MACHINE LEARNING AND AI

plain concepts

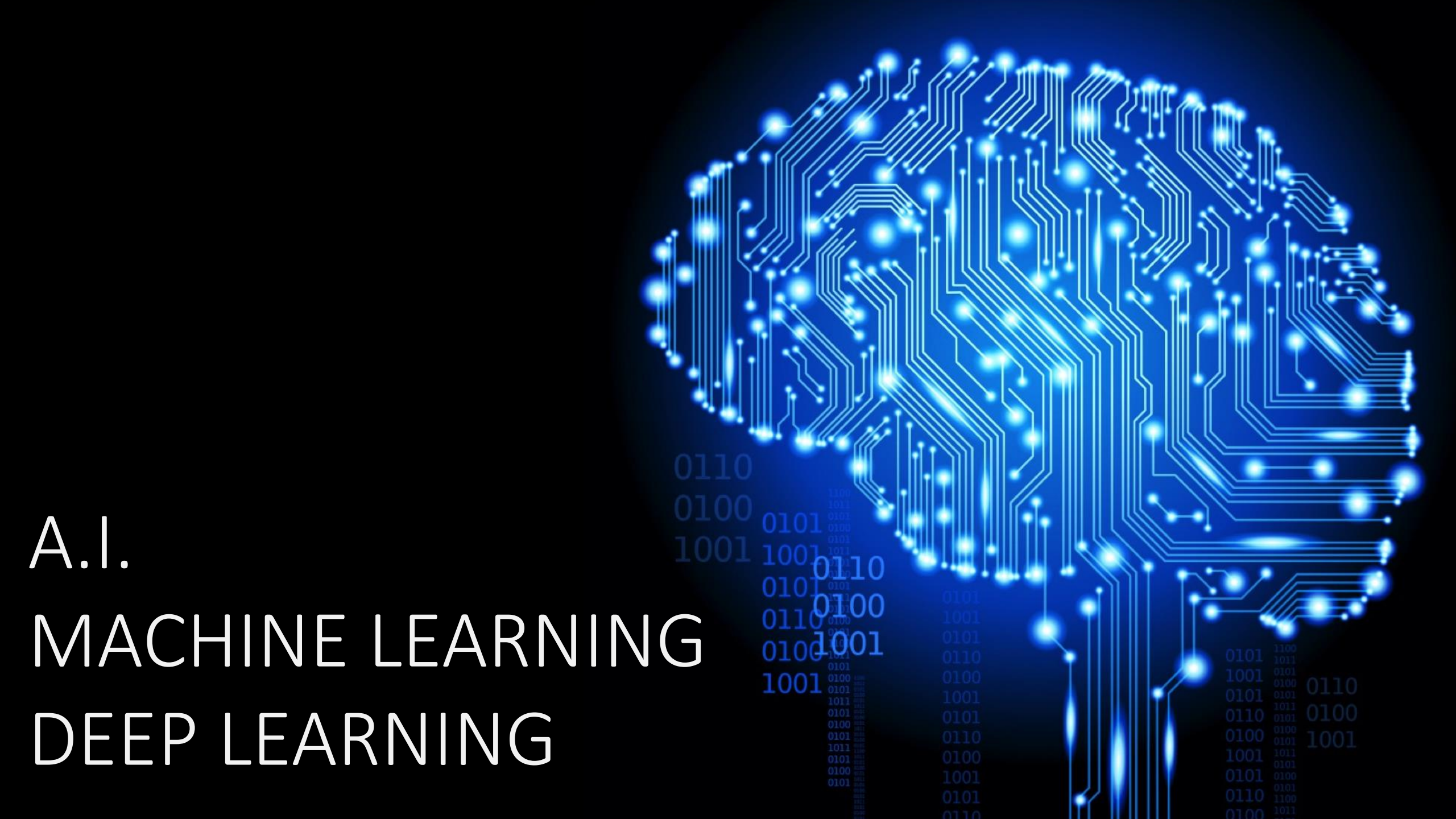
Pablo Doval

DATA PONTIFEX @Plain Concepts

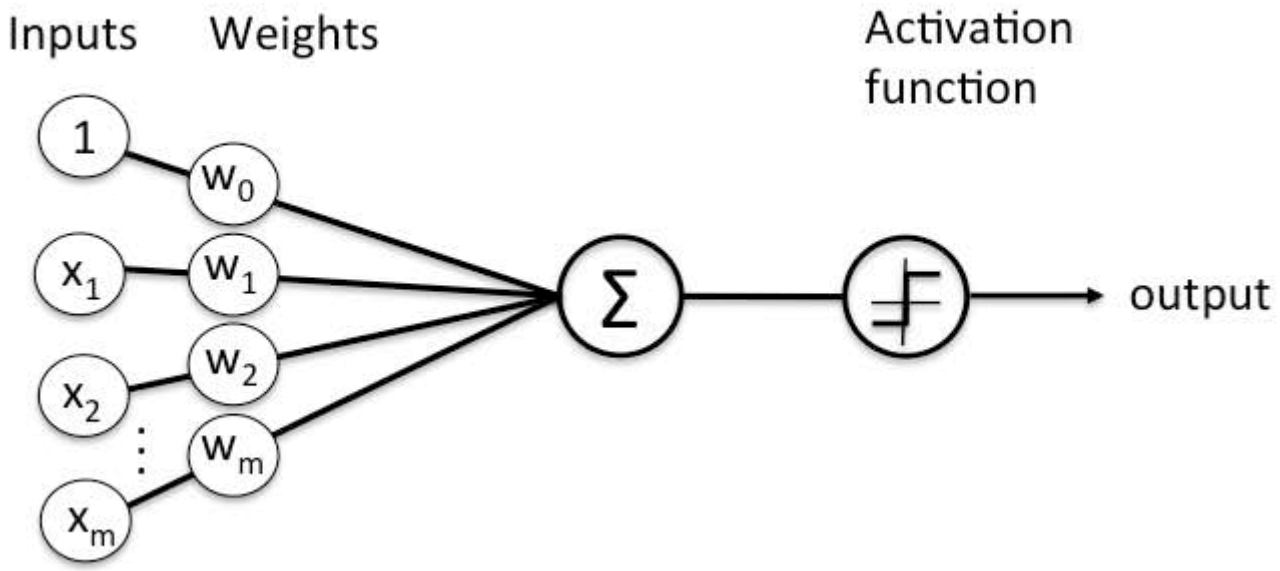
I work with code and data, but don't tell my mom; she thinks I'm a piano player in a whorehouse.

plain concepts

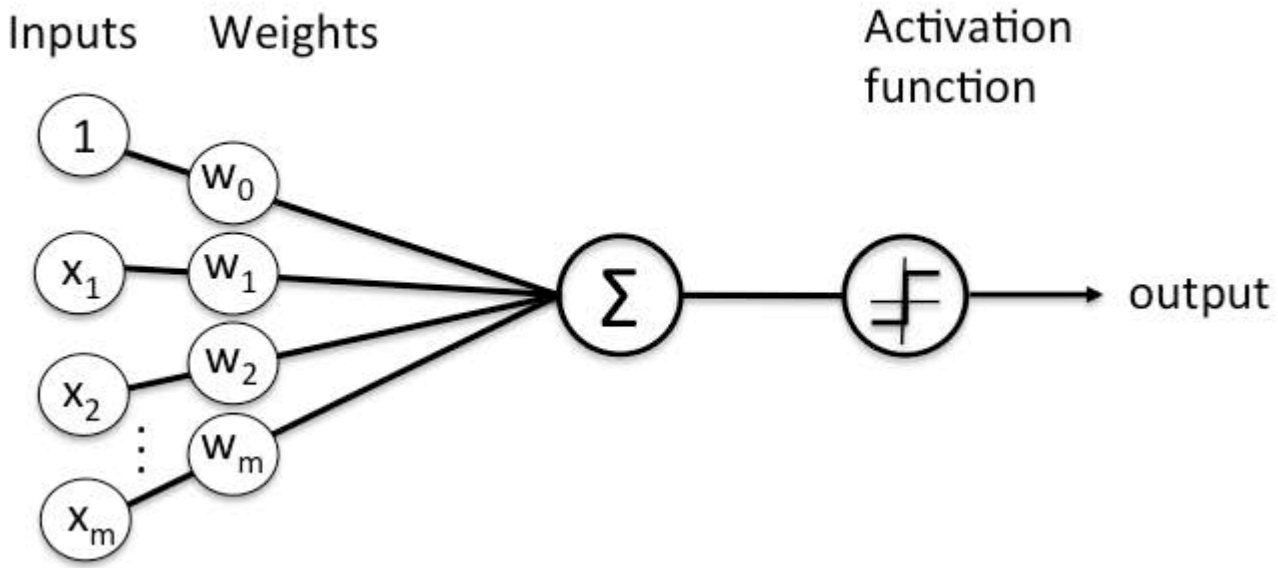
A BIT OF STORY

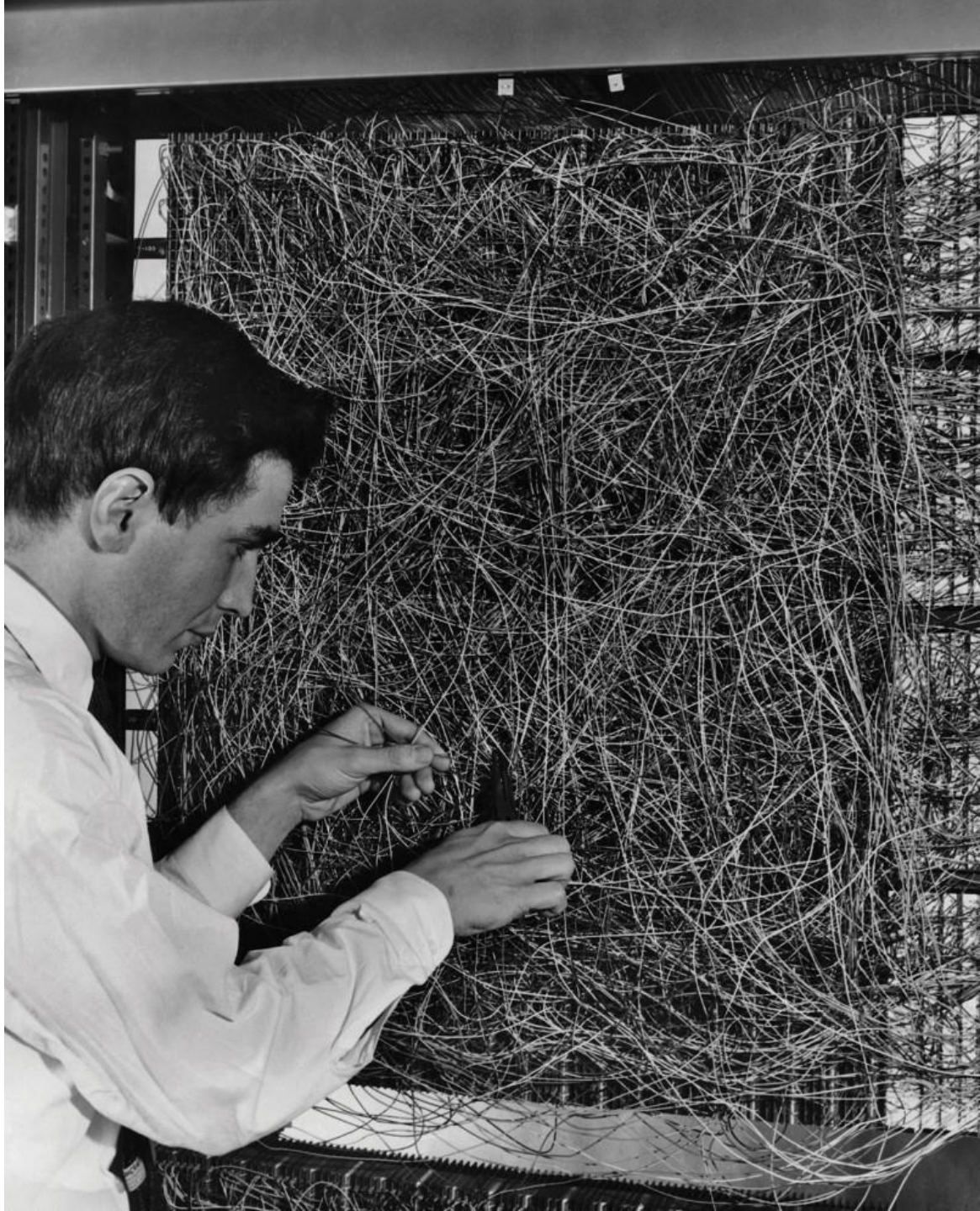

A.I.
MACHINE LEARNING
DEEP LEARNING

PERCEPTRON


Inputs
Weights
Activation function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

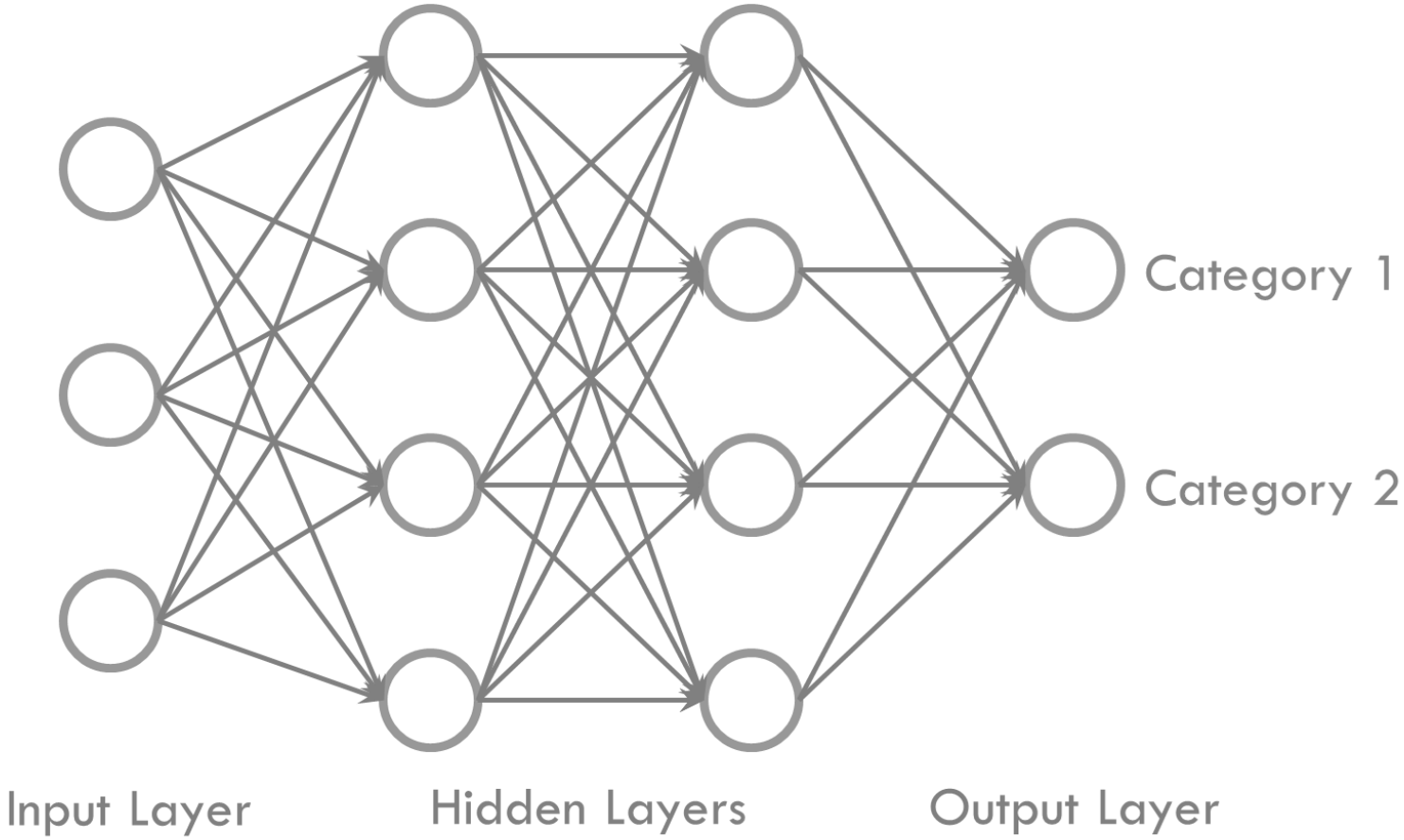
PERCEPTRON


Inputs
Weights
Activation function
1
w_0
x_1
w_1
w_2
x_2
w_m
x_m
Σ
output

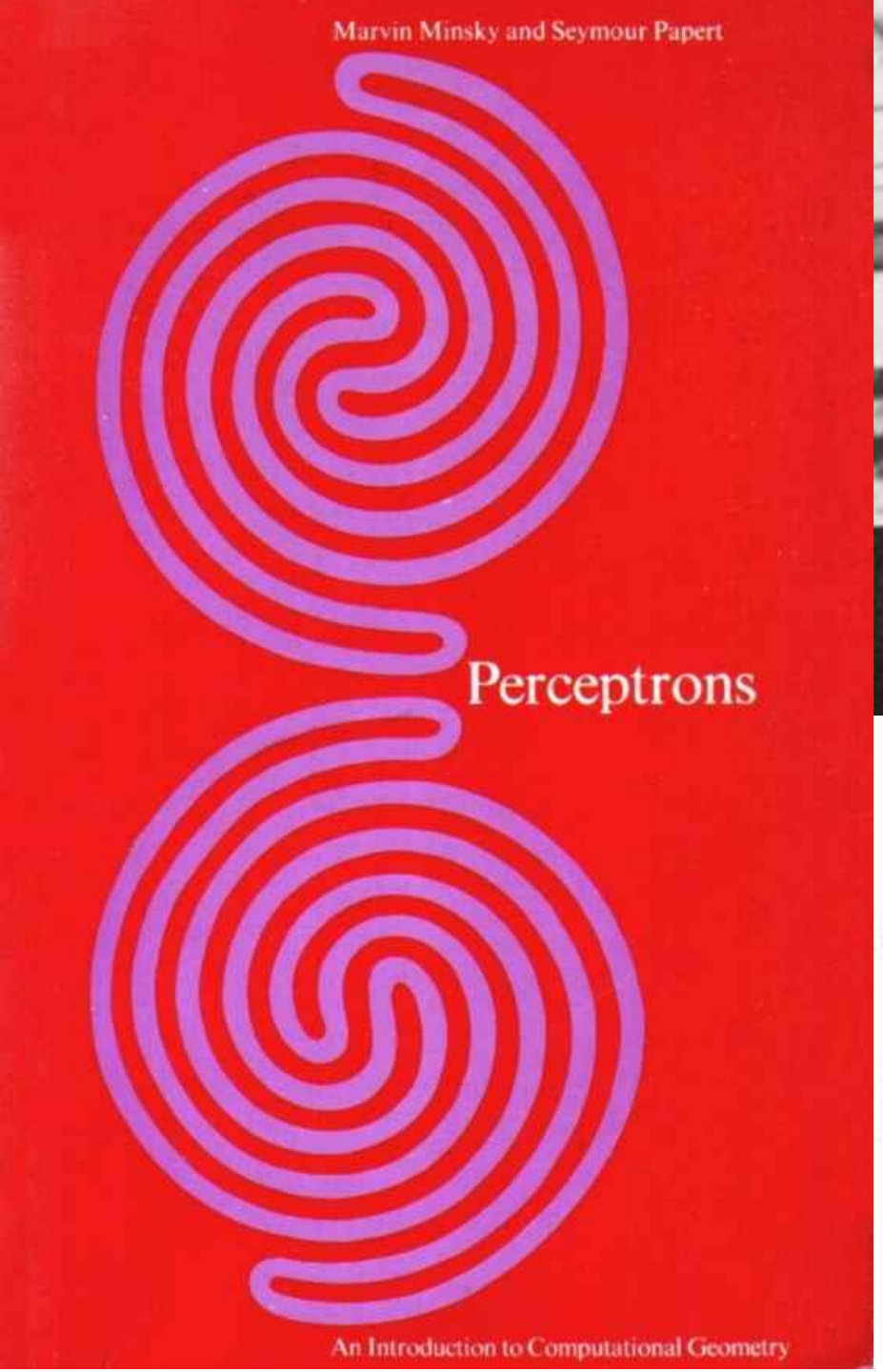


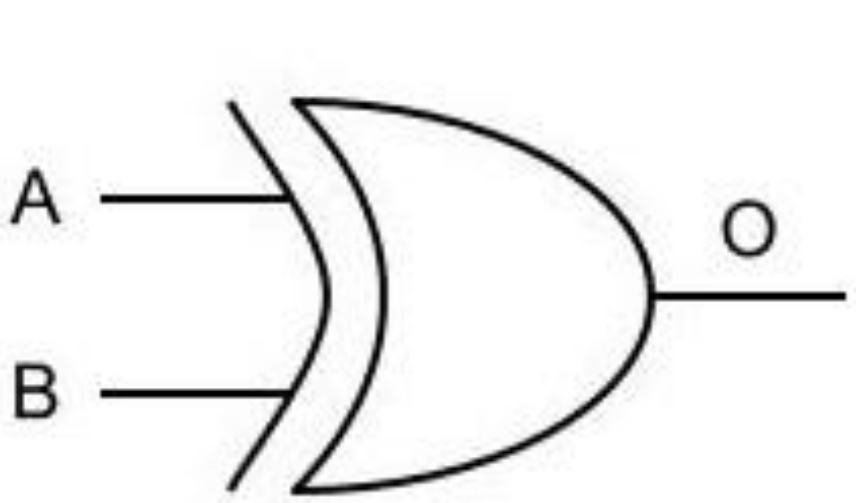
ARTIFICIAL INTELLIGENCE

MULTILAYER PERCEPTRON



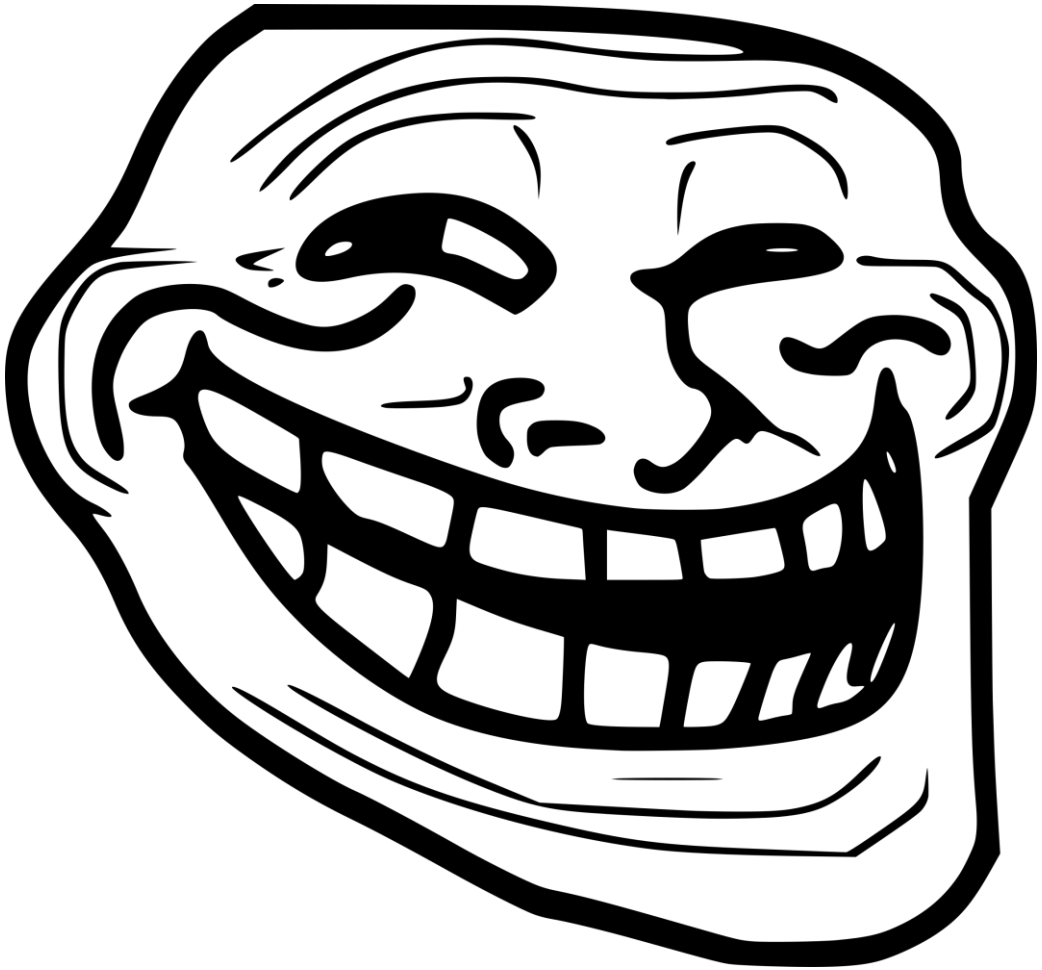
Winter is Coming

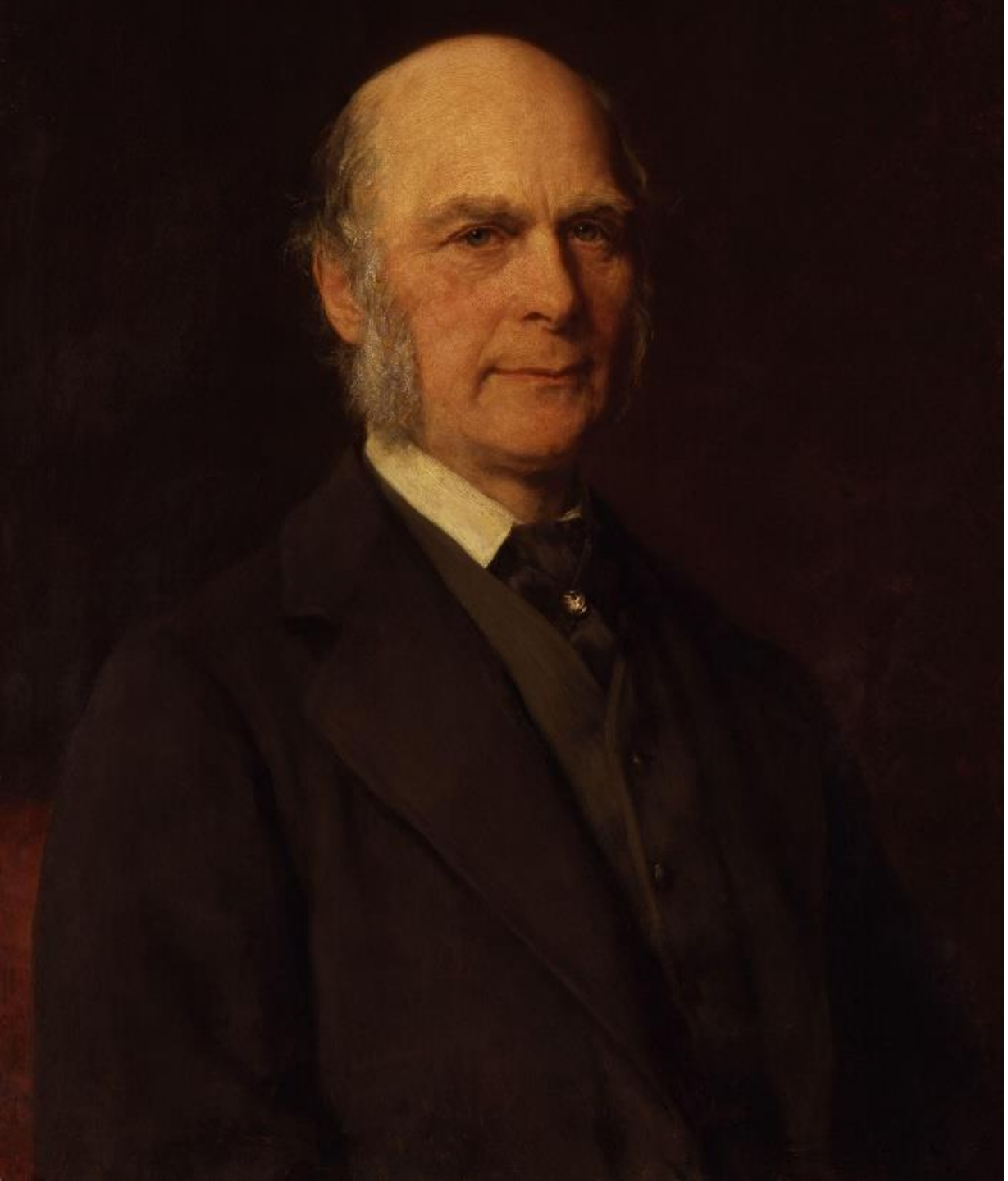




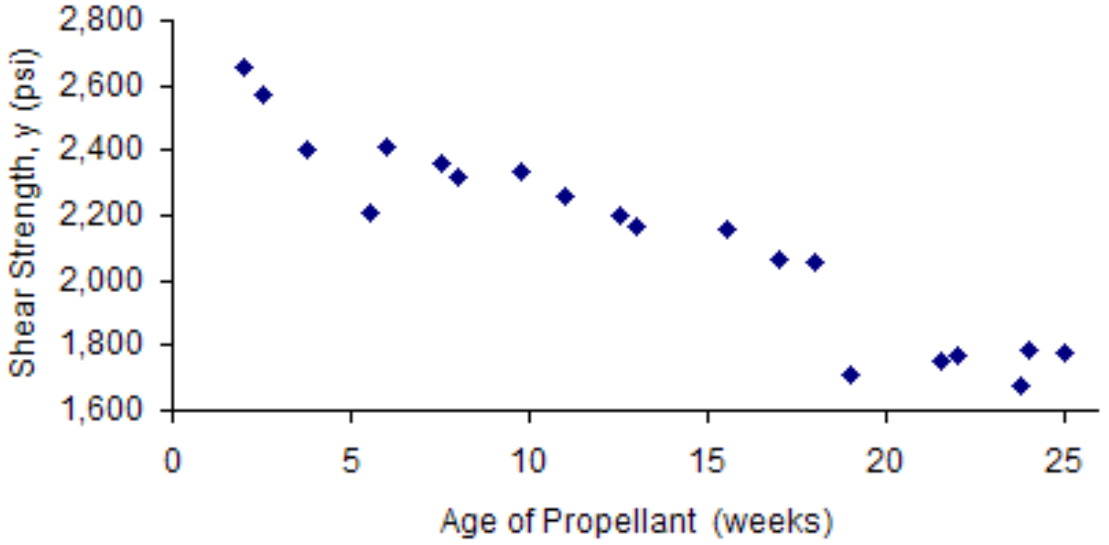
A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

EXPERT SYSTEMS?

