Vectorization

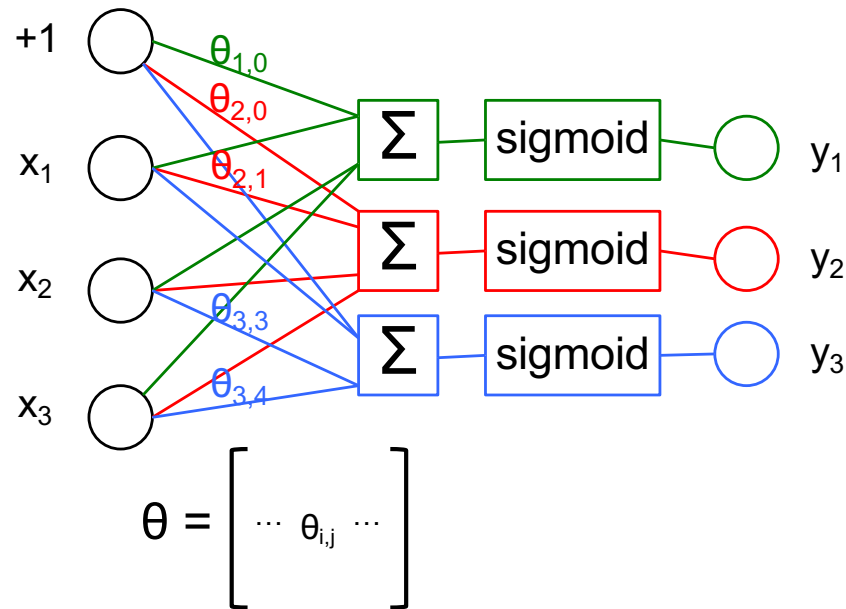
$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \end{bmatrix} = \begin{bmatrix} \theta_{1,0} + \theta_{1,1}x_1 + \theta_{1,2}x_2 + \theta_{1,3}x_3 \\ \theta_{2,0} + \theta_{2,1}x_1 + \theta_{2,2}x_2 + \theta_{2,3}x_3 \\ \theta_{3,0} + \theta_{3,1}x_1 + \theta_{3,2}x_2 + \theta_{3,3}x_3 \end{bmatrix} = \theta * x = \begin{bmatrix} \theta_{1,0} & \theta_{1,1} & \theta_{1,2} & \theta_{1,3} \\ \theta_{2,0} & \theta_{2,1} & \theta_{2,2} & \theta_{2,3} \\ \theta_{3,0} & \theta_{3,1} & \theta_{3,2} & \theta_{3,3} \end{bmatrix} * \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Computing predictions for a set of examples X :

$$\theta * X = \begin{bmatrix} \theta_{1,0} & \theta_{1,1} & \theta_{1,2} & \theta_{1,3} \\ \theta_{2,0} & \theta_{2,1} & \theta_{2,2} & \theta_{2,3} \\ \theta_{3,0} & \theta_{3,1} & \theta_{3,2} & \theta_{3,3} \end{bmatrix} * \begin{bmatrix} 1 & \dots & 1 \\ x_1^{(1)} & \dots & x_1^{(m)} \\ x_2^{(1)} & \dots & x_2^{(m)} \\ x_3^{(1)} & \dots & x_3^{(m)} \end{bmatrix} = \begin{bmatrix} h_1^{(1)} & \dots & h_1^{(m)} \\ h_2^{(1)} & \dots & h_2^{(m)} \\ h_3^{(1)} & \dots & h_3^{(m)} \end{bmatrix} : \begin{bmatrix} \text{✗} & \triangle & \text{✗} & \dots & \circ & \triangle \end{bmatrix}$$

matrix(3x4) * matrix(4xm) = matrix(3xm) : vector(1xm)

Multivariate Multiple Logistic Regression (Classification)

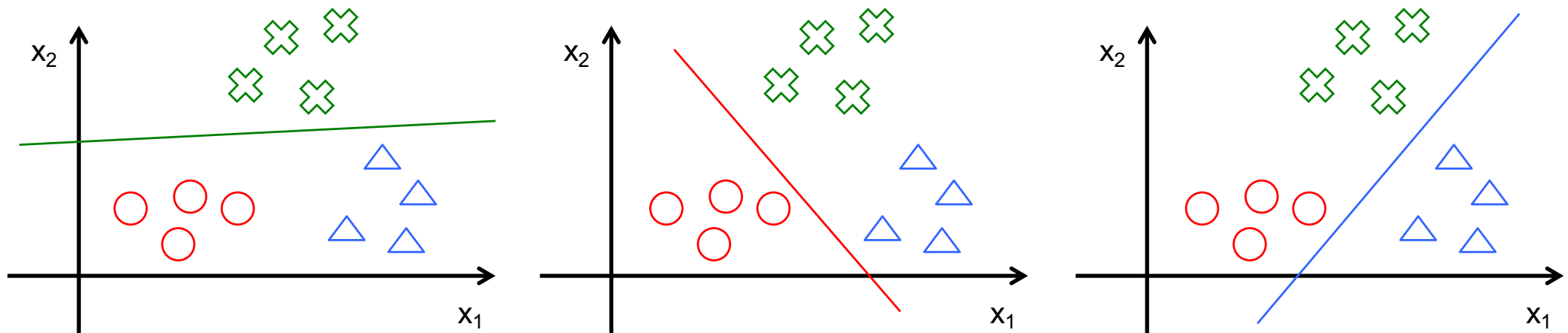


$$h(x) = \text{sigmoid}(\theta * x)$$

$$X : \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} \quad \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} \quad \dots \quad \begin{bmatrix} x_1^{(m)} \\ x_2^{(m)} \\ x_3^{(m)} \end{bmatrix}$$

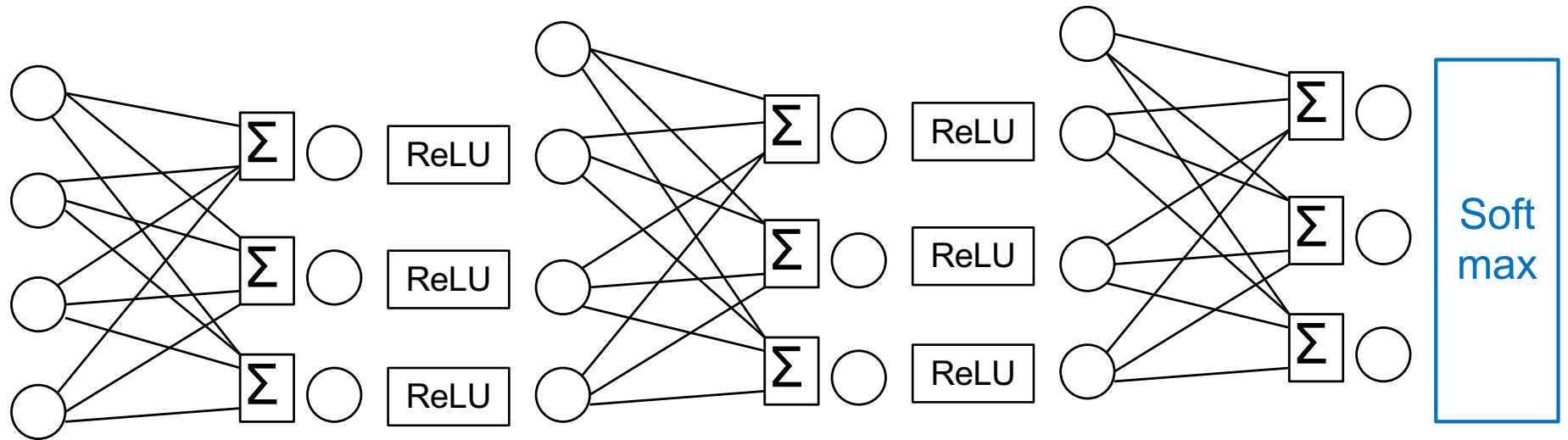
$$y : \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_3^{(1)} \end{bmatrix} \quad \begin{bmatrix} y_1^{(2)} \\ y_2^{(2)} \\ y_3^{(2)} \end{bmatrix} \quad \dots \quad \begin{bmatrix} y_1^{(m)} \\ y_2^{(m)} \\ y_3^{(m)} \end{bmatrix}$$

One vs All Algorithm: higher score = class recognized



Current Neural Networks use **Softmax**

Modern Neural Network



Softmax transforms values into probabilities

$$\sigma(z)_i = e^{z_i} / \sum e^{z_j}$$

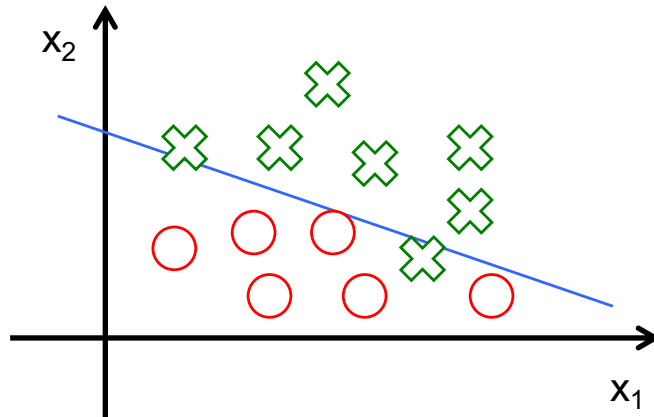
*Generalization of **sigmoid***

*Associated cost function is **cross entropy***

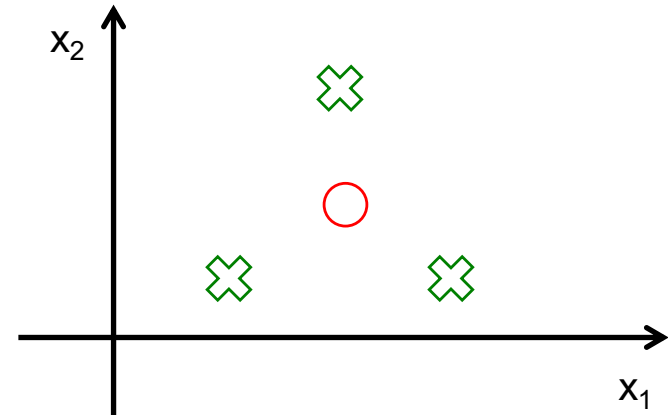
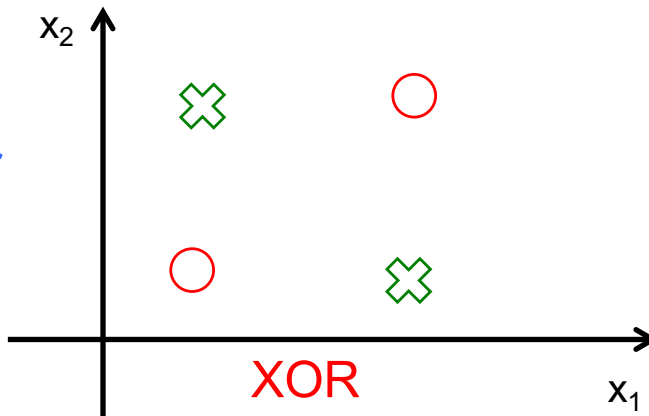
Non Linearity

Linear vs Non Linear Decision Boundary

- Linear



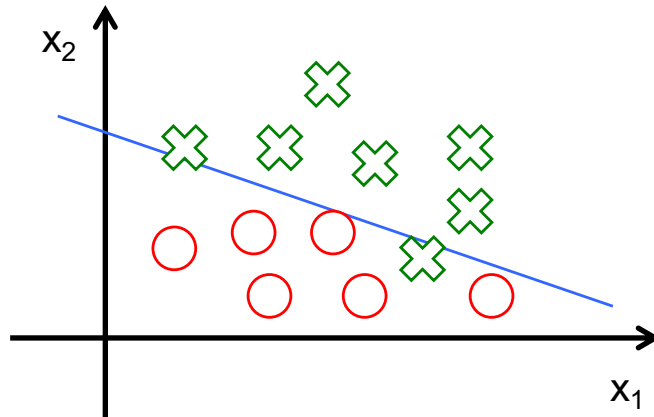
- Non Linear



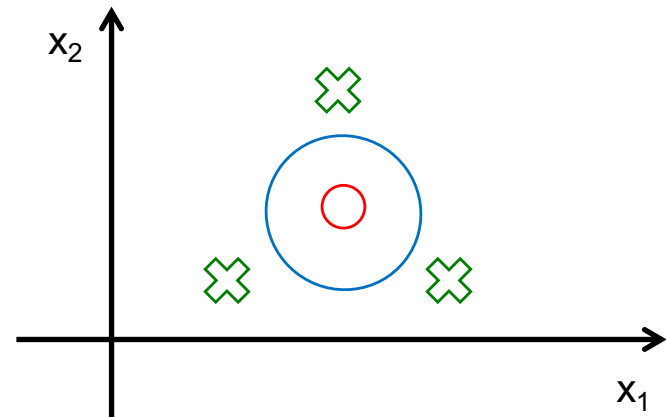
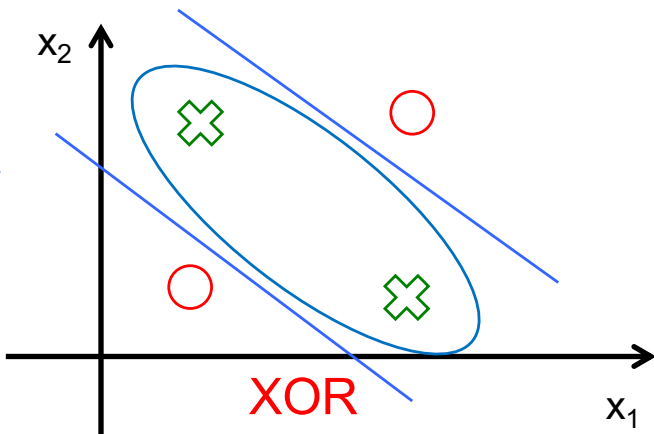
- Argument (XOR) used by [Minsky & Papert 1969] to criticize Perceptrons [Rosenblatt 1957] (and advocate Symbolic Artificial Intelligence)
- This stopped research on Perceptrons/Neural Networks for a long while
 - until Hidden Layers and Backpropagation or/and Kernel Trick (see later)

Linear vs Non Linear Decision Boundary

- Linear



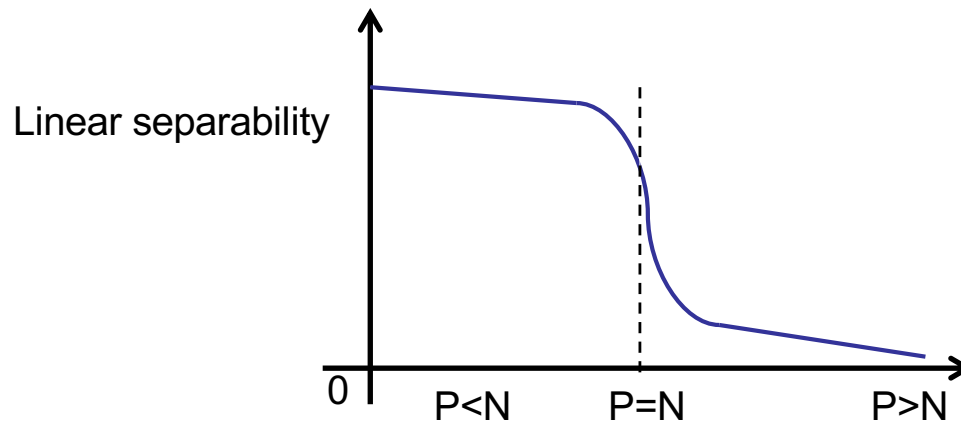
- Non Linear



- Argument (XOR) used by [Minsky & Papert 1969] to criticize Perceptrons [Rosenblatt 1957] (and advocate Symbolic Artificial Intelligence)
- This stopped research on Perceptrons/Neural Networks for a long while
 - until Hidden Layers and Backpropagation or/and Kernel Trick (see later)

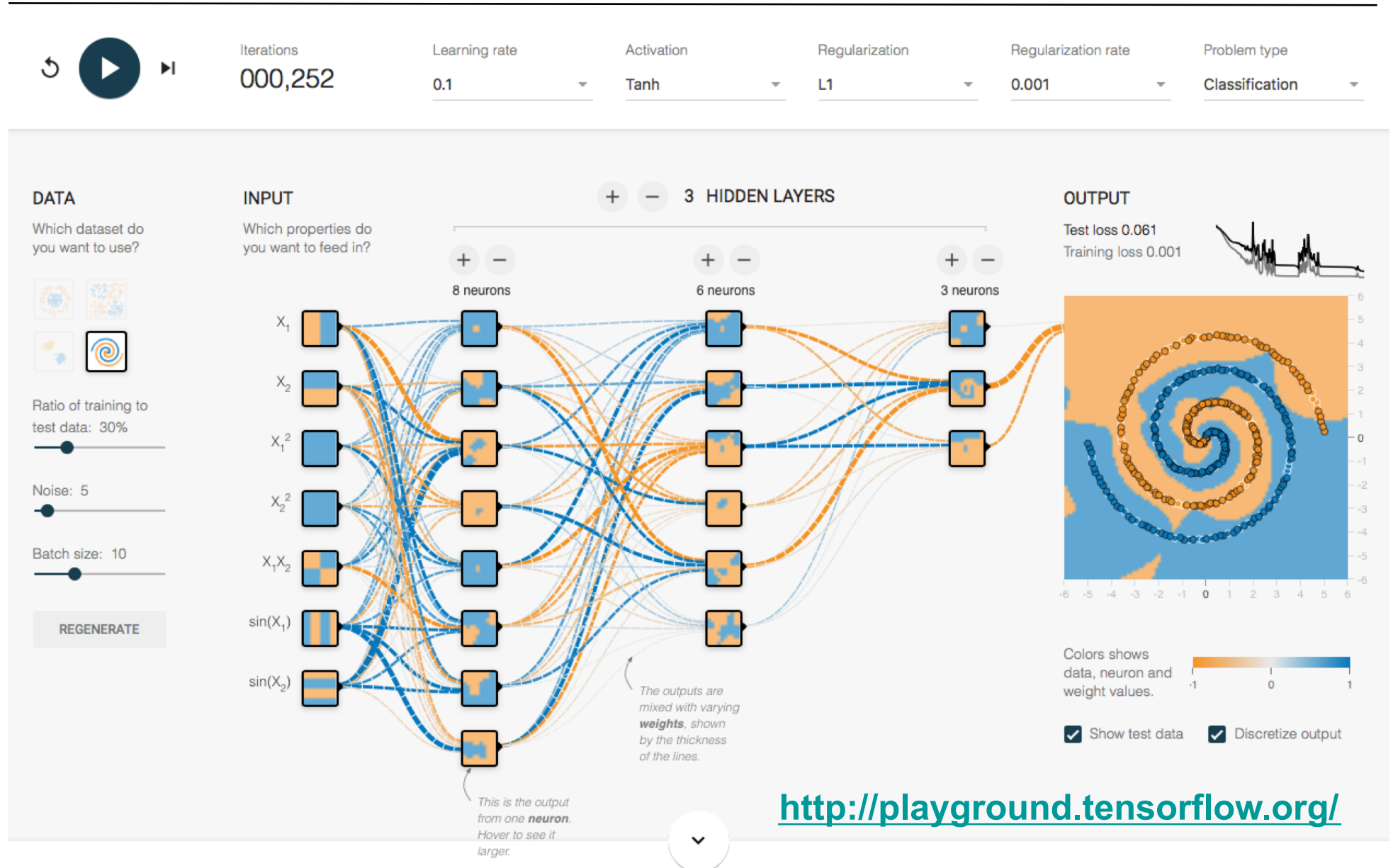
Linear vs Non Linear Decision Boundary

- A consequence of Cover's Theorem [Cover, 1966]
 - The probability that P samples (examples) of dimension N (number of parameters) are linearly separable goes to zero very quickly as P grows larger than N

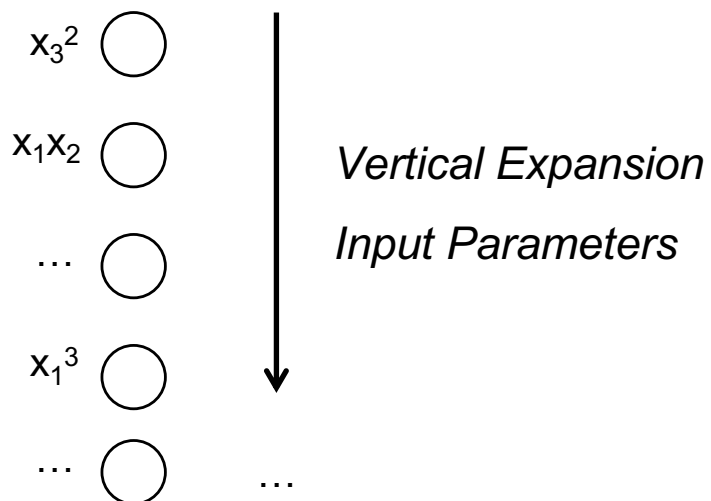
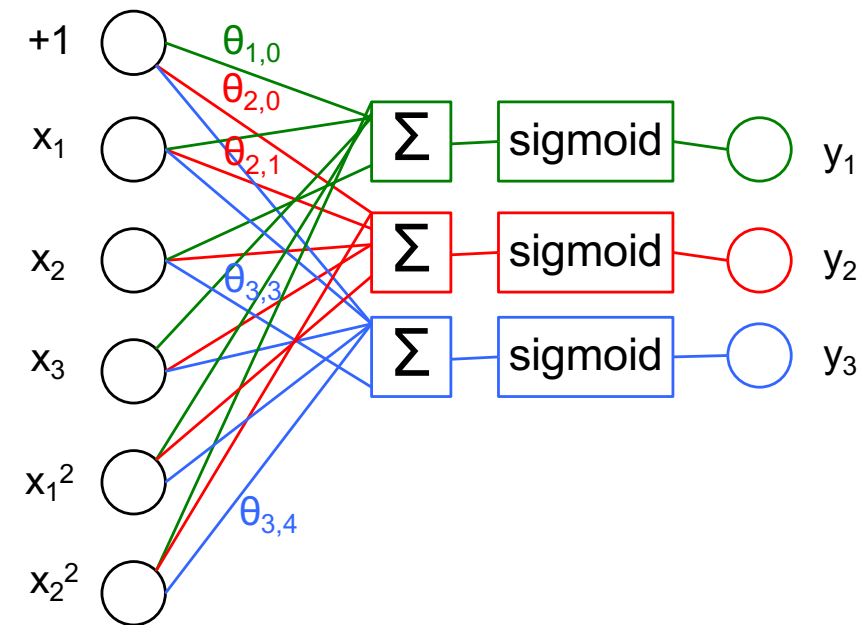


[LeCun 2016]

Example (TensorFlow Playground)



1st Approach: Polynomial Logistic Regression

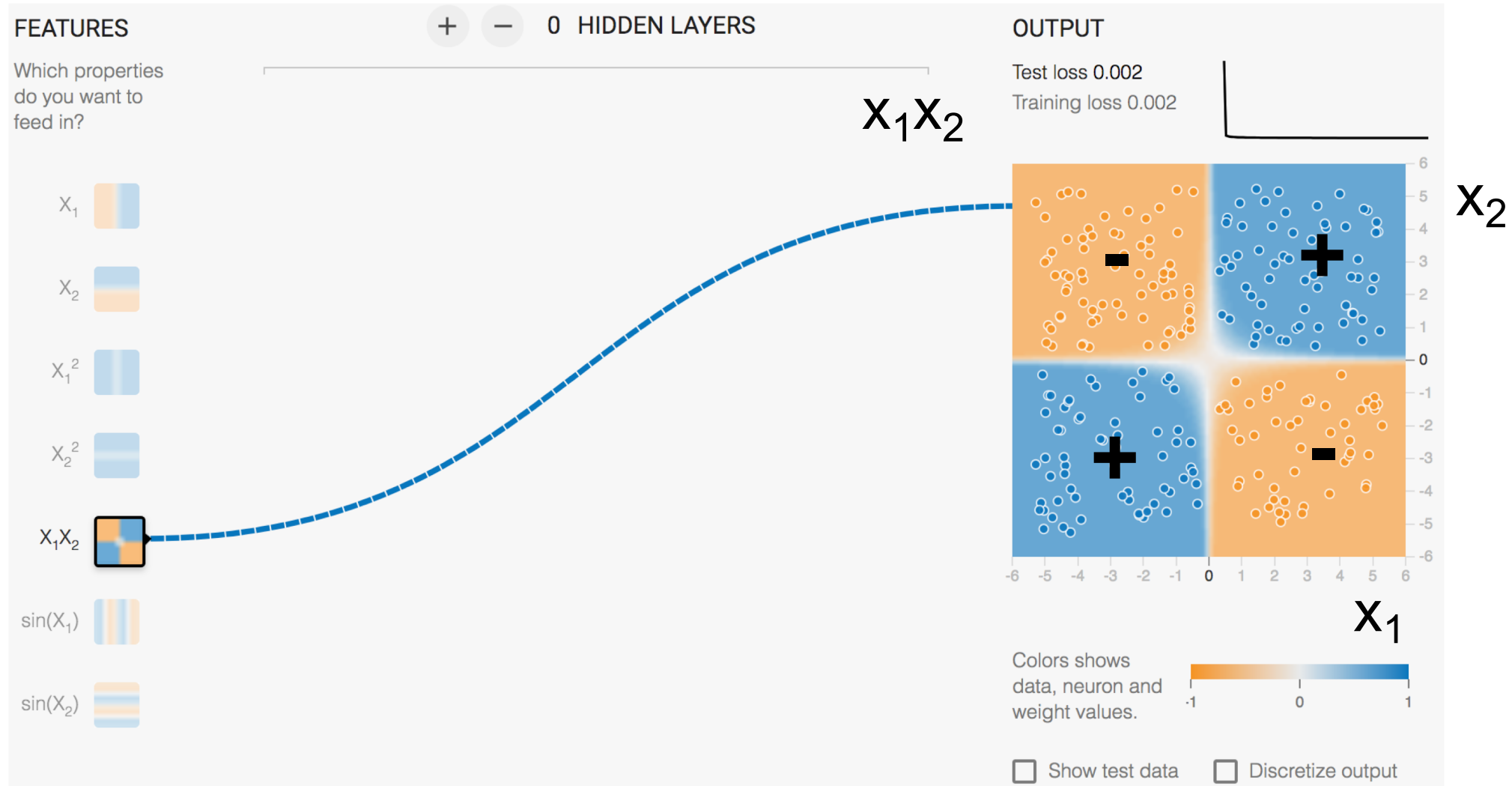


1st Approach: Polynomial Logistic Regression

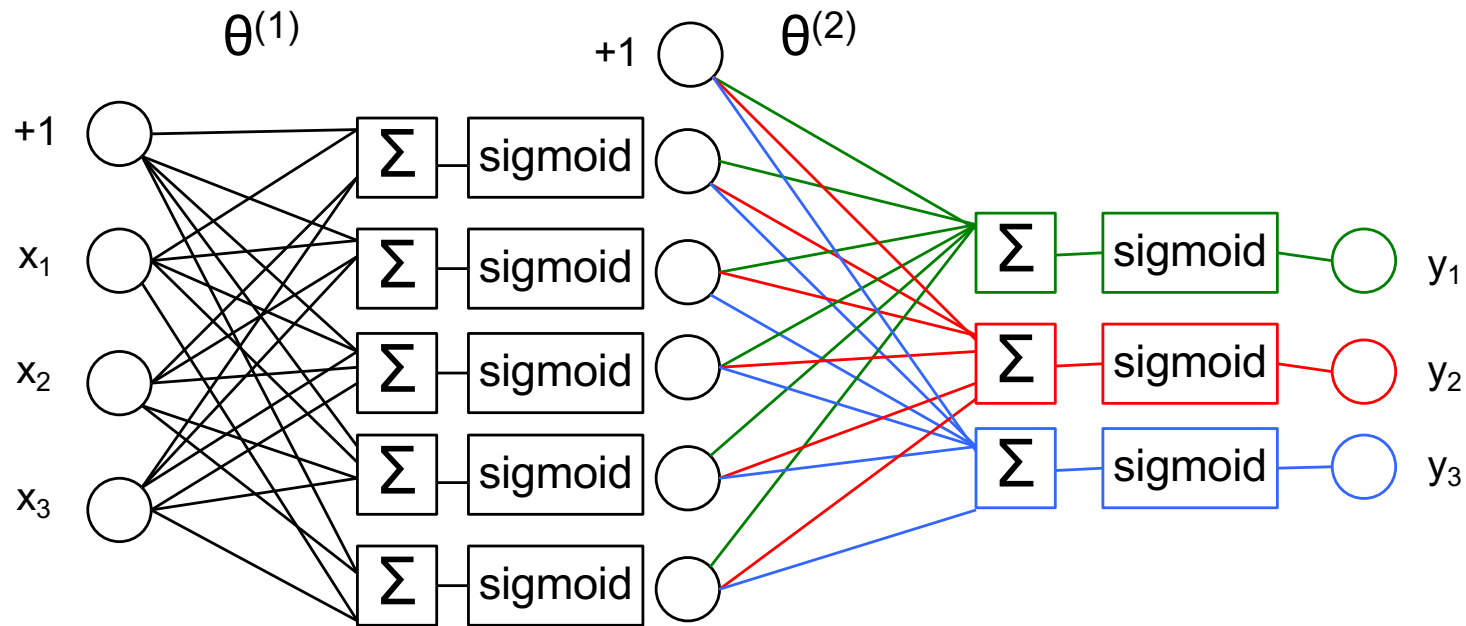
- Idea:
 - Increase the number of Dimensions
 - By adding Polynomial Inputs
 - » Quadratic: $x_1^2, x_2^2 \dots$
 - » Dot products: $x_1x_2, x_1x_3 \dots$
 - » Cubic: $x_1^3, x_2^3 \dots$
 - » Sinus: $\sin(x_1), \sin(x_2) \dots$
 - » ...
- In order to find (explore) a higher Dimension Space in which could be found a Linear Decision Boundary
- Problems:
 - Select the Additional Inputs
 - Add them in a combinatorial way
 - The Number of Inputs could become Very Large
 - Computation and Data becomes more Heavy