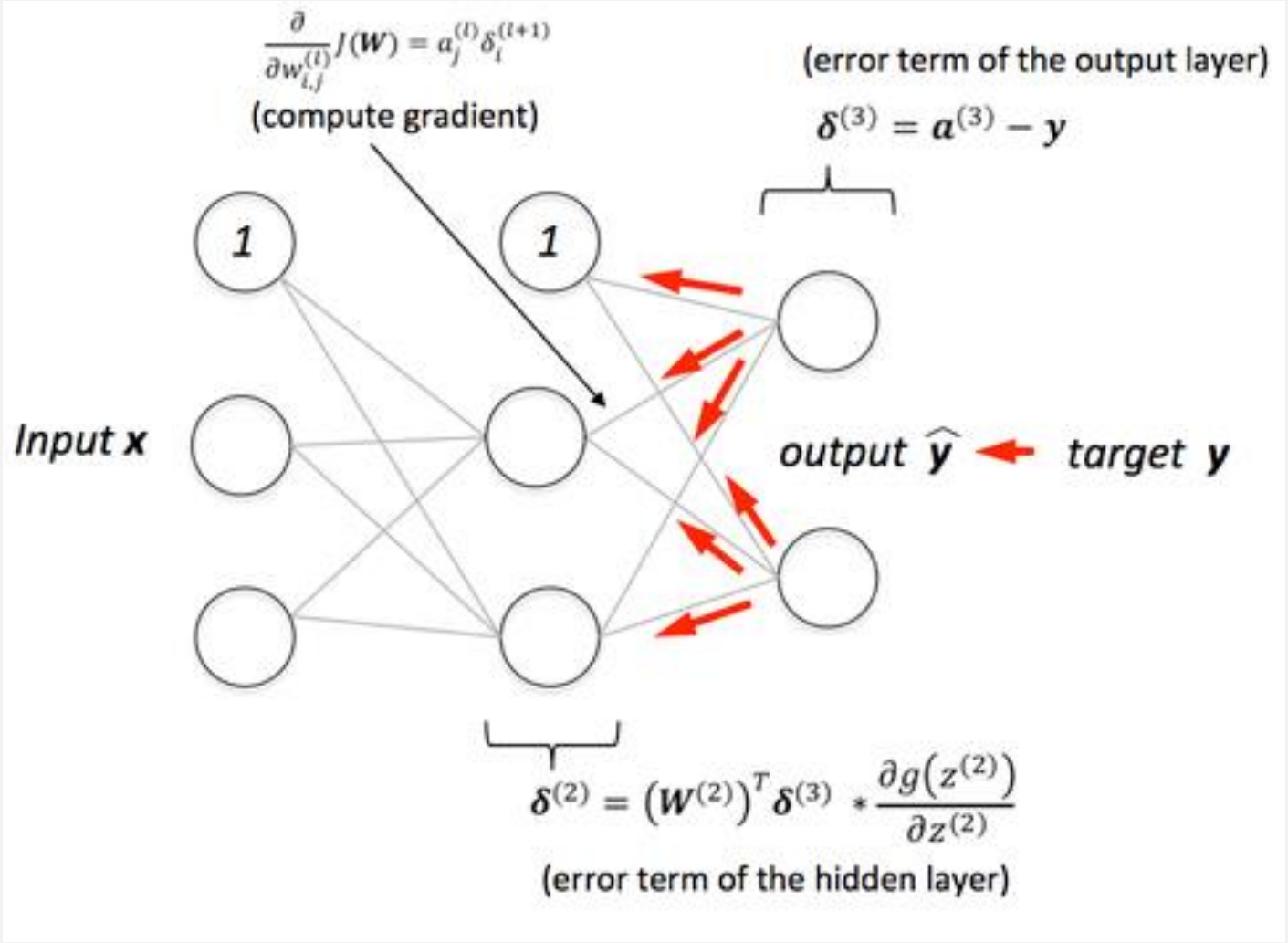




MACHINE LEARNING

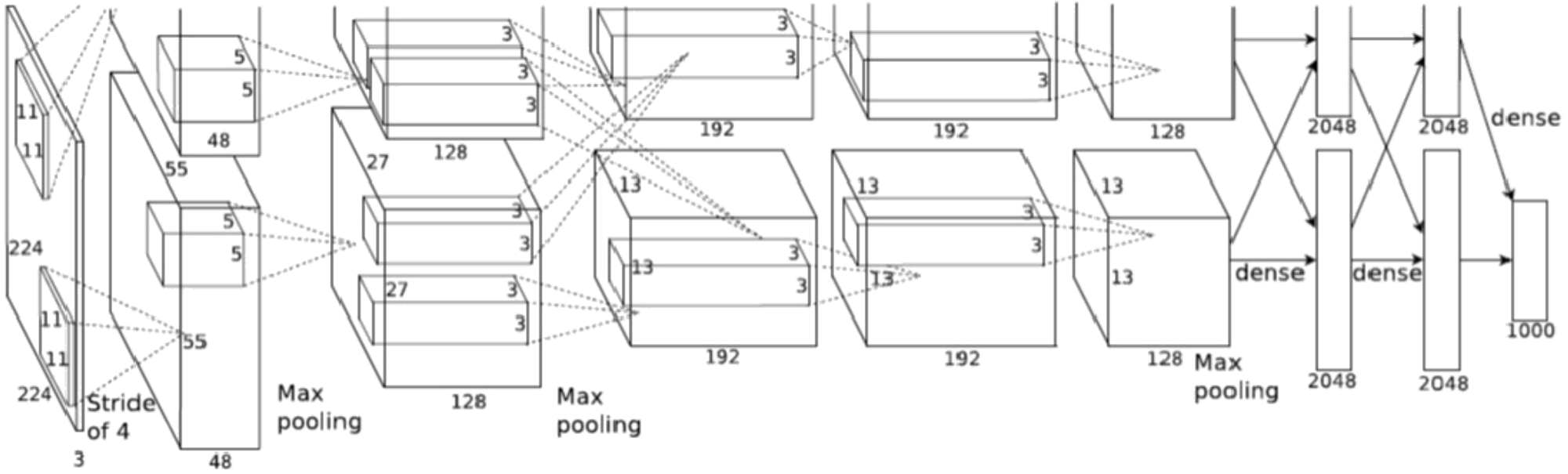


BACKPROPAGATION



DEEP LEARNING

A Machine Learning technique



plain concepts

MACHINE LEARNING

MACHINE LEARNING

Algorithms to find patterns in data

Intelligent and completely automatic *(the dream!)*

WHERE CAN IT BE USEFUL?

Where…

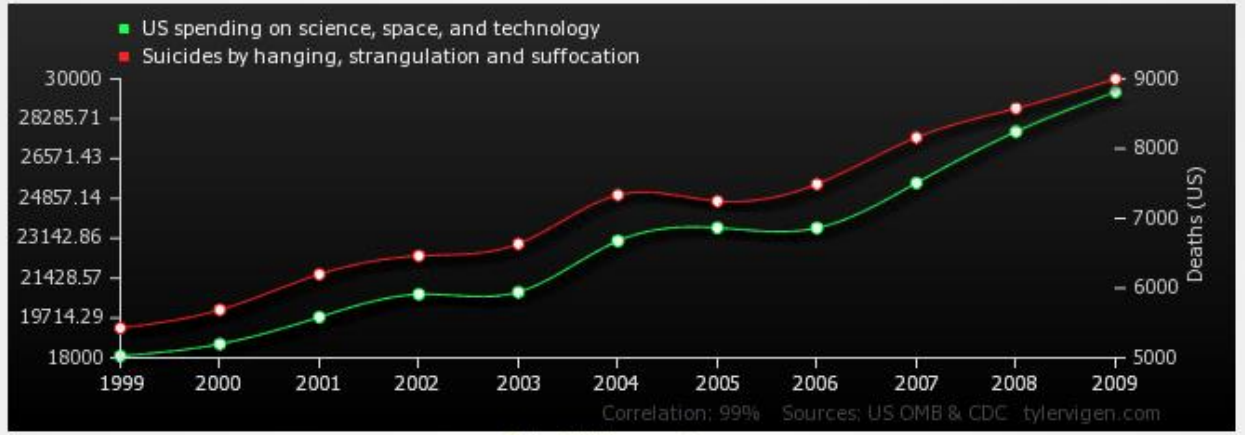
- There is no previous human experience
- Human experience exists, but it cannot be easily reduced to a set of rules (voice recognition)
- The scenario changes rapidly (spam email)
- Human training is expensive
- The amount of data is massive

COMMON SCENARIOS

Domain	Scenario
Financial services	Modelling risks
	Threat and fraud detection
Media y entertainment	Segmented advertisement
	Recommendation engine
Commerce	Sentiment analysis
	Transaction analysis in point of sales (POS)
IT	Call Detail Records
Government	Environment monitorization
	Traffic clogging and re-routing
Healthcare	Research (genome, cancer, etc..)
	Pandemic early detection
Engineering	Predictive maintenance

CORRELATION != CAUSALITY

US spending on science, space, and technology
correlates with
Suicides by hanging, strangulation and suffocation



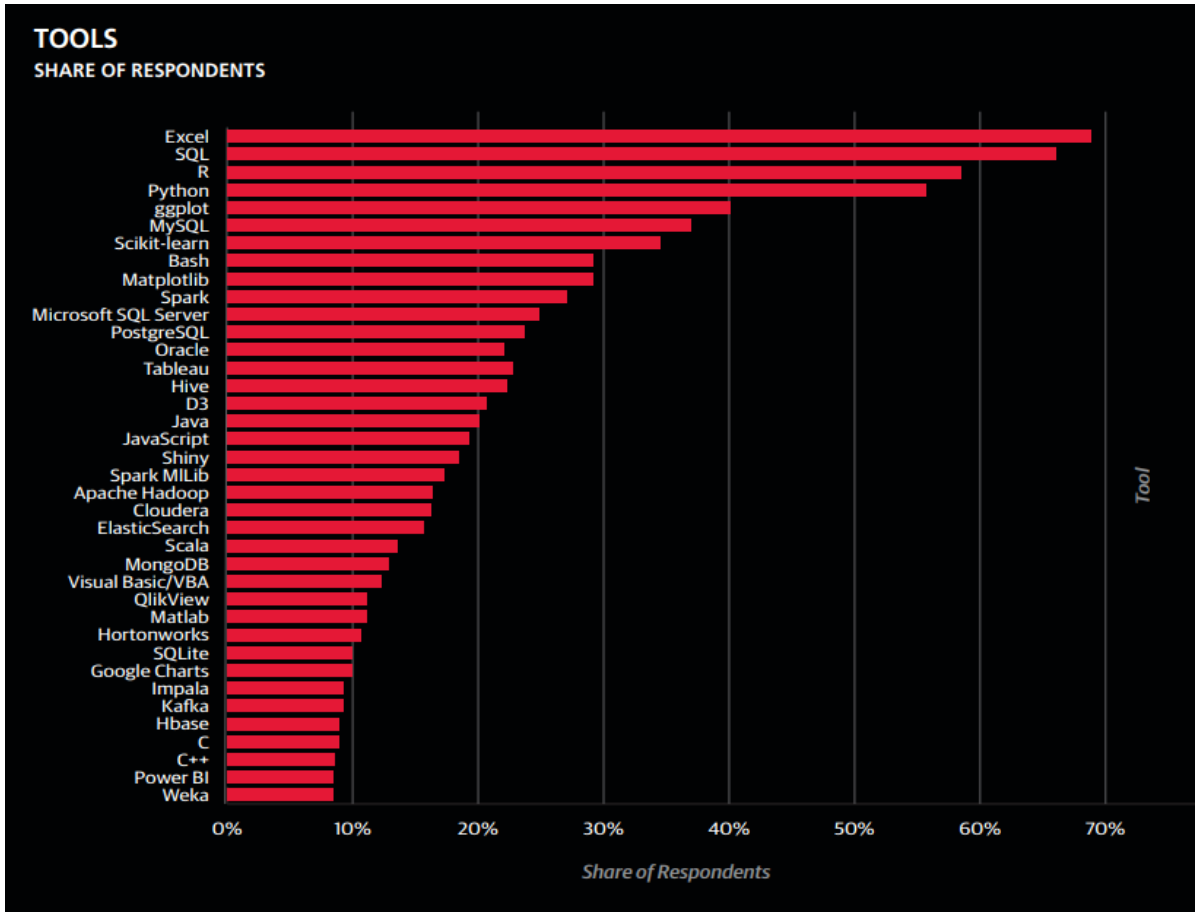
Upload this image to imgur

	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
US spending on science, space, and technology *Millions of todays dollars (US OMB)*	18,079	18,594	19,753	20,734	20,831	23,029	23,597	23,584	25,525	27,731	29,449
Suicides by hanging, strangulation and suffocation *Deaths (US) (CDC)*	5,427	5,688	6,198	6,462	6,635	7,336	7,248	7,491	8,161	8,578	9,000

Correlation: 0.992082

Source: http://www.tylervigen.com

TOOLS



"2017 European Data Science Salary Survey“ [O’Reilly]

OUR TOOLS

Mainly

- SQL Server
- Python
- R (with Rstudio and Microsoft R)
- Excel + PowerBI
- HDInsight (Hive and Spark)
- CNTK

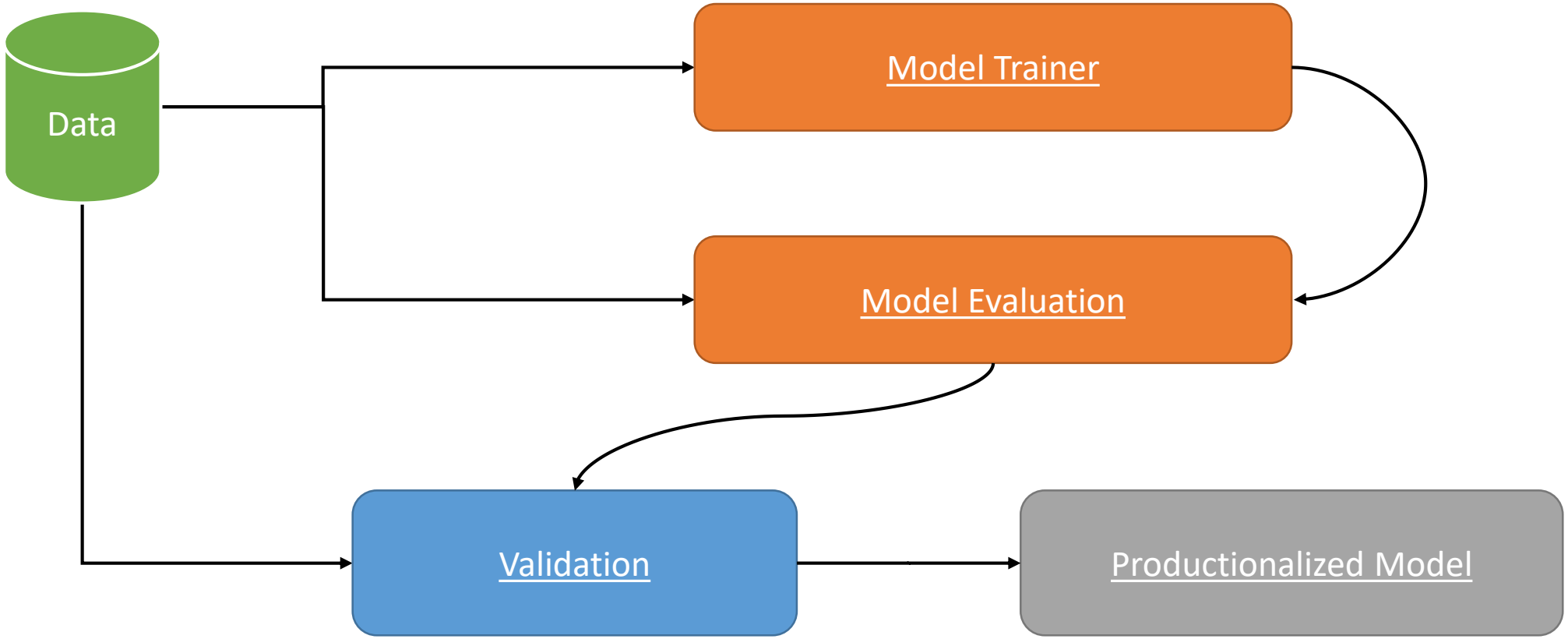
Secondly

- TensorFlow
- AzureML

We try to avoid ☺

- Mahout

BASIC CONCEPTS



MACHINE LEARNING CATEGORIES

Classification

- Assign a category
- i.e.: Restaurant (Chinese | Indian | Italian | Japanese)

Regression

- Predicting a real value for each element
- i.e.: Amount of a purchase, temperature, etc.

Ranking

- Ordering elements according to a criteria
- i.e.: web search

Clustering

- Partitioning of elements in heterogeneous groups
- i.e.: Tweets clustering by theme

Dimensionality reduction

- Transforming something complex to something simpler
- i.e.: Image pre-processing, voice recognition, signals

LEARNING / TRAINING

Task	• Applying a machine learning algorithm to domain data (training data) • This domain represents a series of features that we want to model
Goal	• Minimize errors
Types of learning	• Supervised • Not supervised • Others (semi-supervised, active)

METHODS

Supervised

Non Supervised

SUPERVISED LEARNING

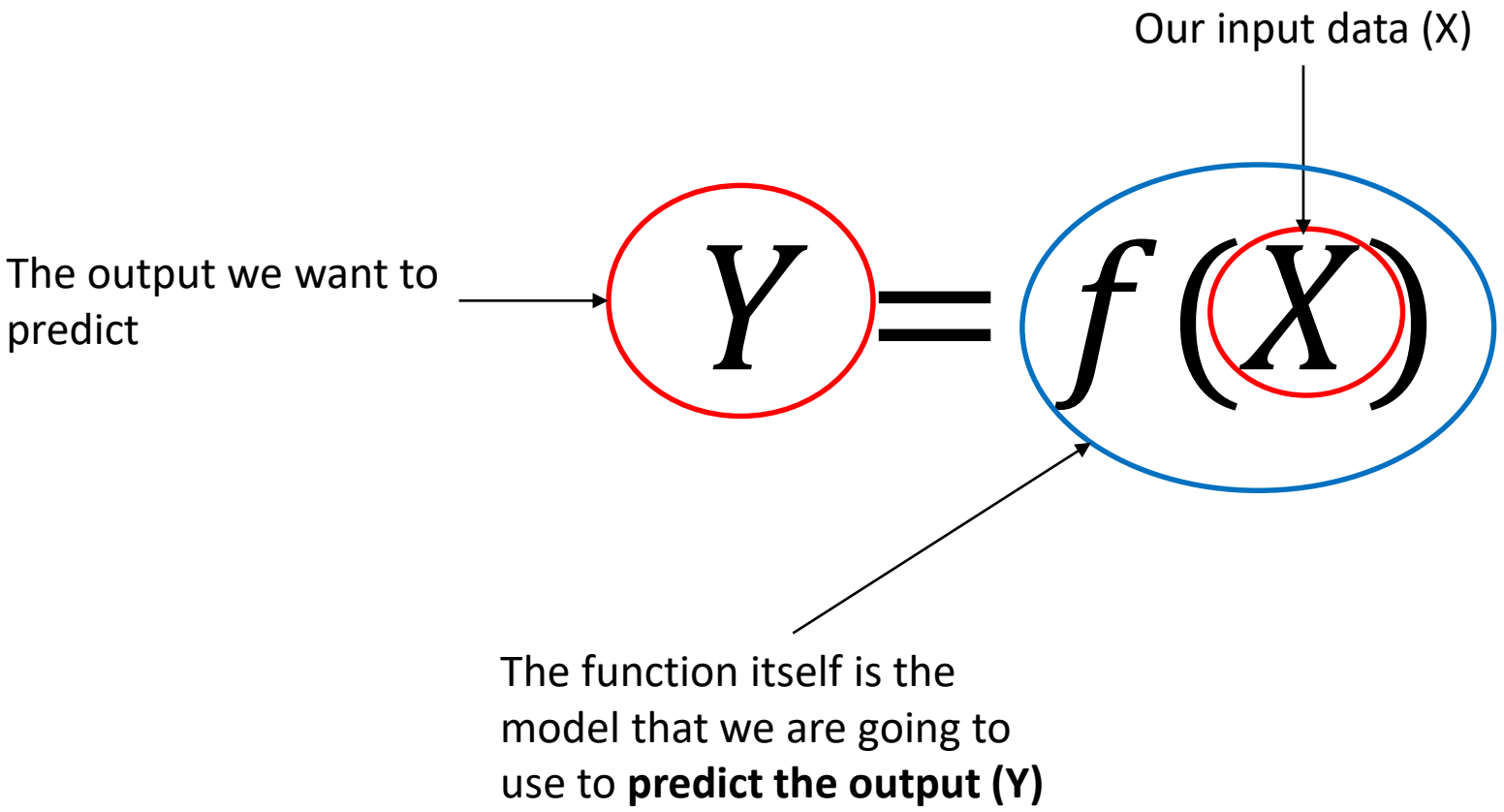
*Training a model with data that has been previously **labeled***

SUPERVISED LEARNING

Labeled means that we have **historical data** about what we want to **predict**

SUPERVISED LEARNING

- Basically:



SO, WHERE IS THE LEARNING PROCESS?

The learning process consists on adjusting the model (*f(x)*) parameters in order to minimize the error when predicting

PREDICTING CAR TYPE USING DIMENSIONS

$f($  $) =$

NON SUPERVISED LEARNING

- Unsupervised training is when you have input data but no output
- The goal of this kind training is not predicting an output but modelling the underlying structure or distribution in the input data
- It is called non supervised because there is no correct answers or teacher

TYPES OF NON SUPERVISED LEARNING

Clustering

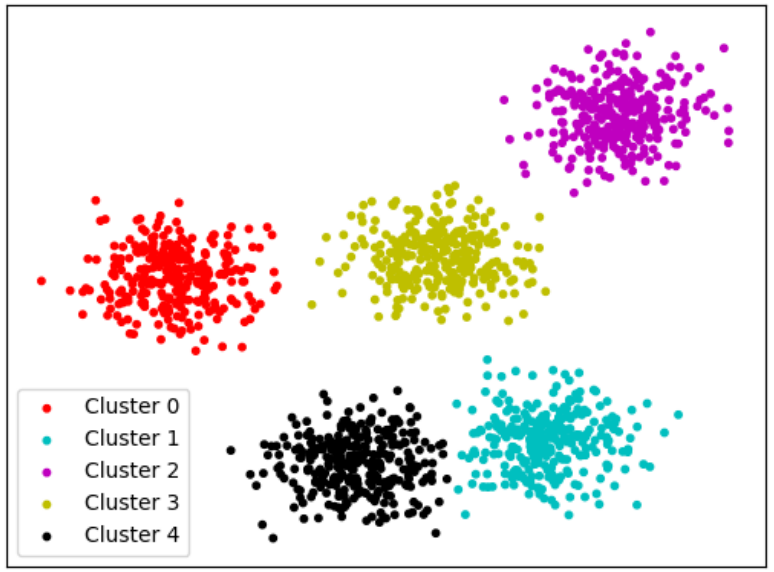
- Discover groups in data
- Example: Grouping cars by horsepower

Association

- Search rules for large portion of you data
- Example: Amazon's "other customers also bought…"

EXAMPLE OF CLUSTERING

$$f(horsepower, weight) =$$



WHEN IS IT USEFUL?

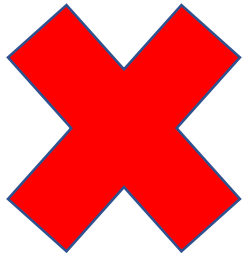
- When data cannot be labeled
  - Or when it is expensive to achieve
- When the amount of data is huge
  - Would anyone label/tag 10M of tweets?
- When there is no previous experience

SEMI SUPERVISED LEARNING

- Used combining labelled an unlabelled data to achieve better model results
- Consists on:
  1. Train a model using labelled data (supervised training)
  2. Predict using unlabelled data
  3. Add the best predictions (based in a threshold) to the labelled data set
  4. Repeat

EXAMPLE?