Example: x_1x_2

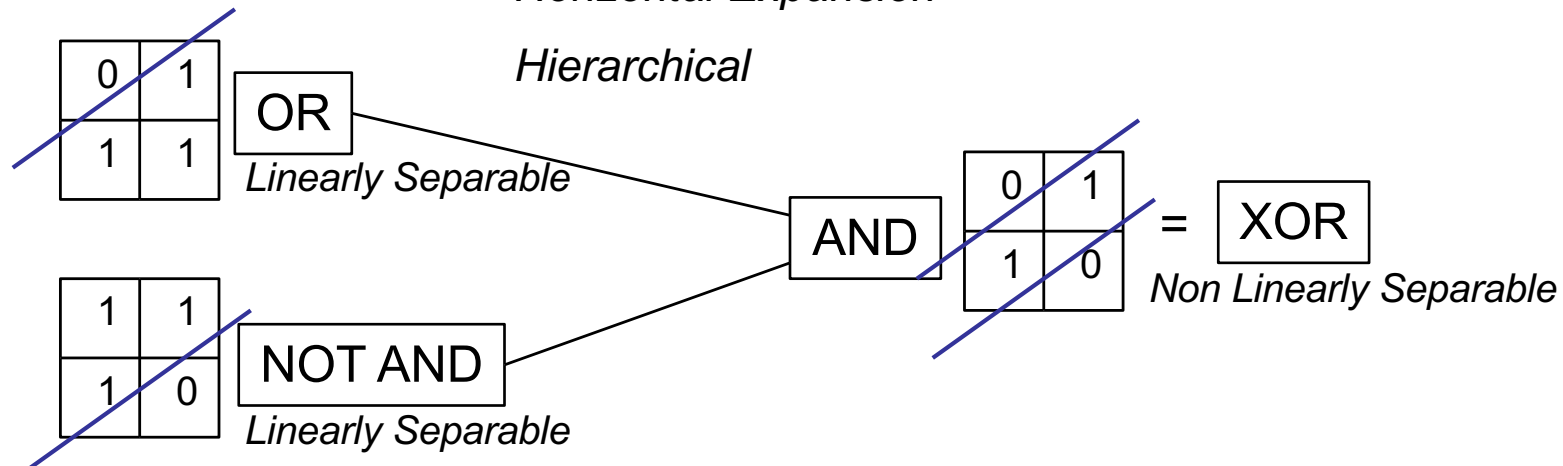
- Initial Representation $x_1 \times x_2$
- New Representation x_1x_2



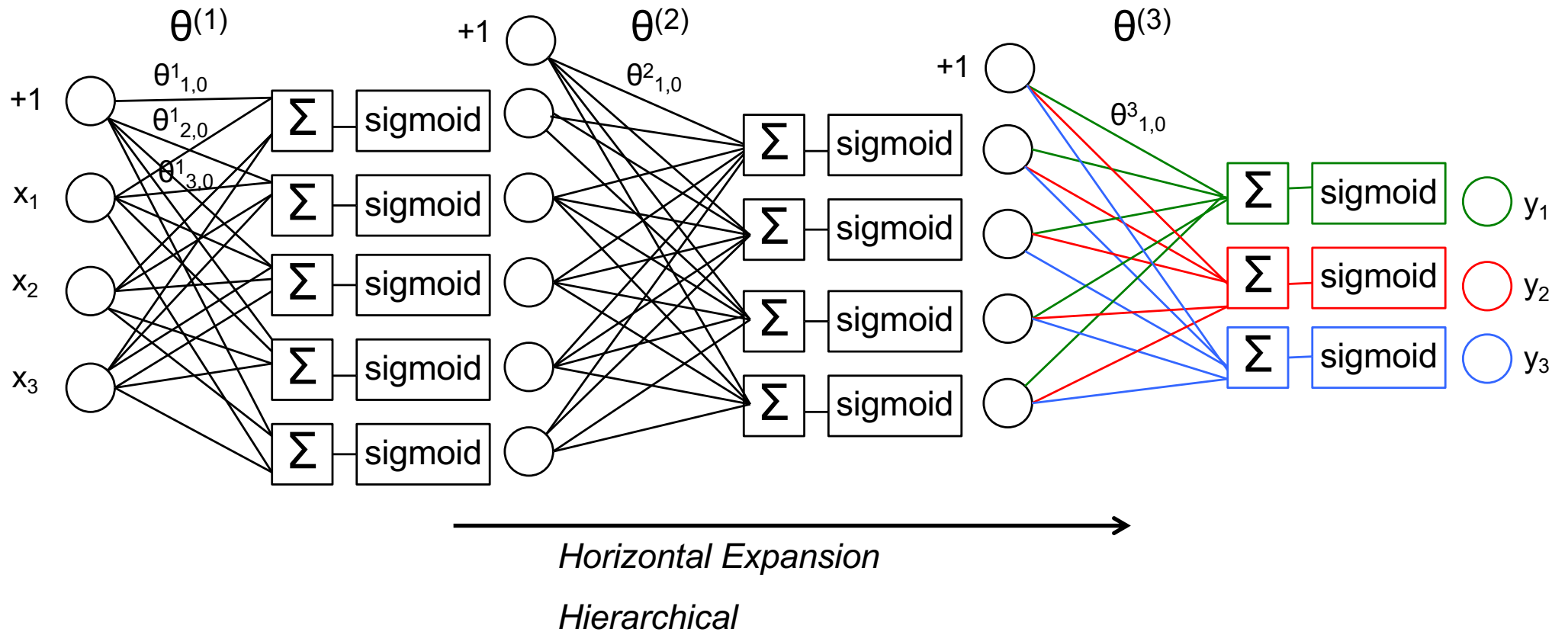
2nd Approach: Hidden Layers



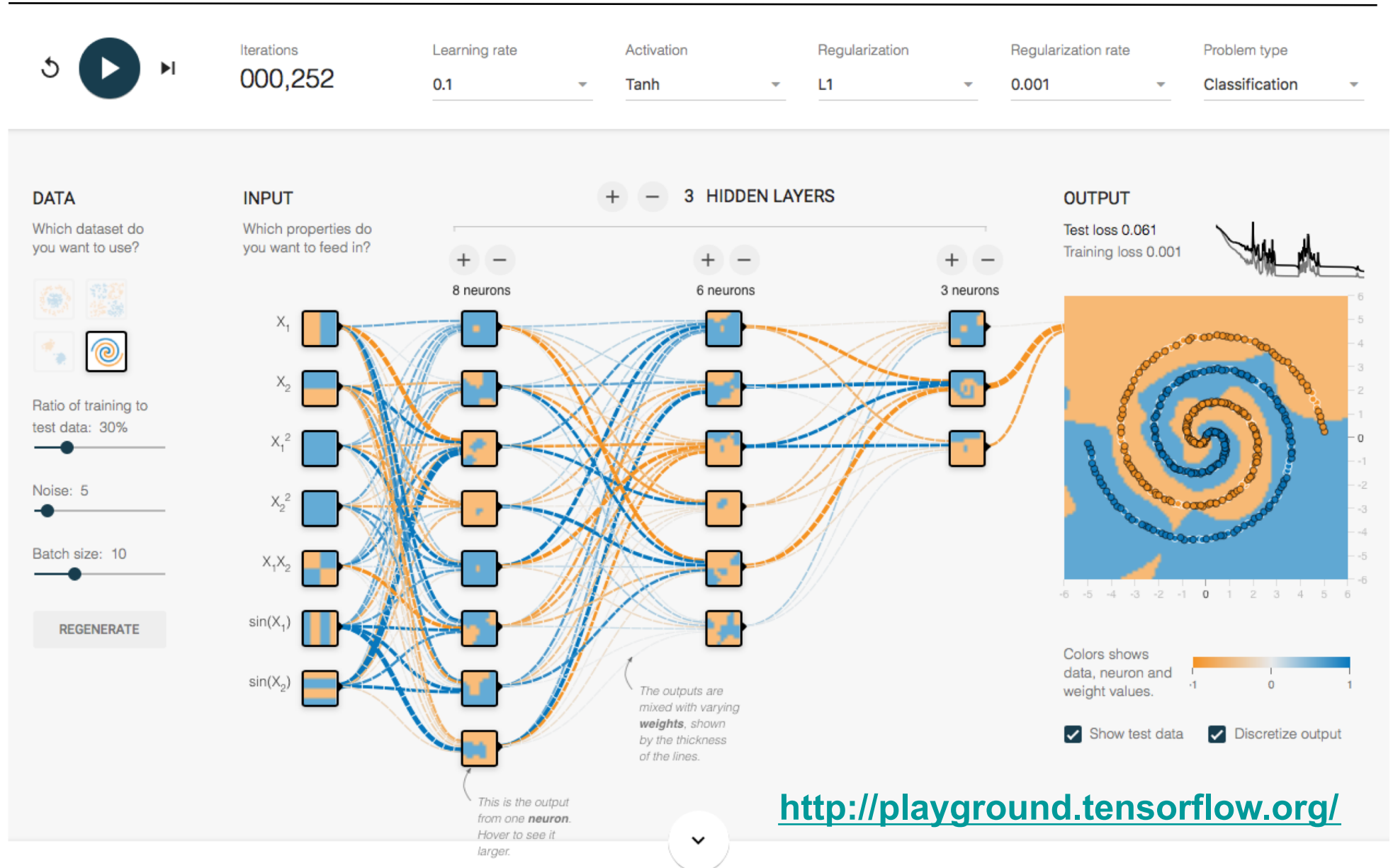
Horizontal Expansion



2nd Approach: Hidden Layers



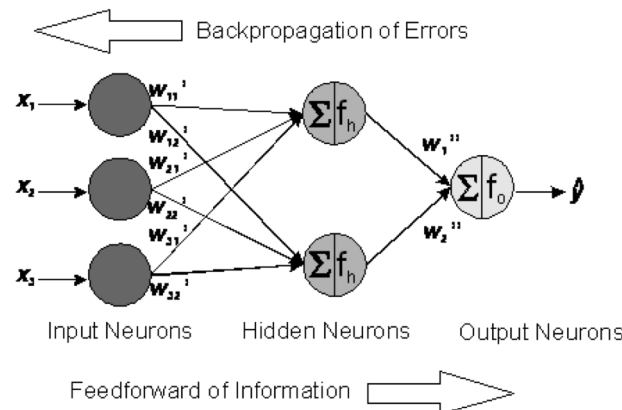
Example (TensorFlow Playground)



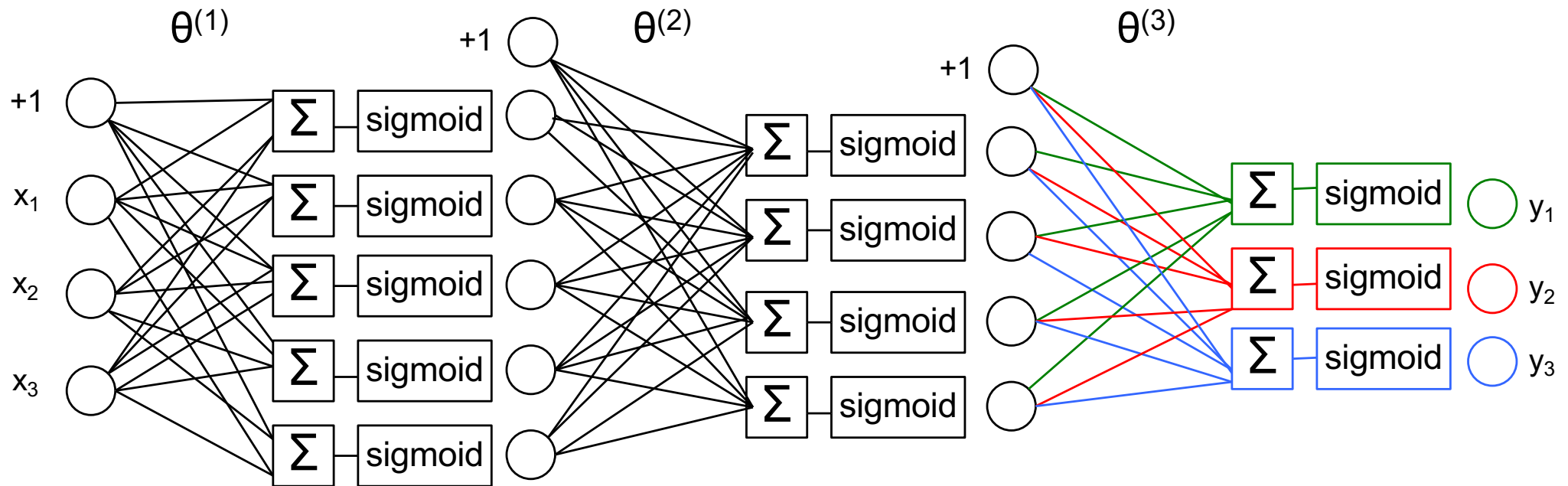
Backpropagation

Issue:

- How to compute **Gradients** (partial derivatives respective to each $\theta^{(k)}_{i,j}$ (weight)) ?
- Now that there are several hidden layers...
- **Backpropagation** (backward propagation of errors) algorithm [Rumelhart et al. 1986, initial ideas since 1960] estimates the gradients for each weight $\theta^{(k)}_{i,j}$ (in each layer k), through a **chain rule** approach:
 - Feedforward the network to compute the output
 - Compute output layer units errors (deltas between predicted and actual values)
 - Propagate backward the errors (deltas) to each unit of previous layer
 - And then compute (accumulate) the gradients (relative to each weight)



2nd Approach: Hidden Layers – Limits

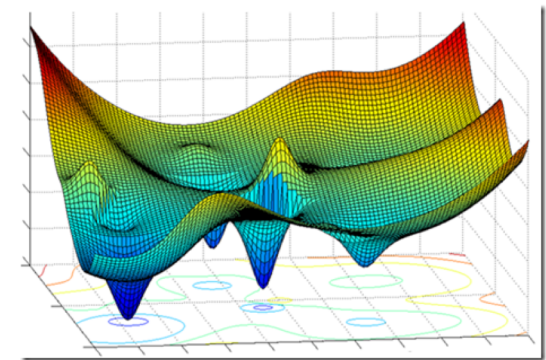


- **Non Convex** Optimization

- Importance/Difficulty of Initialization of Weights ($\theta^{(n)}_{i,j}$)
- Local Minimum

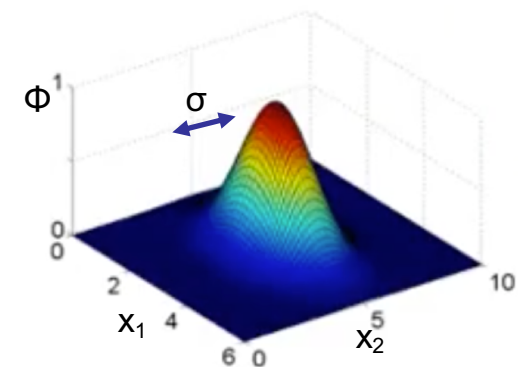
- **Gradient Vanishing** [Hochreiter 1991]

- Backpropagation algorithm (chain rule based) to propagate gradients (from output errors)
- Thinner Estimation (Increasing errors) of gradients while traversing layers



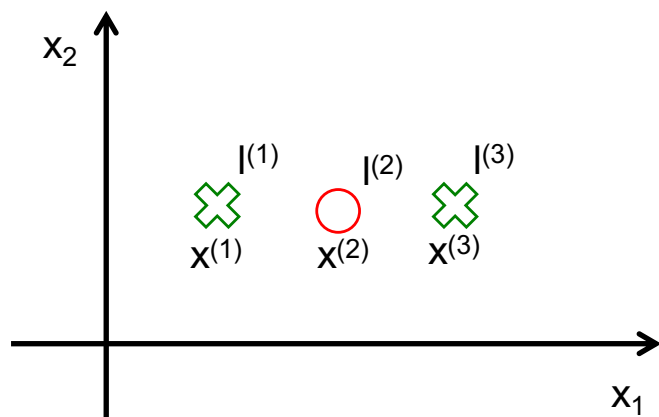
3rd Approach: Kernel (Trick)

- Increase the number of Dimensions
- By a Transformation of Input Parameters into a Linearly Separable Higher Dimension Feature Space
- A Change of Representation
- This Transformation Function K is Named a *Kernel Function*
- A Kernel function $K(x^{(1)}, x^{(2)})$ represents the *Similarity* between two examples ($x^{(1)}$ and $x^{(2)}$)
- Example of Kernel: Gaussian Kernel
 - also named Radial Basis Function (RBF) Kernel
 - $K(x^{(i)}, l^{(i)}) = \text{similarity}(x^{(i)}, l^{(i)}) = \exp(- \|x^{(i)} - l^{(i)}\|^2 / 2\sigma^2)$
 - Other Kernels:
 - » Linear (Without) Kernel, Polynomial Kernel, Sigmoid Kernel...
 - » Homemade Kernel (must satisfy Mercer property [Mercer 1909])



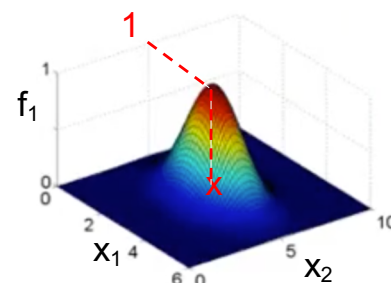
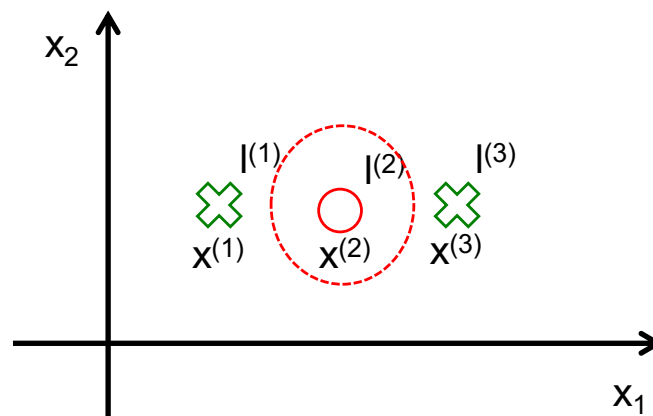
Kernel = Samples-Centered Feature Transformation

- Let's consider all ($m = 3$) examples $x^{(i)}$ as *Landmarks* $l^{(i)}$

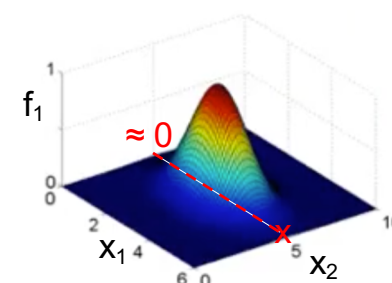


- Features are defined as following:

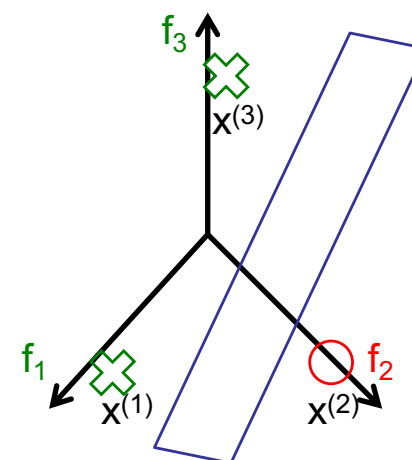
- $f_1(x) = K(x, l^{(1)}) = \exp(- \|x - l^{(1)}\|^2 / 2\sigma^2)$
- $f_2(x) = K(x, l^{(2)})$
- ...
- $f_m(x) = K(x, l^{(m)})$



$$f_1(x^{(1)}) = K(x^{(1)}, l^{(1)}) = 1$$



$$f_1(x^{(2)}) = K(x^{(2)}, l^{(1)}) \approx 0$$



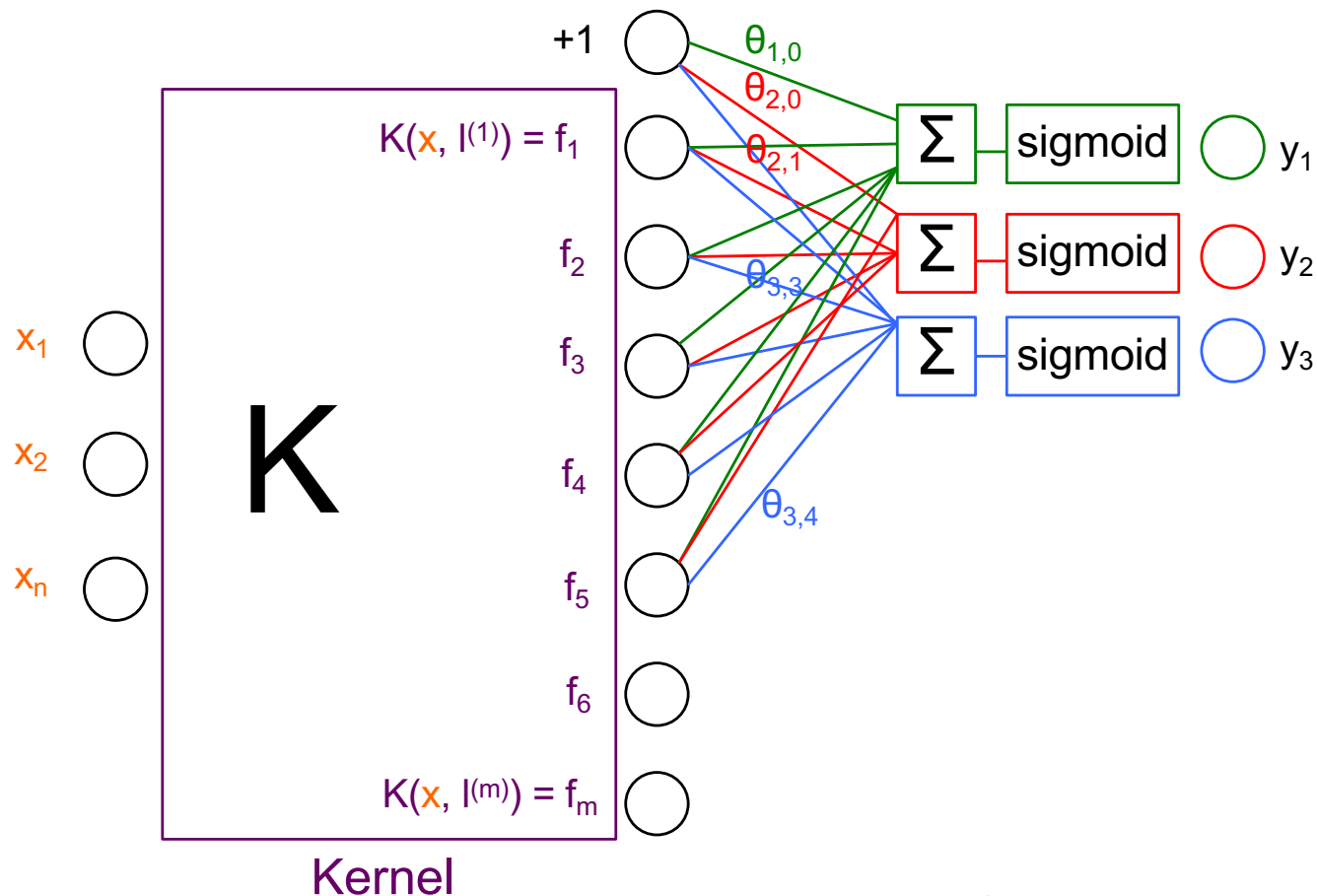
Kernel Trick

- Computing the Kernel is much easier (and more efficient) when it can be expressed as an inner product (in the feature space) of a kernel function f
- $K(x^{(1)}, x^{(2)}) = f(x^{(1)}) \cdot f(x^{(2)})$
- Ex: $K(x, y) = (\langle x_1, x_2 \rangle \cdot \langle y_1, y_2 \rangle)^2$
 $= (x_1 y_1 + x_2 y_2)^2$
 $= x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2$
 $= \langle x_1^2, \sqrt{2} x_1 x_2, x_2^2 \rangle \cdot \langle y_1^2, \sqrt{2} y_1 y_2, y_2^2 \rangle$
 $= f(x) \cdot f(y)$
with $f(x) = \langle x_1^2, \sqrt{2} x_1 x_2, x_2^2 \rangle$

Mercer's Theorem: Characterization of a function as a kernel function

Kernel (Trick)

- The initial n dimension space (number of parameters) is transformed into a m (number of Training examples) dimension space of features (new parameters), linearly separable
- This new space is independent from the number of parameters (n) and so depending on the number of training examples (m)



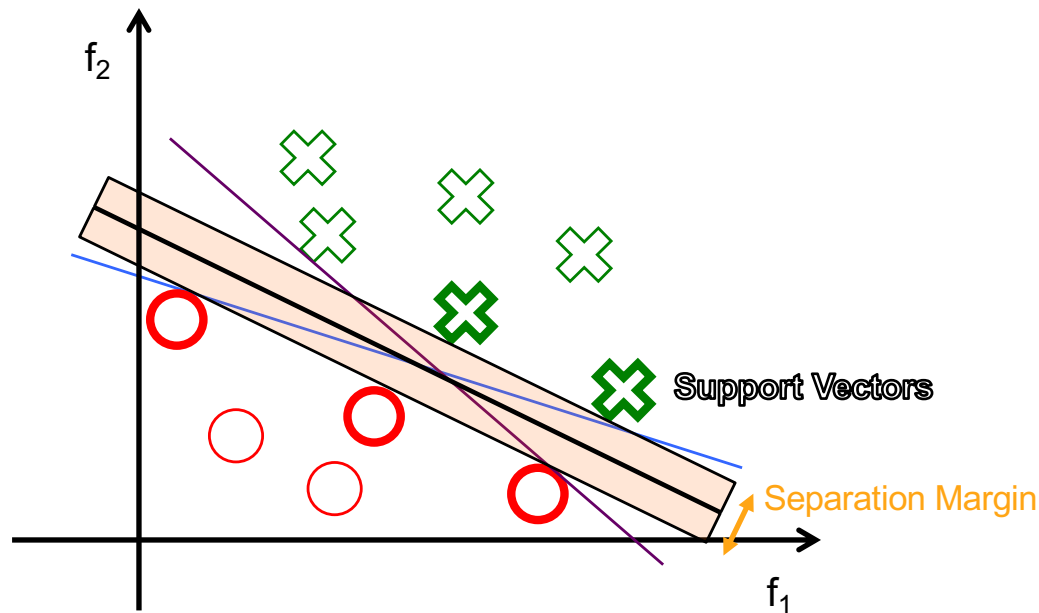
Note:

There are 2 phases:

- Configuration of the Kernel component with Training examples – landmarks, $l^{(1)}, \dots, l^{(m)}$ are set and stored into the Kernel component
- Application of the Kernel component as a dimension transformation on examples ($X \rightarrow F$)

Support Vector Machines

- The same Kernel Trick is used by Support Vector Machines
- Support Vector Machine [Boser et al. 1992] [Vapnik 1995]
= **Kernel** [Mercer 1909] + **Linear Support Vector Machine** [Vapnik & Lerner 1963]
- A Linear Support Vector Machine is good at maximizing the **Separation Margin**

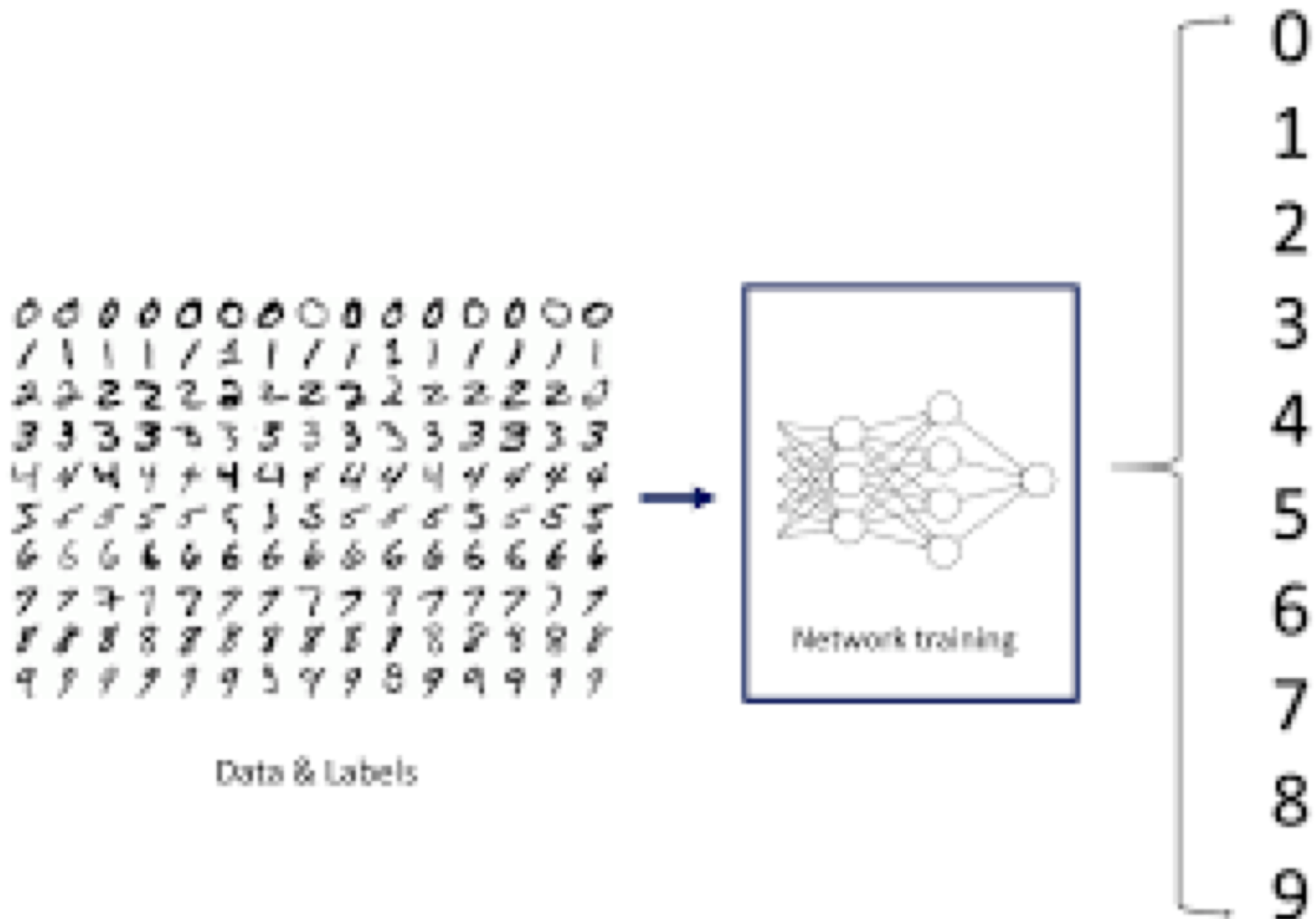


- From 2000's, SVMs outperformed Neural Networks and led to the decrease of research in Neural Networks
- Until the 2010's, when they returned through the idea of Deep Networks/Learning (see later)

MNIST Handwritten Character Recognition Test Case

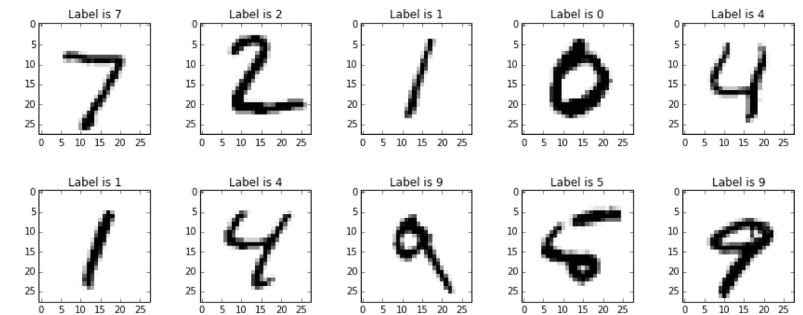


MNIST Handwritten Character Recognition Test Case



MNIST Handwritten Character Recognition Test Case

- Program in Keras (Python)
- Dataset/Examples: 70.000 characters
 - Each character: 28x28 pixels
 - Associated label: $\in \{0, 1, \dots, 9\}$
- Dataset split into:
 - Training Set (60.000)
 - Evaluation (Testing) Set (10.000)
- Scaling of data
 - Pixel: Grayscale $\in (0, 255)$
 - Divide by 255
- Input : X = vector of vectors of characters
- Output : y = vector of corresponding labels
 - One-hot encoding of label



label encoding labels

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

MNIST Handwritten Character Recognition Test Case Programs

- Standard
 - 2 Hidden Layers (512 units)
 - Regularization (Dropout)
 - 10 Epochs
 - Accuracy: 98.33 %
- Simplified
 - 1 Hidden Layer (100 units)
 - No Regularization
 - 5 Epochs
 - Accuracy: 92.25 %
- Convolutional
 - 4 Hidden Layers
 - » 2 Convolutional (30 and 15 units) and 1 Pooling
 - » Dense (128, 50)
 - Regularization (Dropout)
 - 10 Epochs
 - Accuracy: 99.11 %

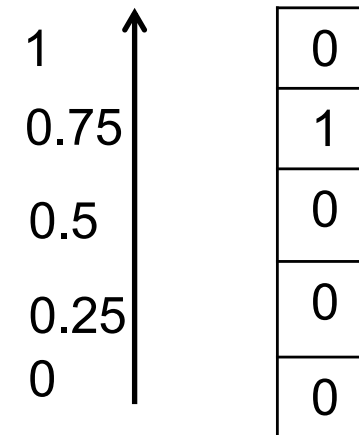
Determinism vs Stochastic (Sampling)

- Neural Networks are **Deterministic**
- As opposed to ex. Markov Chains, MDPs and RBMs, where Production (**Sampling**) is **Stochastic**, with Distribution Properties Expected

	<i>Learning</i>	<i>Production</i>
<i>Deterministic</i>	<ul style="list-style-type: none">• Gradient Descent	<ul style="list-style-type: none">• Neural Networks
<i>Stochastic</i>	<ul style="list-style-type: none">• Stochastic Gradient Descent (SGD)	<ul style="list-style-type: none">• Neural Networks with Sampling• Markov Chains• Markov Decision Processes (MDP) and Reinforcement Learning• Restricted Boltzmann Machines (RBM)

Representation (Embeddings)

- Importance of the choice of Representation
- Direct
- Features
 - Manual
 - Extracted by autoencoders
- Encoding
 - Value encoding (Analog) 0.75
 - Item (One-hot) encoding (Discrete)
- Ex: Case of Music
 - Audio
 - Spectrum (Fourier Transform)
 - MIDI
 - Piano Roll
 - Text
 - Time
 - Quantization
 - Fermata, Tonality...



Universal Approximator

- The combination of hidden layer and non linear activation function makes the neural network an **universal approximator**, able to overcome the non linear separability limitation
- The universal approximation theorem [Hornik, 1991] states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate a wide variety of interesting functions when given appropriate parameters (weights)
- **Meanwhile, there is no guarantee that the neural network will *learn* it !**

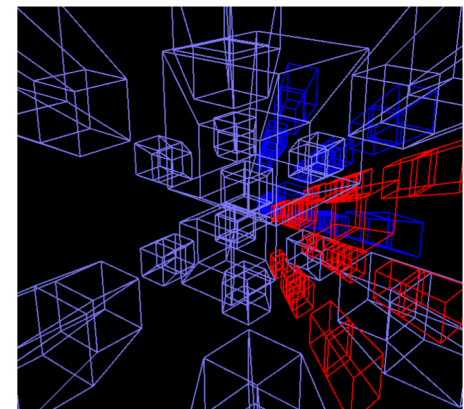
No Free Lunch !

- The no free lunch theorem for machine learning [Wolpert, 1996] states that, averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.
- No machine learning is universally any better than any other.
- The most sophisticated algorithm we can conceive has the same average performance, over all possible tasks, as merely predicting that every point belongs to the same class [Goodfellow *et al.*, 2016].
- But, these results hold when we average over all possible data-generating distributions.
- Real applications/data are specific, as well as related algorithms, and have some regularities and inner structure that an algorithm can exploit.

The Malediction of High Dimensionality [Mallat, 2018]

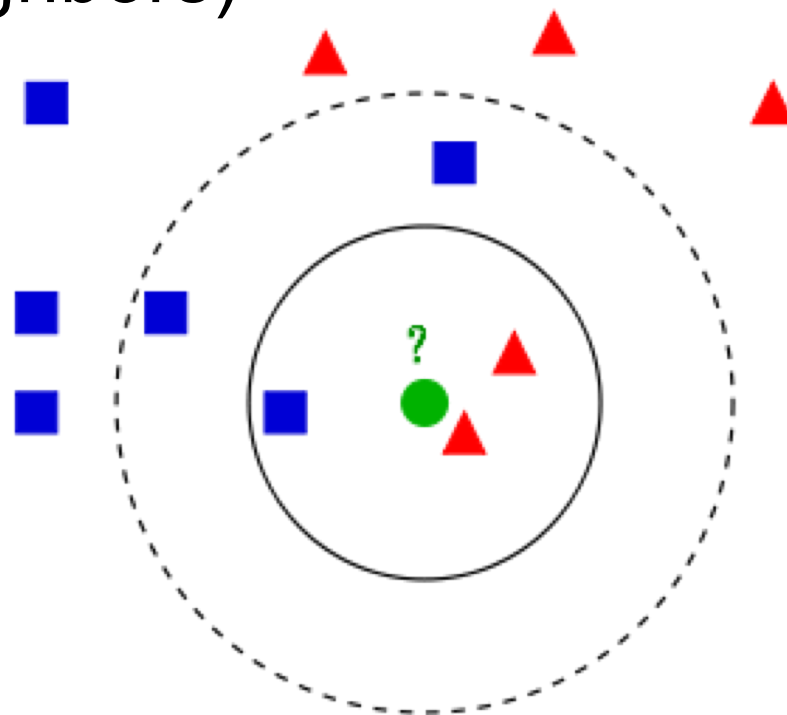


- Ex: Image with 2,000 x 1,000 pixels with color
- = 6.000.000 bits
- **Space of dimension 6.000.000 !!**



The Malediction of High Dimensionality [Mallat, 2018]

- Ex. of Task: Image recognition (Classification Task)
- kNN algorithm (k Nearest Neighbors)
- The class of an element = majority class of k nearest neighbors



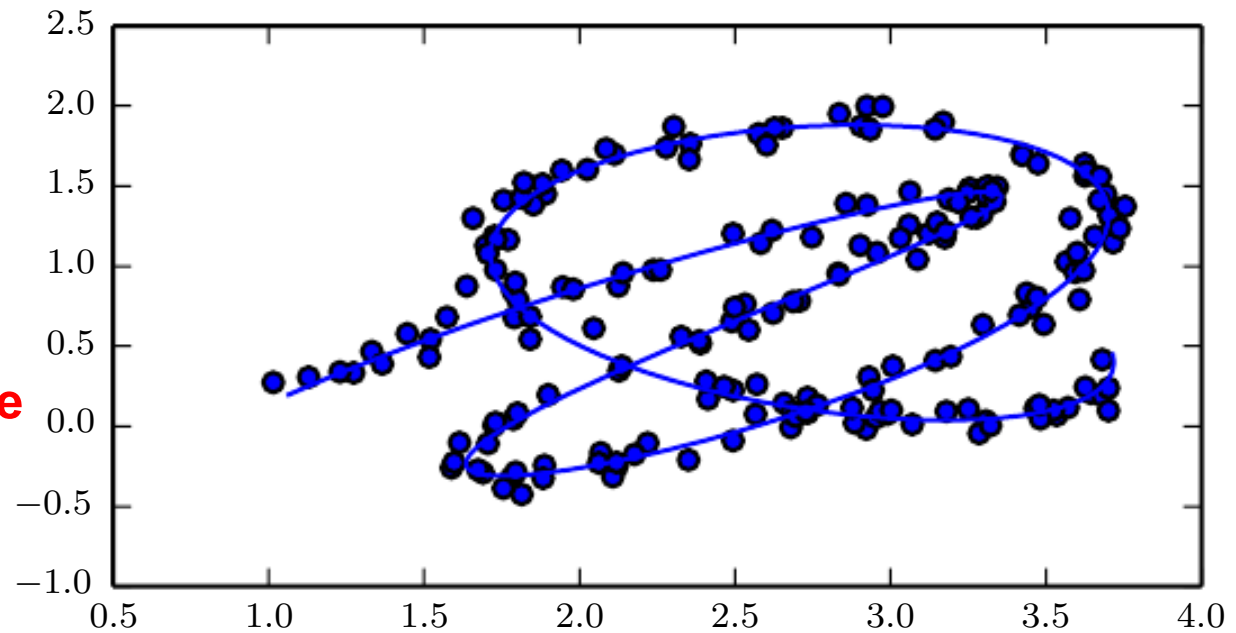
Problem:

- Euclidean distance is unhelpful in high dimensions because all vectors are almost equidistant to the query vector

Representation/Manifold Learning

Manifold :
Set of connected points

In a **high-dimensional space**

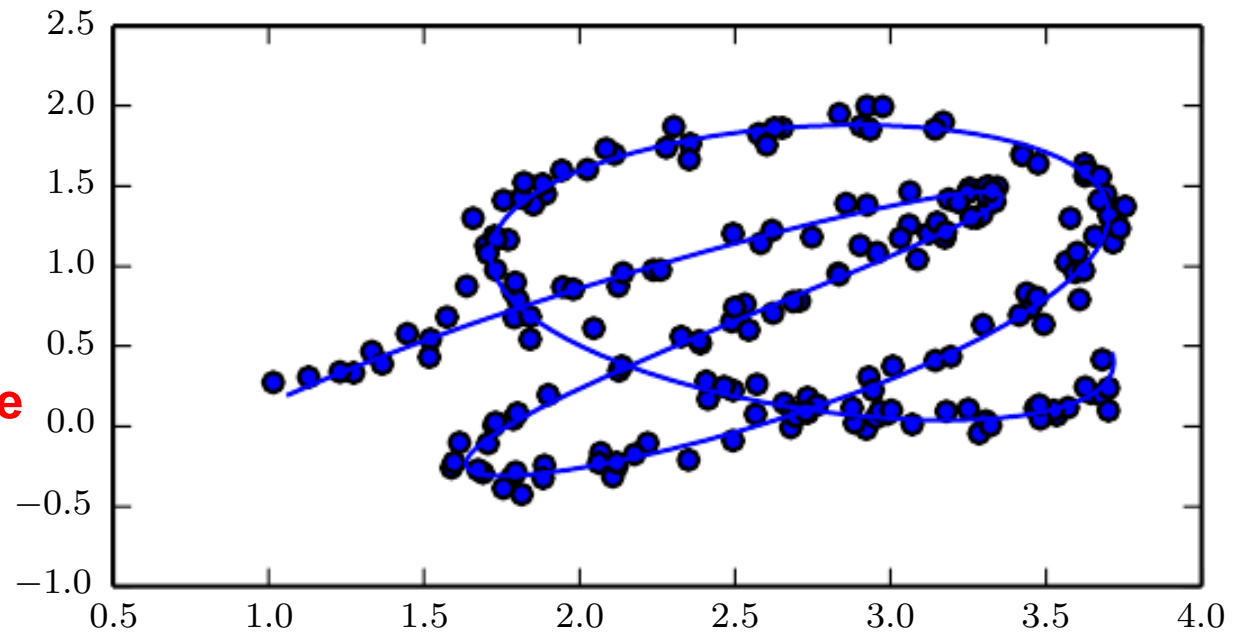


[Goodfellow et al., 2016]

Representation/Manifold Learning

Manifold :
Set of connected points

In a **high-dimensional space**

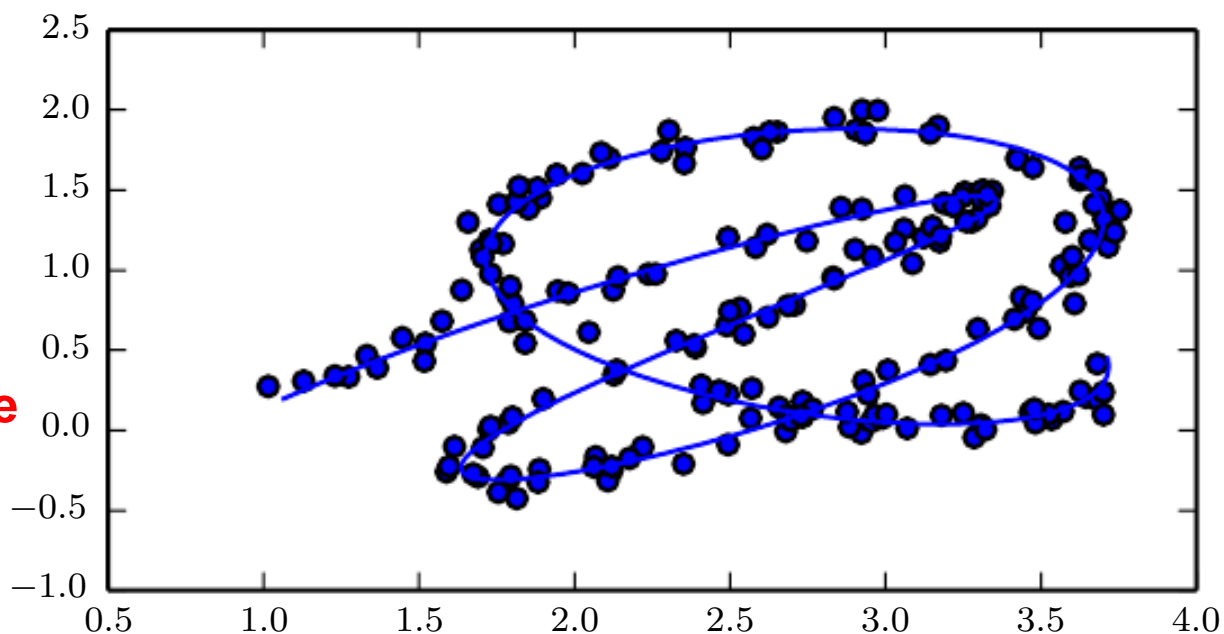


But can be approximated by
a **smaller number of dimensions**,
each dimension corresponding
to a **local variation**

Representation/Manifold Learning

Manifold :
Set of connected points

In a **high-dimensional space**

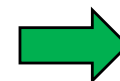


But can be approximated by
a **smaller number of dimensions**,
each dimension corresponding
to a **local variation**

Analogy:



3D Earth

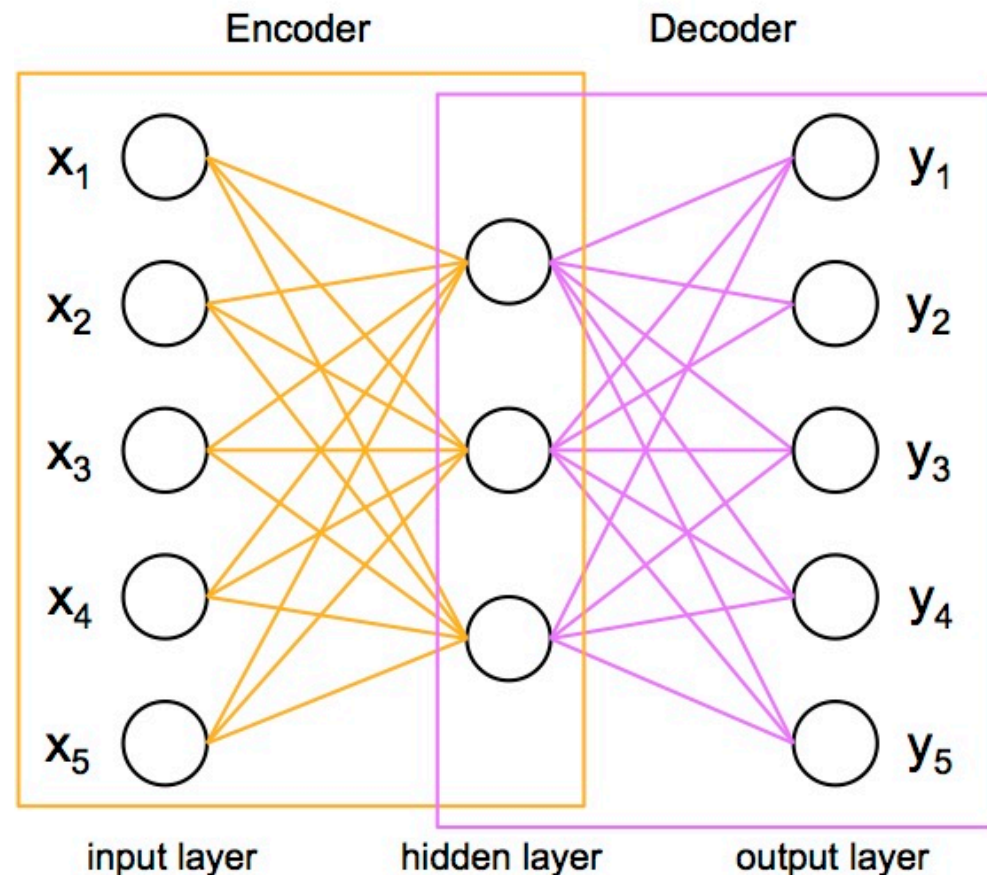


2D Map



Autoencoder

- Symmetric Neural Network
- Trained with examples as input and output
- Hidden Layer will Learn a **Compressed Representation at the Hidden Layer (Latent Variables)**

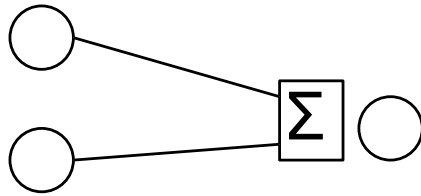


Autoencoder

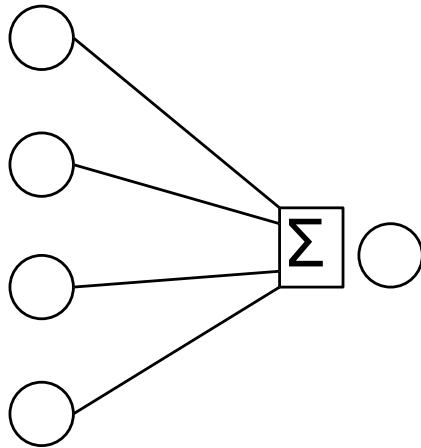
- Autoencoder can do such representation transformation
- Sparse autoencoder
 - Constraint: Sparse activation
 - -> Specialization of each unit/latent variable
 - Feature extractor
- Variational Autoencoder (VAE)
 - Constraint: latent variables follow a Gaussian distribution
 - -> Smooth latent space mapping
 - Representation learning

Deep Learning **Bio-inspired** or/and **Regression-based** ?