- Let’s say we want to categorize websites
- Labelled data would force to
    1. Crawl data from the website
    2. Someone (human) would have to tag the site: “Forum”, “News”, “E-commerce”, etc
- The bigger the sample the better the results



NOT AFFORDABLE

TAGGING WEBS

- But writing a crawler is a “simple” thing
- We could crawl the data, evaluate the website data using our “economic” model
- Use the best prediction as if they were labelled data
- Repeat

AFFORDABLE

TYPES OF SEMI SUPERVISED LEARNING

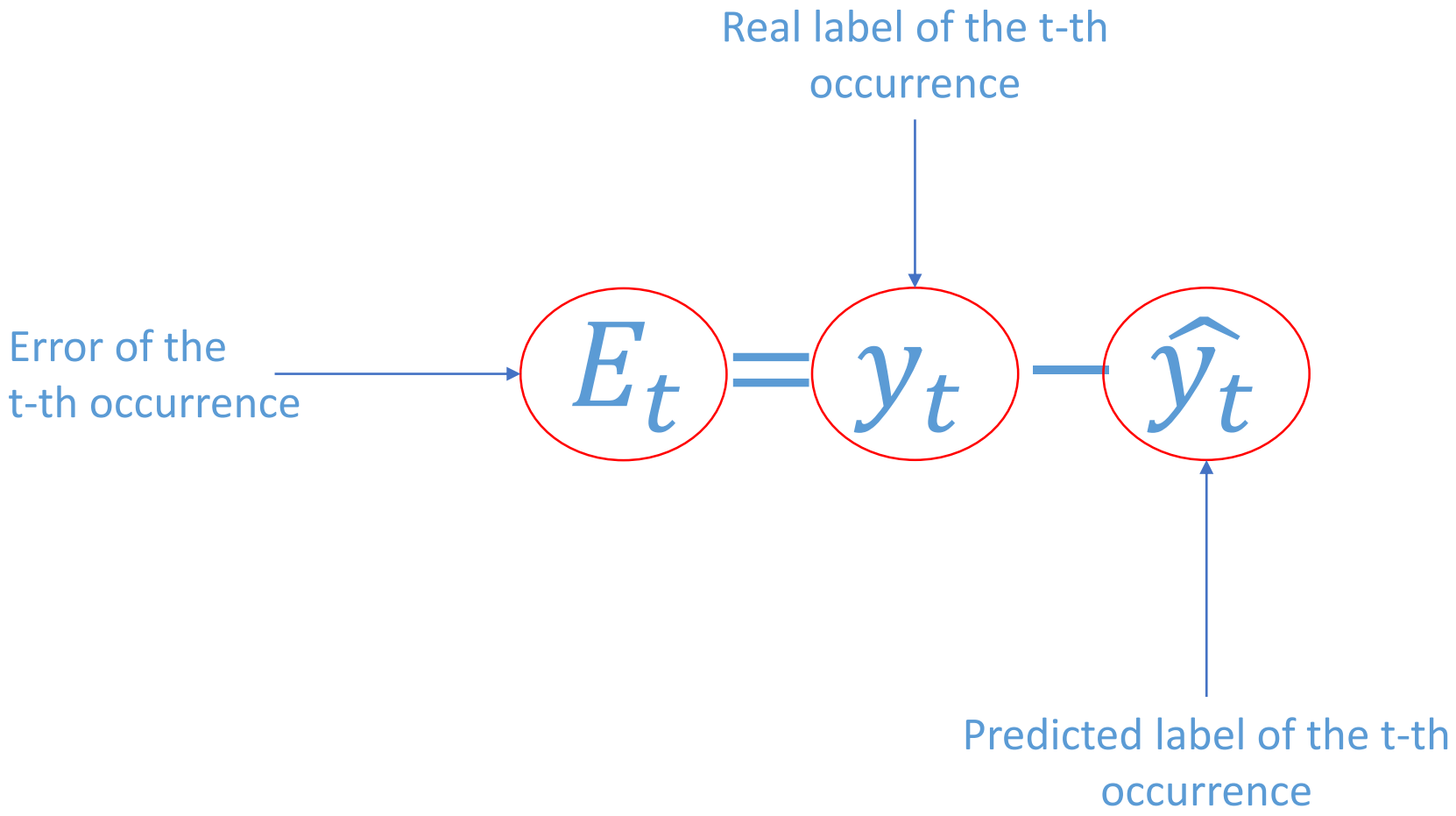
- Self training (the simplest one)
- Generative Models
- S3VMs
- Graph-Bases Algorithms
- Multiview Algorithm

A QUICK INTRODUCTION TO ERROR MEASURING

- After the model is created and evaluated
- How can we know if it was a good model?

- We need to measure the **error and be able to compare it**

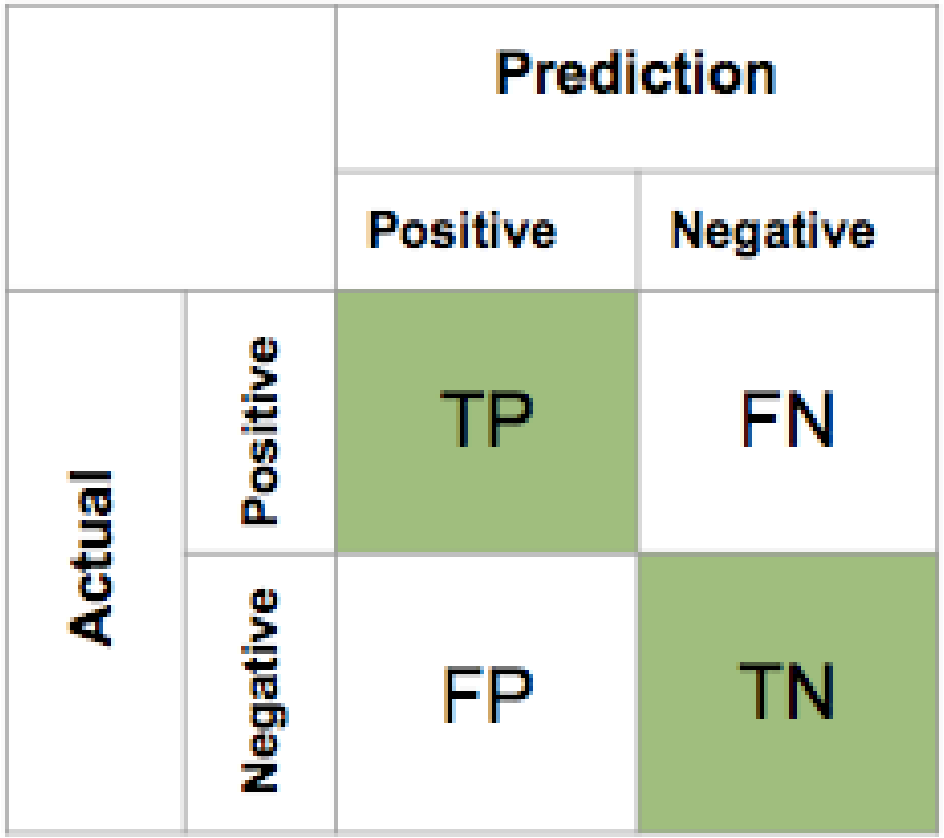
DEFINITION OF ERROR



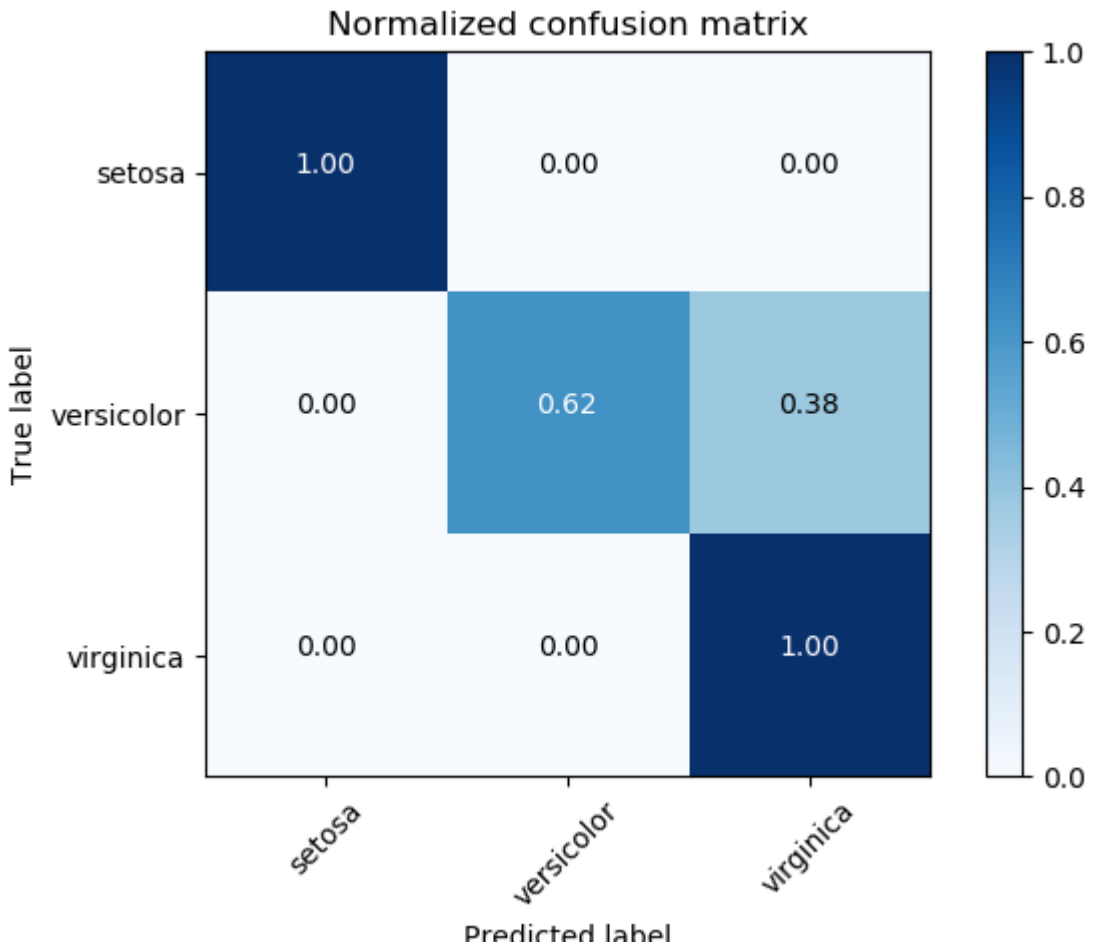
STATISTICS TOOLS

- Confusion matrix
- Recall and precision
- Error measurement:
  - Mean absolute error
  - Mean squared error
  - Root mean squared error
  - Mean absolute percentage error

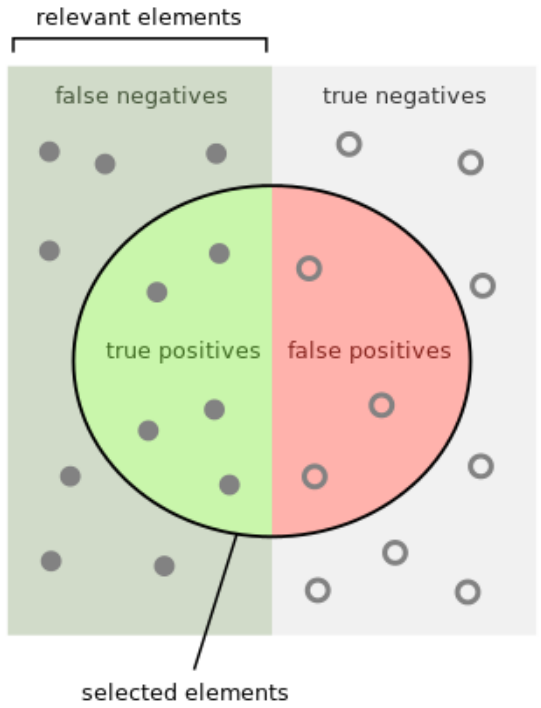
CONFUSION MATRIX



CONFUSION MATRIX



RECALL AND PRECISION



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

ERRORS MEASUREMENT

Mean squared error	MSE	=	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	RMSE	=	$\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	MAE	=	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	MAPE	=	$\frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

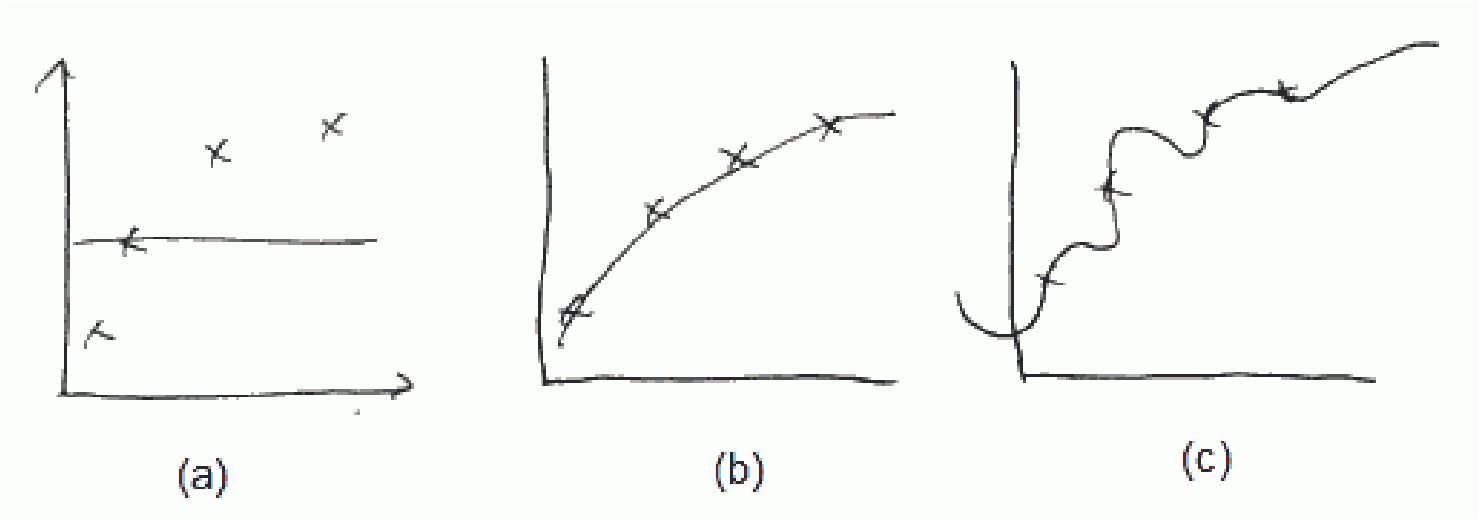
plain concepts

AZURE ML STUDIO

https://studio.azureml.net/

OVERFITTING AND UNDERFITTING

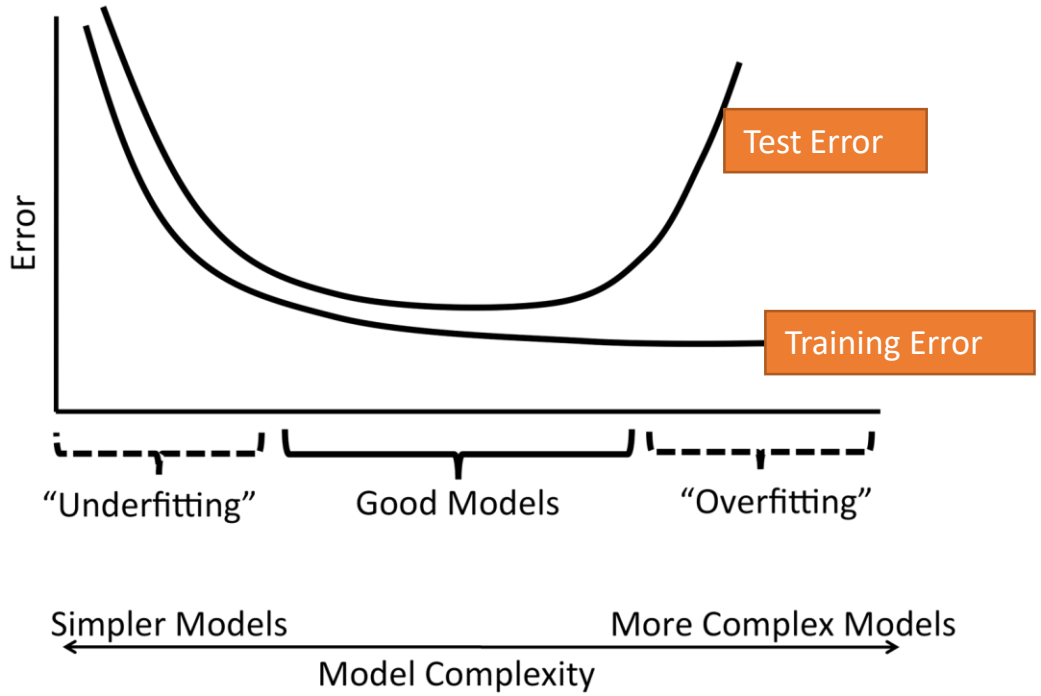
OVERFITTING AND UNDERFITTING



OVERFITTING AND UNDERFITTING

Occam Razor:

- Best models are the simplest which fit the data properly



CLASSIFICATION

plain concepts

CLASSIFICATION

- There is something we want to predict (a class or a label)
- That class or label is categorical
  - That is, it's a value in a set of categories, not a number
  - True/False, A/B/C/D, …
- There is a set of features associated to that class
- We want to know what features are able to predict that class

CLASSIFICATION

- Example: Manhole explosion analysis

Manhole model: [5 3 120 12 1 0 …..]

Number of events last year
Number of serious incidents last year
Number of wires installed before 1930
Number of electric wires
Manhole with air circulation?
Inspected?

CLASSIFICATION

- Each observation is represented by a feature vector

Manhole model:

[	5	3	120	12	1	0	] -1
[	0	0	89	5	1	1	] 1
[	1	0	20	0	0	1	] -1

Features

(predictors, covariables
or independent variables)

labels, Y

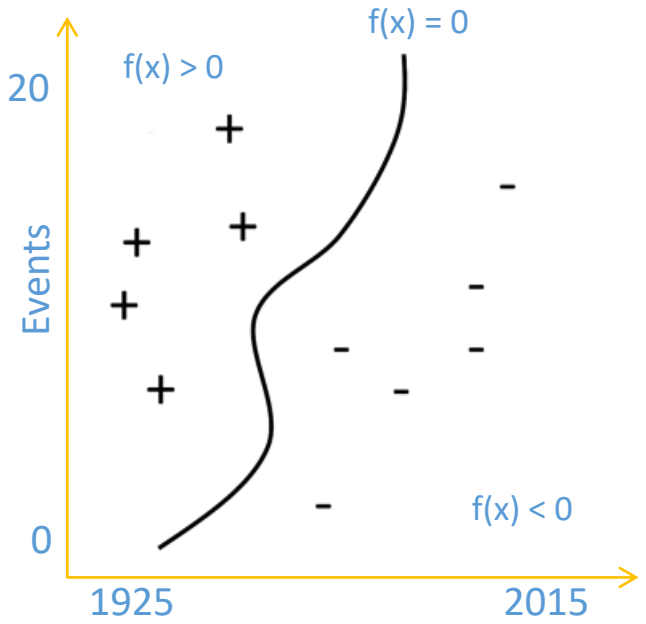
CLASSIFICATION

- Given a training set (x_i, y_i) for $i=1…n$, we want to create a classification model which is able to predict a label ***y*** for the next value of **x**

Manhole model : [1925 15]

Wire installation date

Number of events last year



CLASSIFICATION

Binary

Multivariate

Use cases

- Automatic handwritten recognition
- Spam detection
- Fraud detection
- Customer churn
- Voice recognition
- Image recognition
- Etc.

plain concepts

CLASSIFICATION

DECISION TREES

plain concepts

DECISION TREES

- Classification and prediction tools
- Can be represented using a tree structure, hierarchy and recursive
  - Easy to understand, even for not expert users
- Each node is divided in branches
- Each leaf represents a decision

TENNIS MATCH

Day	Weather	Temperature	Humidity	Wind	Match?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Warm	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
7	Cloudy	Cold	Normal	Strong	Yes
8	Sunny	Warm	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rainy	Warm	Normal	Weak	Yes
11	Sunny	Warm	Normal	Strong	Yes
12	Cloudy	Warm	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rainy	Warm	High	Strong	No

What will happen day 15?

Rain, High temperature, High humidity and weak wind.

TENNIS MATCH

Day	Weather	Temperature	Humidity	Wind	Match?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Warm	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
7	Cloudy	Cold	Normal	Strong	Yes
8	Sunny	Warm	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rainy	Warm	Normal	Weak	Yes
11	Sunny	Warm	Normal	Strong	Yes
12	Cloudy	Warm	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rainy	Warm	High	Strong	No

Training Data: 14 rows

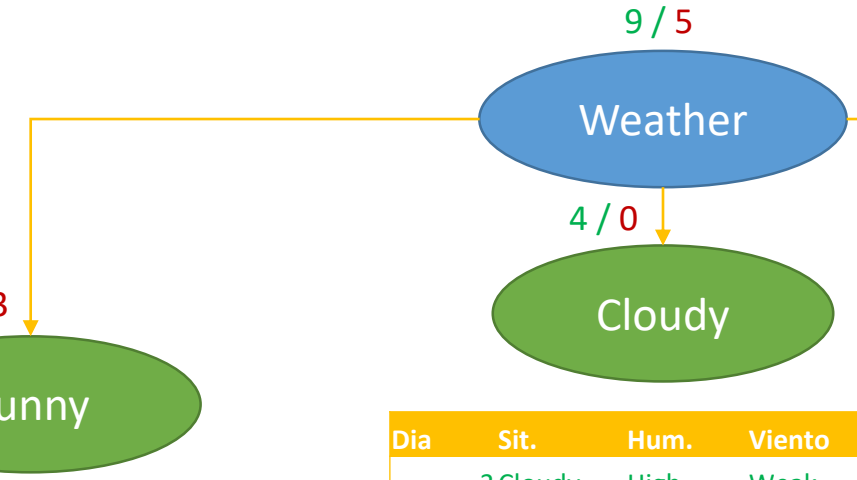
TENNIS MATCH

Day	Weather	Temperature	Humidity	Wind	Match?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Warm	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
7	Cloudy	Cold	Normal	Strong	Yes
8	Sunny	Warm	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rainy	Warm	Normal	Weak	Yes
11	Sunny	Warm	Normal	Strong	Yes
12	Cloudy	Warm	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rainy	Warm	High	Strong	No

Training Data:

9 Yes
5 NO

TENNIS MATCH



Dia	Sit.	Hum.	Viento	Partido
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	Yes
11	Sunny	Normal	Strong	Yes

Keep splitting

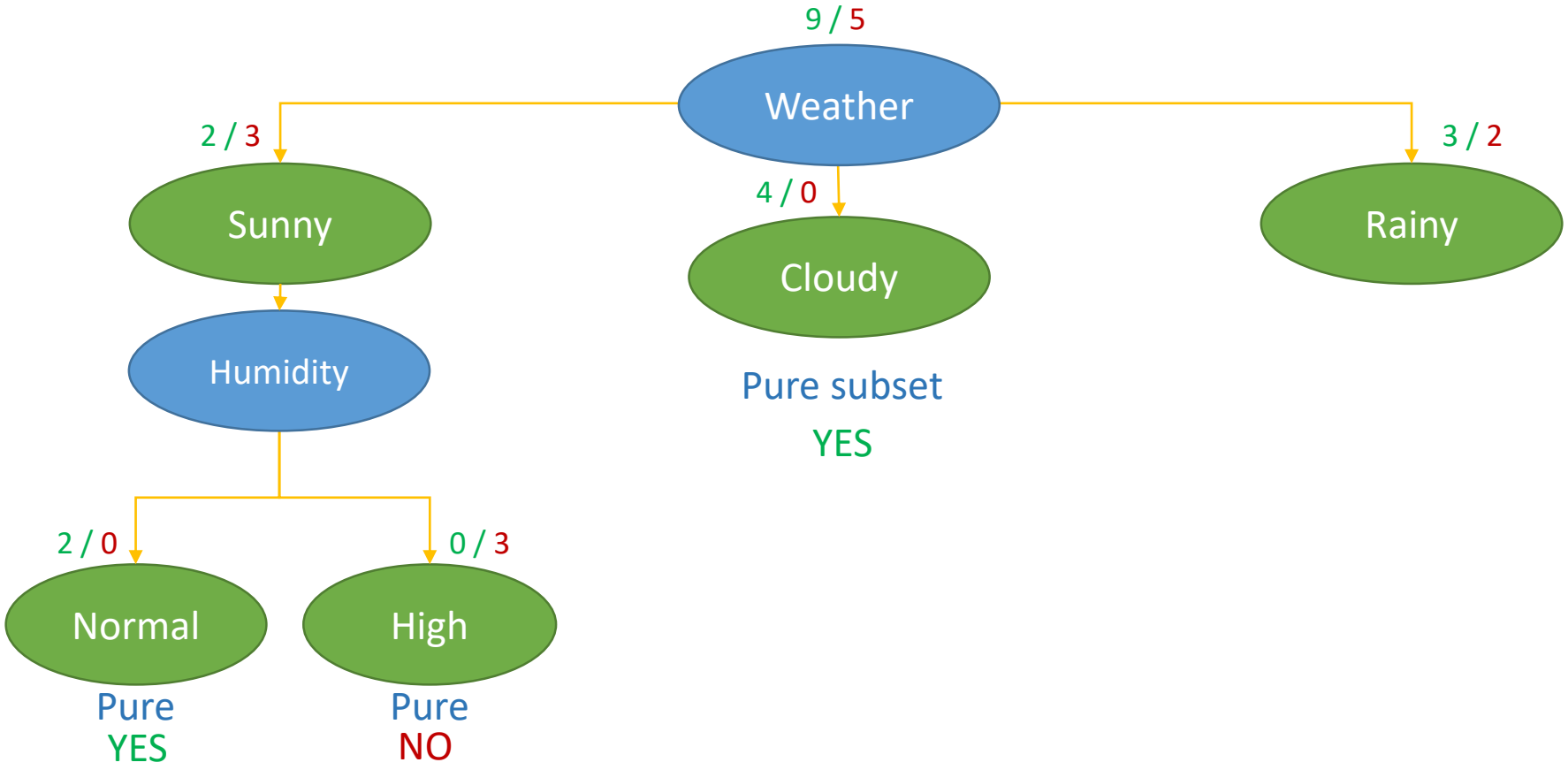
Dia	Sit.	Hum.	Viento	Partido
3	Cloudy	High	Weak	Yes
7	Cloudy	Normal	Strong	Yes
12	Cloudy	High	Strong	Yes
13	Cloudy	Normal	Weak	Yes

Pure subset

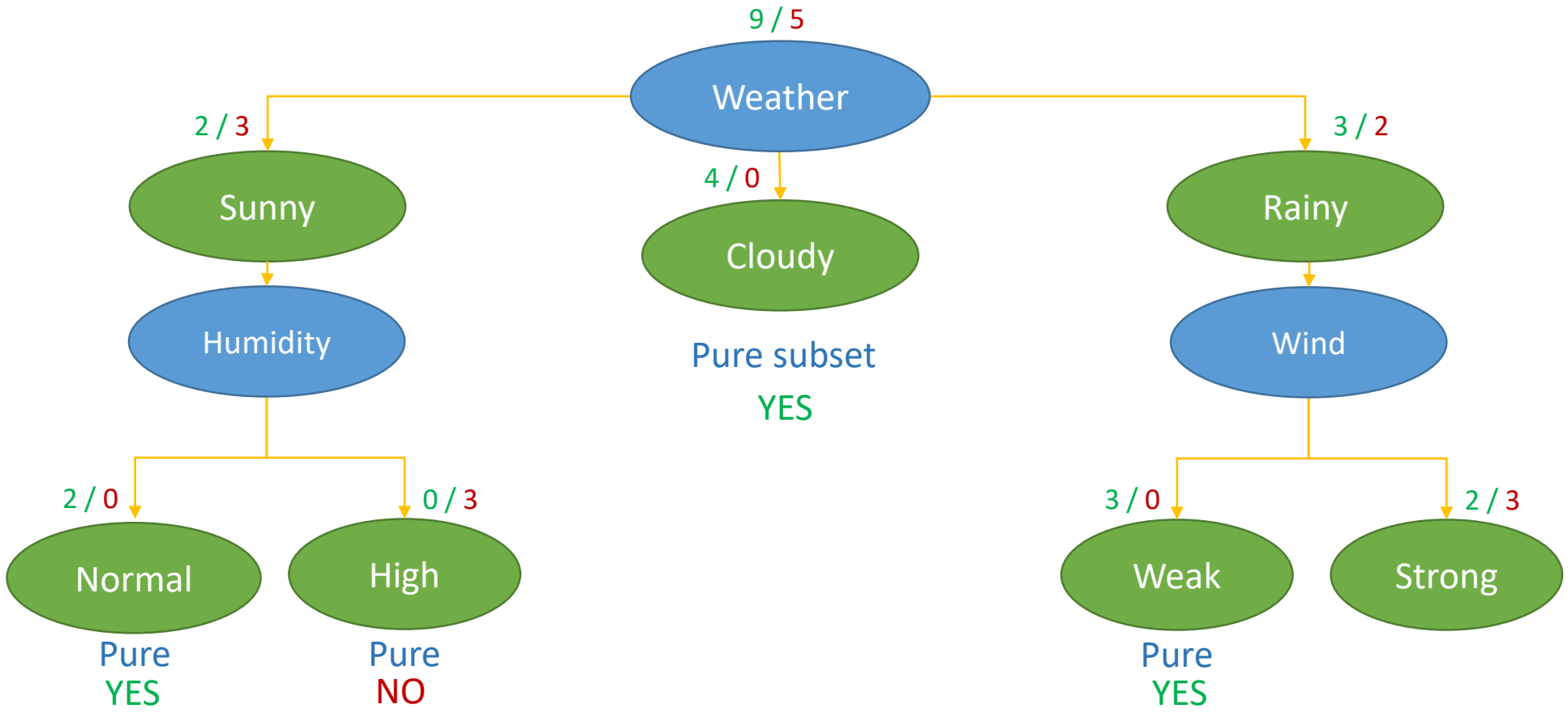
Dia	Sit.	Hum.	Viento	Partido
4	Rainy	High	Weak	Yes
5	Rainy	Normal	Weak	Yes
6	Rainy	Normal	Strong	No
10	Rainy	Normal	Weak	Yes
14	Rainy	High	Strong	No

Keep Splitting

TENNIS MATCH



TENNIS MATCH



REGRESSION

plain concepts

REGRESSION

- There is something we want to predict (class or label)
- It’s a numeric/real value
  - How many conversions are we going to get for this campaign in this week?
  - How many televisions will sell next year?
  - What is the salary of someone given his demographic information?
- There is a set of features associated to that class
- We want to know which features can predict that class

REGRESSION

- Each observation is represented by a feature vector

We model a person like this:

[	5	3	120	12	1	0 .....]	83
[	0	0	89	5	1	1 .....]	32
[	1	0	20	0	0	1 .....]	-10

Features, or X

labels, Y

(Predictors, covariables
or independent variables)

REGRESSION

- Each observation is represented like:

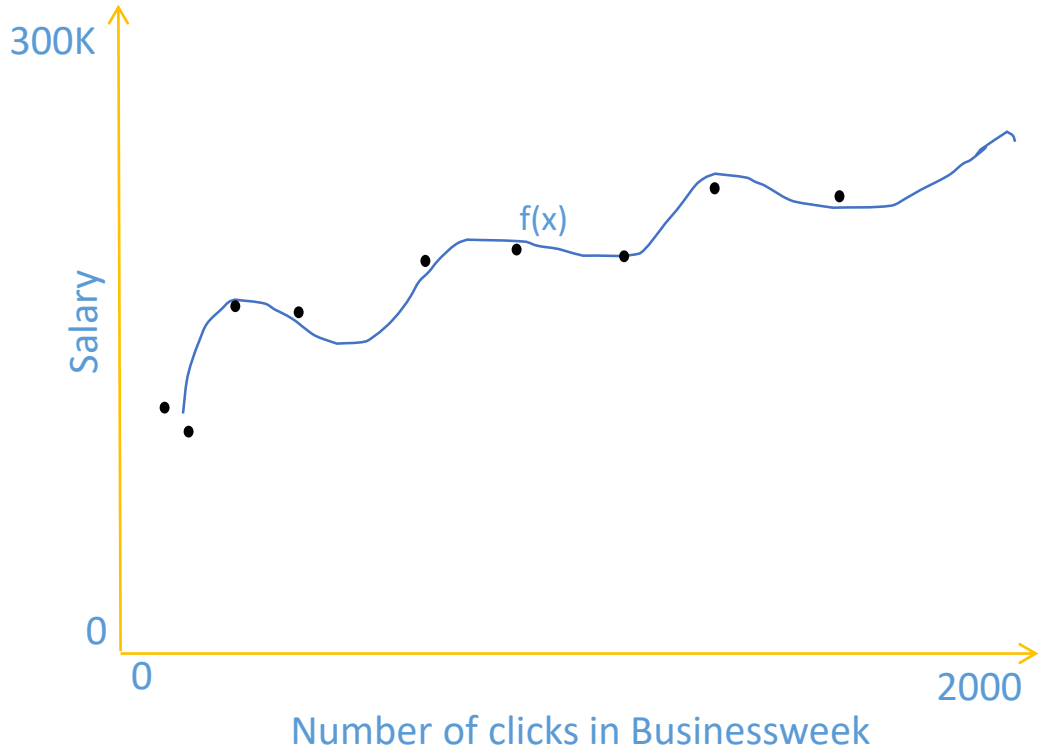
Modelling a person like this:

[5]	83
[0]	32
[1]	-10
	
Only one feature	Y

REGRESSION

f(x) = function(Number of clicks in BusinessWeek)

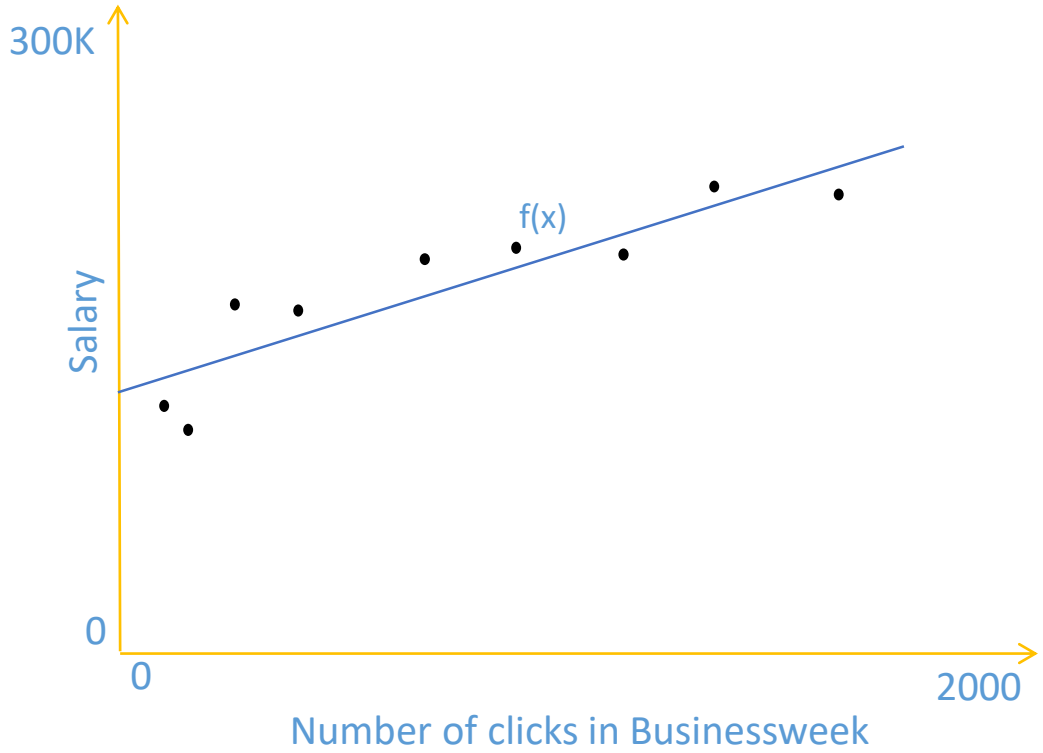
(Overfitting?)



REGRESSION

f(x) = function(Number of clicks in BusinessWeek)
= 5K Number of clicks in BusinessWeek + 100K*

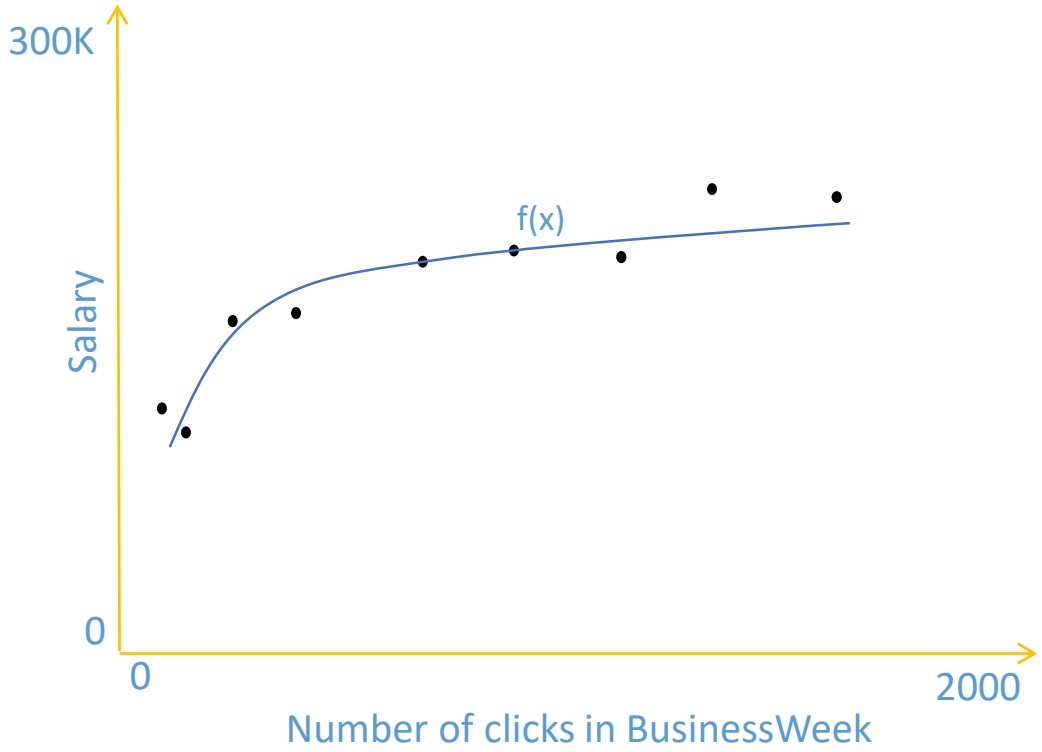
(Underfitting?)



REGRESSION

f(x) = function(Number of clicks in BusinessWeek)

(Perfect?)



REGRESSION

Estimated salary

- *f(x) = function(number of visits to a furniture shops, Number of clicks in BusinessWeek, number of different people who mails are sent per day, number of purchases above 5000€ within the last month, flights taken lately)*

Example:

*f(x) = 3 * number of visits to a furniture shops*
*+10 * Number of clicks in BusinessWeek*
*+100 * number of different people who mails are sent per day*
*+2 * number of purchases above 5000€ within the last month*
*+10 * flights taken lately*

plain concepts

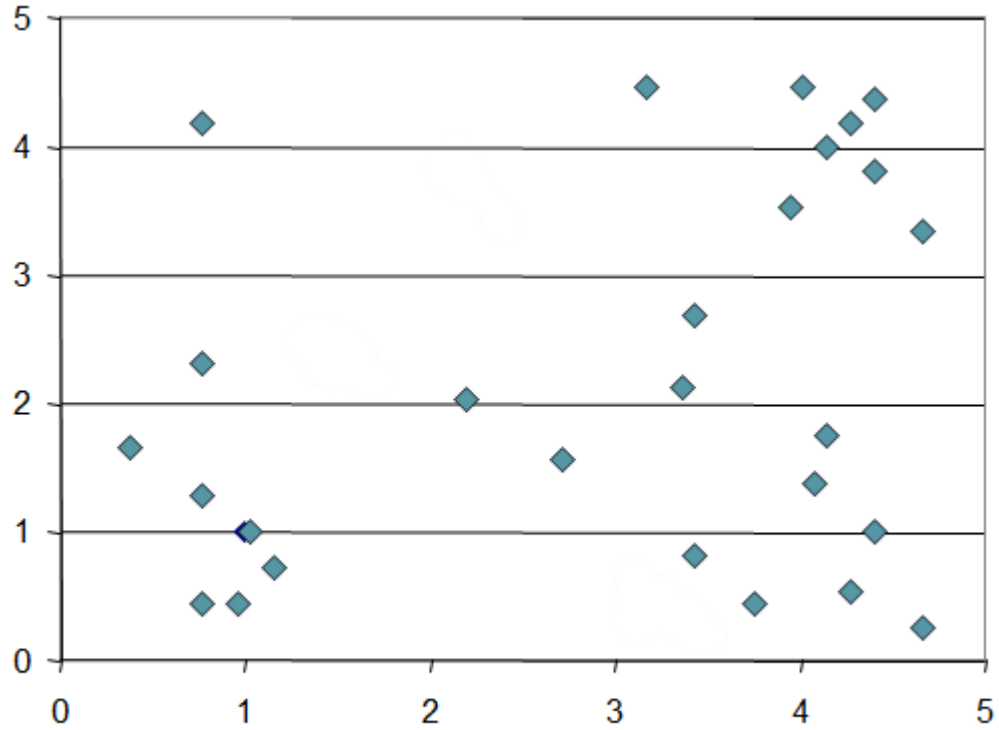
LINEAR REGRESSION

CLUSTERING

plain concepts

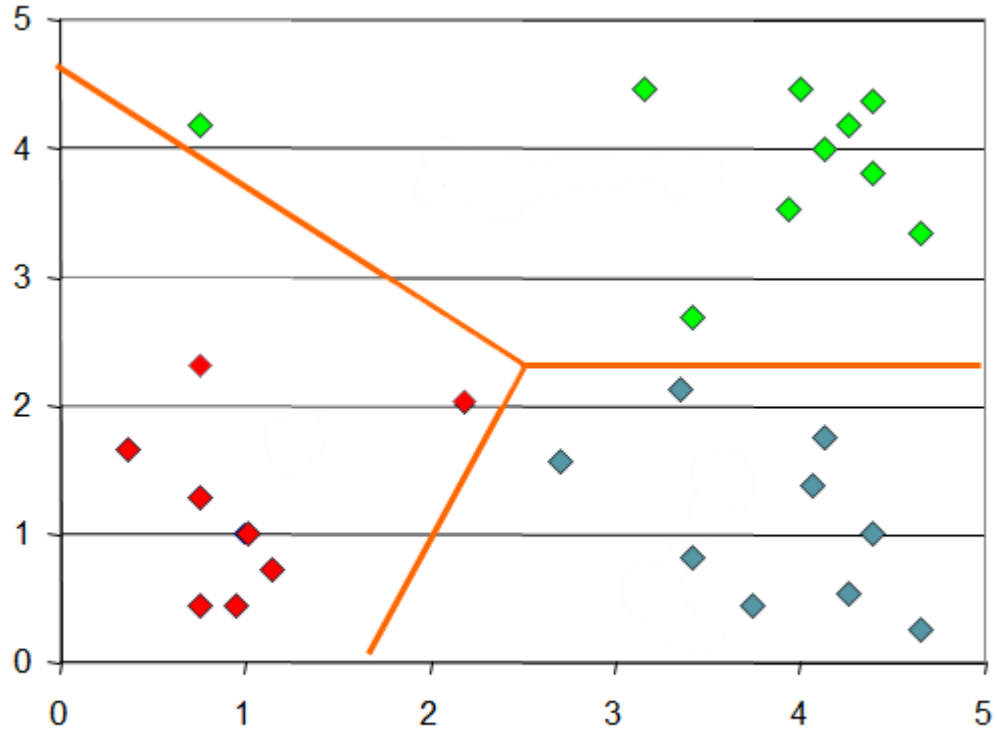
K-MEANS

- One of the most used clustering algorithms
- Groups data depending on its features
- Popular
- Fast
- Conceptually simple



K-MEANS

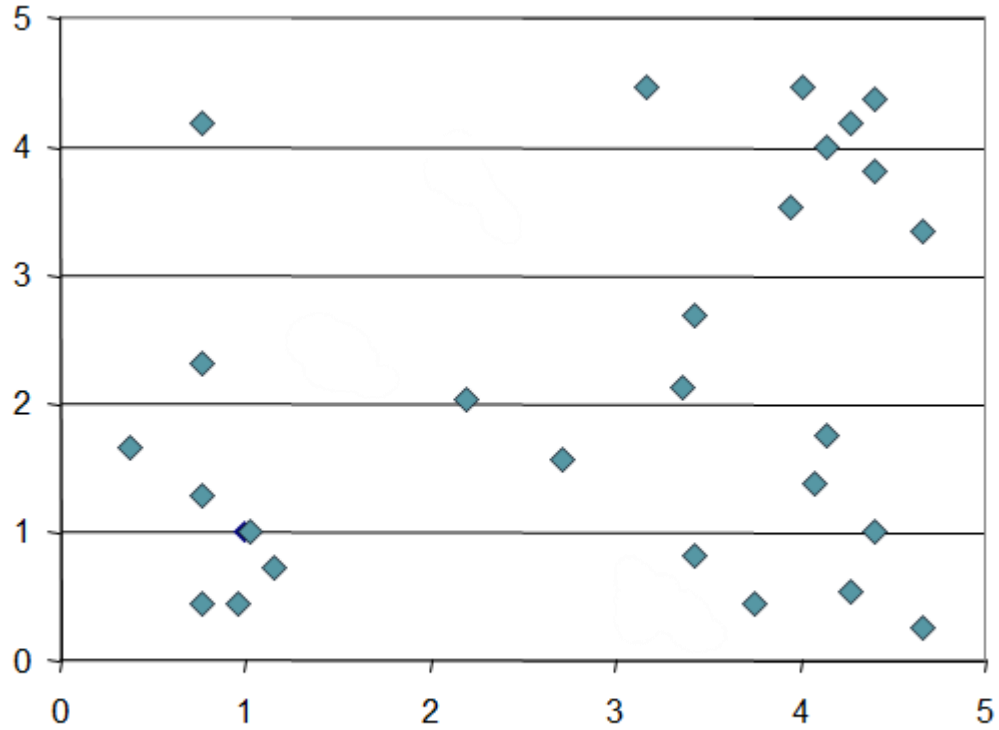
- One of the most used clustering algorithms
- Groups data depending on its features
- Popular
- Fast
- Conceptually simple



K-MEANS

Given a collection of values

```
data = lines.map(line =>
                    parseVector(line))
```

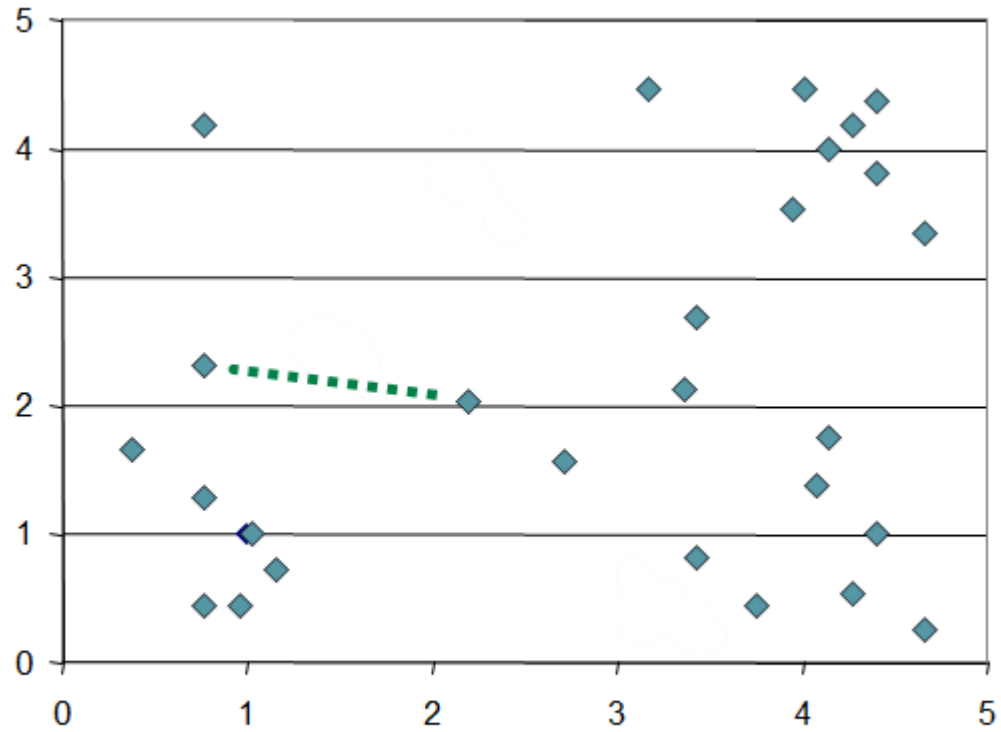


K-MEANS

We define the difference between points

as the square of the euclidean distance

```
dist = p.squaredDist(q)
```

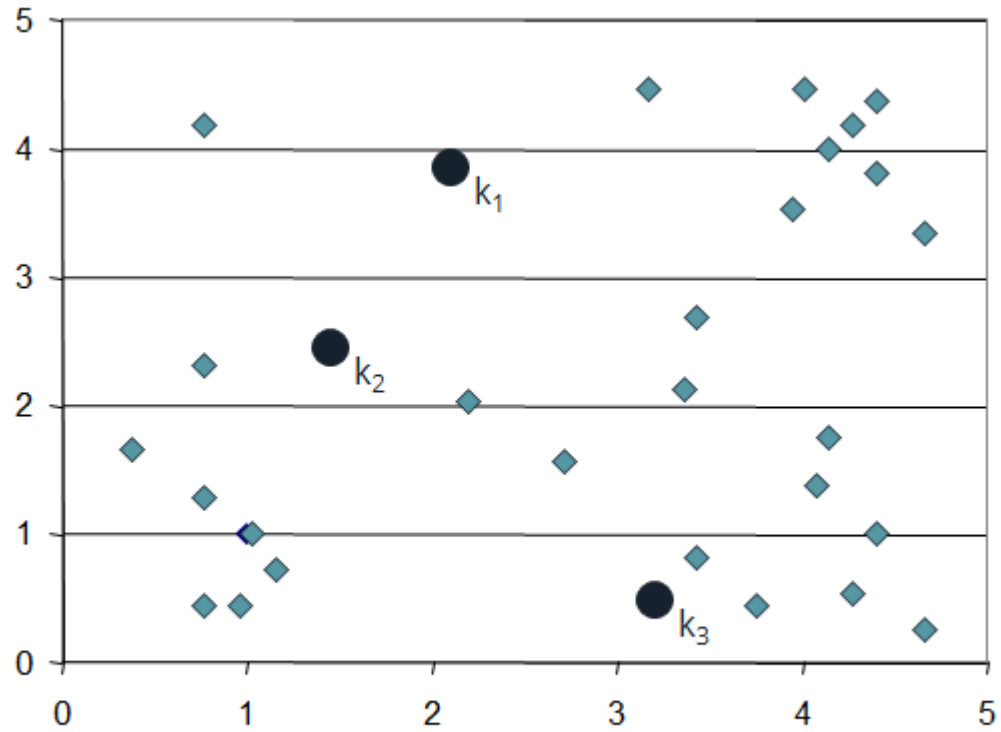


K-MEANS

With n clusters: k1, k2, k3, …, kn

We initialize the cluster centers

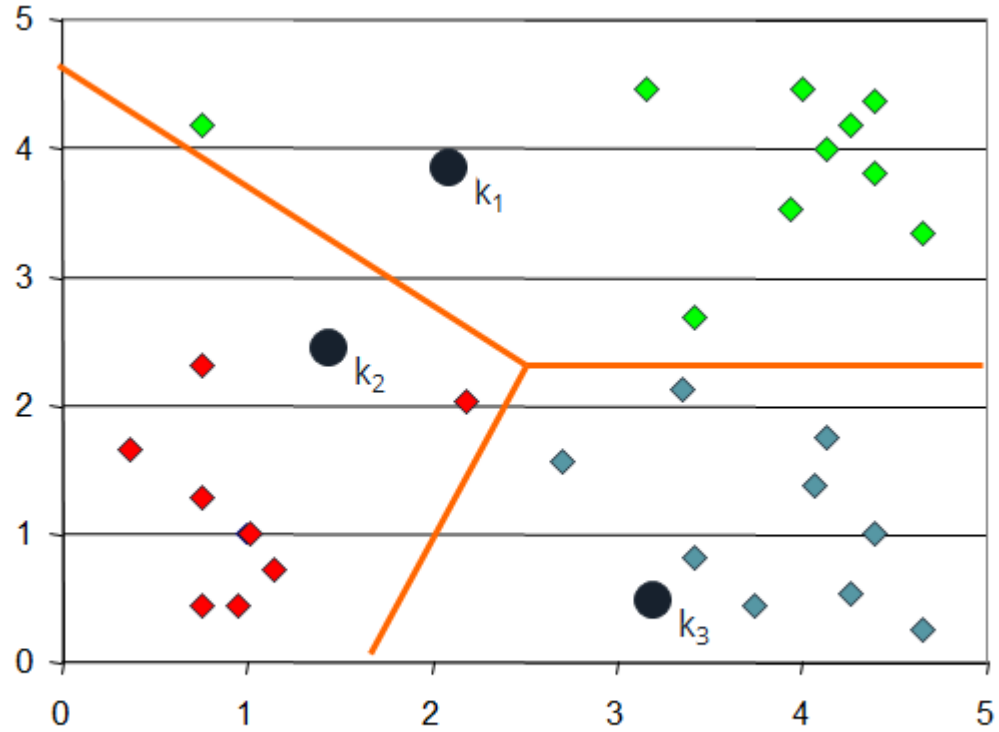
```
centers = data.takeSample(false,
K, seed)
```



K-MEANS

Assign each point with the nearest cluster centroid

```
closest = data.map(p =>
        (closestPoint(p,centers),p))
```

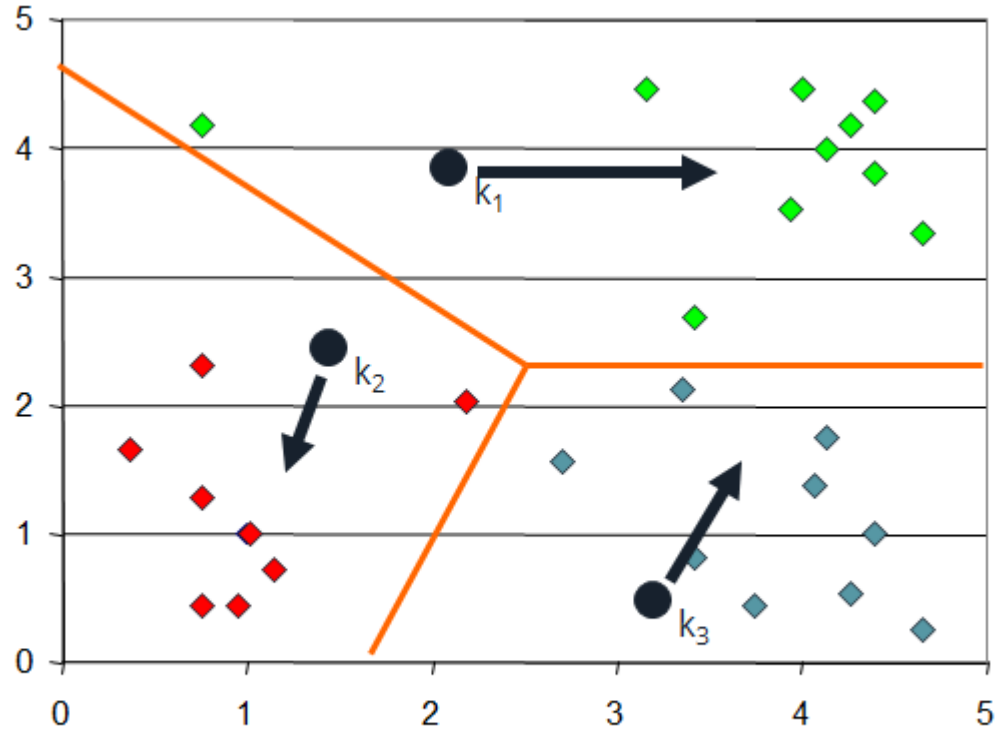


K-MEANS

We move the centroid to the average distance of its points

```
pointsGroup = closest.groupByKey()

newCenters = pointsGroup.mapValues(ps
                => average(ps))
```