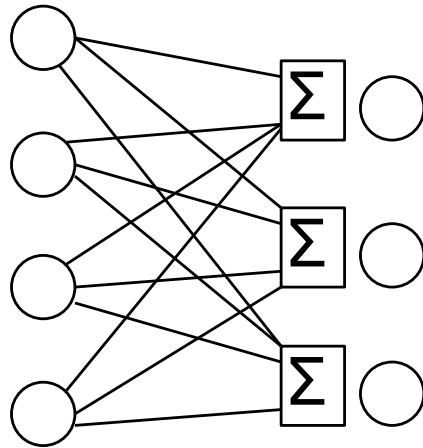
Simple Linear Regression



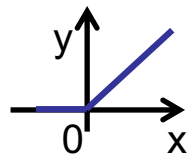
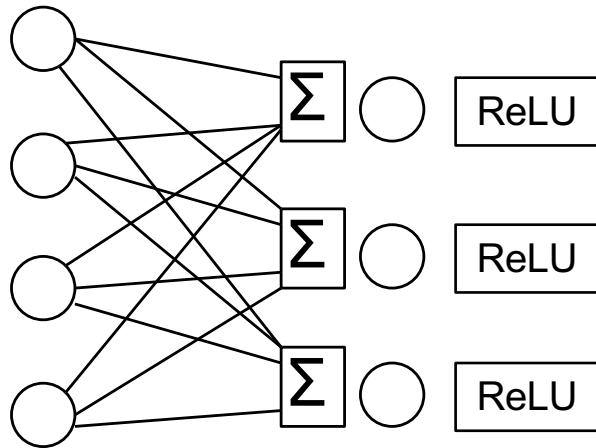
Multiple Linear Regression



Multivariate Multiple Linear Regression

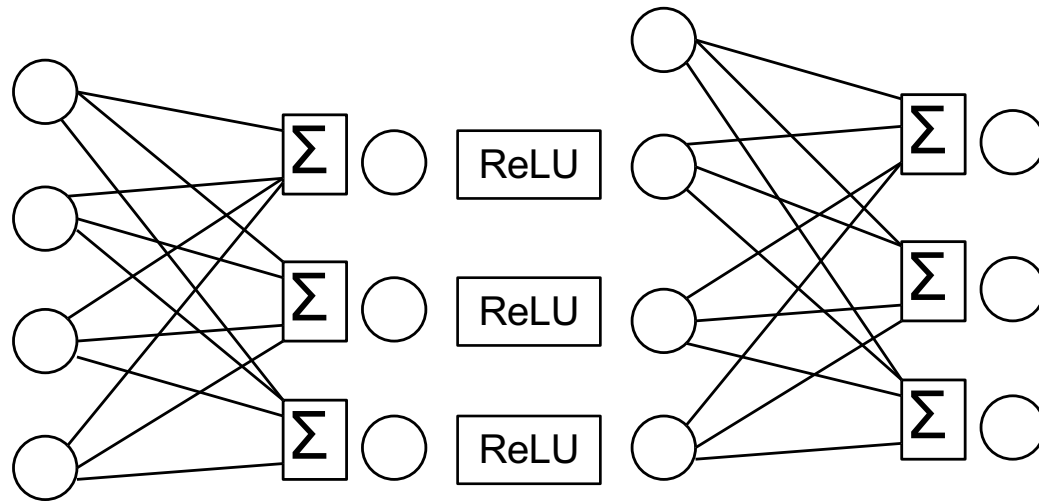


Non Linear Multivariate Multiple Linear Regression

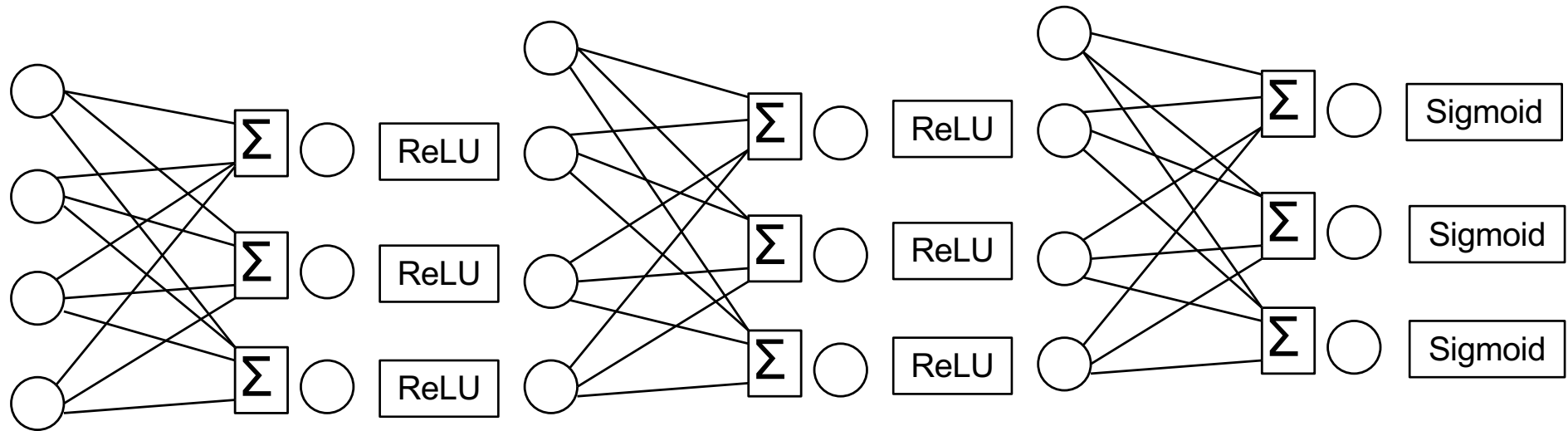


$\text{ReLU} = \max(x, 0)$
Rectified Linear Unit

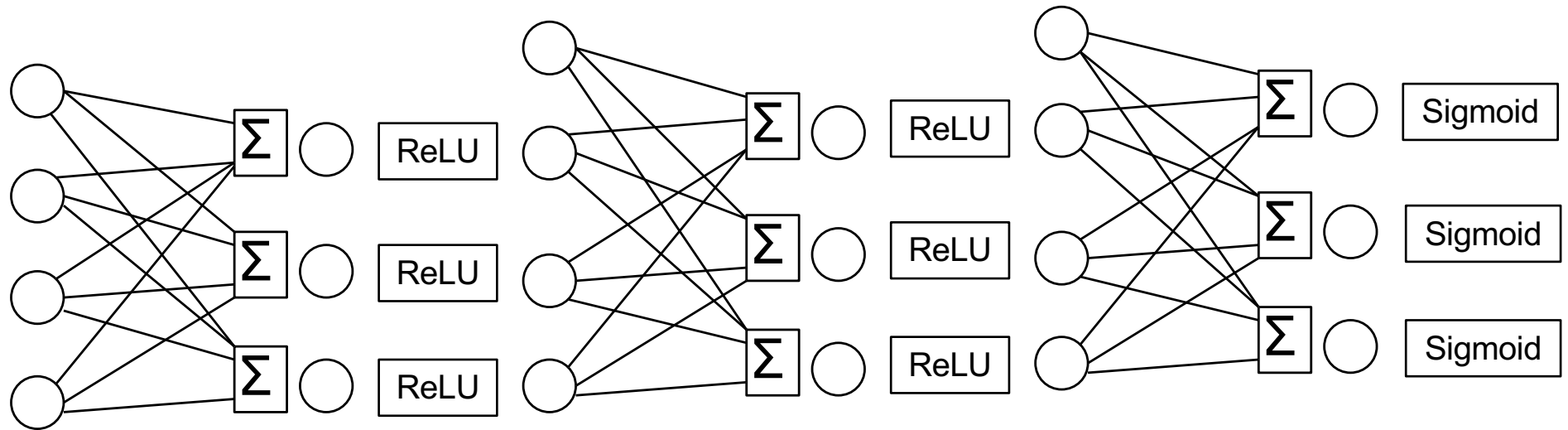
Multilayer Non Linear Multivariate Multiple Linear Regression



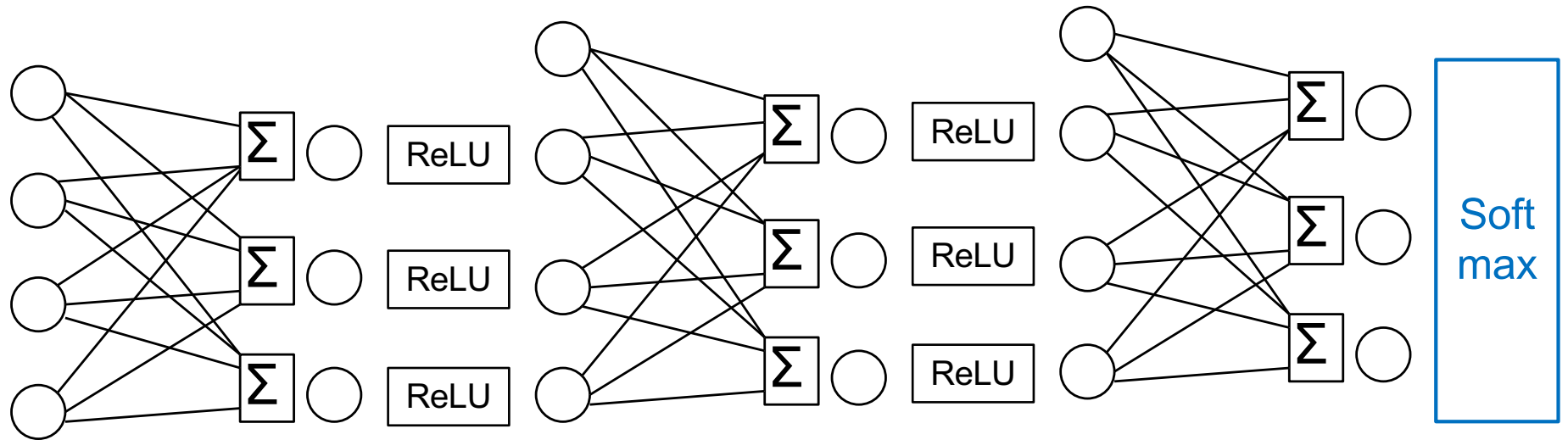
Multilayer Non Linear Multivariate Multiple Linear Regression



= Neural Network



Modern Neural Network



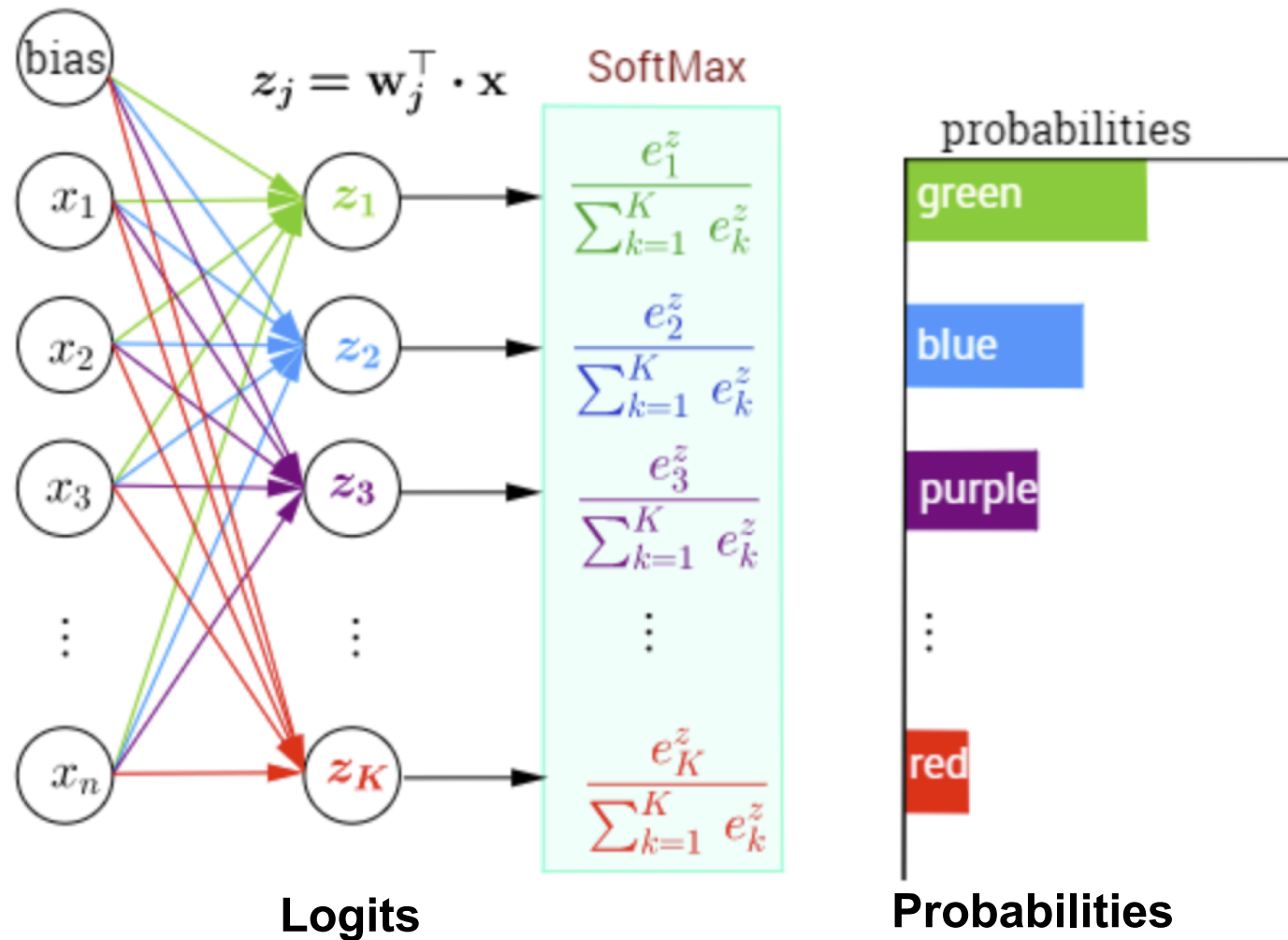
Softmax transforms values into probabilities

$$\sigma(z)_i = e^{z_i} / \sum e^{z_j}$$

*Generalization of **sigmoid***

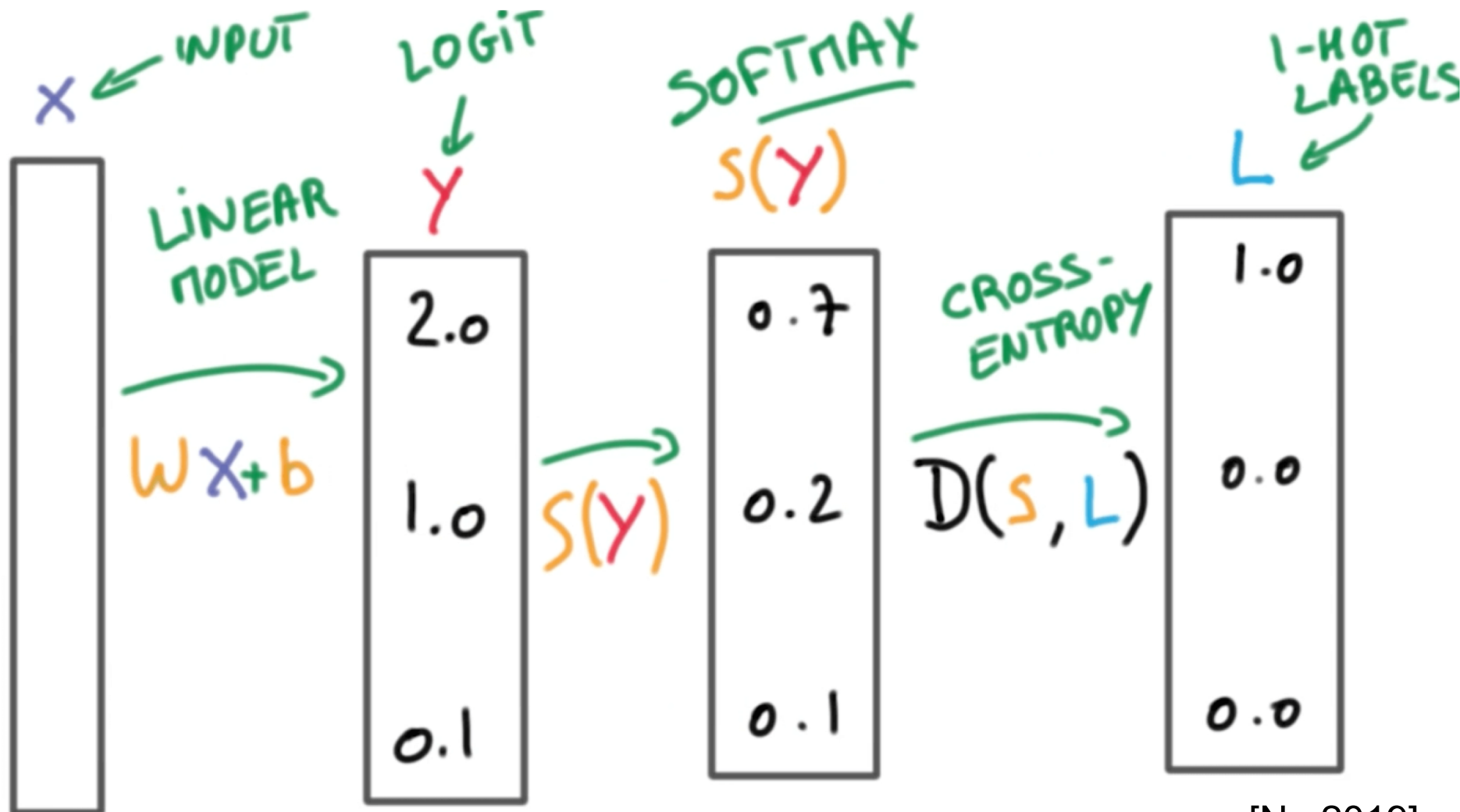
*Associated cost function is **cross entropy***

Softmax



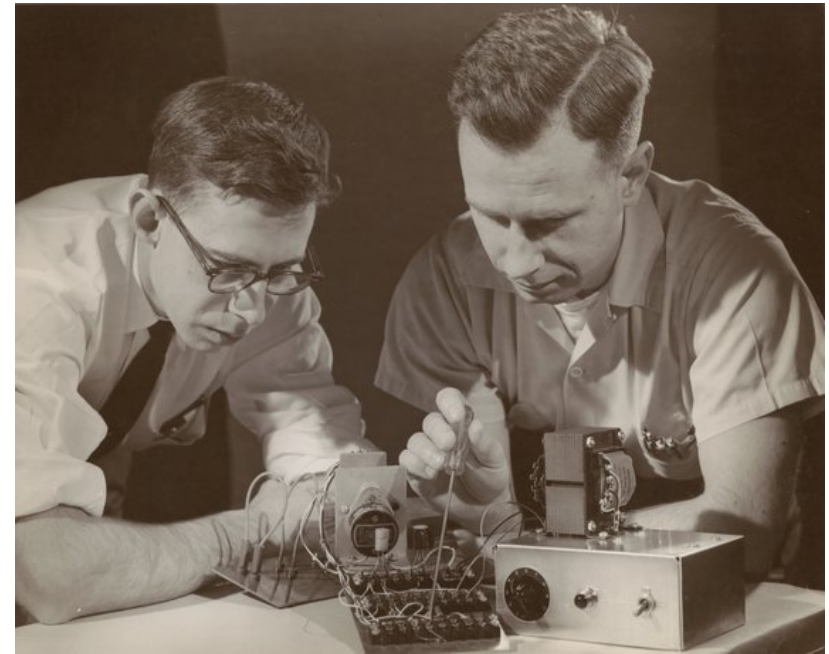
Softmax + Cross-Entropy

- Cross-Entropy measures dissimilarity between two probability distributions (prediction and target/true value)

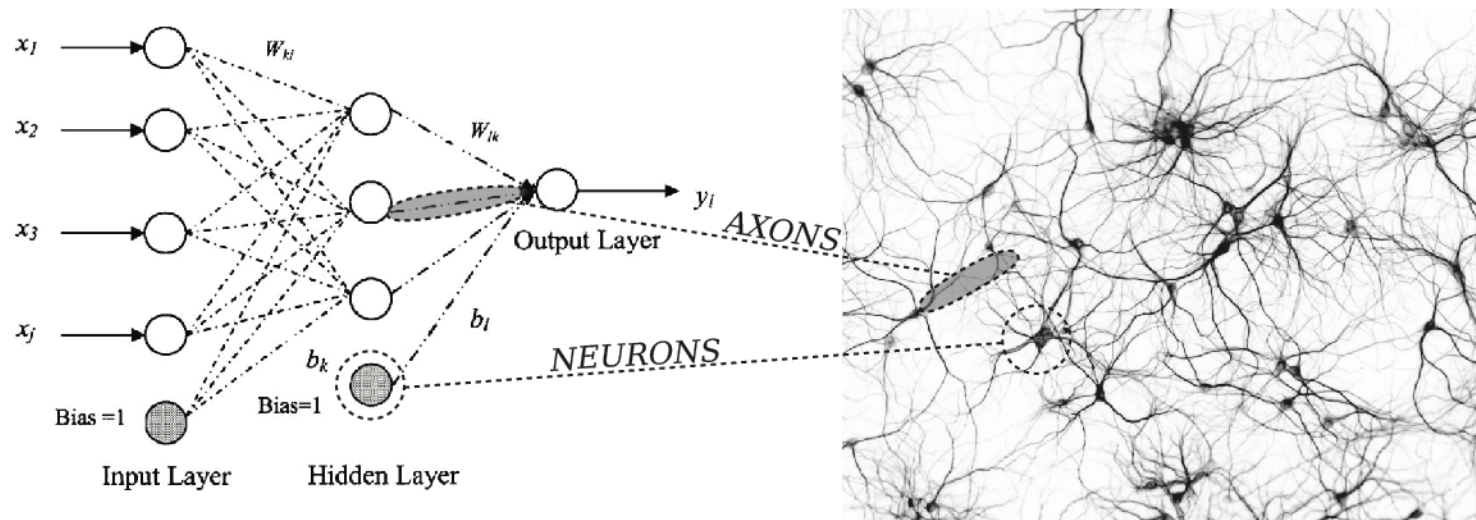


Deep Learning **Bio-inspired** or/and **Regression-based** ?

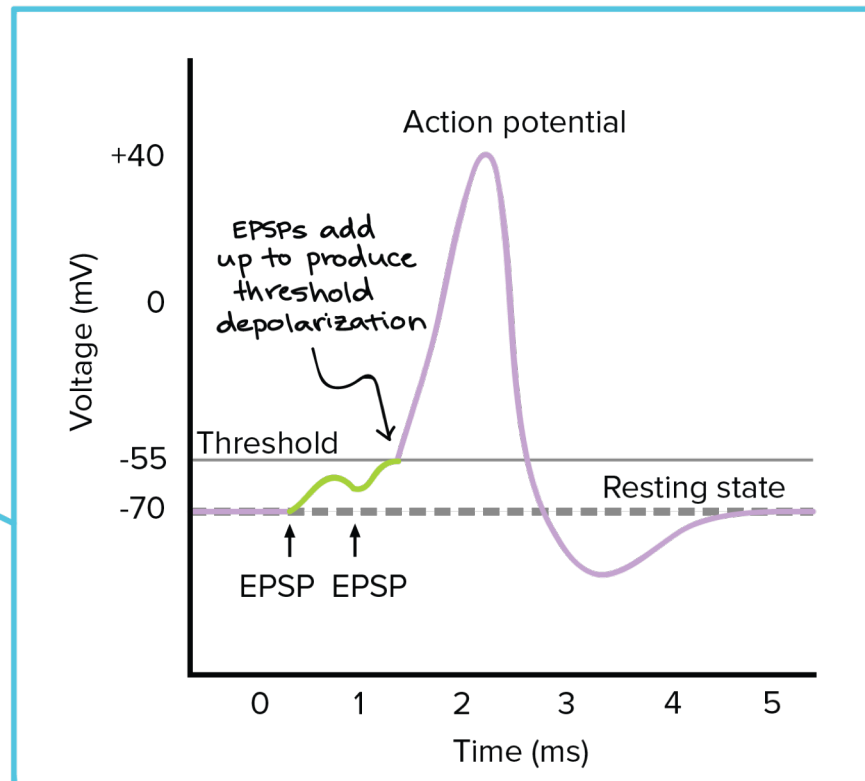
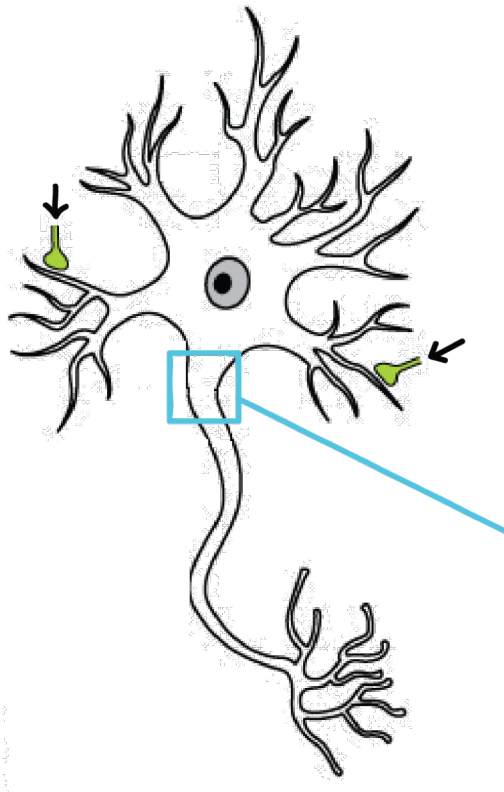
- Historically Perceptron bioinspired



NEURAL NETWORK MAPPING

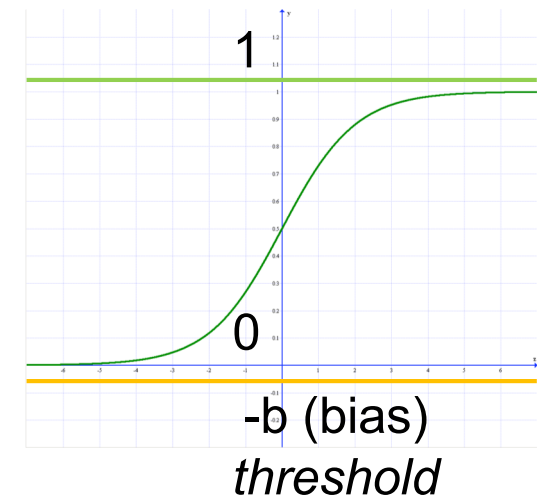


Deep Learning **Bio-inspired** or/and **Regression-based** ?



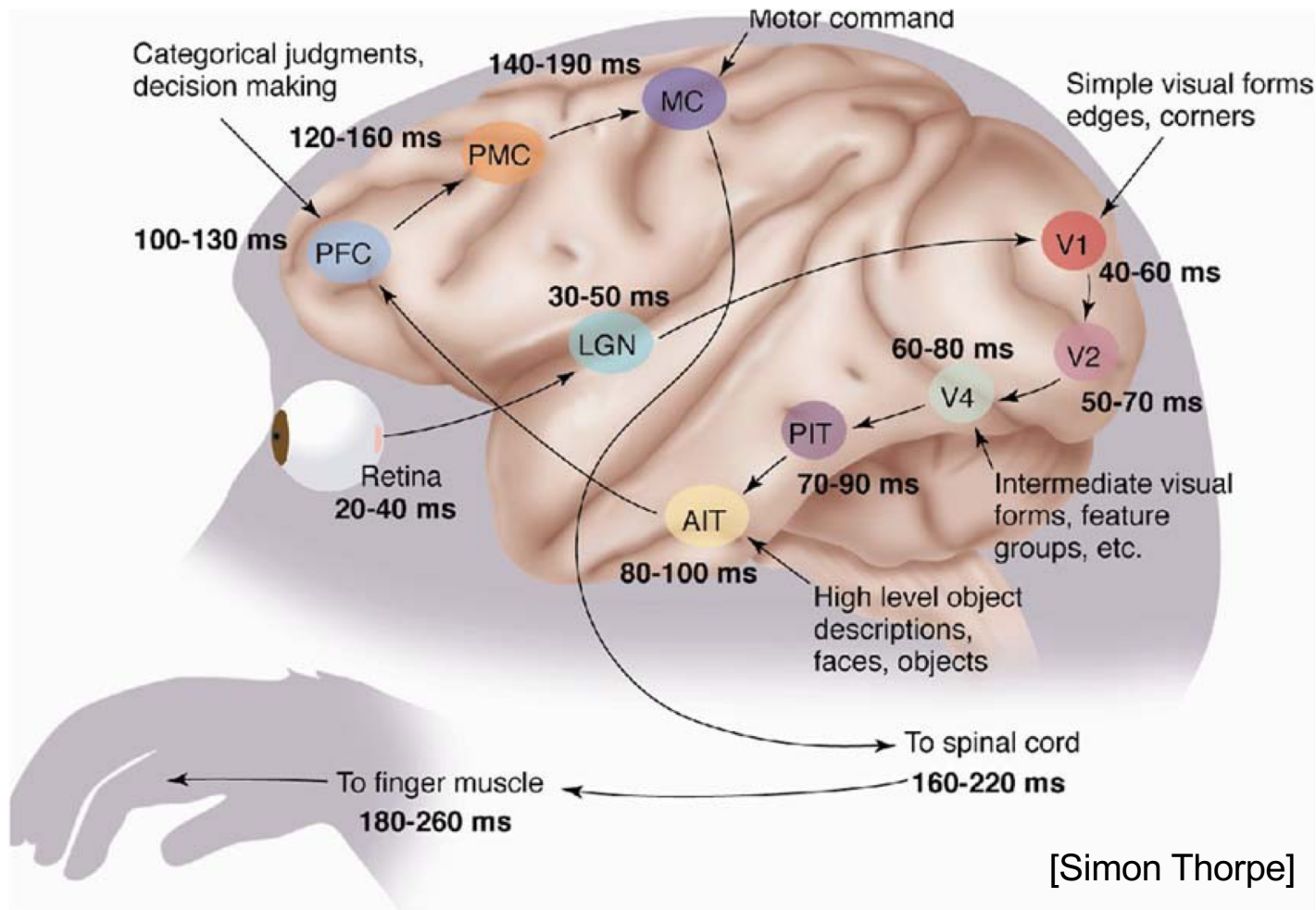
[Khan Academy]

$$\text{sigmoid}(\Sigma + b)$$



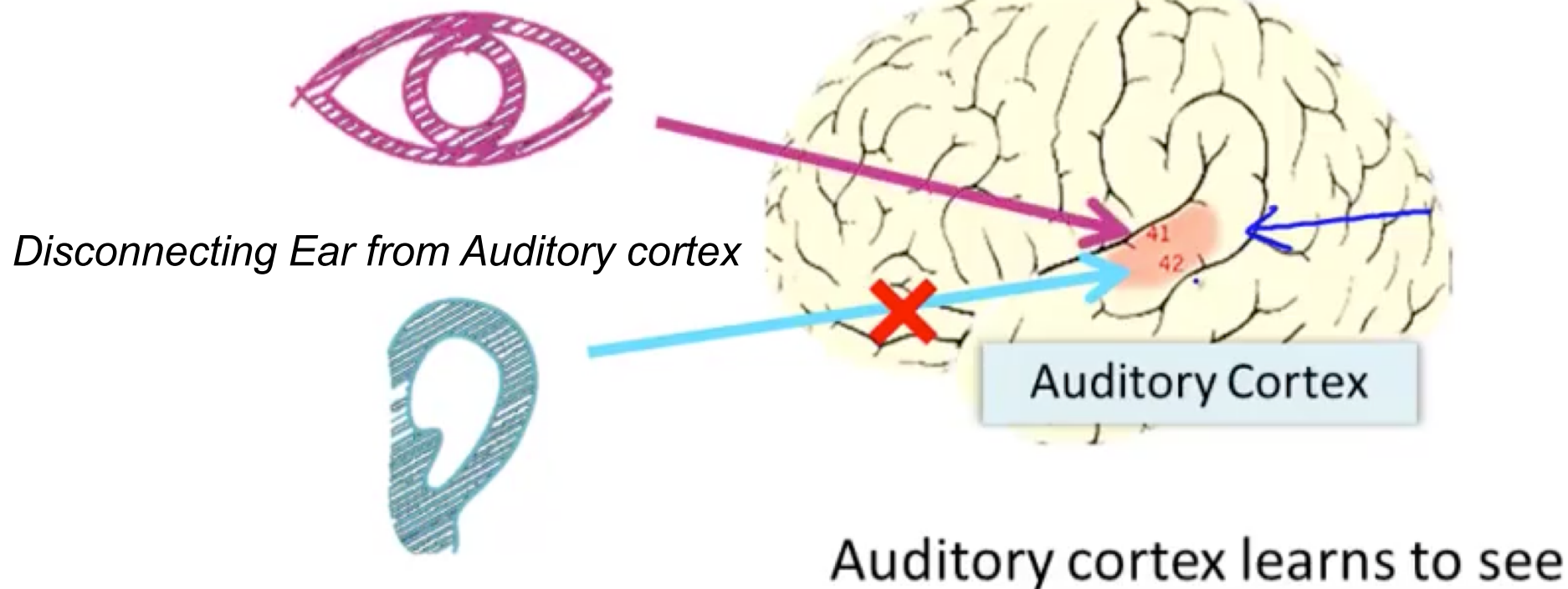
Analogy with Mammalian Visual Cortex

- Hierarchical
- Multiple Successive Stages
- Intermediate Representations (Features)



"Universal Learning" Biological Neural Network Hypothesis

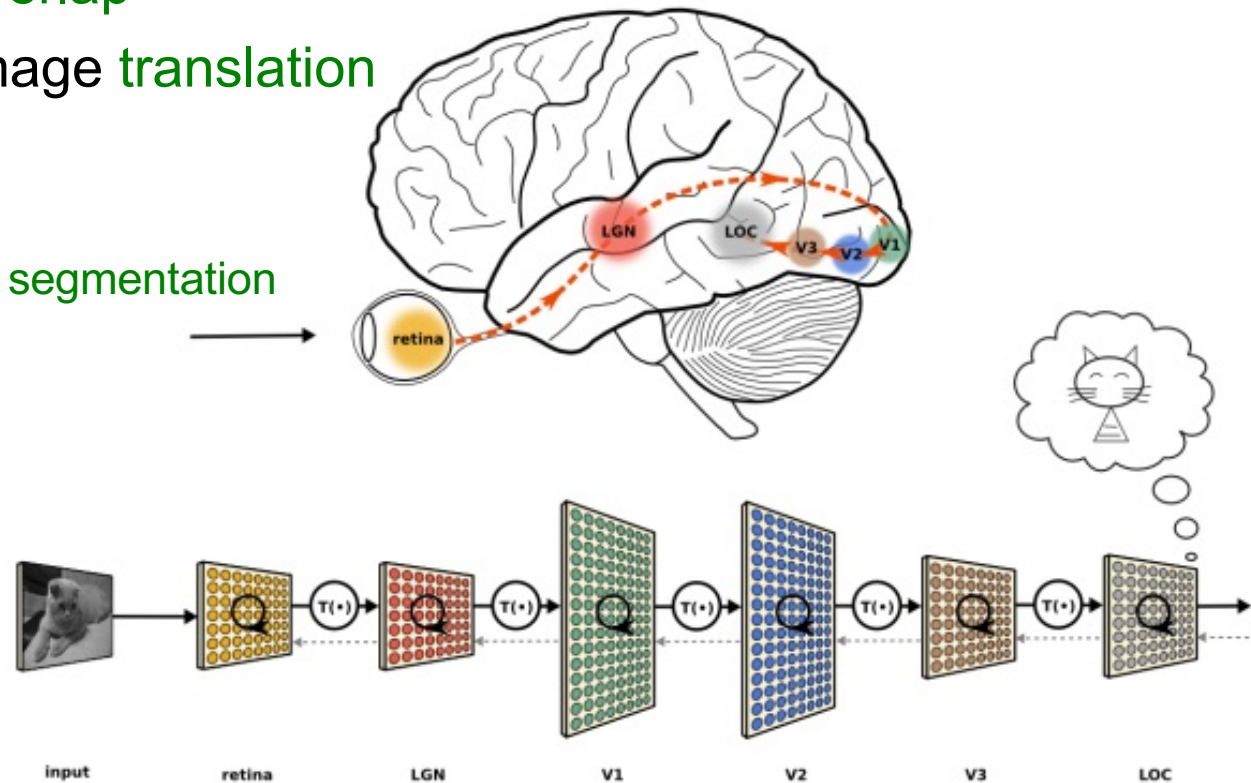
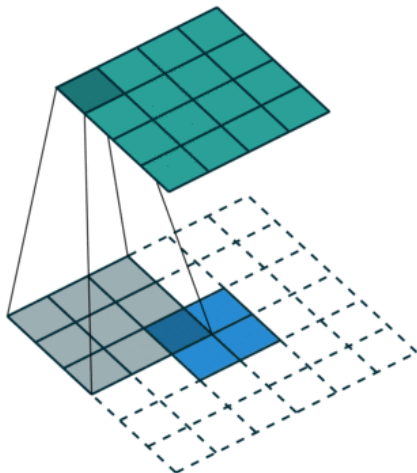
Reconnecting Eye to Auditory cortex



[Ng 2011]

Deep Learning **Bio-inspired** or/and **Regression-based** ?

- Convolutional Neural Networks (ConvNets [LeCun et al. 1989]) are bio-inspired
 - Inspired by animal visual cortex [Fukushima1980]
 - Complex neuron cells respond to stimuli in **receptive fields** (restricted region)
 - Receptive fields partially **overlap**
 - Allows tolerance to input image **translation**
 - » Weight sharing
 - » Translation invariance
 - » Subsumes (and thus avoids) segmentation
 - » For locally correlated data



Convolution

Convolution

- Slide a matrix (named **filter**, **kernel**, or **feature detector**) through the entire image
- For each mapping position:
 - compute the dot product of the filter with each mapped portion of the image
 - then add all elements of the resulting matrix
- Resulting in a new matrix (named **convolved feature**, or **feature map**)

0	1	1	1
1	0	1	0
0	1	0	0
1	1	1	1

Image

1	0	1
0	1	0
1	0	1

Filter

0	1 _{x1}	1 _{x0}	1 _{x1}
1	0 _{x0}	1 _{x1}	0 _{x0}
0	1 _{x1}	0 _{x0}	0 _{x1}
1	1	1	1

Convolution

1	4
5	2

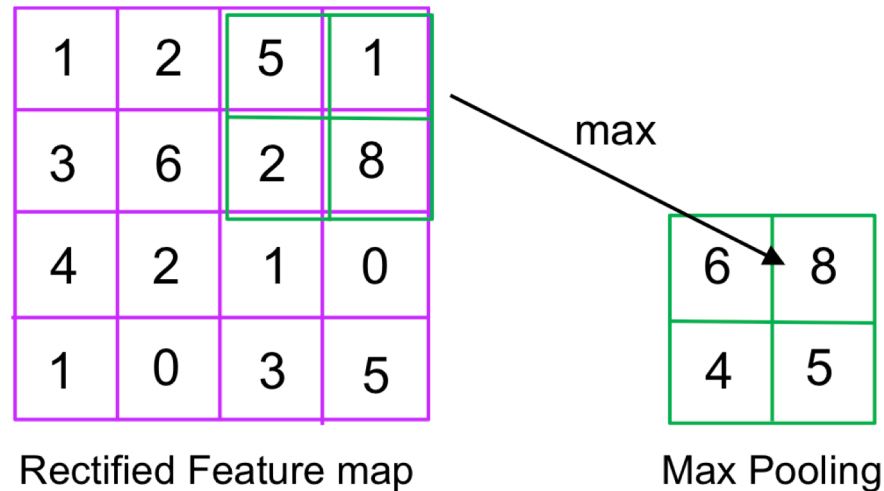
Convolved feature
(Feature map)

- The size of the feature map is controlled by three hyperparameters:
 - **Depth**: the number of filters used
 - **Stride**: the number of pixels by which we slide the filter matrix over the input matrix
 - **Zero-padding**: if the input matrix is padded with zeros around its border

Pooling

- Reduce the dimensionality of each feature map, while retaining significant information
- Standard operations:

- max
- average
- sum

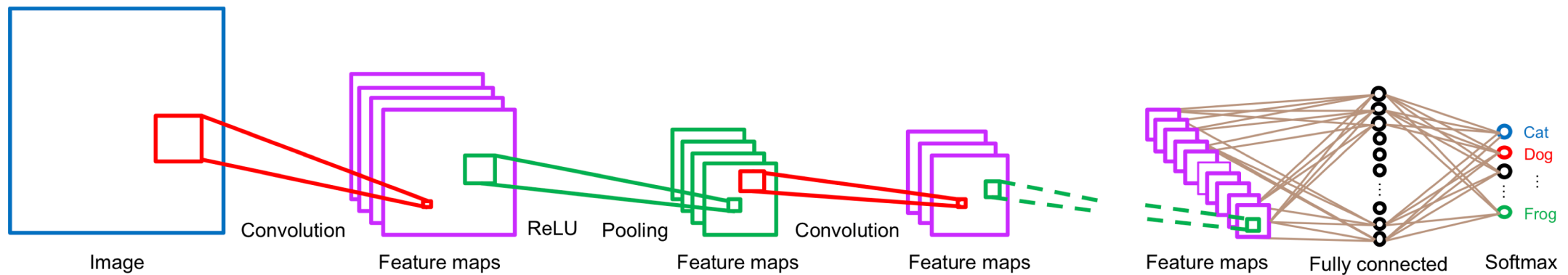


- Parameter sharing used by convolution (shared fixed filter) brings **equivariance** to translation: a motif in an image can be detected independently of its location
- Pooling brings **invariance** to small transformations, distortions and translations in the input image



Convolution

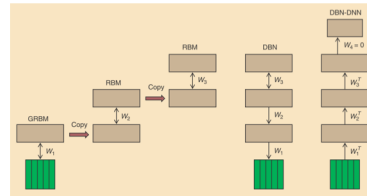
- Full Convolutional architecture:
 - Successive Layers, each Layer with 3 Stages:
 - » Convolution stage
 - » Non linear rectification (ReLU) stage
 - » Pooling stage



Techniques

Techniques

- Pre-Training



- Efficient Hardware (**GPUs**)



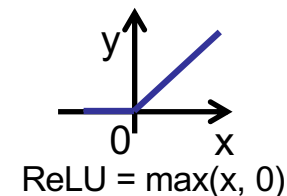
- Trading Space for Time (and Breadth for Depth) [Bengio & LeCun 2007]

- More Layers, Cascade/Sequence

- Additional Techniques and Heuristics:

- **Convolution** Neural Networks, Clustering, Pooling
- **Recurrent Networks**... (See next slides)
- **Memory** (LSTM, etc. See next slides)
- **Dropout** (as a Regularization technique)
- **ReLU** rather than Sigmoid in Hidden Layers
- **SoftMax** final Layer
- **Restricted Boltzman Machines**

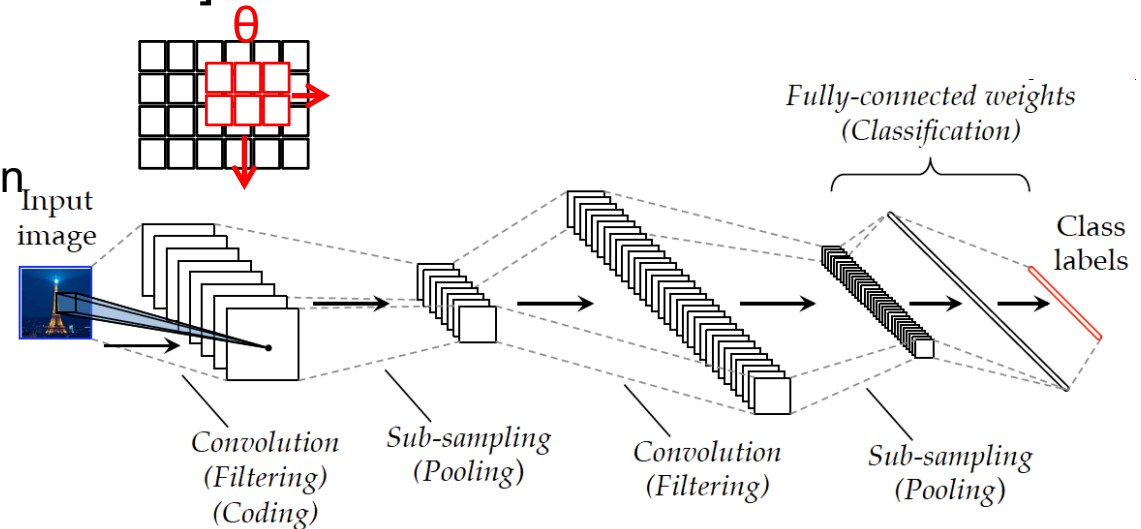
- » Contrastive Divergence Gradient estimation algorithm



Other Variations/Techniques

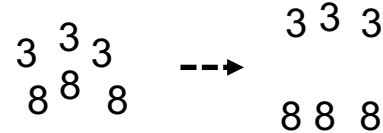
- **Convolutional** Neural Networks [LeCun 2010]

- Weight sharing
- Translation invariance
- Subsumes (and thus avoids) segmentation
- For locally correlated data



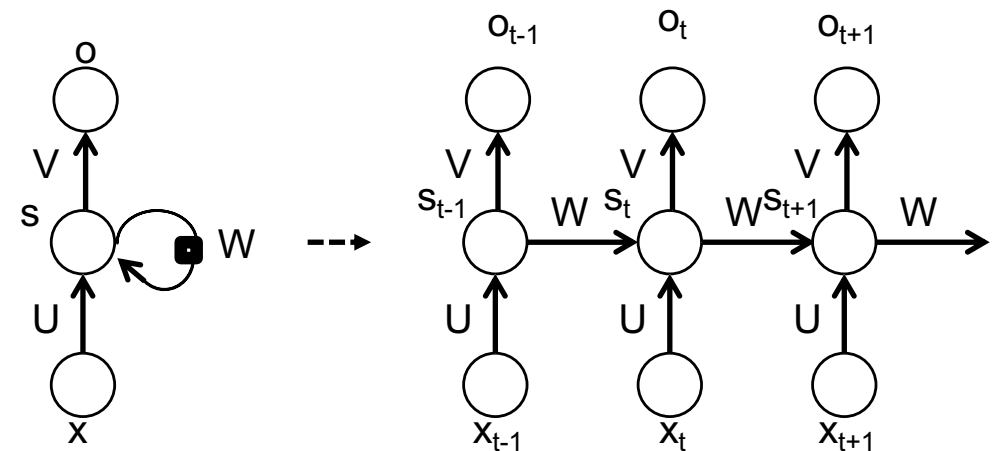
- **Pooling** (Compress/Factorize)

- **Clustering** (Group)



- **Recurrent** Networks

- loop+delay/memory
- for temporal or sequences of data
- ex: Natural Language Processing



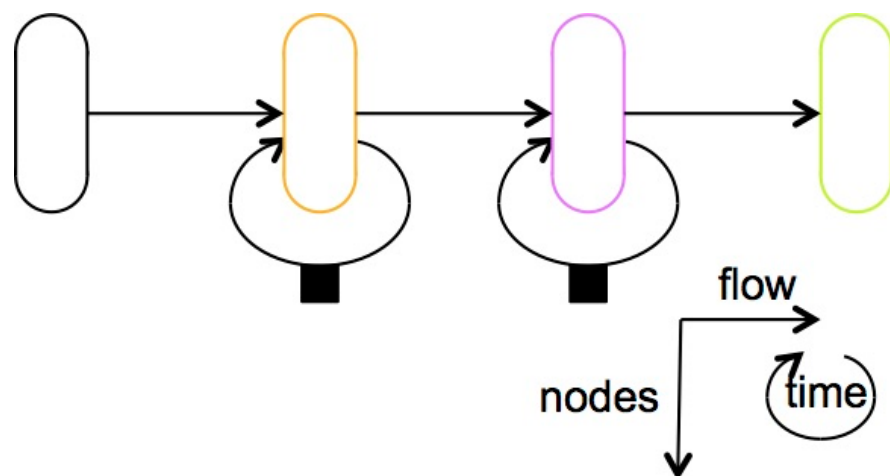
- **Memory-Augmented** Networks

- LSTM, GRU, Neural Turing Machines
- Memory access is **differentiable** thus **trainable**
- **Attention mechanism**

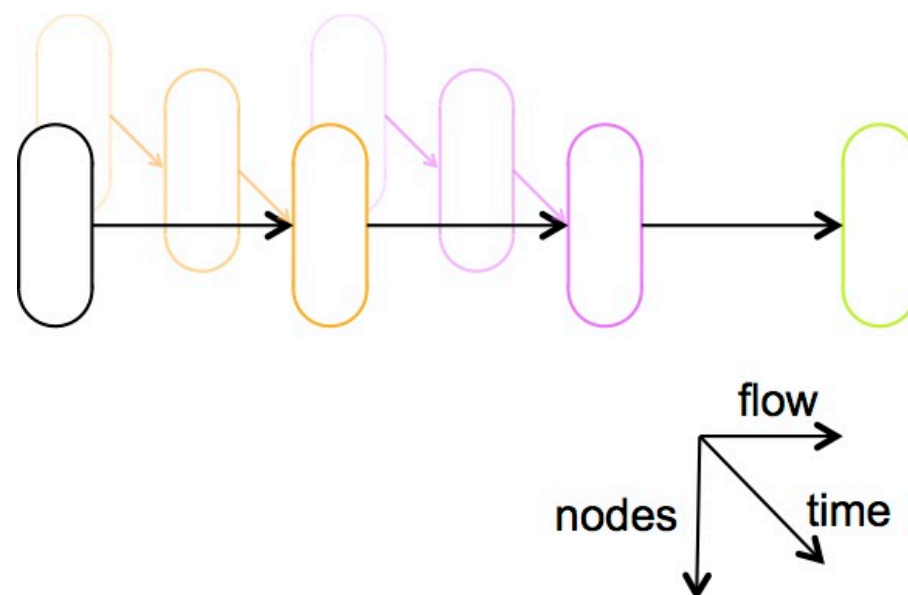
Recurrent Neural Networks (RNN)

- Recurrent connexion from a layer to itself
- Therefore, the layer learns also from its previous state
- It acts as a memory
- The RNN can learn sequences

Folded



Unfolded

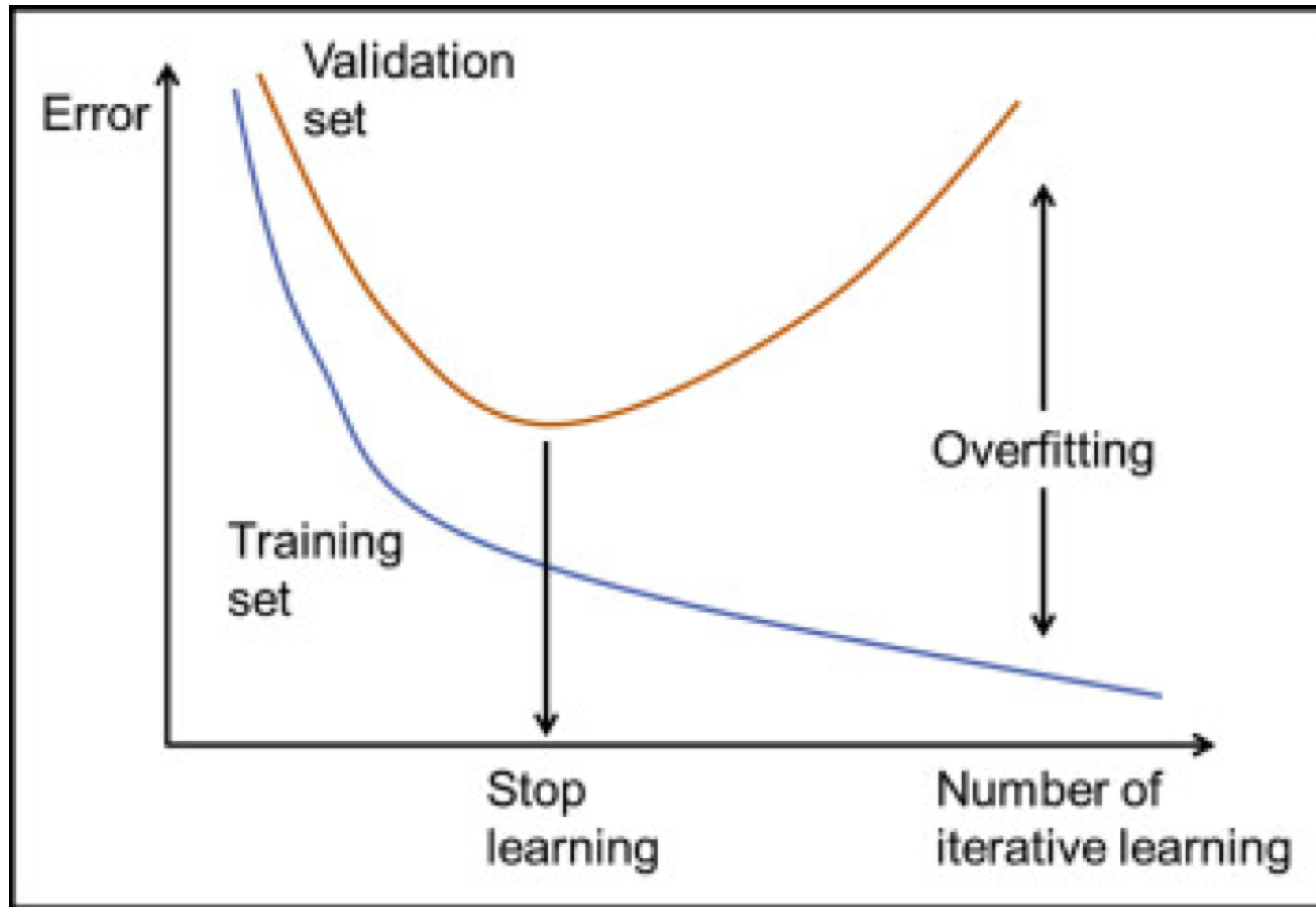


Engineering

Hyperparameters

- Fine Tuning of **Hyperparameters** of the Architecture/Model
- Architecture Structure
 - Number of layers
 - Number of units
 - Non linear Function
- Learning Heuristics
 - **Cost (Prediction Error)**: Quadratic, Log, Cross Entropy...
 - **Optimization**: Gradient Descent, Stochastic, Mini-Batch, Adagrad, Adadelata...
 - Learning rate
 - **Regularization**: L1, L2, Dropout...
- Convolution
 - Depth, Stride, Zero-padding
- LSTM
 - Input gate, Output gate, Forget gate biases
- Autoencoder
 - Sparsity
- Search (Random, Grid, Model-based) for exploring good settings

Learning Curves



(Deep) Networks Platforms/Libraries

2 Levels:

- Modules: C, Matlab, Packages (Linear algebra...)
- Glue/Scripting/ADL: Architectural description for assembling modules (data flow graph) and system/application management

Examples:



- SN, Lush [LeCun & Bottou 1987]
 - Lisp-like (single language)



- Torch (Facebook, Google...)
 - LuaJIT, Python (PyTorch)