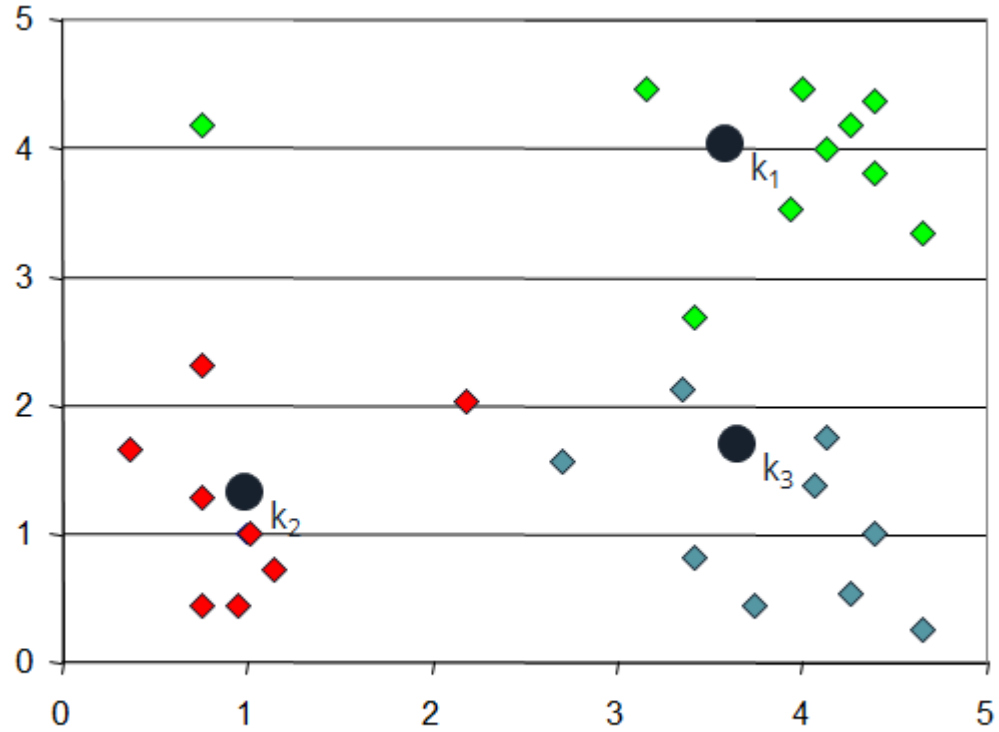


K-MEANS

We move the centroid to the average distance of its points

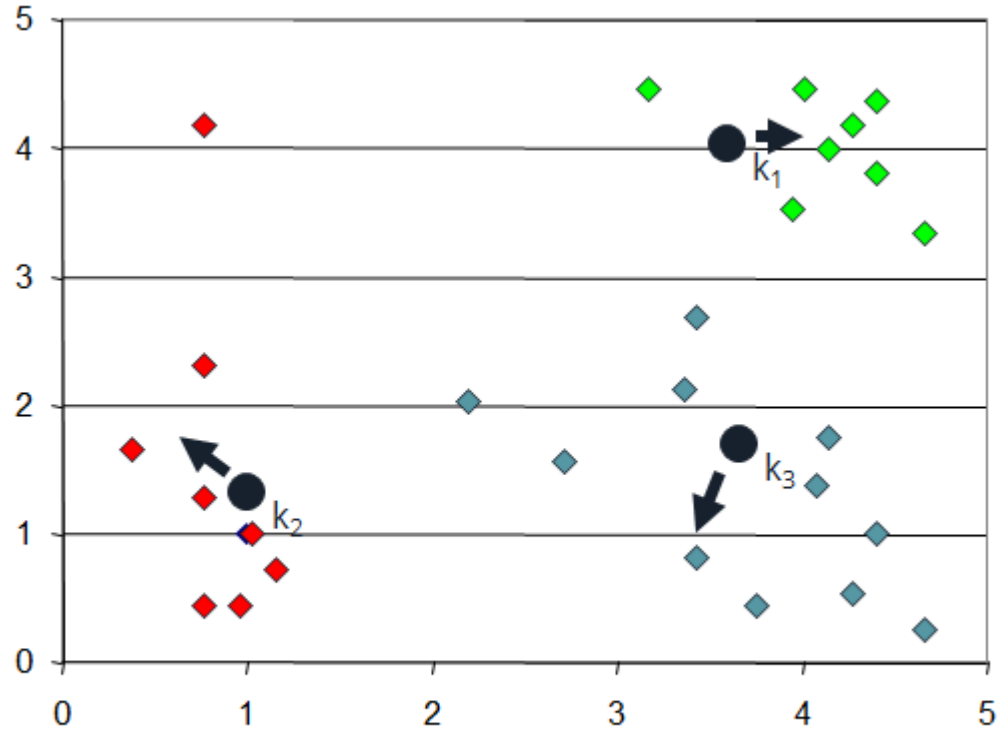
```
pointsGroup = closest.groupByKey()

newCenters = pointsGroup.mapValues(ps
                => average(ps))
```



K-MEANS

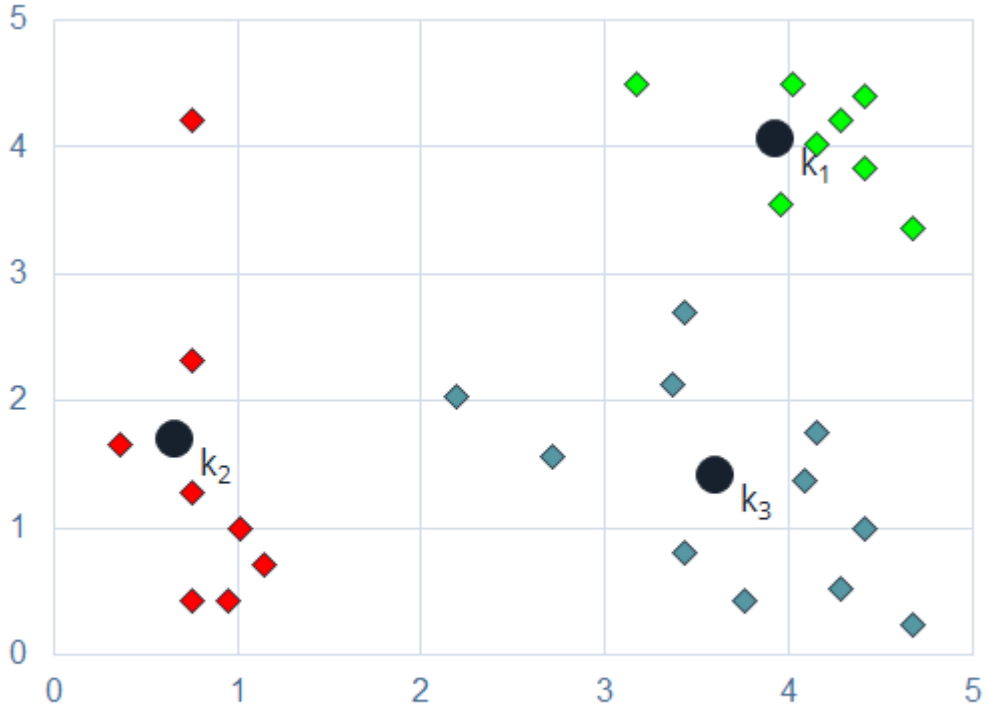
Repeat until convergence



K-MEANS

```
centers = data.takeSample(false, K, seed)
while (d > ε)
        {
                closest = data.map(p => (closestPoint(p,centers), p))
                pointsGroup =  closest.groupByKey()
                newCenters = pointsGroup.mapValues(ps => average(ps))
                d = distance(centers, newCenters)
                centers = newCenters.map(_)
        }
```

K-MEANS



plain concepts

K-MEANS

SUPPORT VECTOR MACHINES

plain concepts

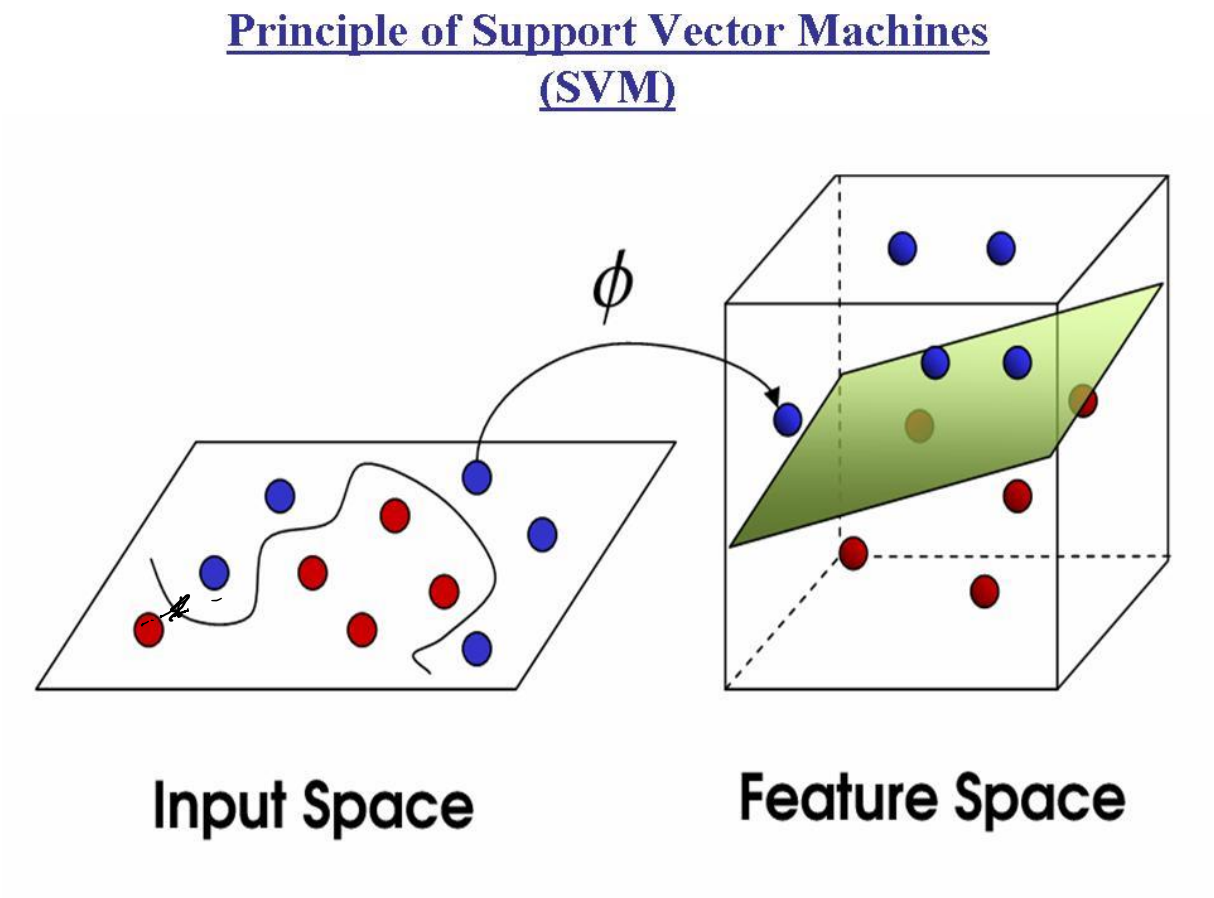
SVM

- Given a point set:
  - Subset of a bigger set (space)
  - Each point belongs to a two possible categories
- SVM builds a model able to predict the category of a new point

SVM

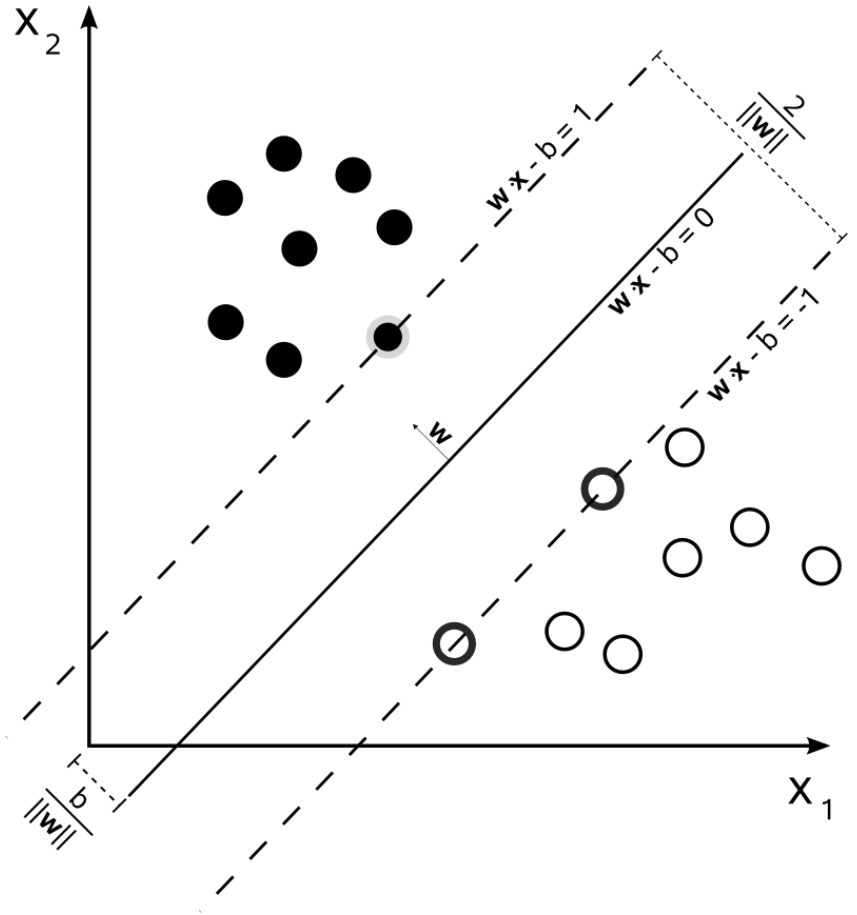
- The goal is look for the plane which divides both classes and, at the same time, maximizes the distance to the nearest point on each side of the plane
- Once the perfect plane is achieved, it is called “perceptron of optimal stability”
- Reminder, this scenario is being simplified (2 classes, 1 plane)
- In fact, when we say plane, we mean hyperplane

SVM

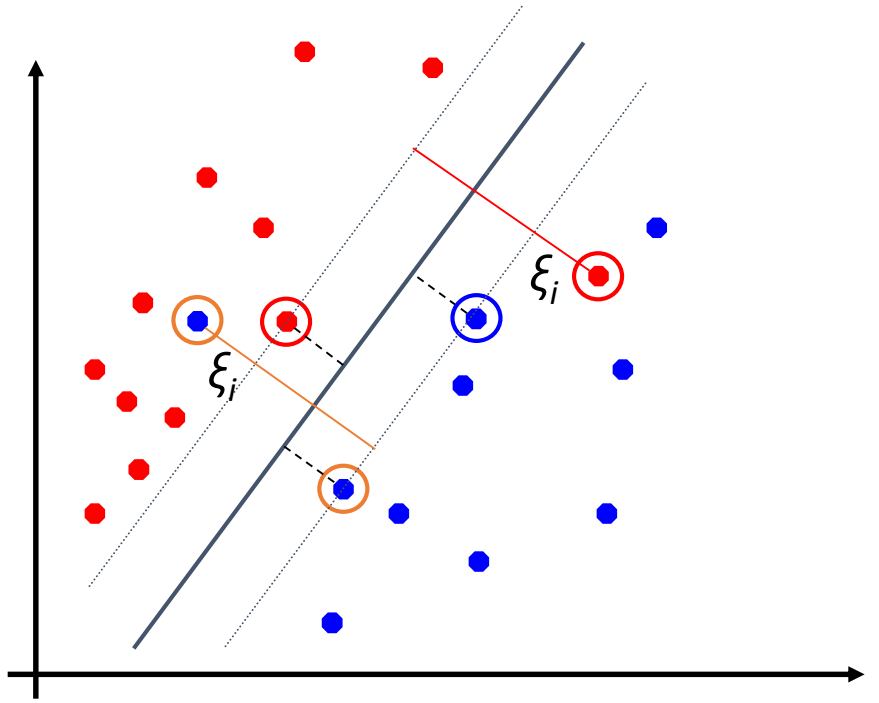

Principle of Support Vector Machines
(SVM)
ϕ
Input Space
Feature Space

CLASSIFICATION MARGIN

- Hyperplane divides de subsets
- Support vectors are defined by the nearest points to the hyperplane
- The classification margin is the distance between hyperplane and support vectors

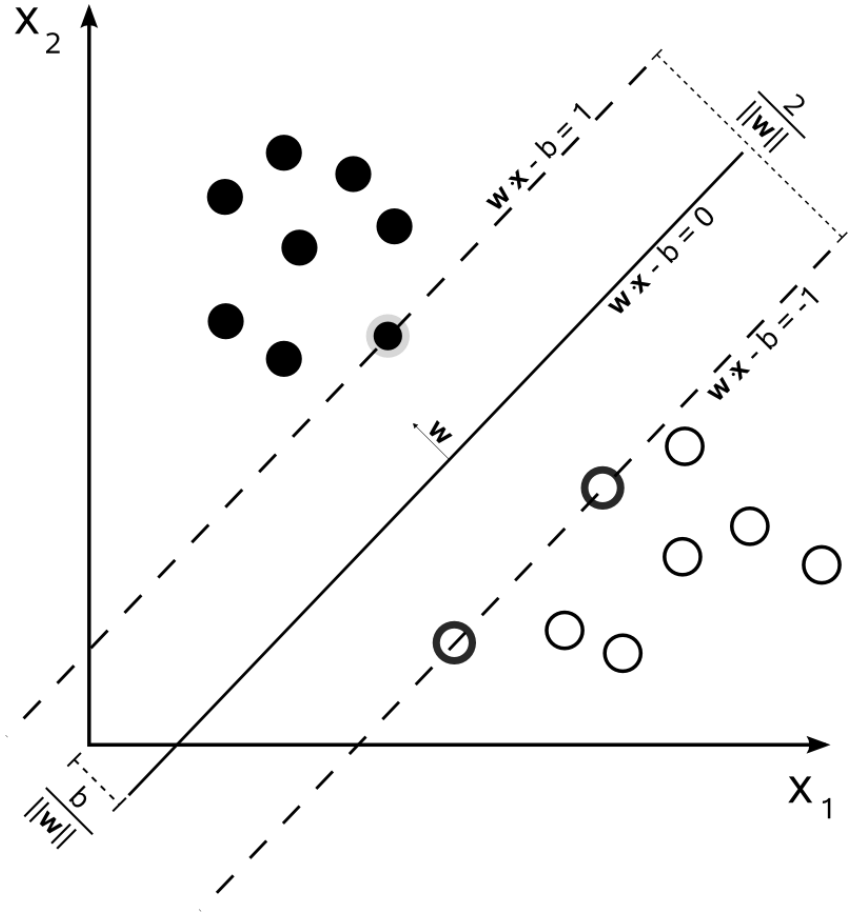


ERROR



WHY MAXIMIZE THE MARGIN?

- Points near the decision surface are inaccurate when predicting
- The bigger the margin, the more accurate the predictions



SVM ADVANTAGES

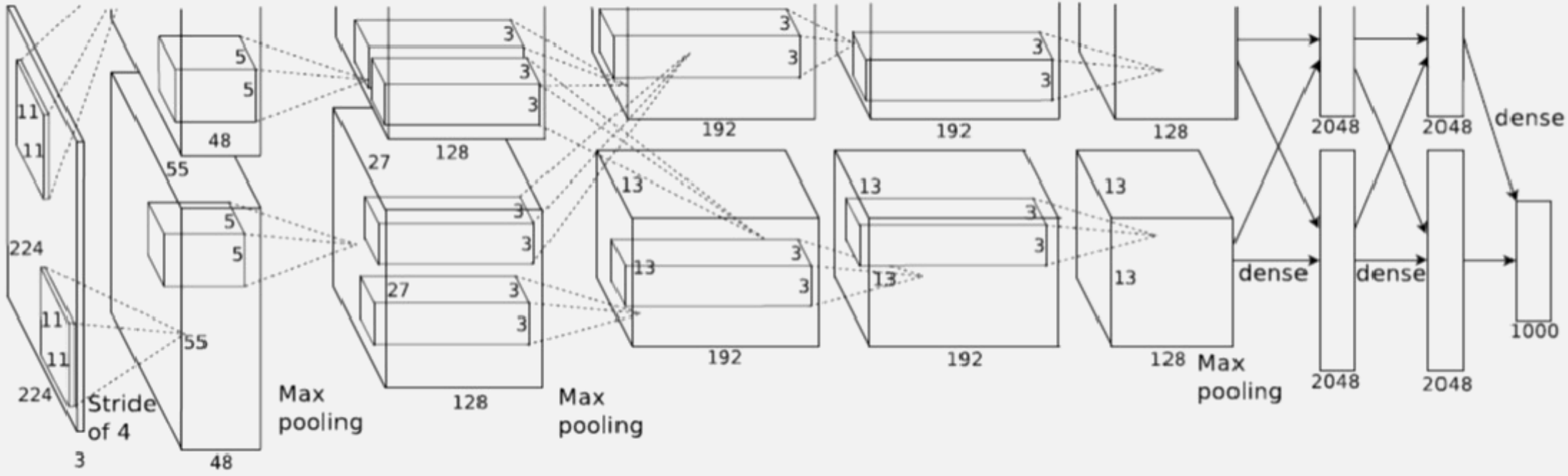
- Maximizing the margin it reduces overfitting
- Efficient: O(n3*m)
- Used for linear and non linear scenarios

DEEP LEARNING

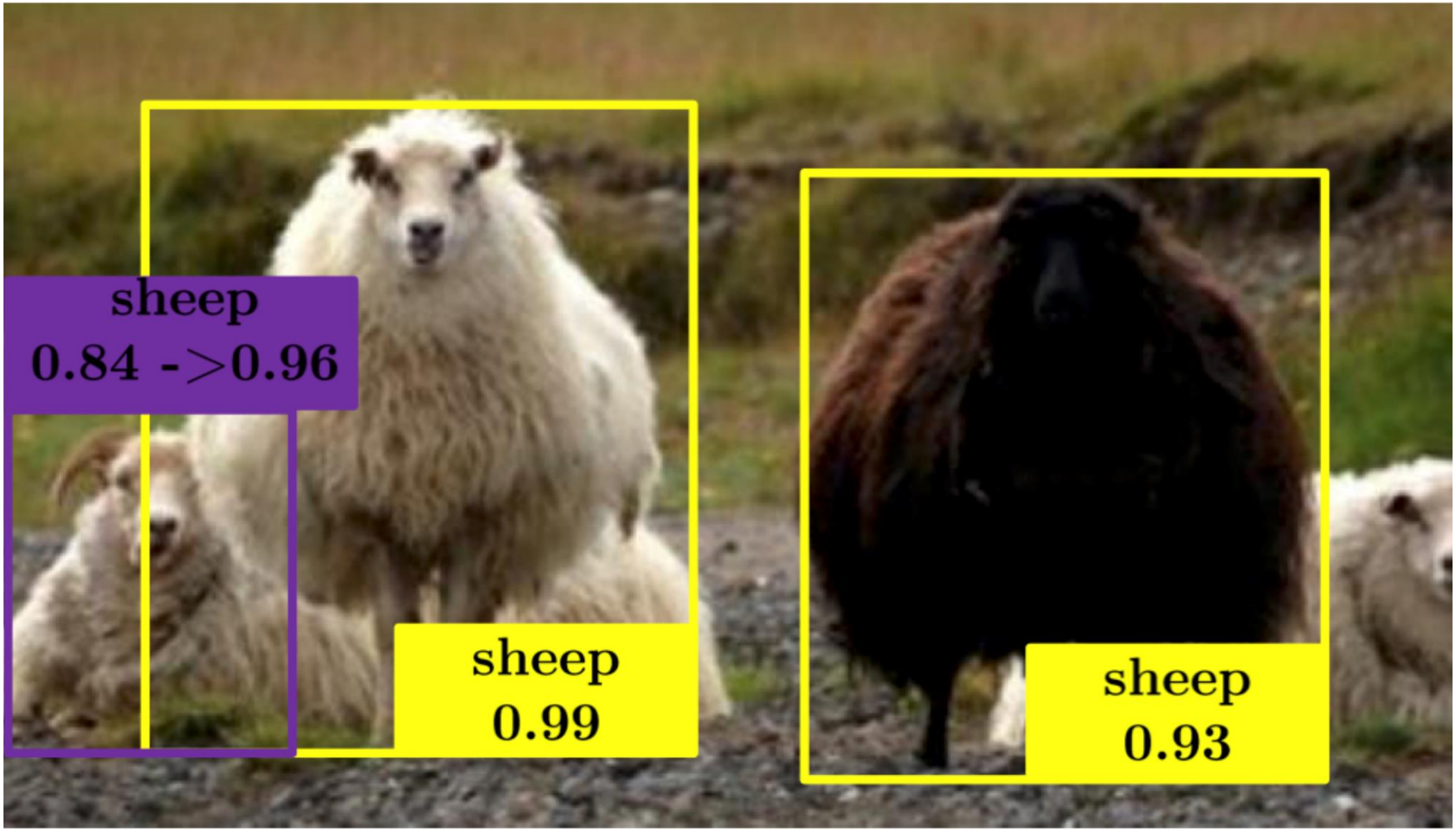
plain concepts

Who are you?

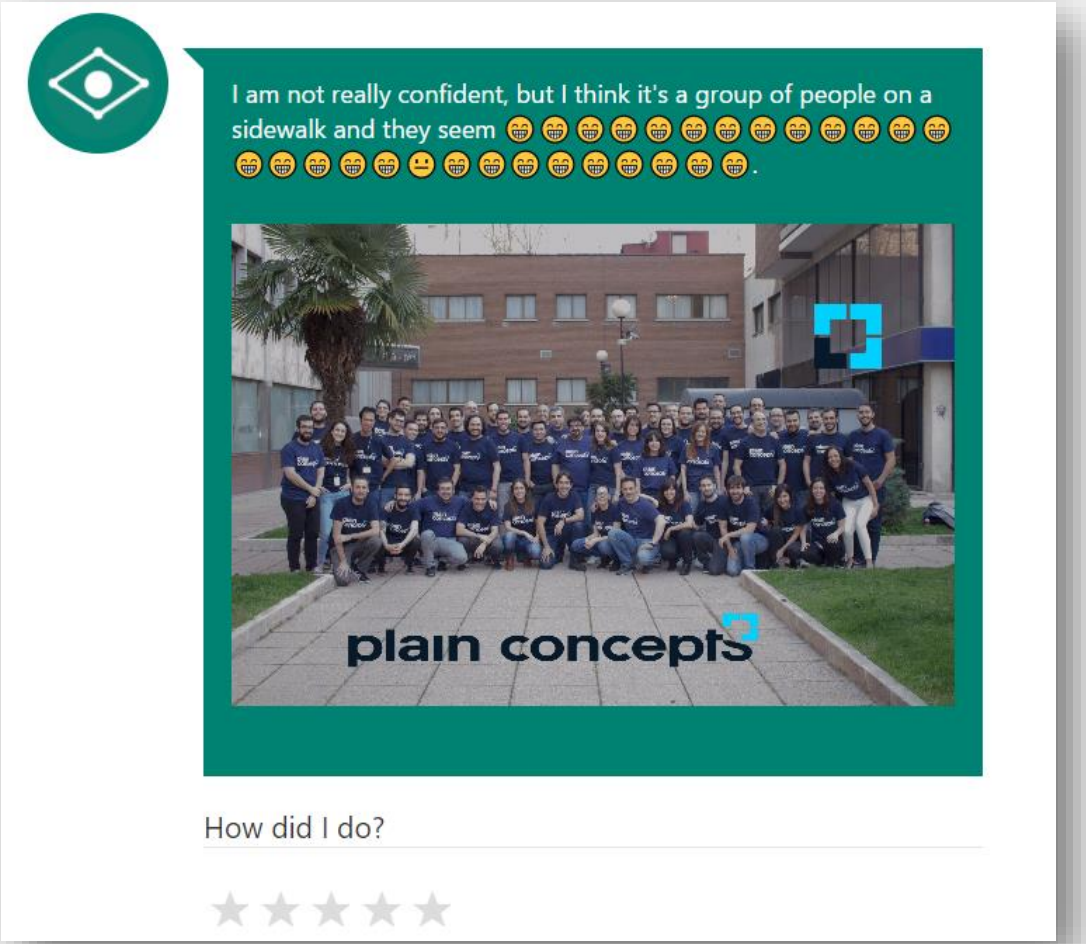
WHAT IS THIS THING?