- TensorFlow (Google)
 - Python

theano

- Theano (U. Montréal)
 - Python



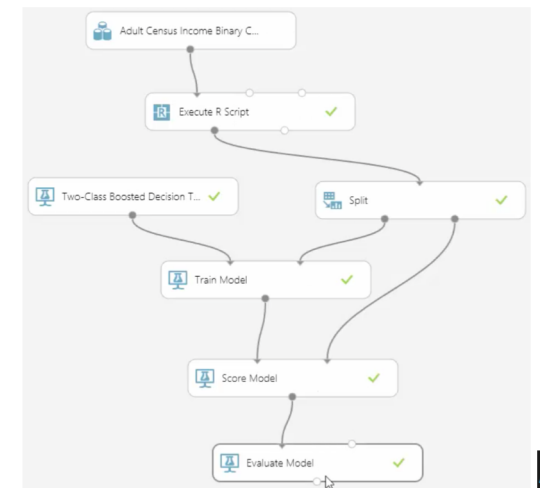
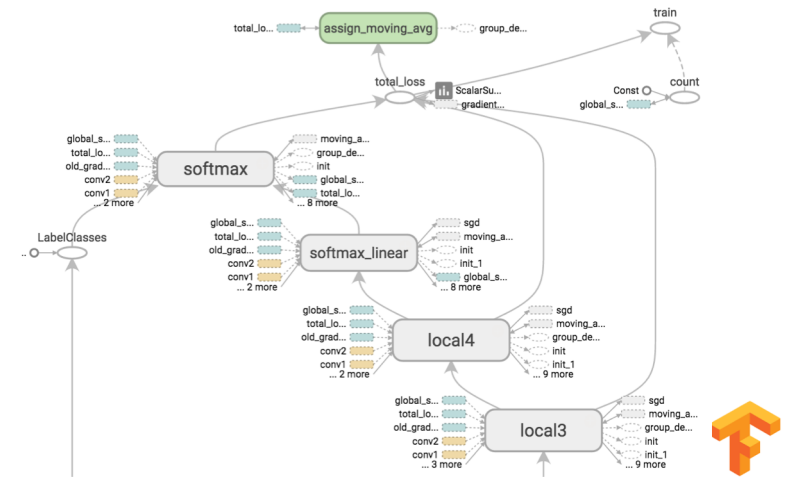
- Keras
 - Superlayer of (both) TensorFlow and Theano
 - Python



- Azure (Microsoft)

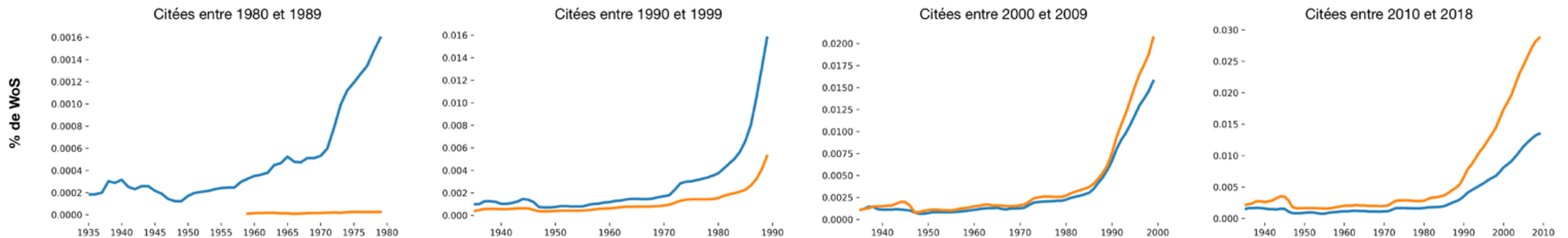
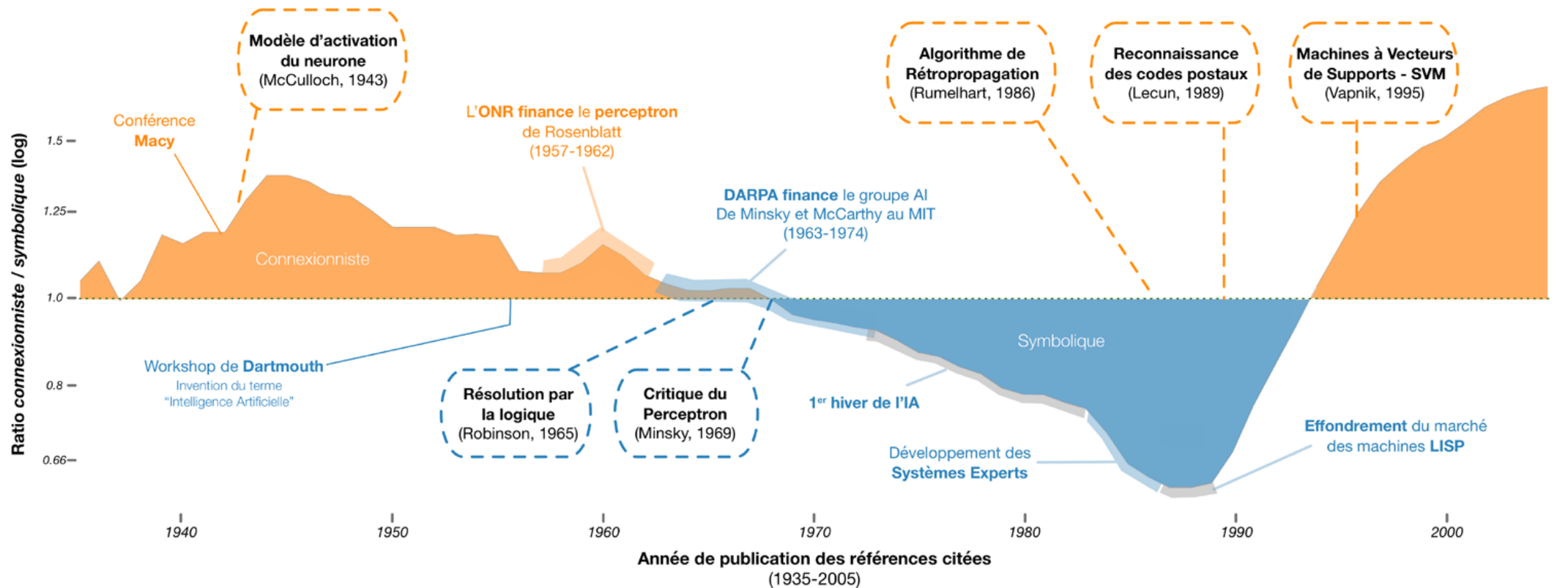
- Cloud
- Drag and Drop

Deep Learning – Music Generation – 2019

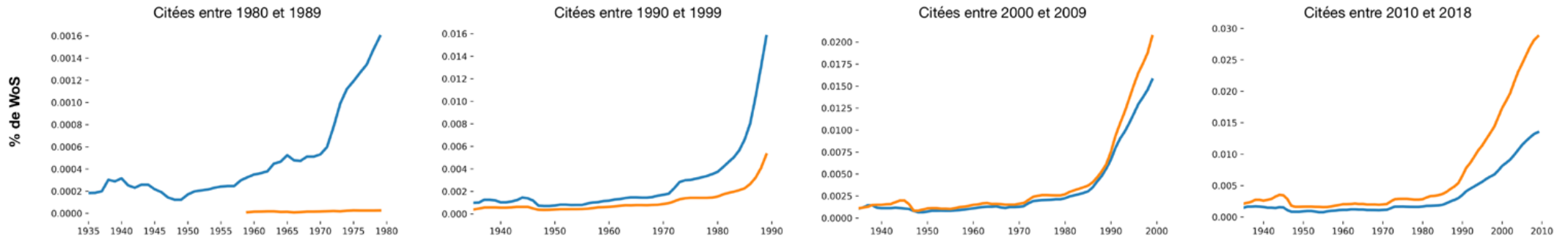
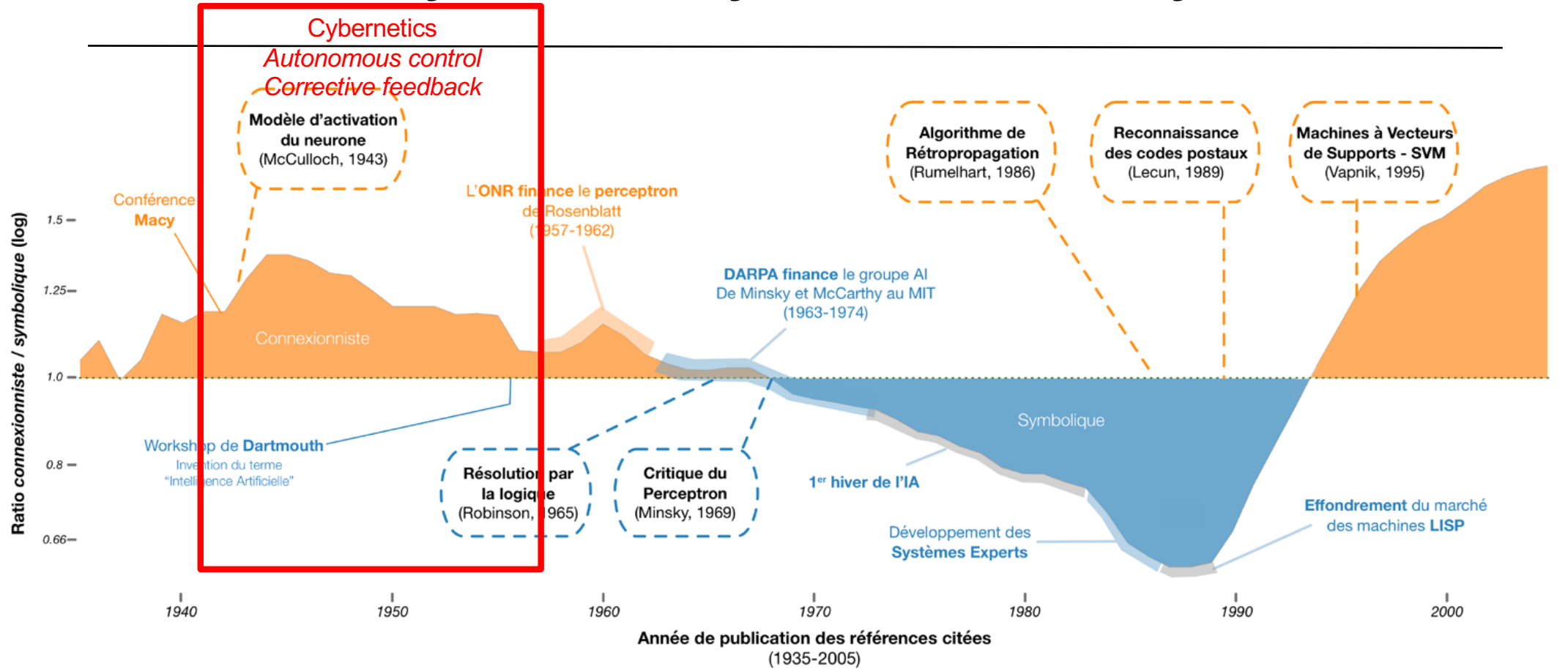


History / Various Facets of AI

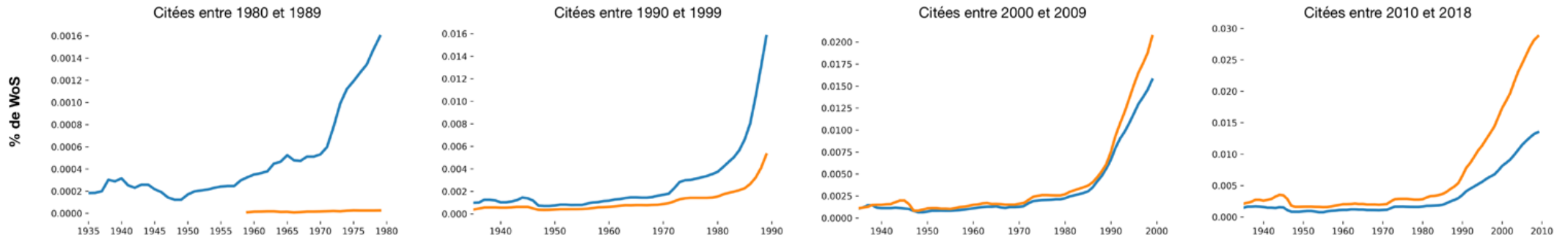
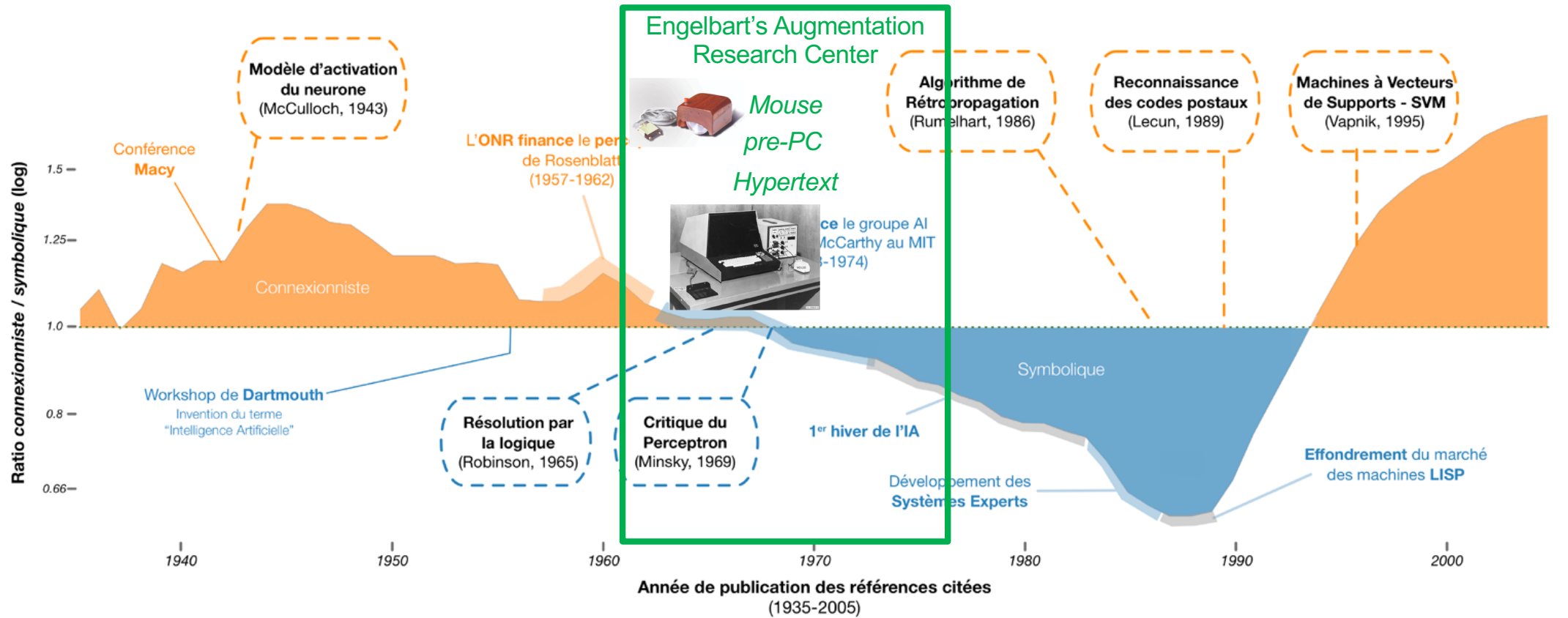
Symbolic vs Connexionist AI – History



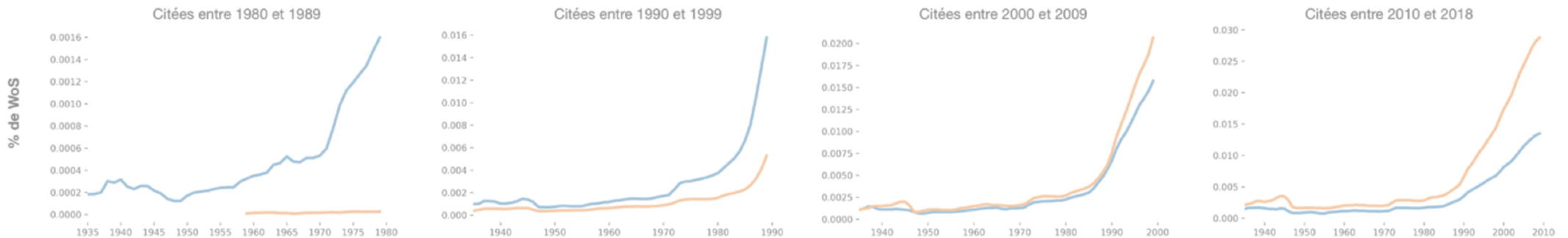
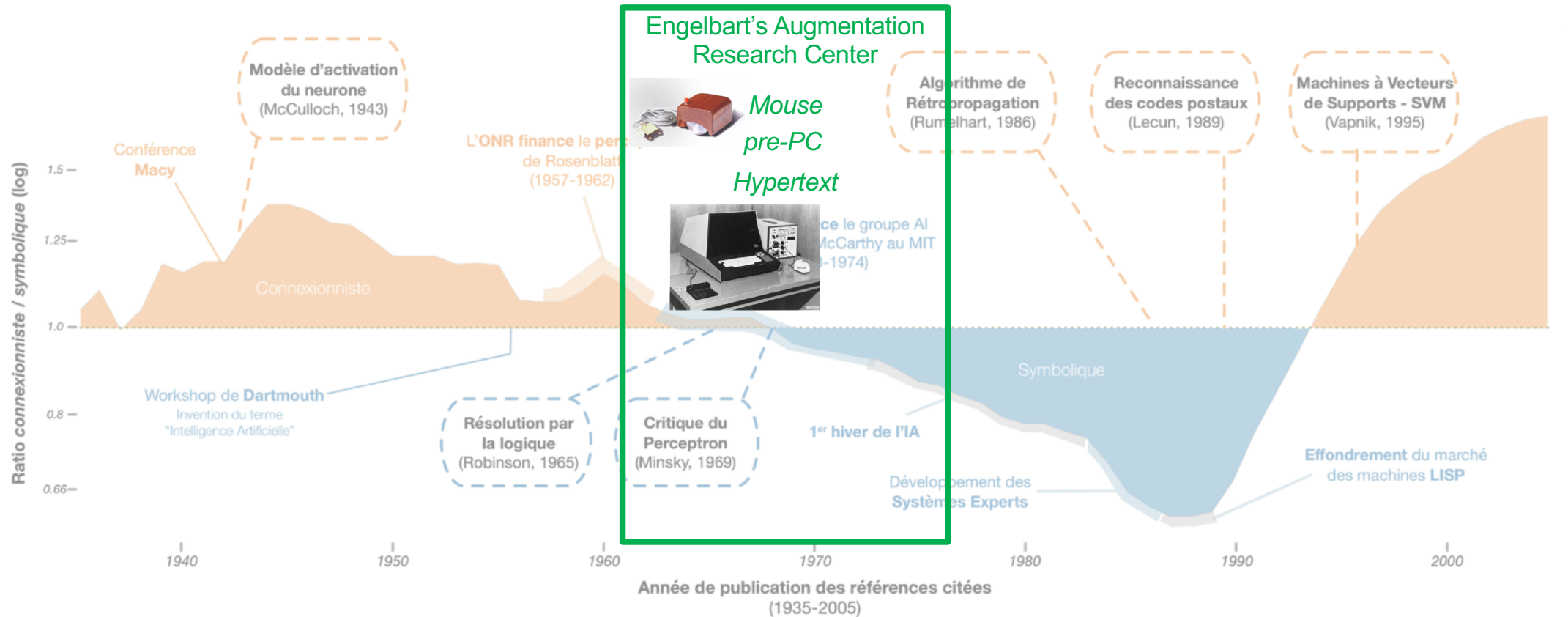
Symbolic vs Cybernetics – History



AI vs IA – History

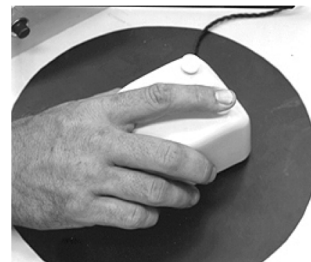


AI (Artificial Intelligence) vs IA (Intelligence Augmentation) – History



Engelbart's Augmentation Research Center

- Augmentation Research Center at Stanford Research Institute (SRI)
- Headed by Douglas Engelbart
- Vision of the Interactive Personal Computer
- Mouse
- Graphical Interfaces
- Hypertext
- Visioconference
- Internet



Douglas Engelbart Team Demonstration (9 December 1969)



monday afternoon

december 9

3:45 p.m. / arena

Chairman:

DR. D. C. ENGELBART

*Stanford Research Institute
Menlo Park, California*

a research center for augmenting human intellect

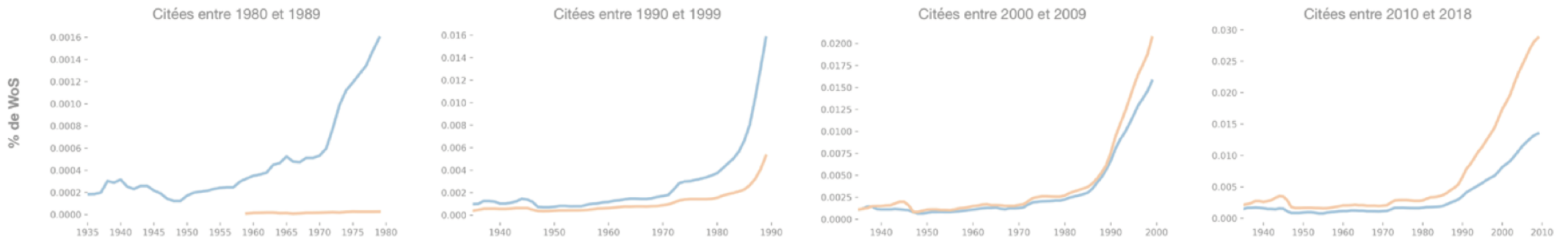
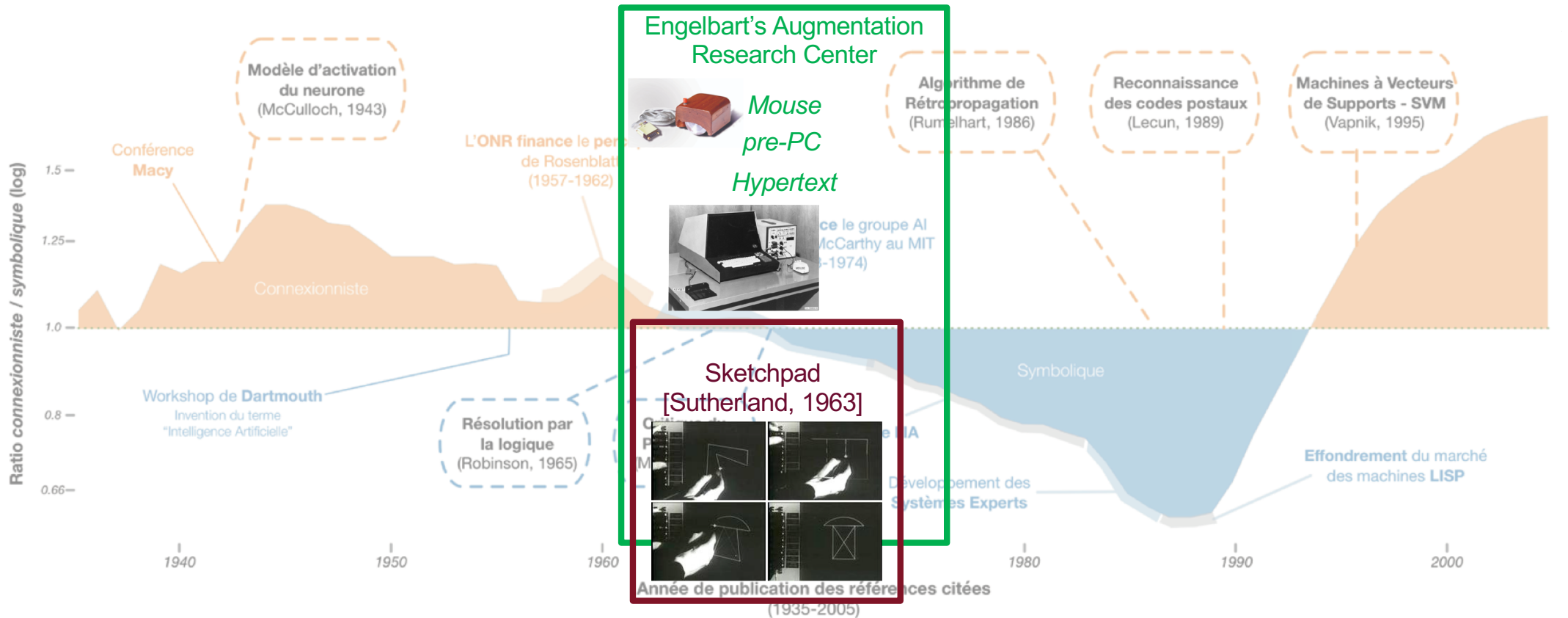
This session is entirely devoted to a presentation by Dr. Engelbart on a computer-based, interactive, multiconsole display system which is being developed at Stanford Research Institute under the sponsorship of ARPA, NASA and RADC. The system is being used as an experimental laboratory for investigating principles by which interactive computer aids can augment intellectual capability. The techniques which are being described will, themselves, be used to augment the presentation.

The session will use an on-line, closed circuit television hook-up to the SRI computing system in Menlo Park.

Following the presentation remote terminals to the system, in operation, may be viewed during the remainder of the conference in a special room set aside for that purpose.

<https://web.stanford.edu/dept/SUL/library/extra4/sloan/MouseSite/1968Demo.html>

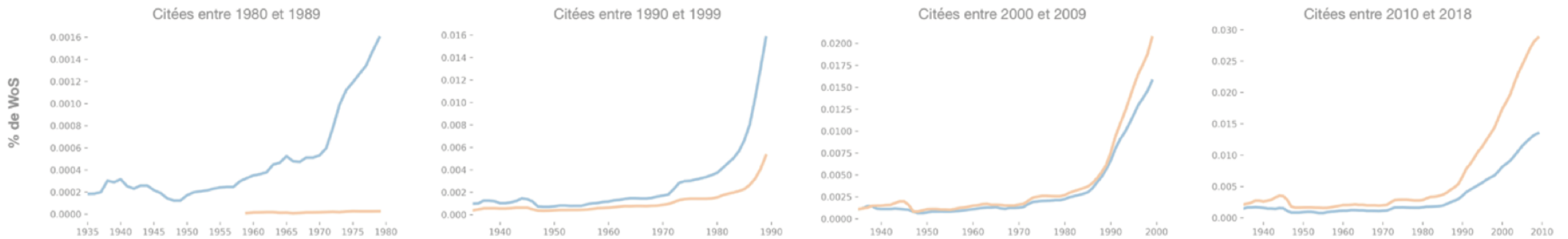
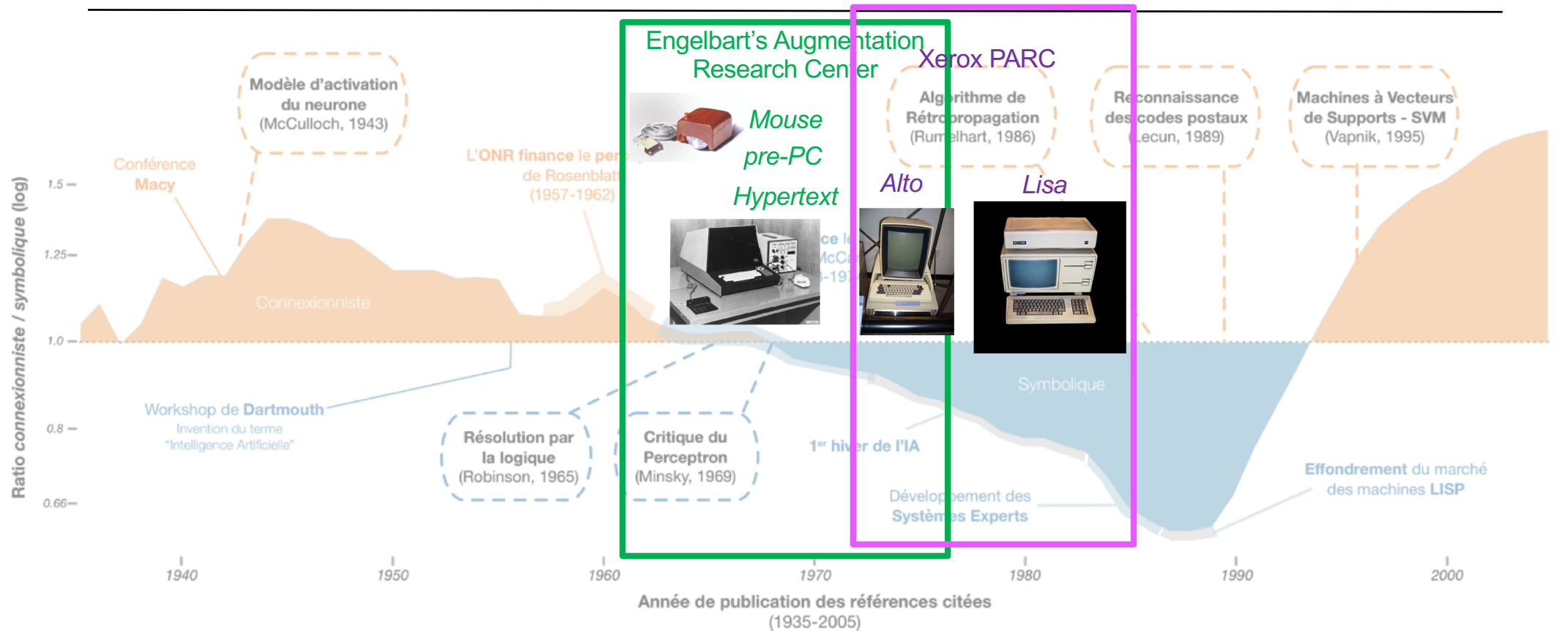
AI vs IA – History



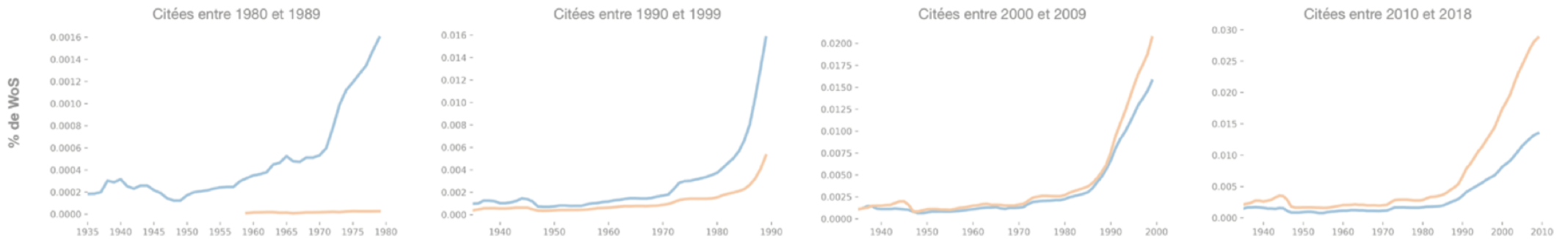
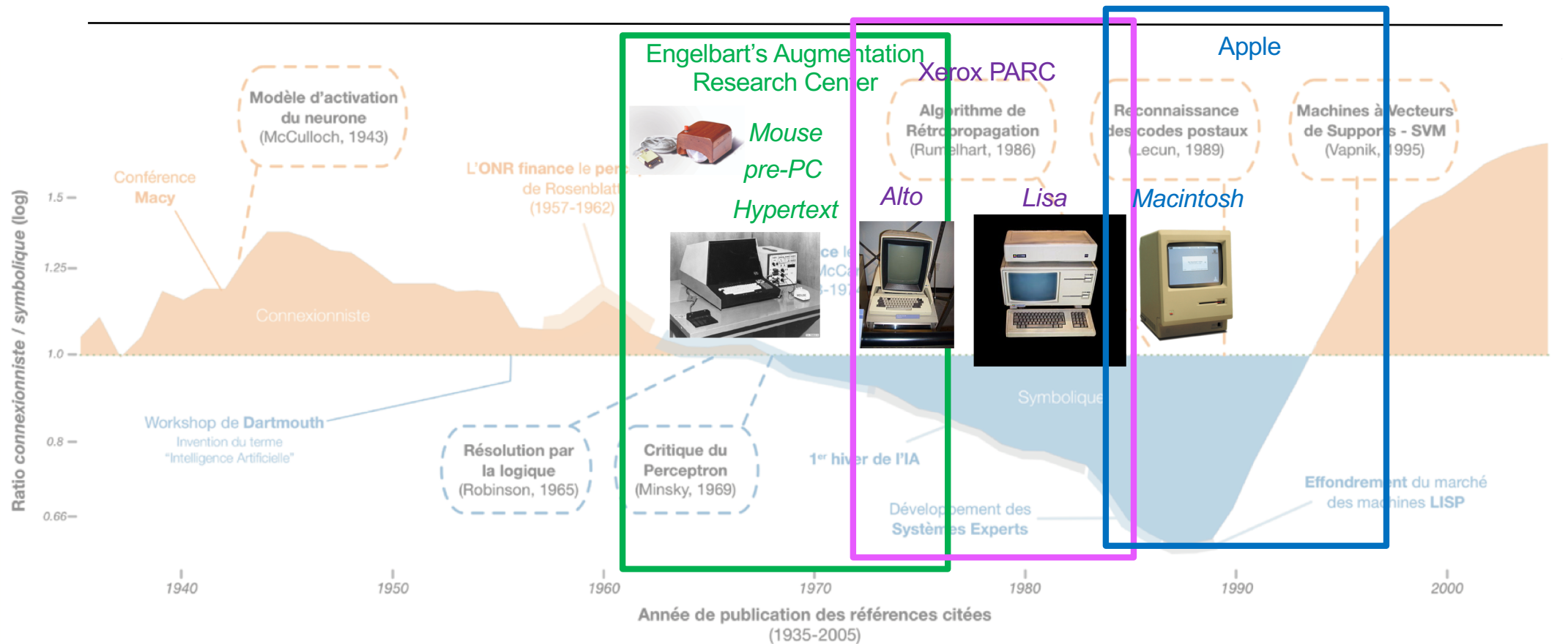
Sketchpad [Sutherland, 1963]



AI vs IA – History



AI vs IA – History



AI and Connectionism with Progressive Incorporation of IA (Intelligence Augmentation & Interaction)

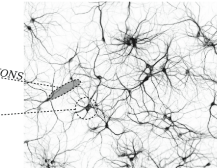
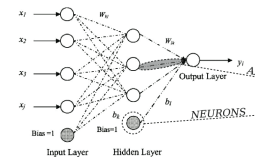
Symbolic AI [1956-]

Knowledge Representation
Reasoning
Planning



Connectionism [1957-]

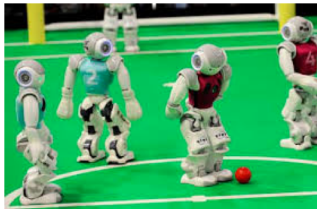
NEURAL NETWORK MAPPING



Data Induction
Machine Learning
Pattern Recognition

Distributed AI [1975-]

Multi-Agent Systems
Coordination
Theory of Mind



Active Learning [2009-]

Interactive User Query

Personal Artificial Assistant [1990-]

Software Agent

Learning over User Shoulder
Chatbot



Autonomous vs Assistance Music Making

- Autonomous Generation/Interpretation

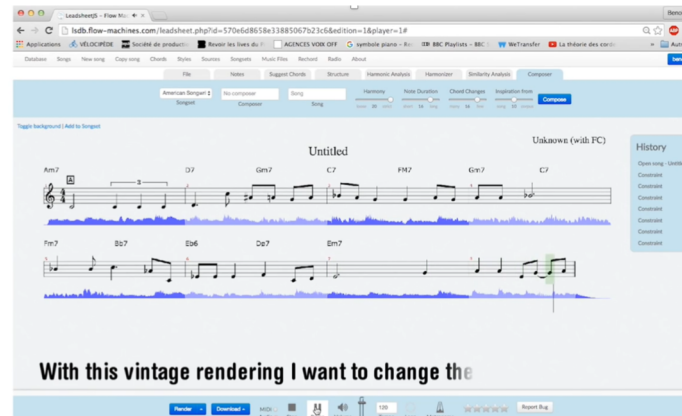
- Turing Test
- Ambient Music



Compressorheads (no AI inside 😊)

- Assistance to Human Composers and Musicians

- Propose
- Refine
- Analyze
- Harmonize
- Produce
- ...



Various Open Issues

- Interpretability (Black Box)
- Reuse (Transfer Learning)
- Theory
- ...

Pentagon Study, 1980

- Automatic Classification Tank/no Tank Images



No Tank



Tank

Further Test



Clear Sky



Cloudy Sky

Missing Context



Missing Context

<https://www.scmp.com/tech/innovation/article/2174564/facial-recognition-catches-chinas-air-con-queen-dong-mingzhu>

Adversarial Neural Networks Generation



x

“panda”

57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_x J(\theta, x, y))$

“nematode”

8.2% confidence

$=$

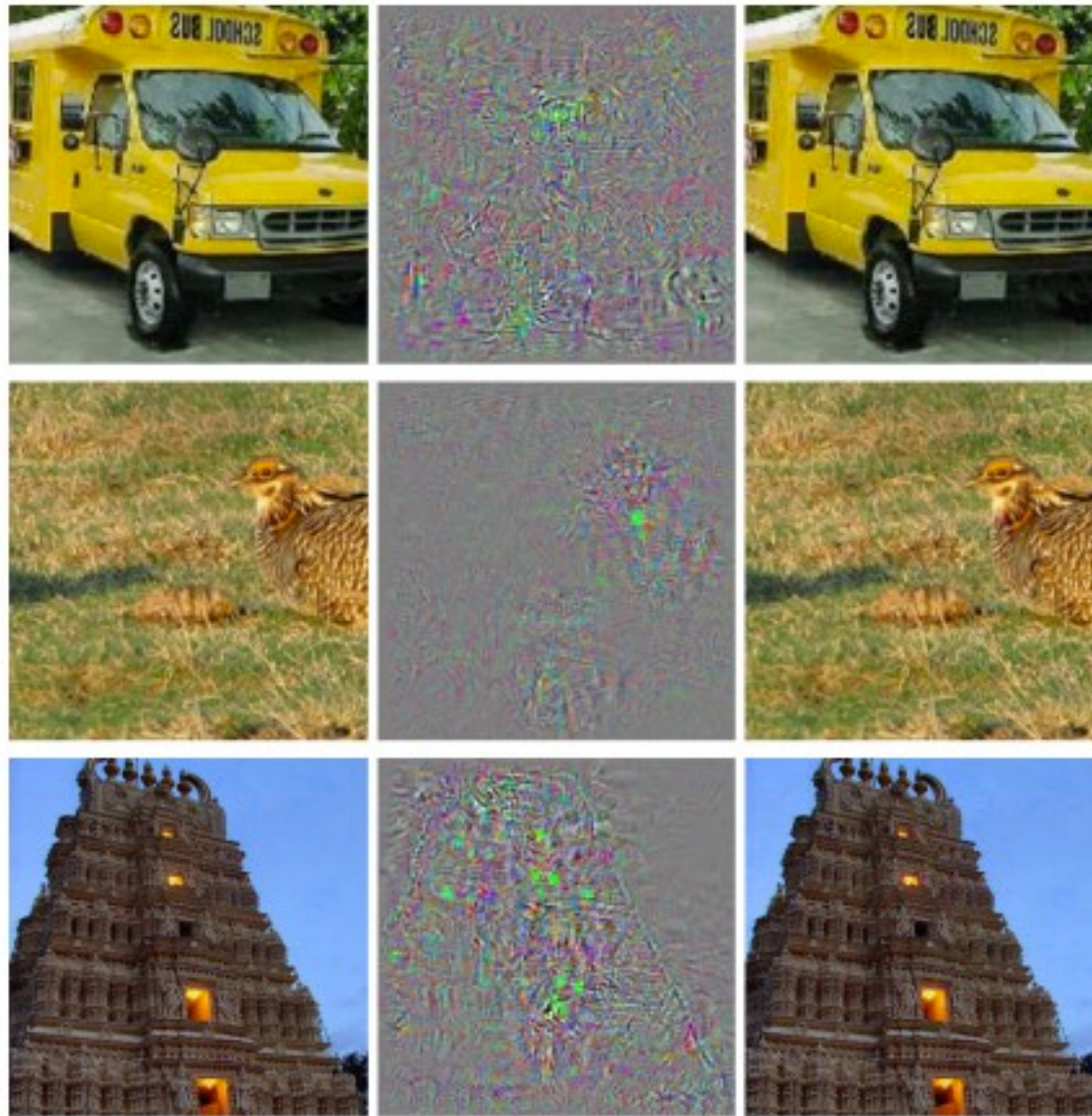


$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$
“gibbon”

99.3 % confidence

[Szegedy et al. 2014]

Adversarial Neural Networks Generation



Correct class Perturbation Incorrect class

[Szegedy et al. 2013]

Adversarial Neural Networks Glasses Generation



Ex. of Handcrafted Features – Bag of Features [Brendel, 2019]

- Bag of Features (BoF) Model

- Define key Visual Features

- Ex: Human eye, Feather



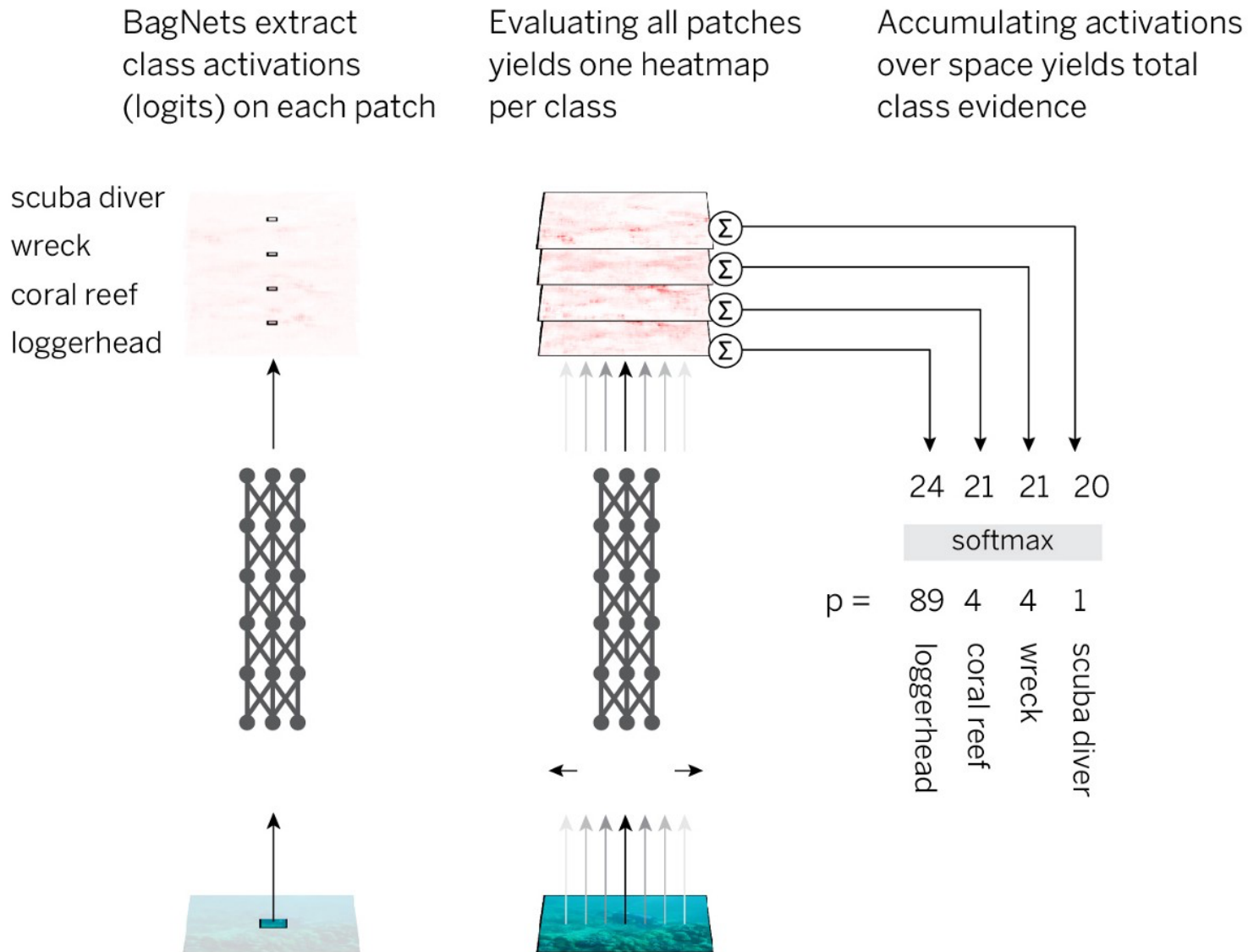
- Count how often each Feature is Present in the Image



- Classify based on these Numbers

- Human: Eye = 2
 - Bird: Feather = 18

Bag of Feature Network (BagNet) [Brendel, 2019]



Bag of Feature Network (BagNet) [Brendel, 2019]

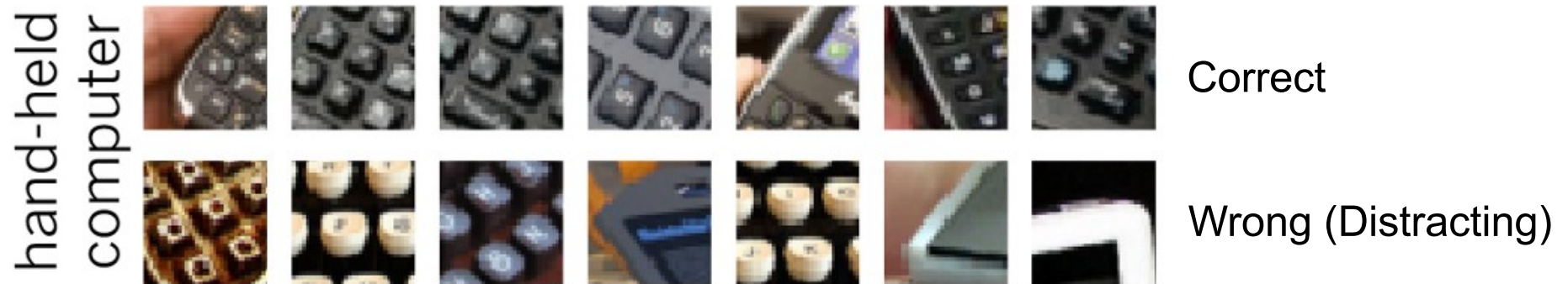
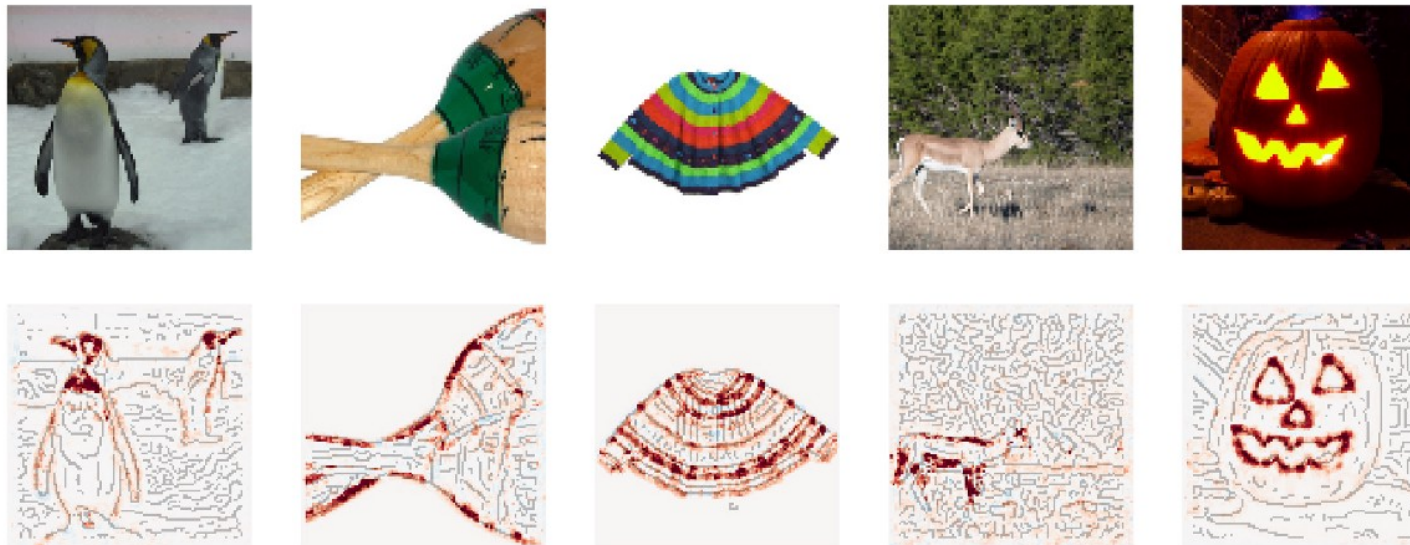


Image Features with most evidence



Feature Visualization and Activation Atlases [Olah et al., 2017]

- Based on DeepDream [Mordvintsev et al., 2015]
 - Iteratively adjust an image to maximize activation



- Visualizing Activations
 - Which pattern triggers activation



neuron



layer

- Helps to Analyze/Discriminate between Patterns and Correlations (e.g., stripes)



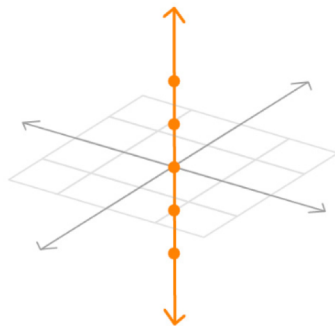
- From One Neuron to Neuron Interactions
& From One Example to Many Images (Next Slide)



Baseball—or stripes?
mixed4a, Unit 6

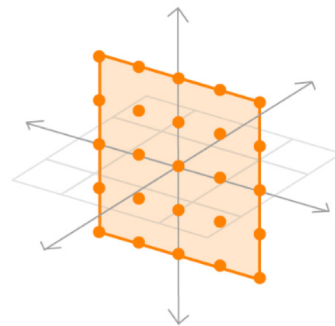
Activation Atlases [Carter et al., 2019]

INDIVIDUAL NEURONS



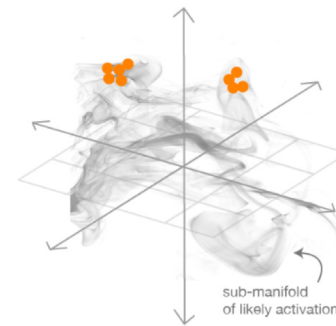
Visualizing individual neurons make hidden layers somewhat meaningful, but misses interactions between neurons — it only shows us one-dimensional, orthogonal probes of the high-dimensional activation space.

PAIRWISE INTERACTIONS



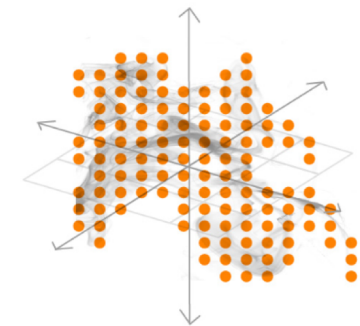
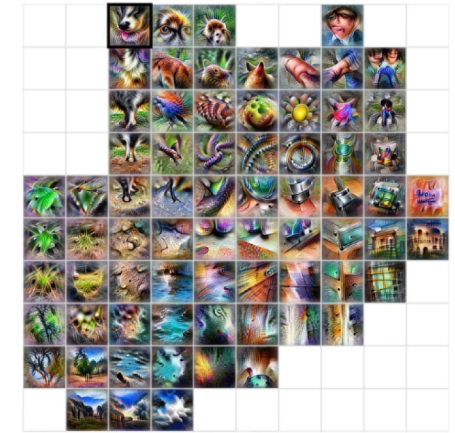
Pairwise interactions reveal interaction effects, but they only show two-dimensional slices of a space that has hundreds of dimensions, and many of the combinations are not realistic.

SPATIAL ACTIVATIONS



Spatial activations show us important combinations of many neurons by sampling the sub-manifold of likely activations, but they are limited to those that occur in the given example image.

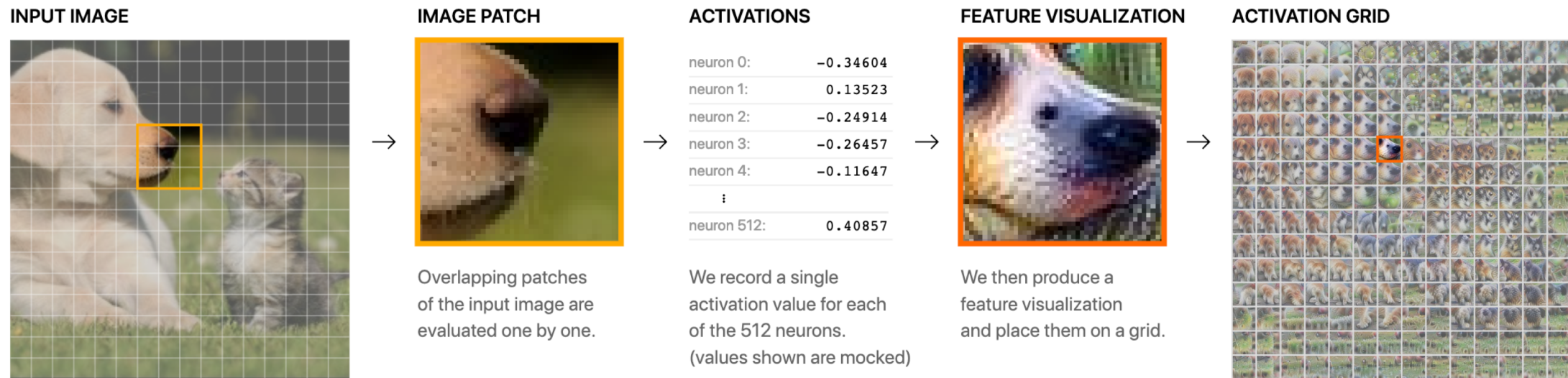
ACTIVATION ATLAS



Activation atlases give us a bigger picture overview by sampling more of the manifold of likely activations.