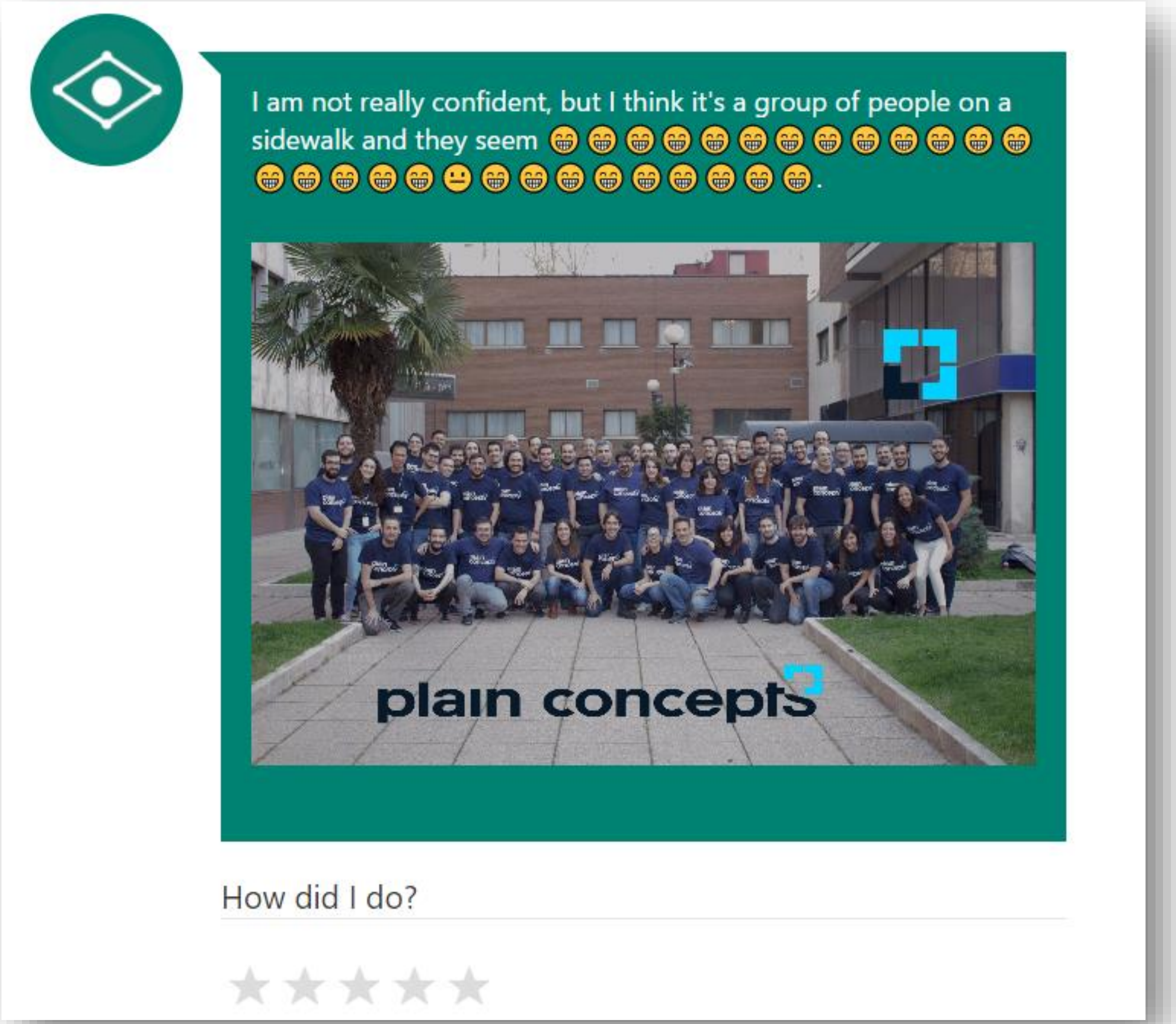
WHAT IS THIS USEFUL FOR?


sheep
0.84 ->0.96
sheep
0.99
sheep
0.93

WHEN NOT TO USE THE COGNITIVE SERVICES?



- Vision
- Speech
- Language
- Knowledge
- Search



Vision

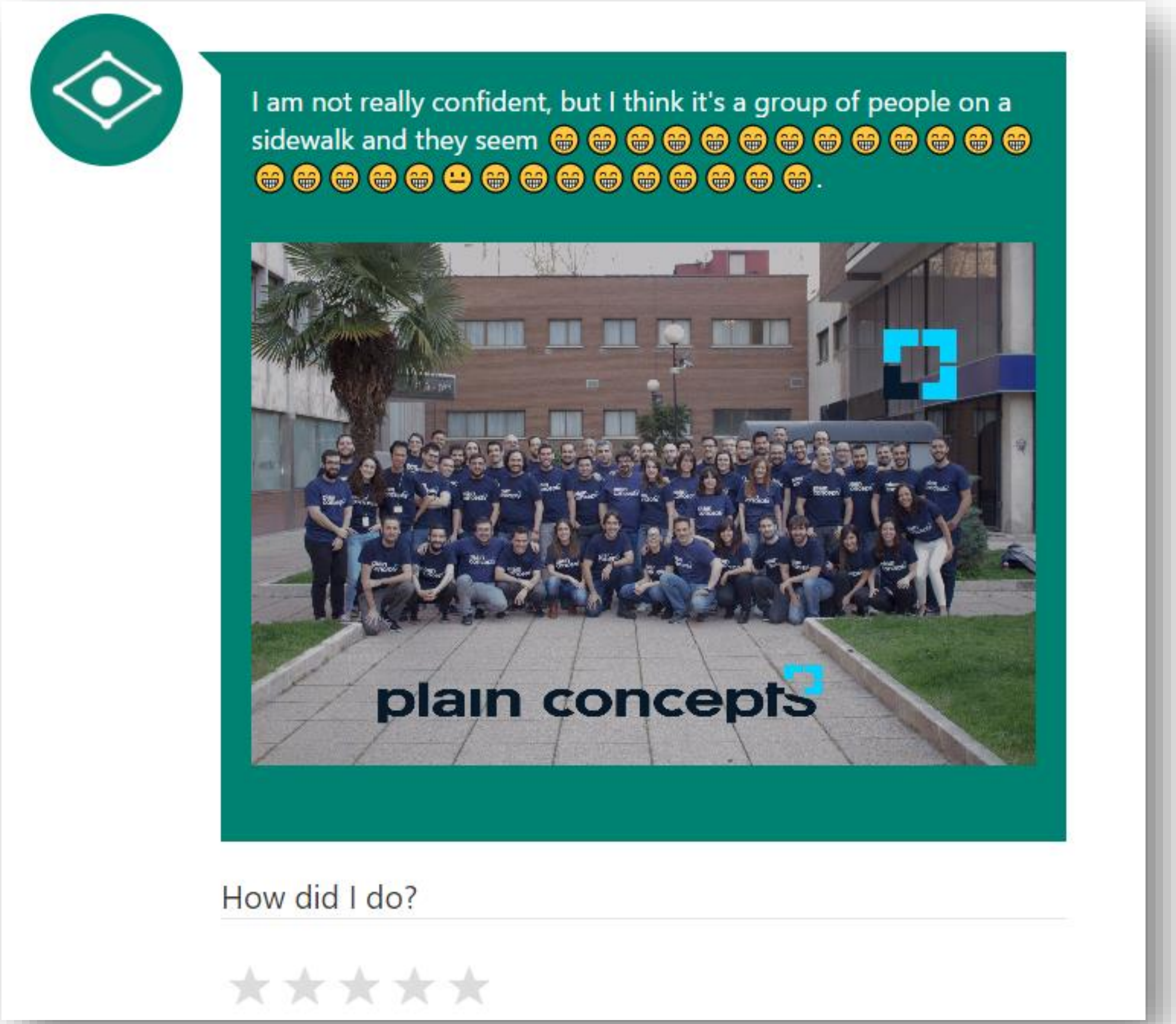
- Computer Vision
- Content Moderator
- Emotion API
- Face API
-

Speech

Language

Knowledge

Search



Vision

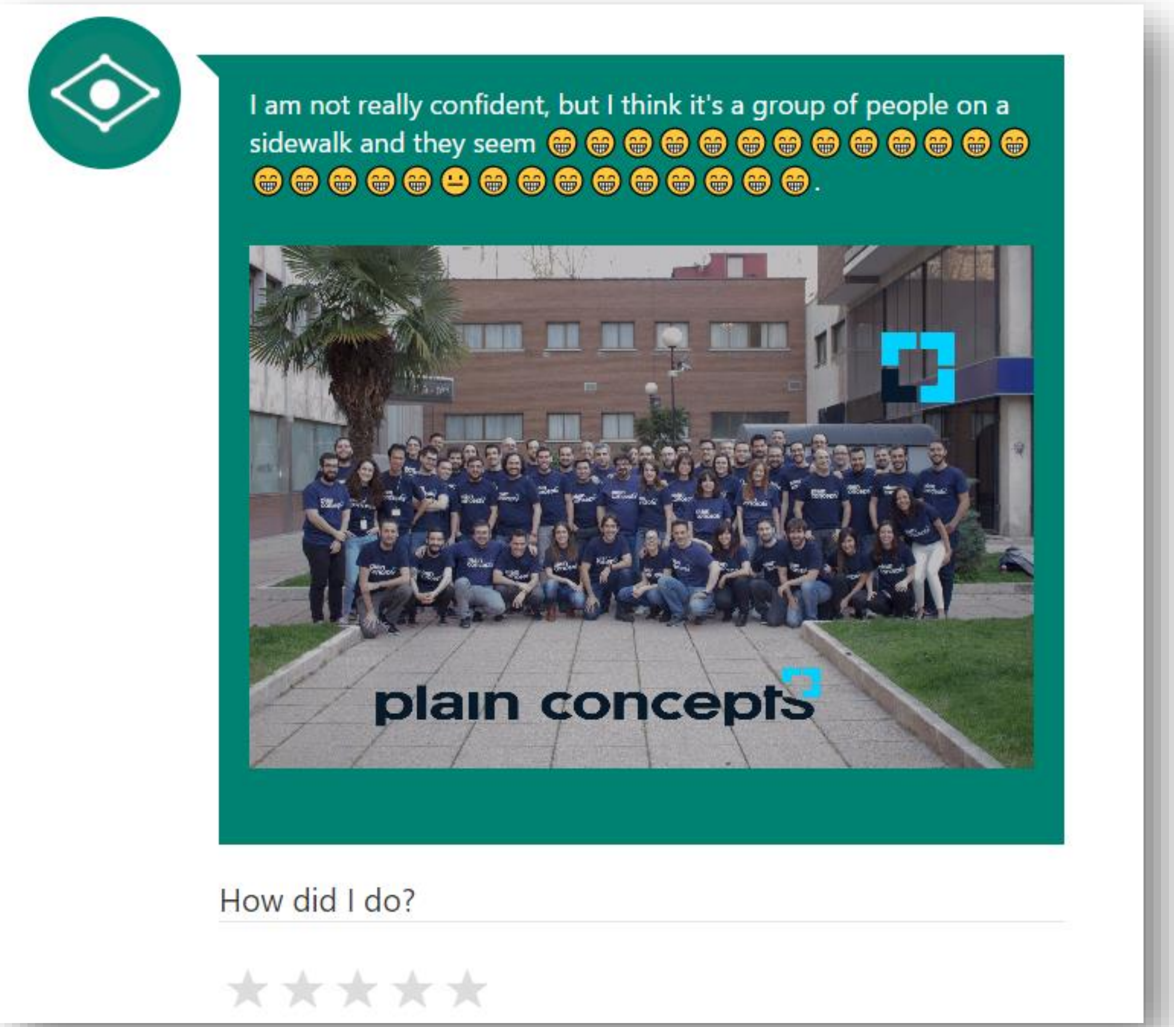
Speech

- Bing Speech API
- Custom Speech Service
- Speaker Recognition Service

Language

Knowledge

Search



Vision

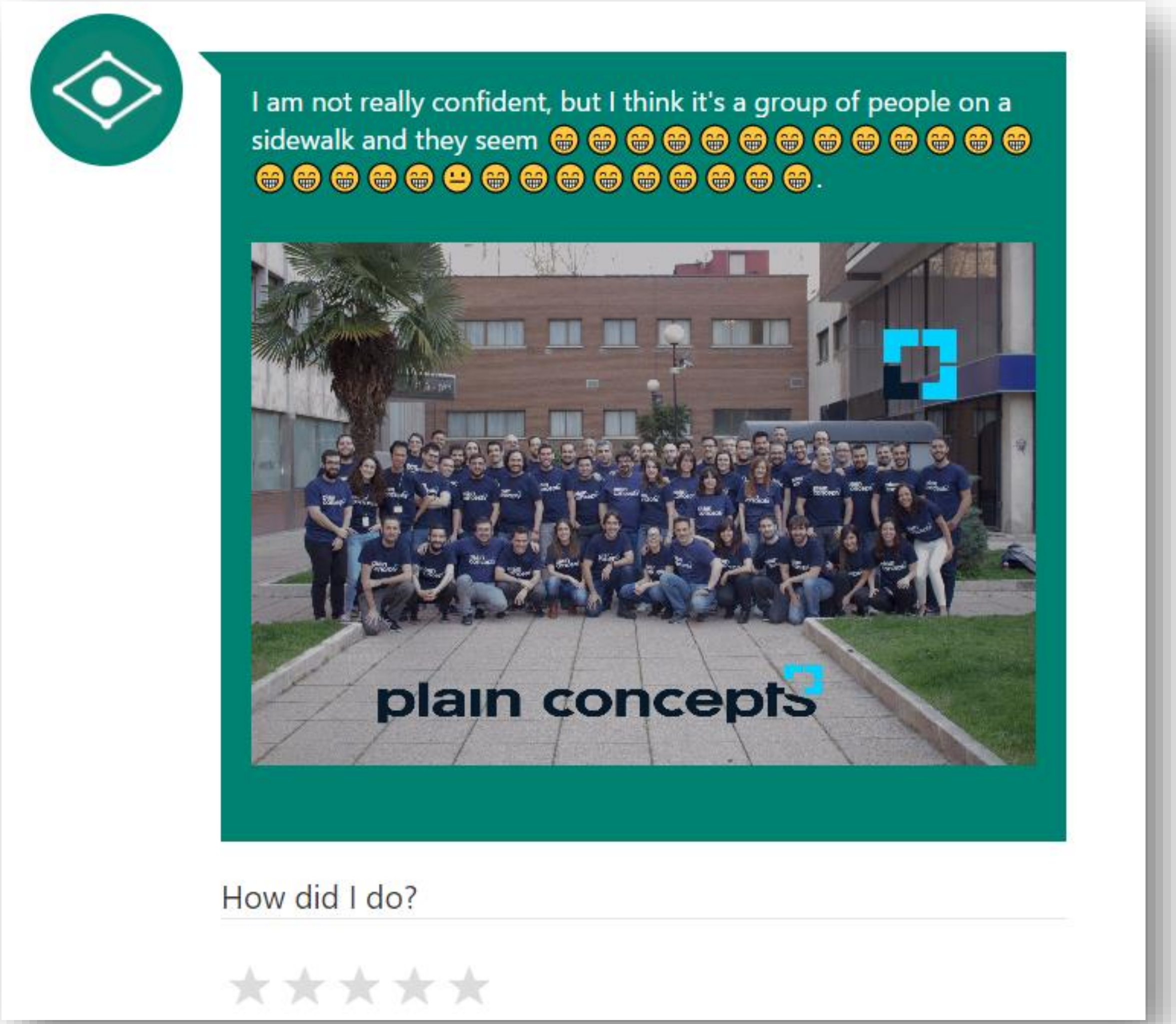
Speech

Language

- Spell Check API
- LUIS
- Linguistic Analysis
- Translator API

Knowledge

Search



Vision

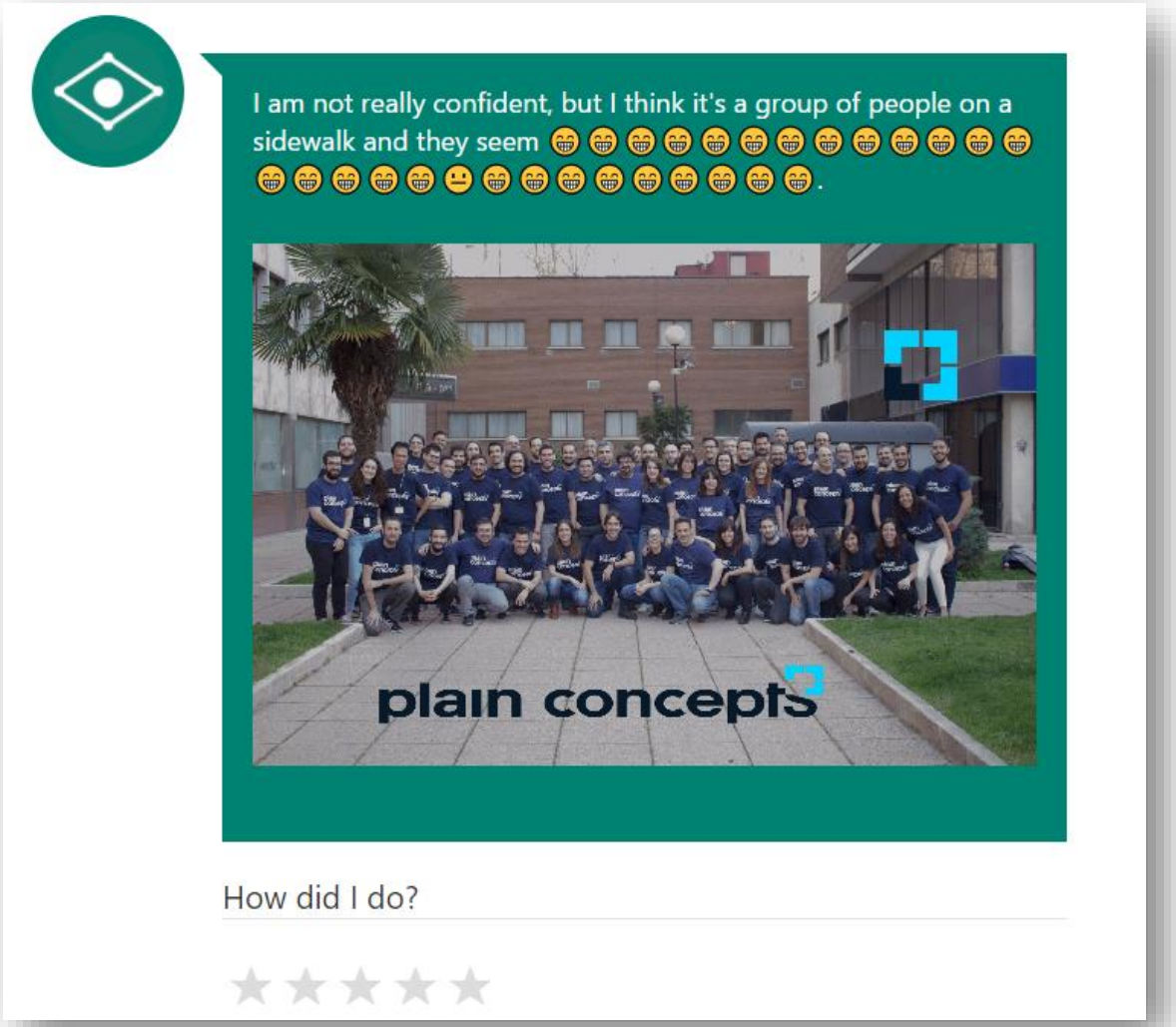
Speech

Language

Knowledge

- Entity Linking Service
- Academic Knowledge Service
- Q&A Maker
-

Search



Vision

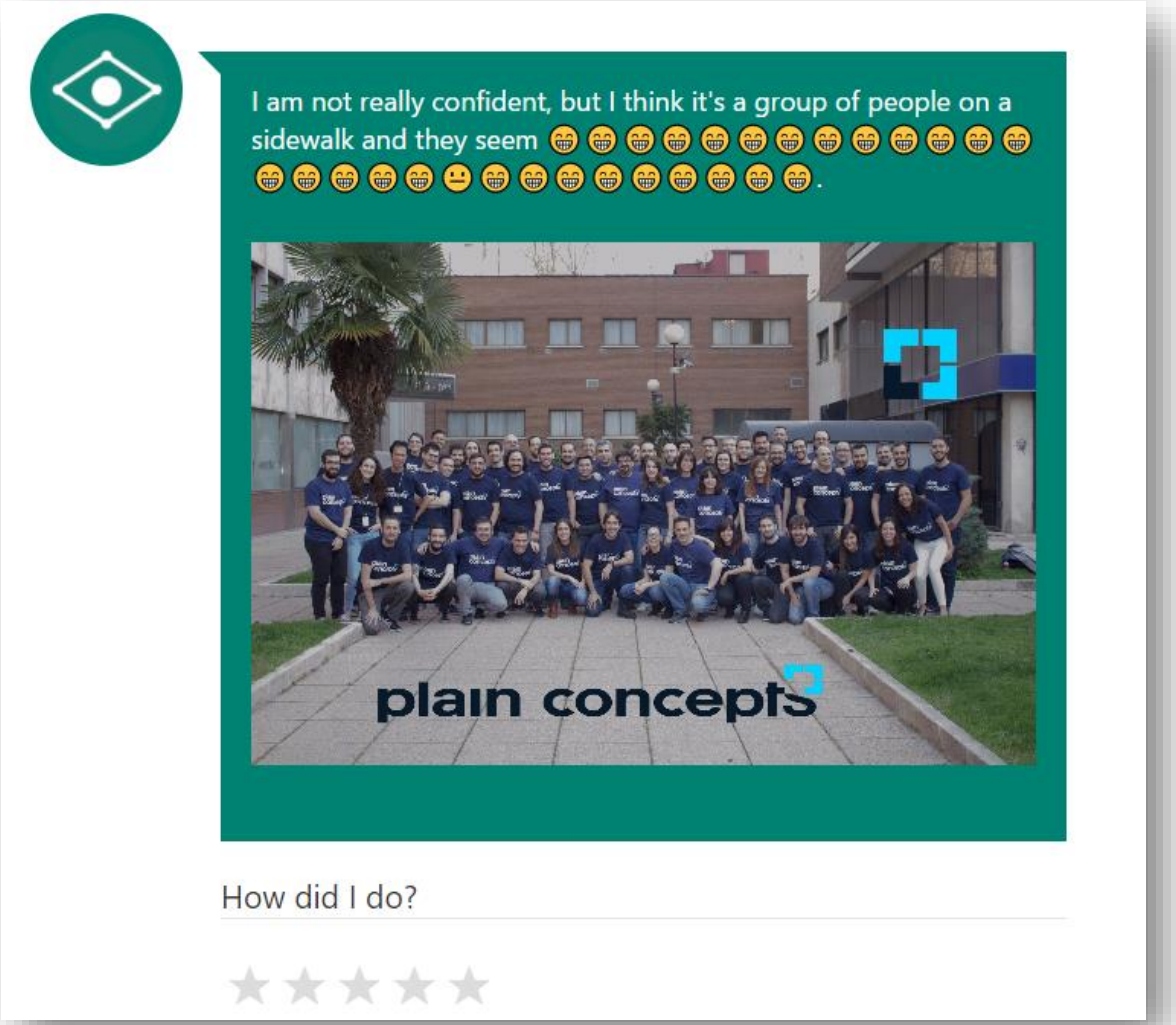
Speech

Language

Knowledge

Search

- Autosuggest API
- Image Search API
- News Search API
- …



WHEN NOT TO USE THE COGNITIVE SERVICES?

WHEN NOT TO USE THE COGNITIVE SERVICES?

WHEN NOT TO USE THE COGNITIVE SERVICES?

WHEN NOT TO USE THE COGNITIVE SERVICES?

- Vienen pre-entrenados



DEEP LEARNING WITH CNTK

plain concepts

DEEP LEARNING

Machine Learning

Based on Neural Networks

More than one hidden layer (*Deep Models*)

CNTK

- Don’t ask me for the acrnoym. Really. Don’t.

Why?

- Faster model development
- Handles múltiple CPUs and GPUs
- Multi-platform
- Internally used at Microsoft. For real.

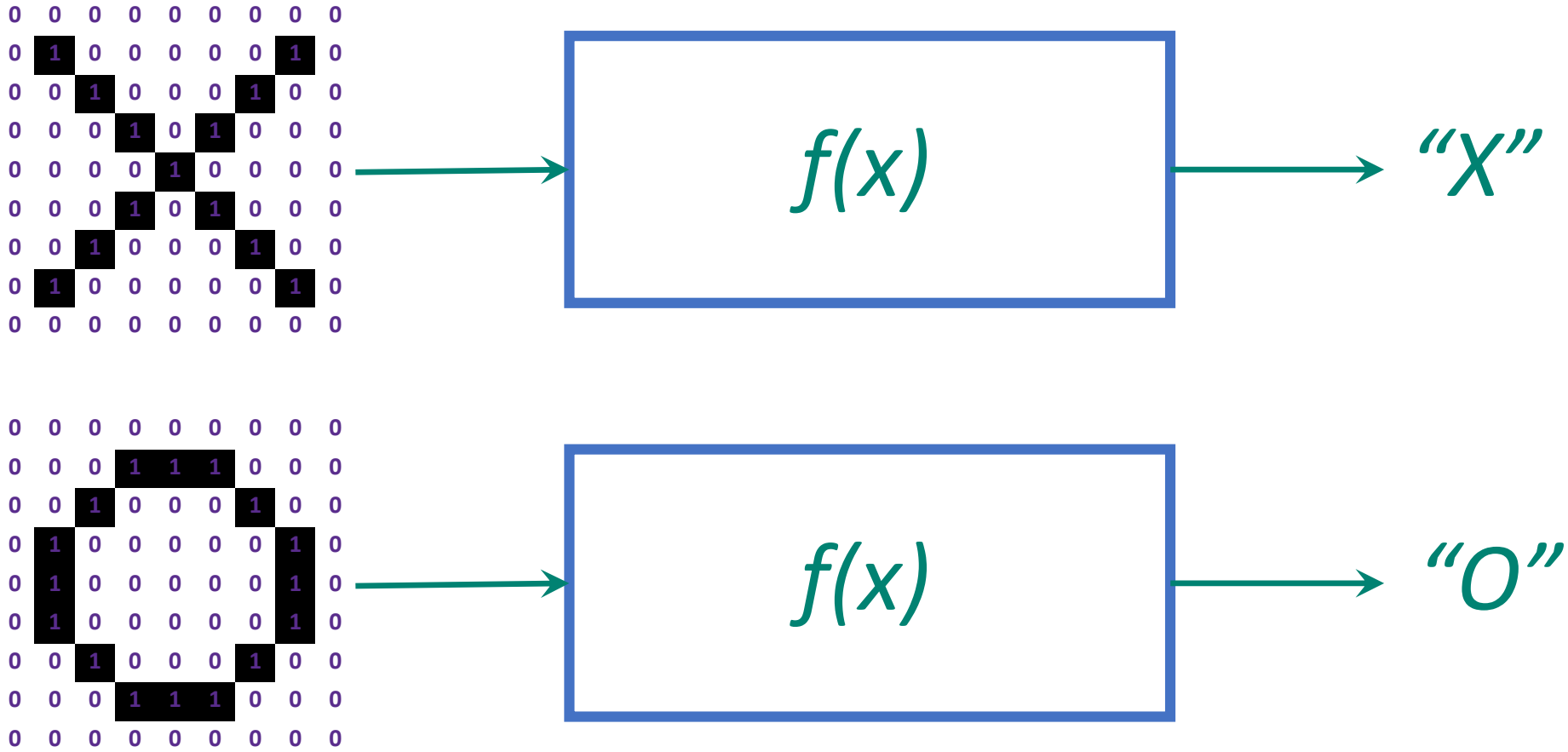
CNTK

- Don’t try. I still don’t want to talk about the meaning of the acronym.

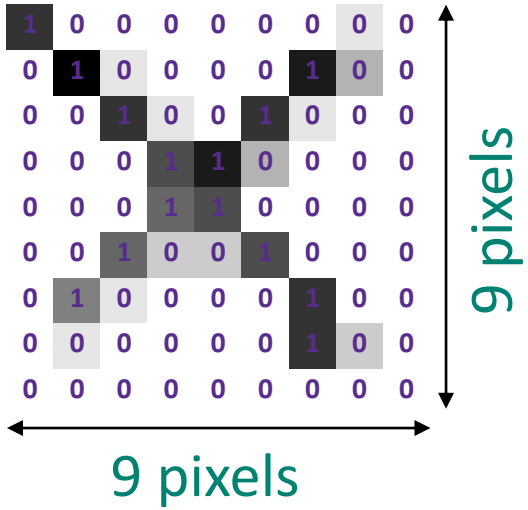
- It is an acronym of something ☺
- Deep Learning framework, open-source with external contributions by MIT and Stanford, among others
- Allows for the expression of complex neural netoworks in a straightforward way, and performs all the steps in the lifecycle: from training to evaluation.
- Development in BrainScript, Python (.NET in progress)

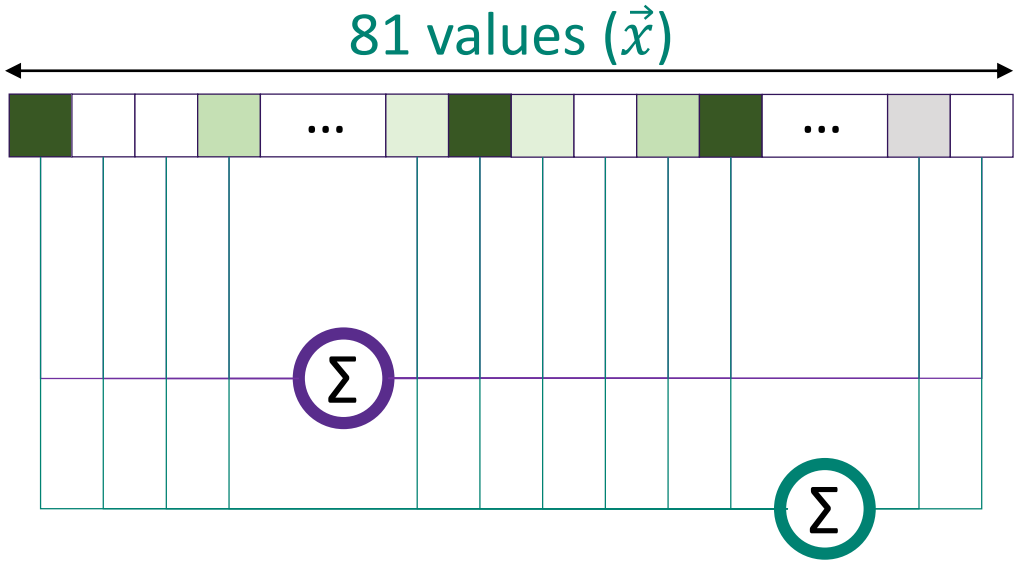
SHALL WE PLAY A LITTLE GAME?

A SIMPLE EXAMPLE

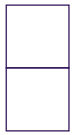


A SIMPLE EXAMPLE





Bias ($\vec{b}$)

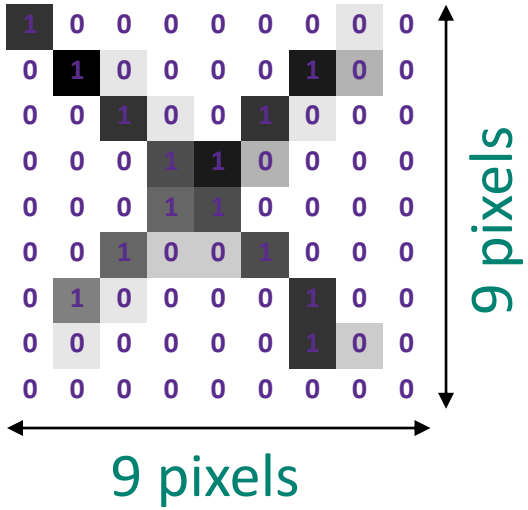


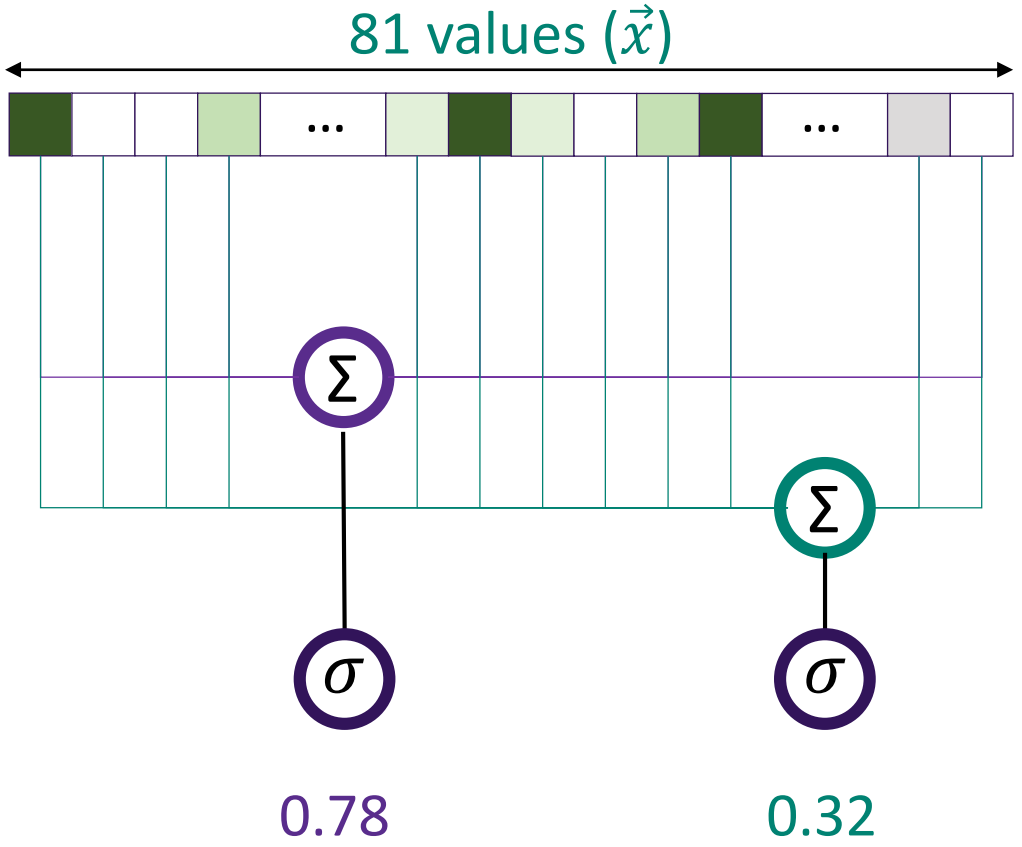
Σ *Sum (weights x pixels) =* $\overrightarrow{w_x} \cdot \vec{x}$

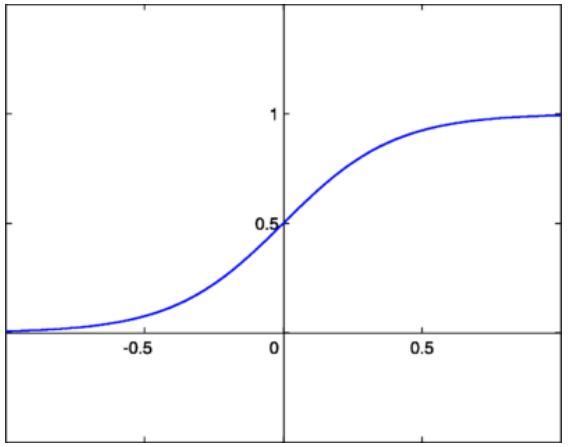
Σ *Sum (weights x pixels) =* $\overrightarrow{w_0} \cdot \vec{x}$

$$z = \vec{w} \cdot \vec{x} + b$$

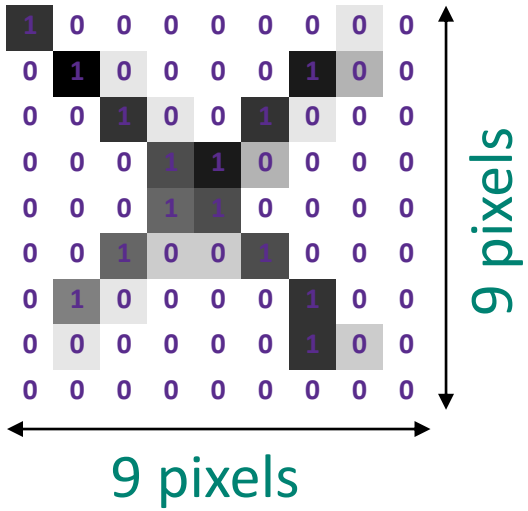
SIGMOID

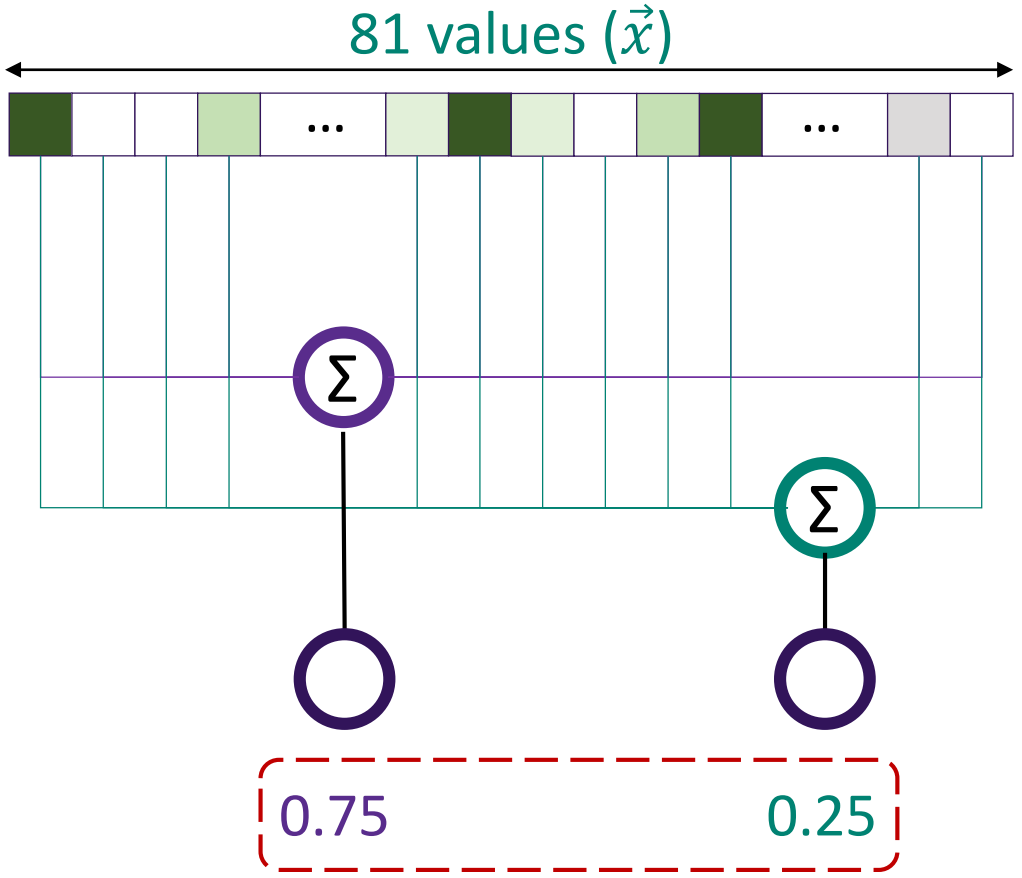






SOFTMAX

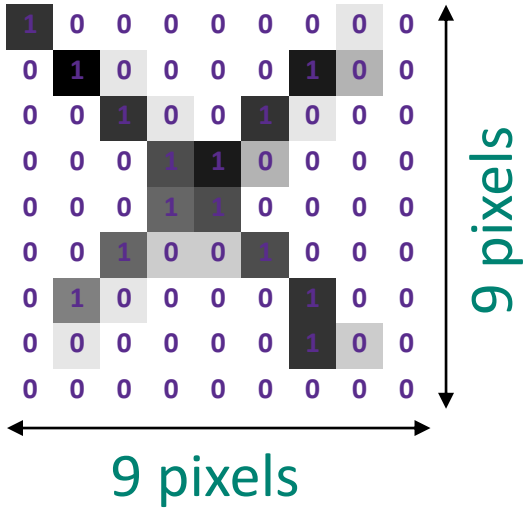




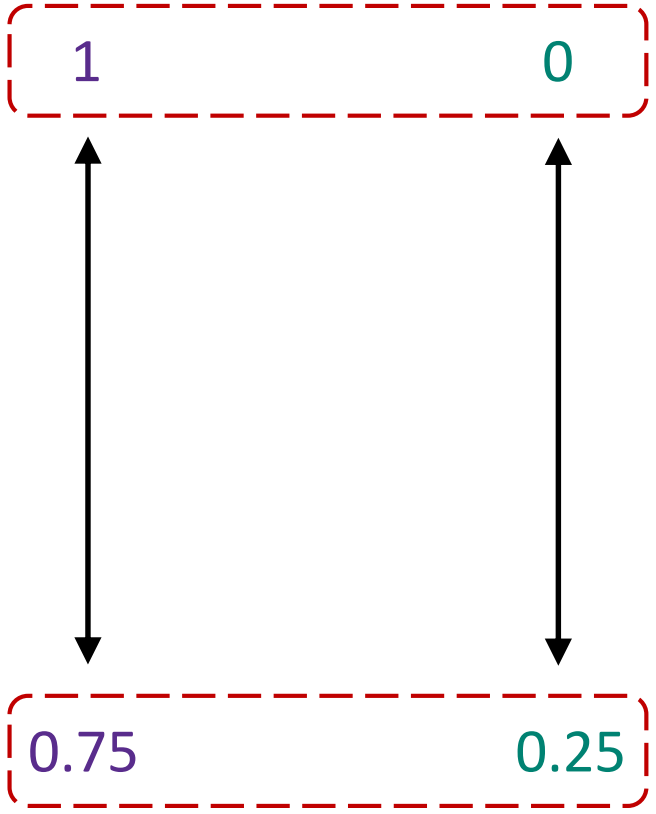
$$S_{(y_i)} = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

These are actual probabilities

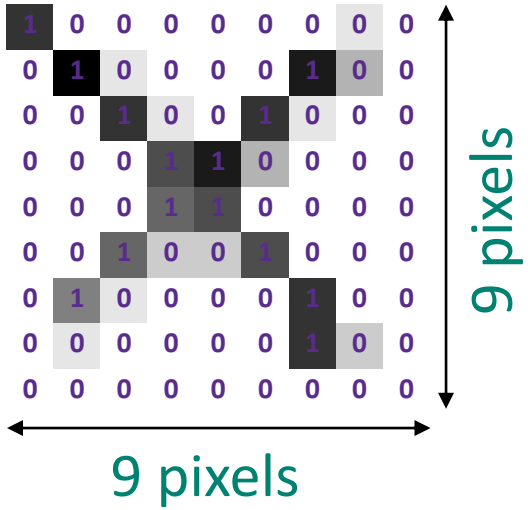
LOSS FUNCTION



One-hot encoding



LOSS FUNCTION



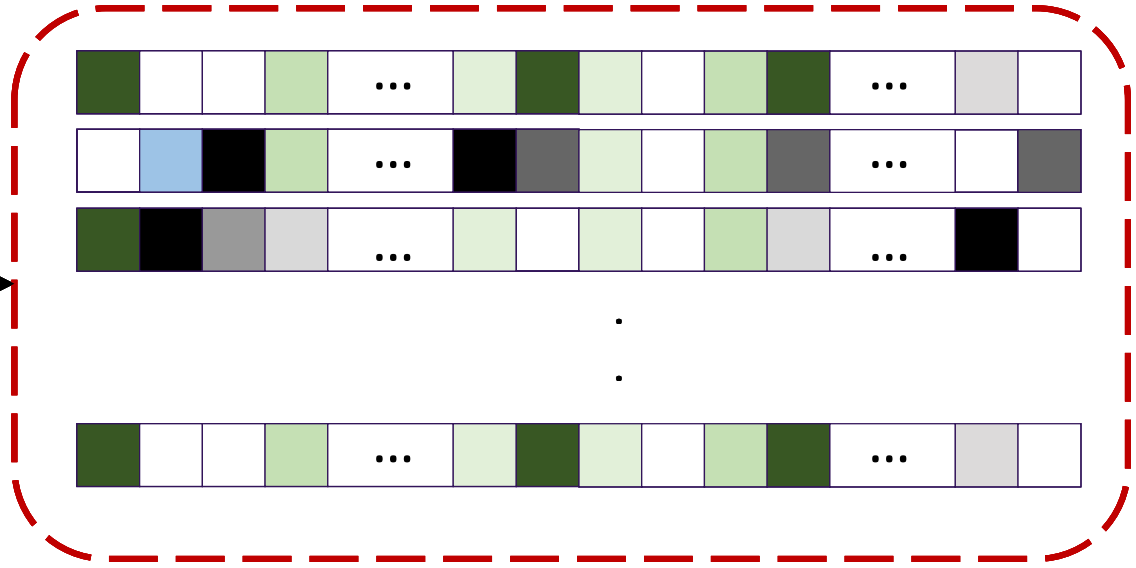
1 0

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}\left(y_j - p_j\right)^2$$

0.75 0.25

TRAINING

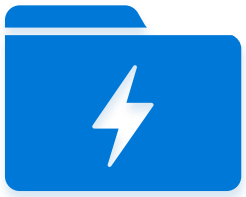
x = vector of pixels (input)



z = times(x, W) + b

cross_entropy_with_softmax(z, y)

classification_error(z, y)



Training Data

(1, 0)

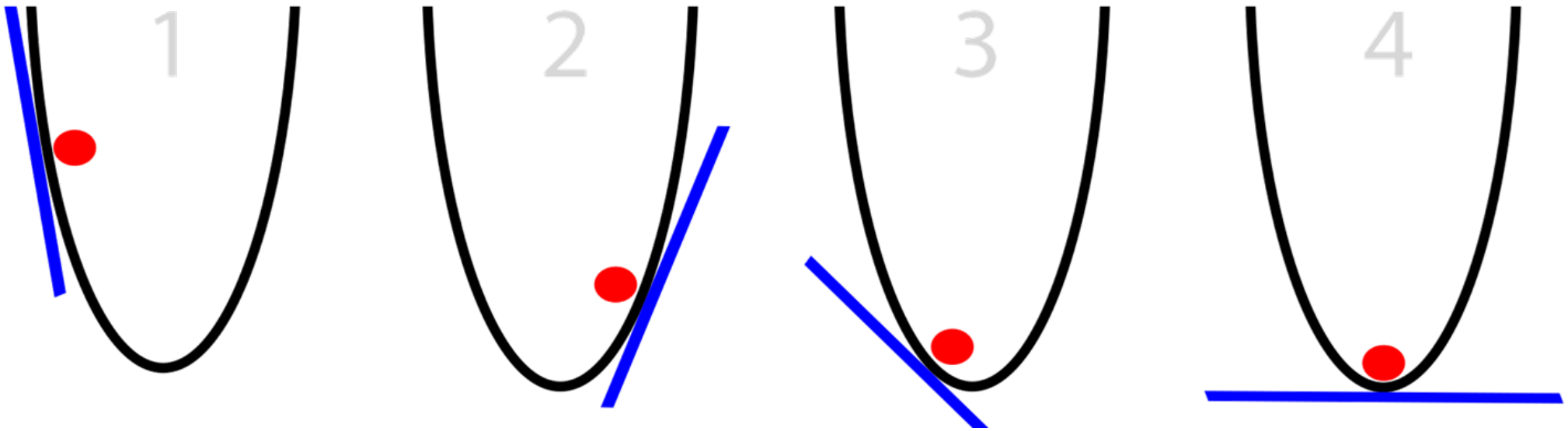
(0, 1)

(1, 0)

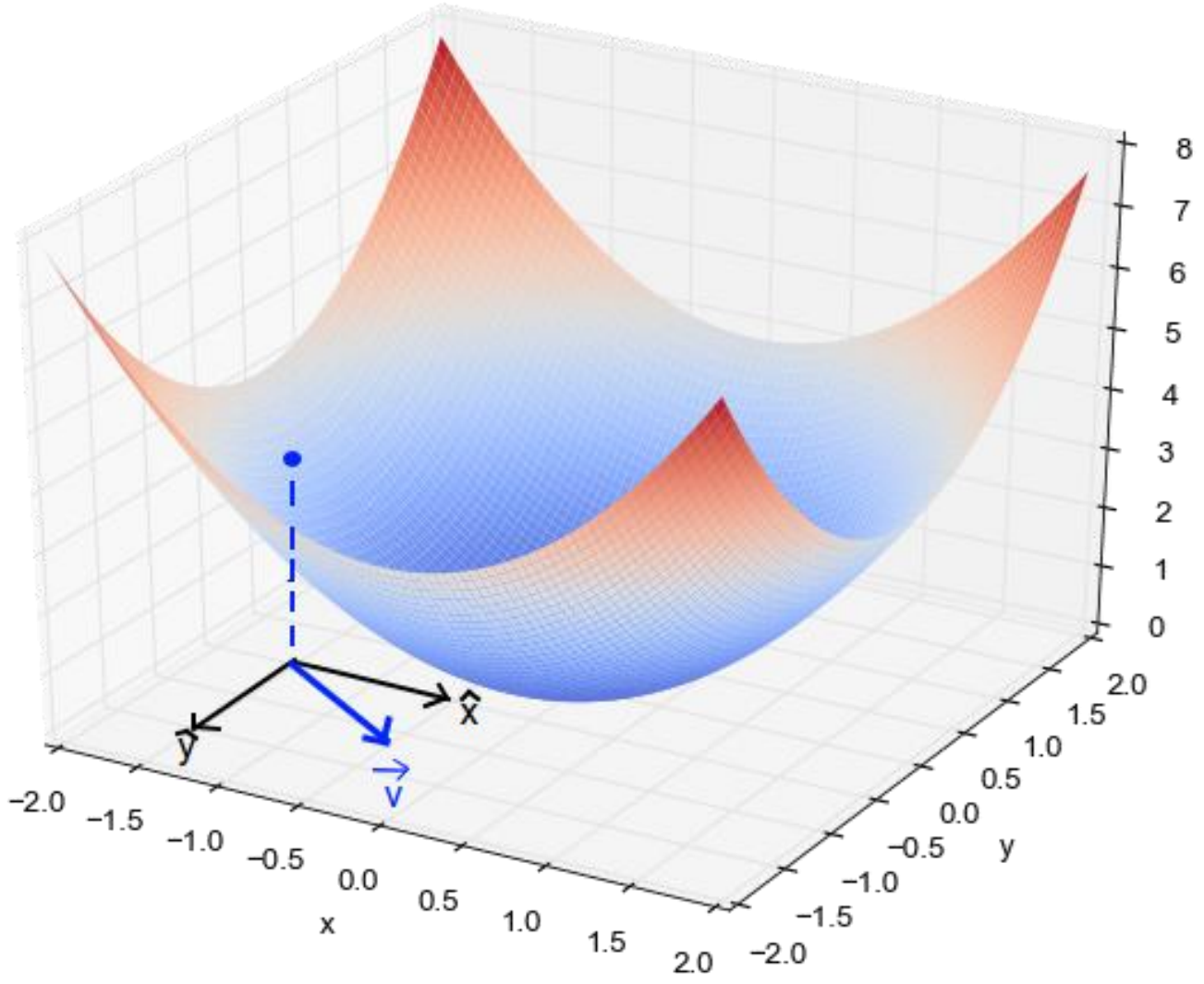
y = label vector

(one-hot)

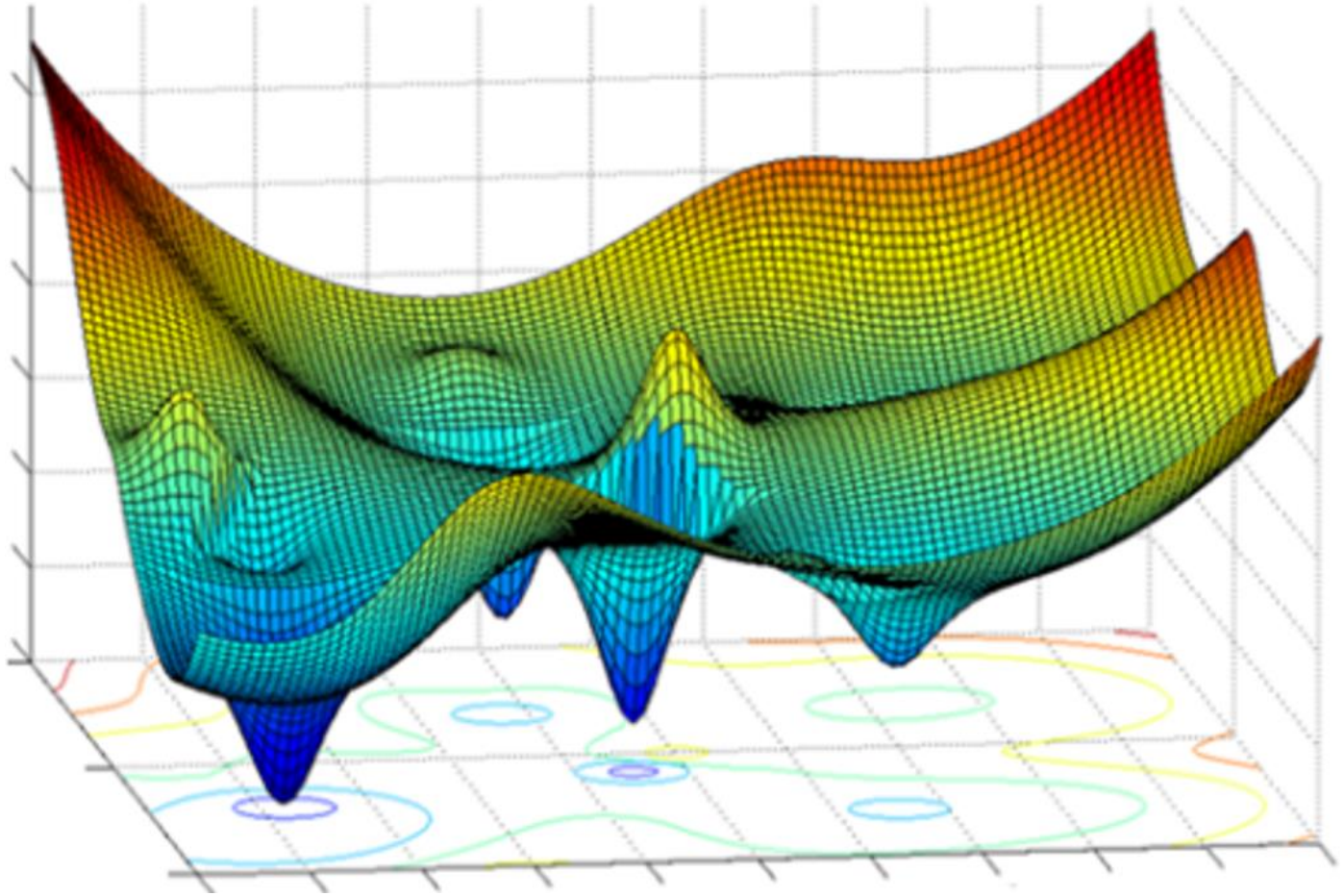
GRADIENT DESCENT



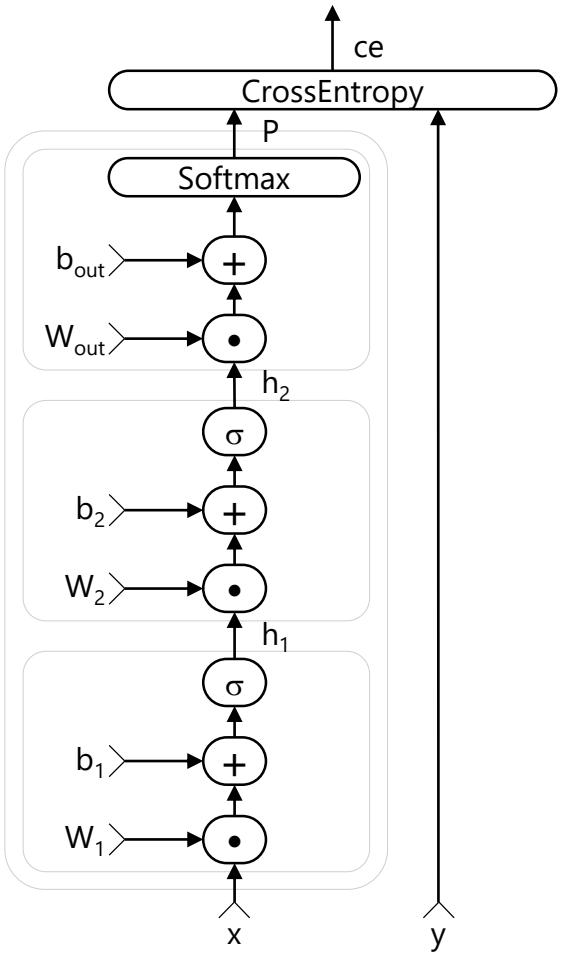
GRADIENT DESCENT



GRADIENT DESCENT



HOW DO WE DESIGN THIS IN CNTK?




```
h1 = Sigmoid (W1   * x  + b1)
h2 = Sigmoid (W2   * h1 + b2)
P  = Softmax (Wout * h2 + bout)
ce = CrossEntropy (y, P)
```