Activation Atlases [Carter et al., 2019]

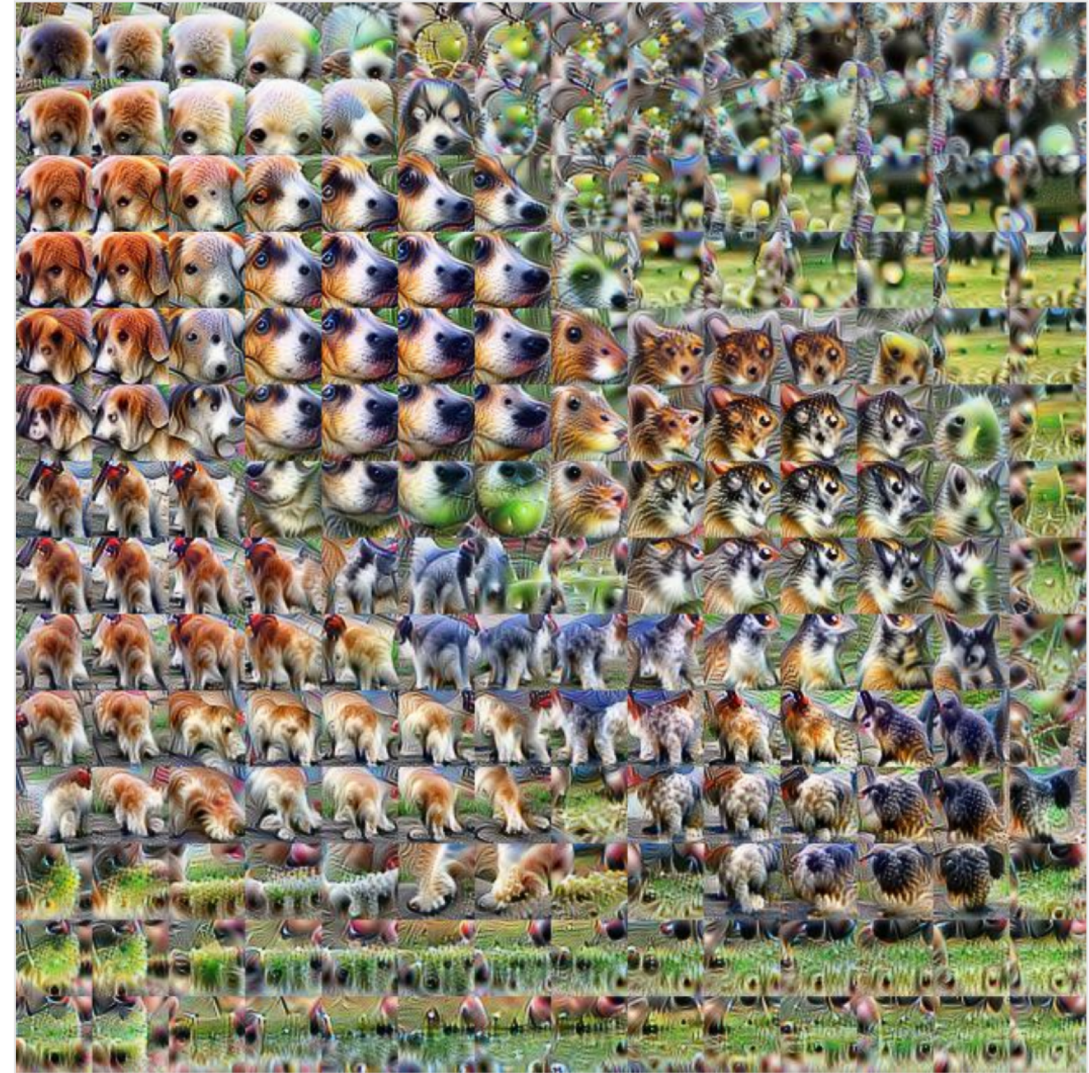
- More details (interactive) in
 - <https://distill.pub/2019/activation-atlas/>



Activation Atlases [Carter et al., 2019]

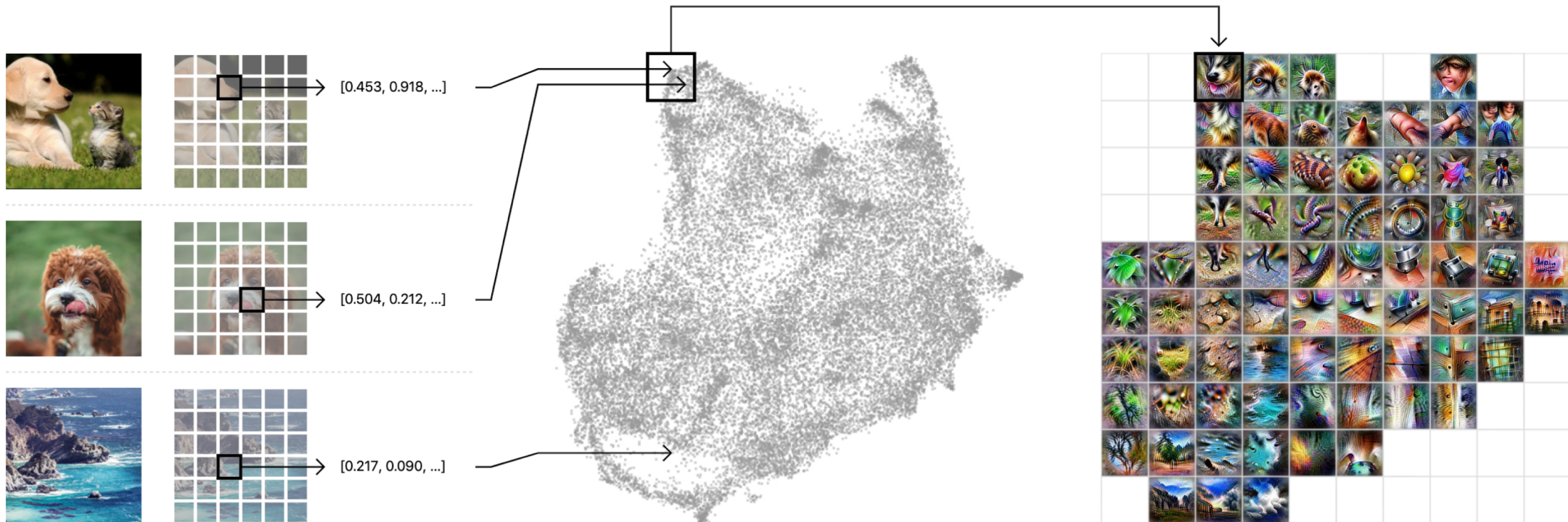


Input image from ImageNet.



Activation grid from InceptionV1, layer mixed4d.

Activation Atlases [Carter et al., 2019]



A randomized set of one million images is fed through the network, collecting one random spatial activation per image.

The activations are fed through UMAP to reduce them to two dimensions. They are then plotted, with similar activations placed near each other.

We then draw a grid and average the activations that fall within a cell and run feature inversion on the averaged activation. We also optionally size the grid cells according to the density of the number of activations that are averaged within.

Activation Atlases [Carter et al., 2019]

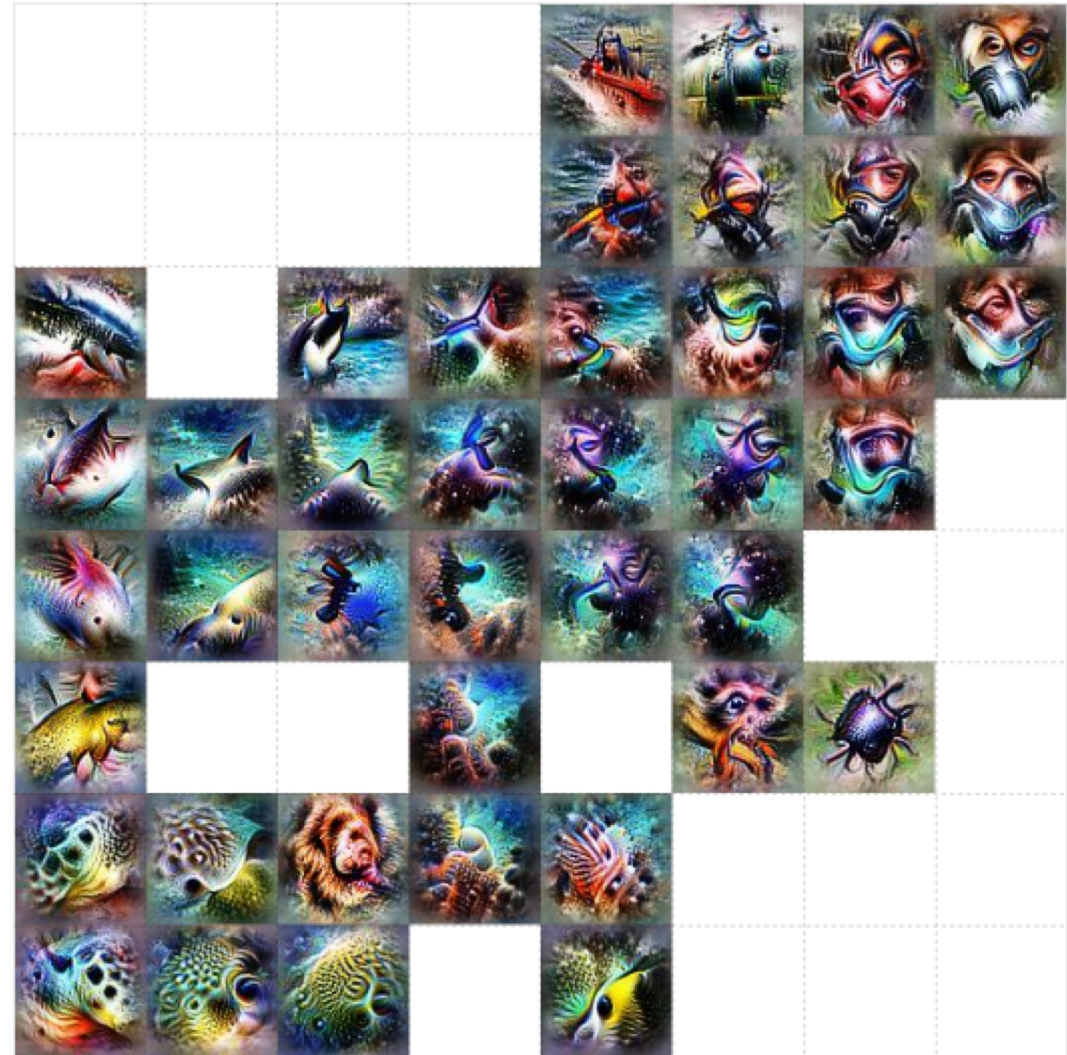


Activation Atlases [Carter et al., 2019]

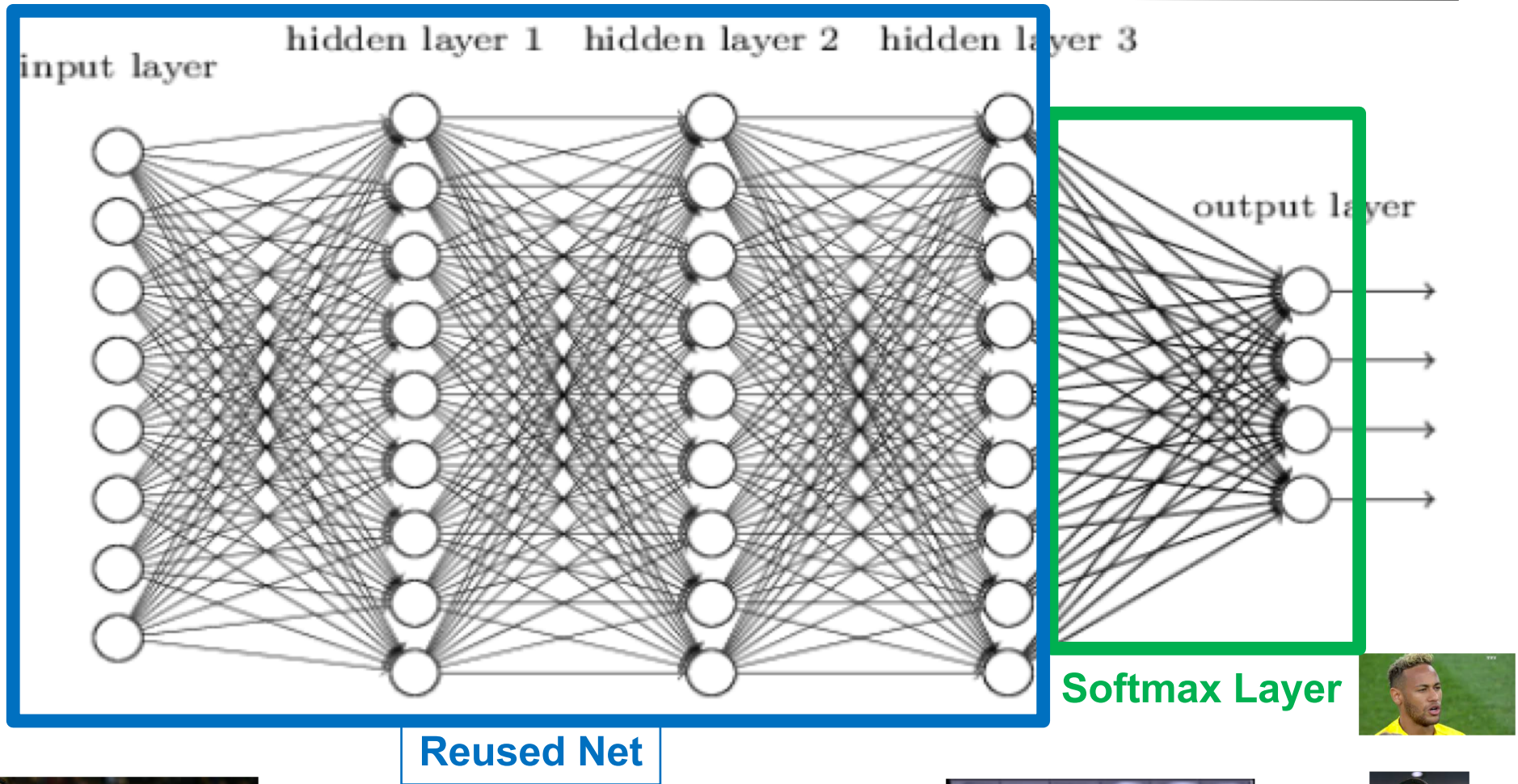
"SNORKEL"



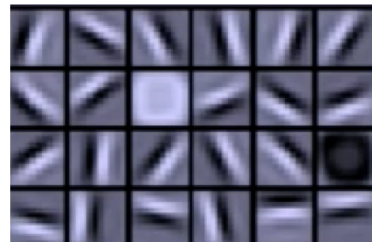
"SCUBA DIVER"



Transfer Learning



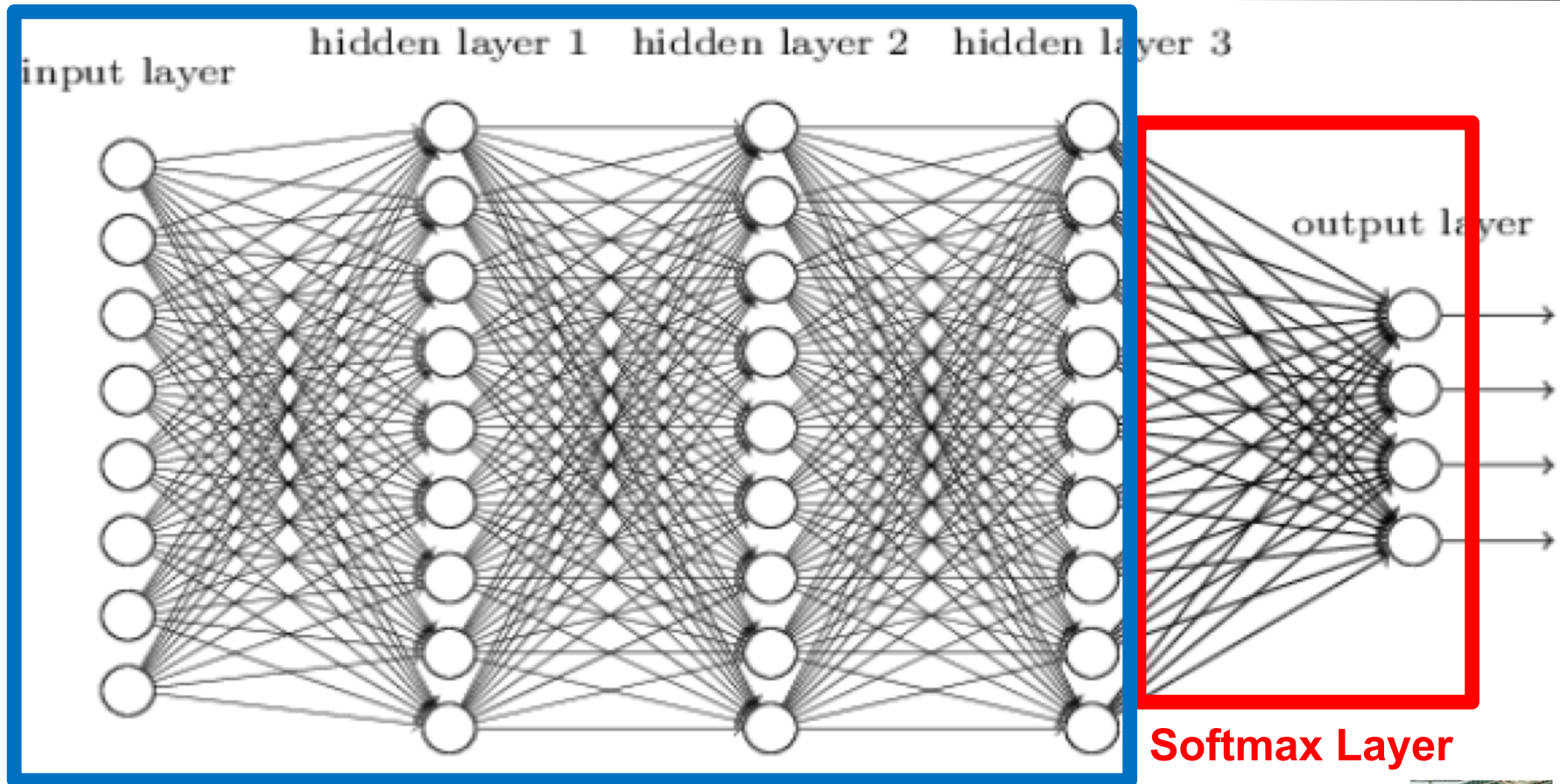
Jean-Pierre Briot



Deep Learning – Music Generation – 2019



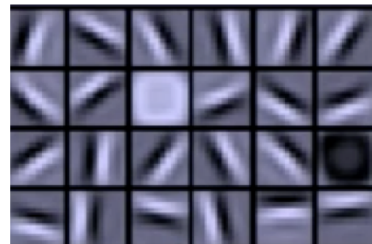
Transfer Learning



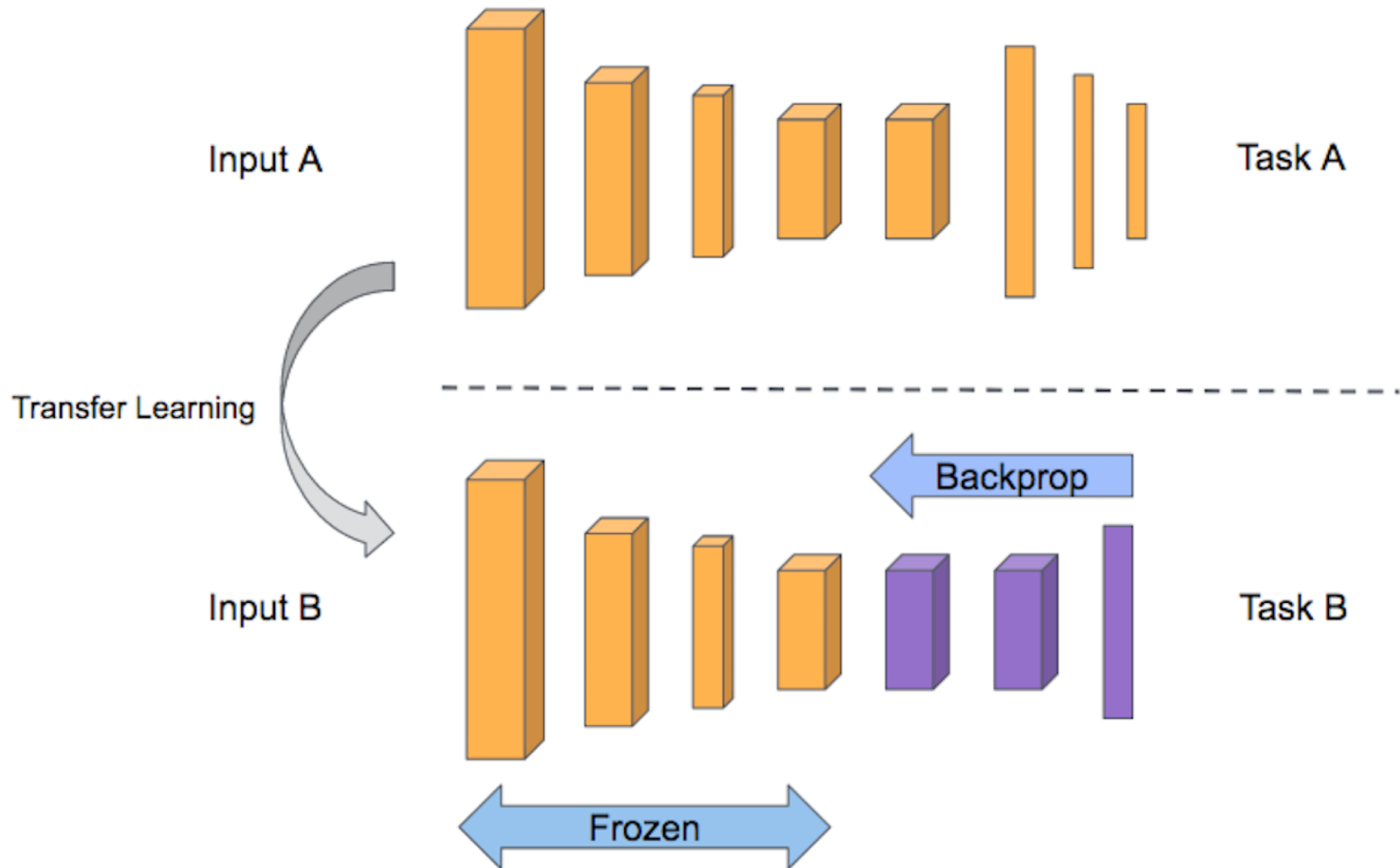
Reused Net



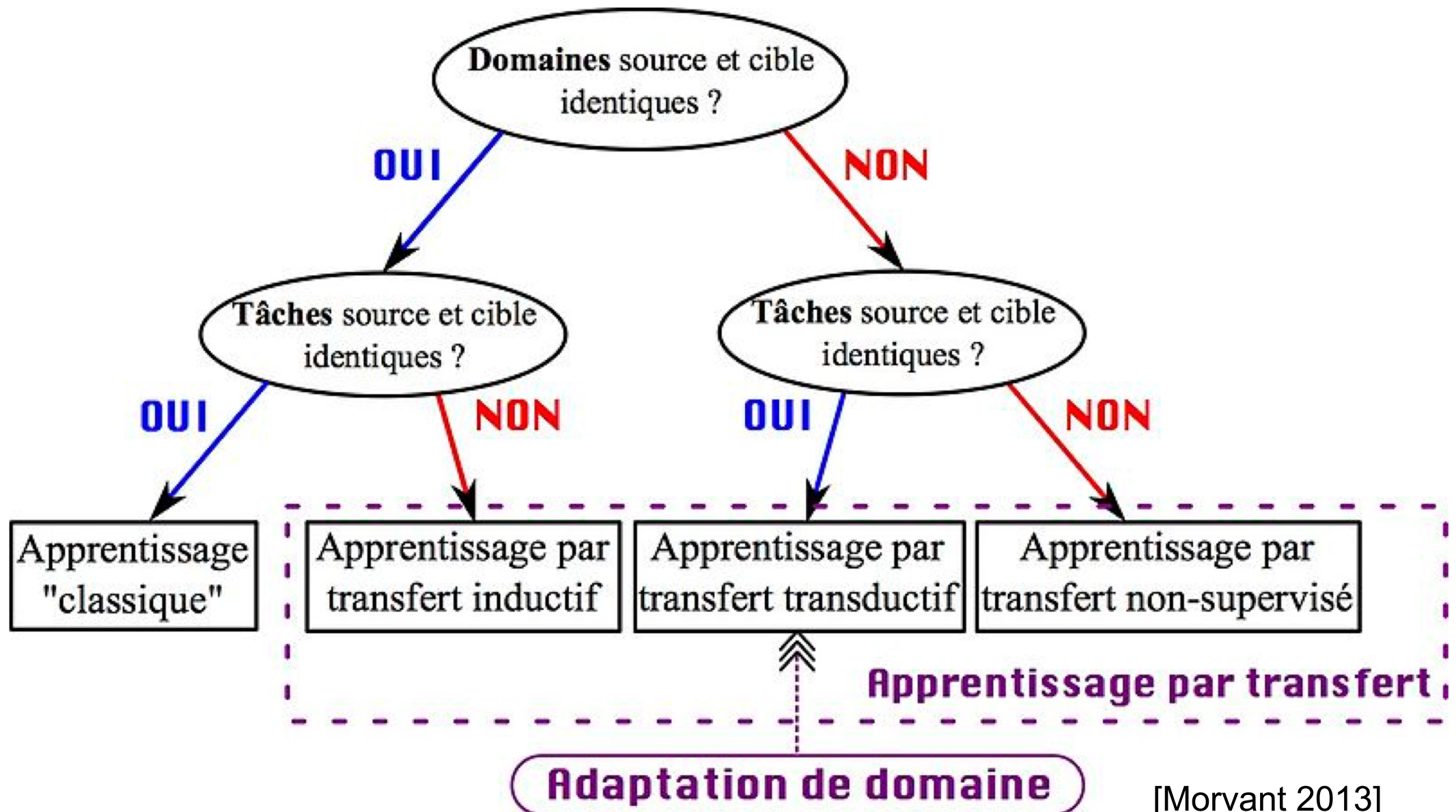
Jean-Pierre Briot



Transfer Learning



Transfer Learning



[Morvant 2013]

Software Architecture View/issues of Deep Learning

- (Feedforward) Deep Network = Pipeline Hierarchical Architecture
- Software Components
 - Layers
 - Patterns (Convolutions, Conditioning...)
 - Hyper-Parameters
 - » Non Linear Operators/Modules: Sigmoid, Tanh, ReLU...
 - » Loss functions: Quadratic, Cross-entropy...
 - » Regularization: Weights-based, Dropout...
 - » Optimization: Batch Gradient descent, Stochastic Gradient, Branch and Bound...
 - » Initialization Algorithms
 - ...
- Software Engineering Issues
 - Interoperability and Assemblage Assistance
 - » Frameworks
 - » Patterns and Anti-Patterns
 - Substituability
 - » Type discipline (but Every data boils down to real numerical computation)
 - » Semantics
 - Validation

Limits

- Specialized
- Data-based Model Generator
- Data-based (Statistical) Machine Learning only an (important) portion of Machine Learning
 - Reinforcement Learning
 - Inductive Logic Programming
 - ...
- Machine Learning only an (important but small) portion of Artificial Intelligence
 - Reasoning
 - Planning
 - Knowledge Representation
 - User Modeling and Interaction
 - Collaboration (Multi-Agent Systems)
 - Natural Language Processing
 - Dialogue
 - Argumentation
 - Speech Processing
 - Decision
 - Game Theory
 - Optimization
 - Robotics
 - ...

References

- Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2018.
- Eugene Charniak, Introduction to Deep Learning, MIT Press, 2019.
- Andrew Ng, Machine Learning Yearning, Online.
- Tom Mitchell, Machine Learning, McGraw Hill, 2017.
- Pedro Domingos, The Master algorithm, Basic Books, 2015.
- Deep Learning, Google, Udacity
- Machine Learning 3 – Markov Decision Processes and Reinforcement Learning, Georgia Tech, Udacity
- Andrew Ng, Machine Learning, Stanford University, Coursera
 - <https://www.coursera.org/learn/machine-learning>
- David Silver, Reinforcement Learning, University College London
 - <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>
- L'apprentissage profond, Yann LeCun, Collège de France
 - <https://www.college-de-france.fr/site/yann-lecun/course-2015-2016.htm>