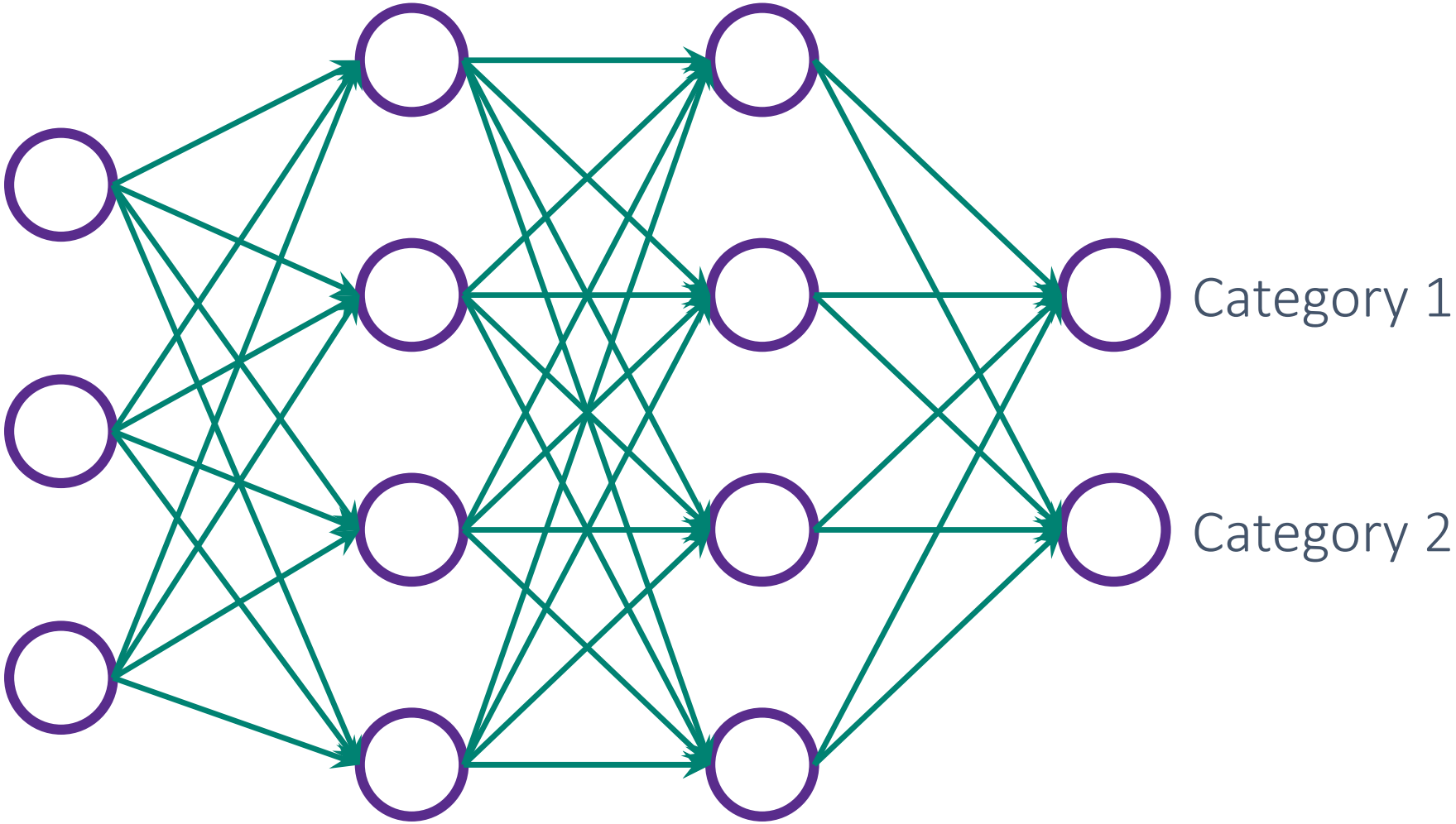
LAB 1: LOGISTIC REGRESSION

plain concepts

NEURAL NETWORKS

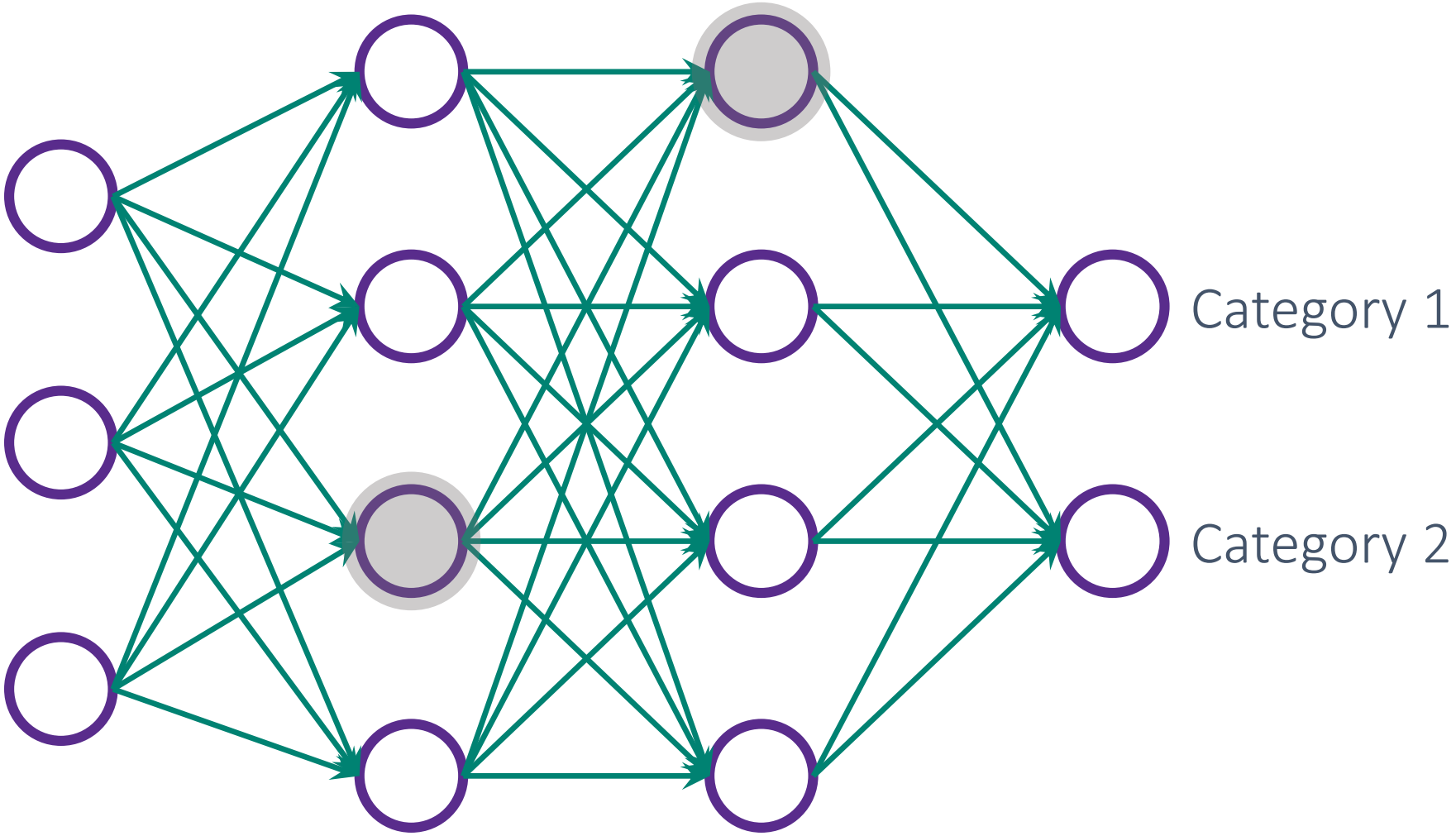
plain concepts

NEURAL NETWORK (CLASSIFIER)



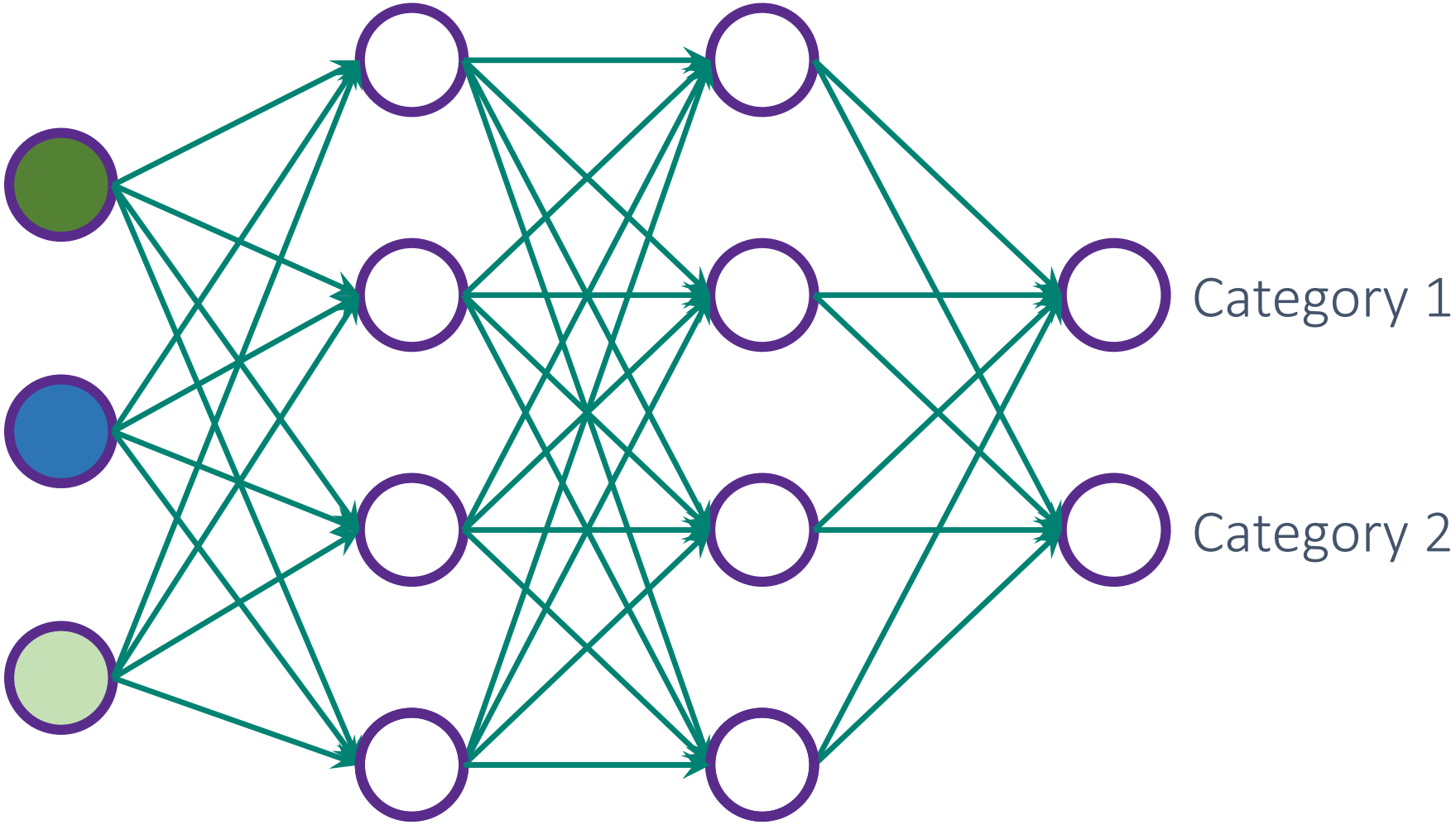
Input Layer Hidden Layers Output Layer

NEURAL NETWORK (CLASSIFIER)



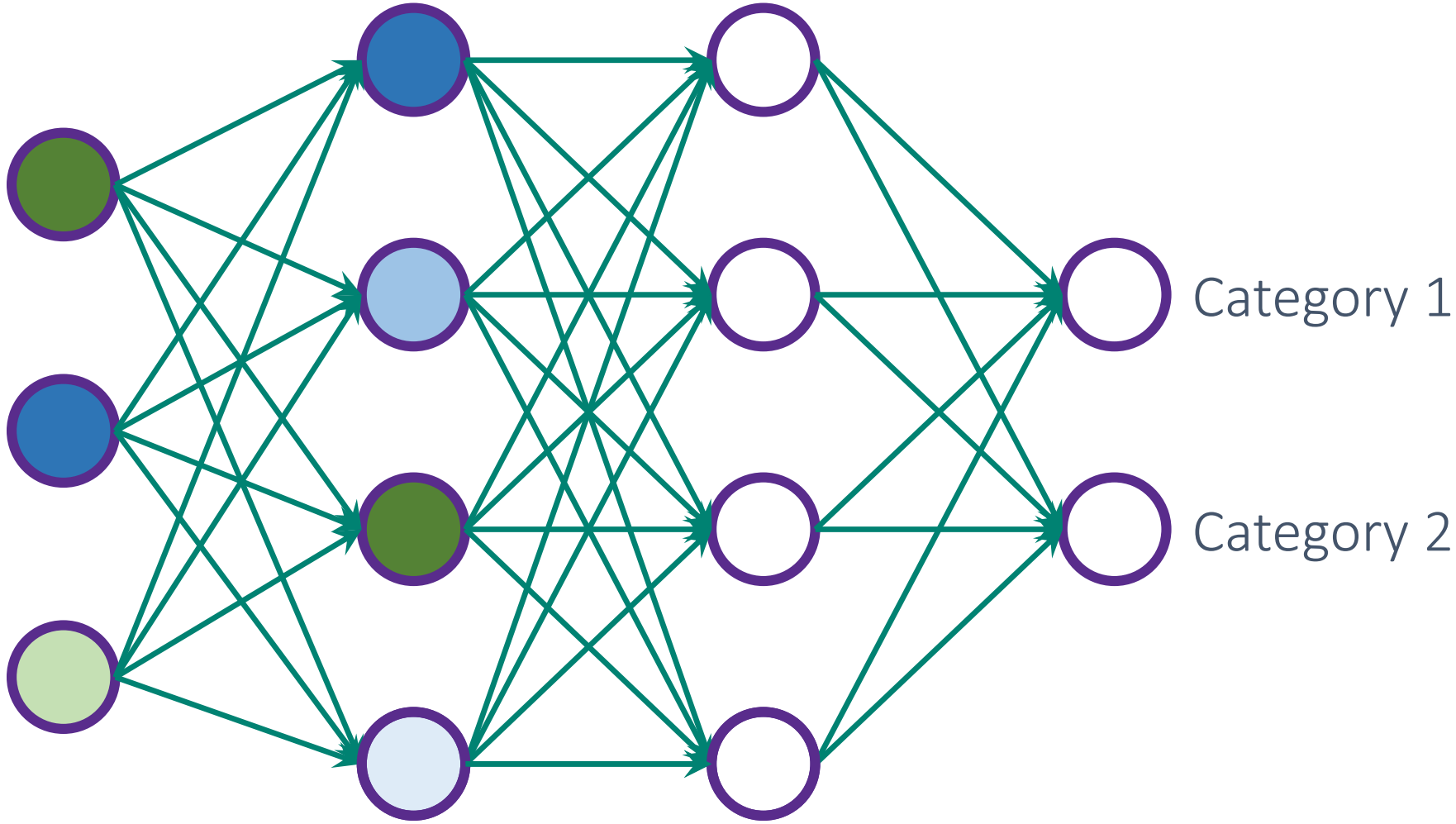
Input Layer Hidden Layers Output Layer

FORWARD PROPAGATION



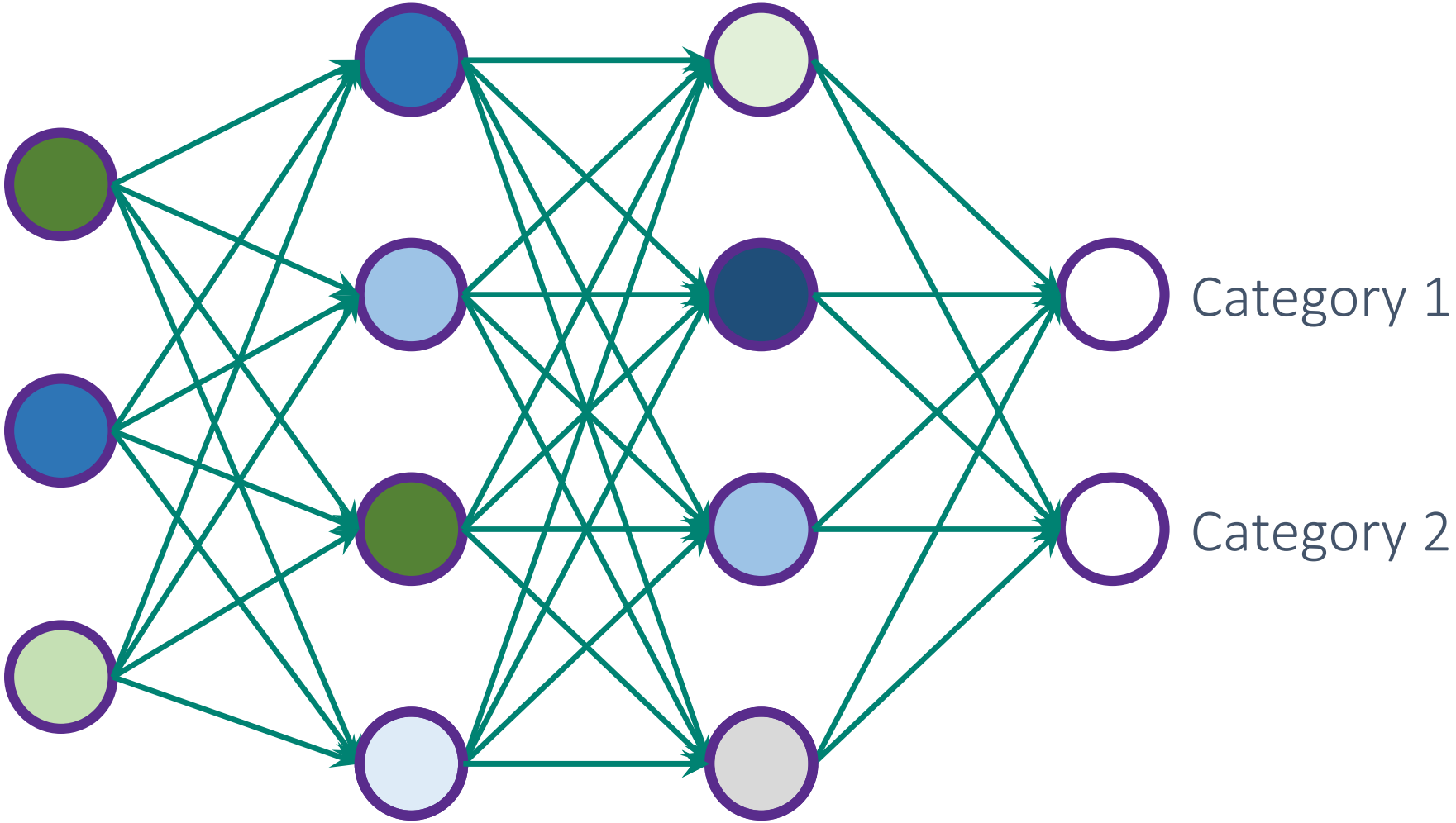
Input Layer Hidden Layers Output Layer

FORWARD PROPAGATION



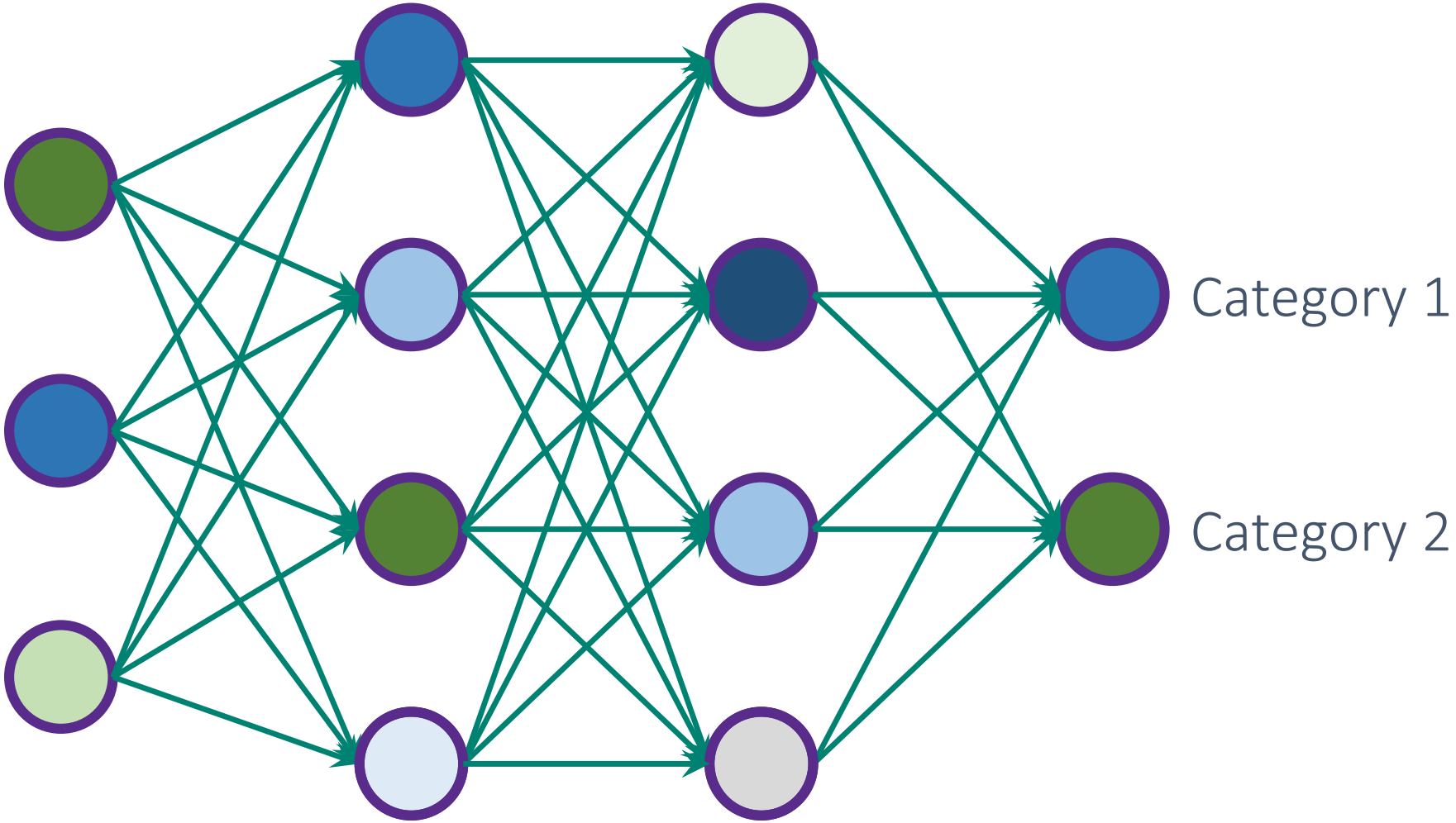
Input Layer Hidden Layers Output Layer

FORWARD PROPAGATION



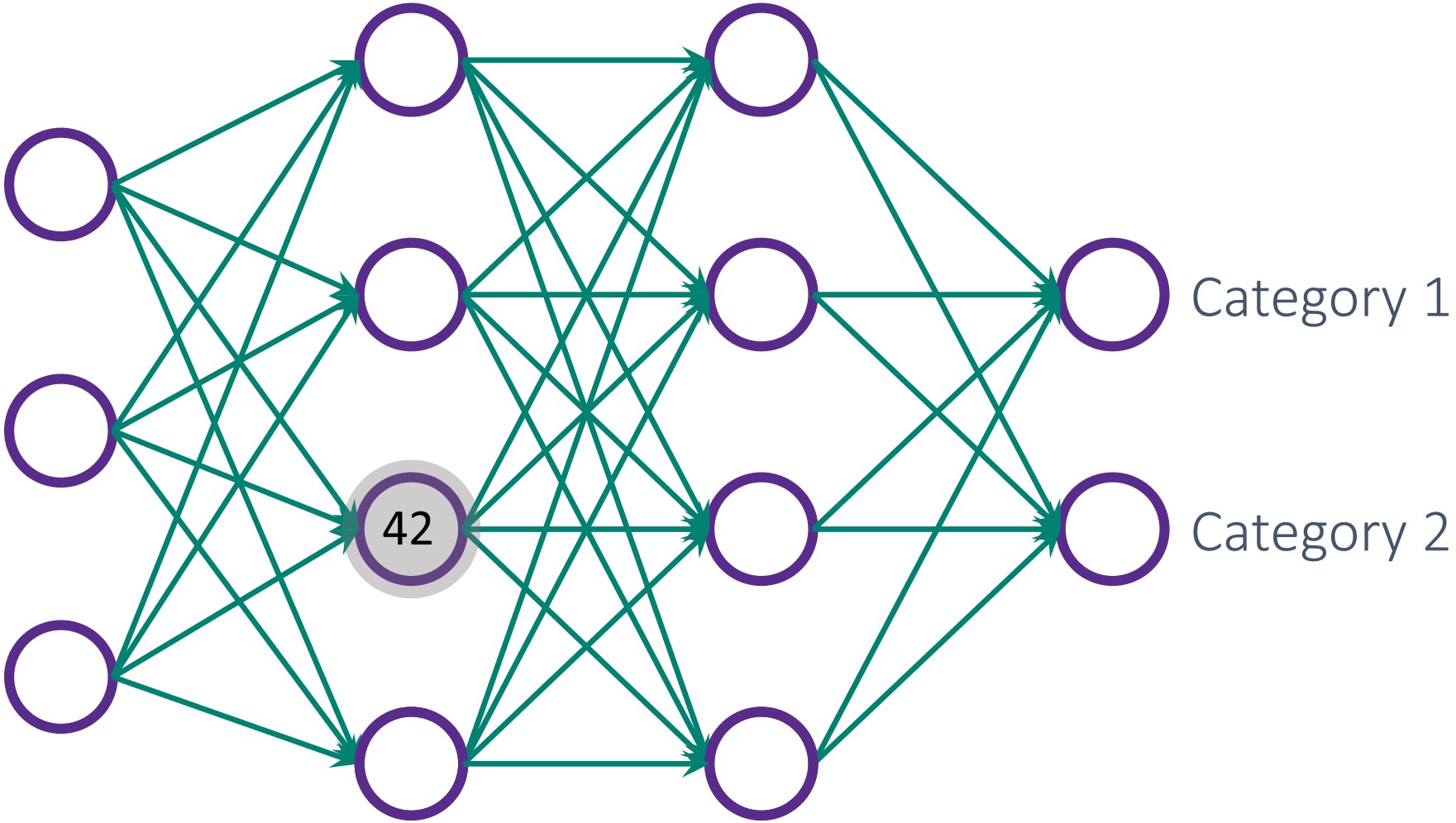
Input Layer Hidden Layers Output Layer

FORWARD PROPAGATION



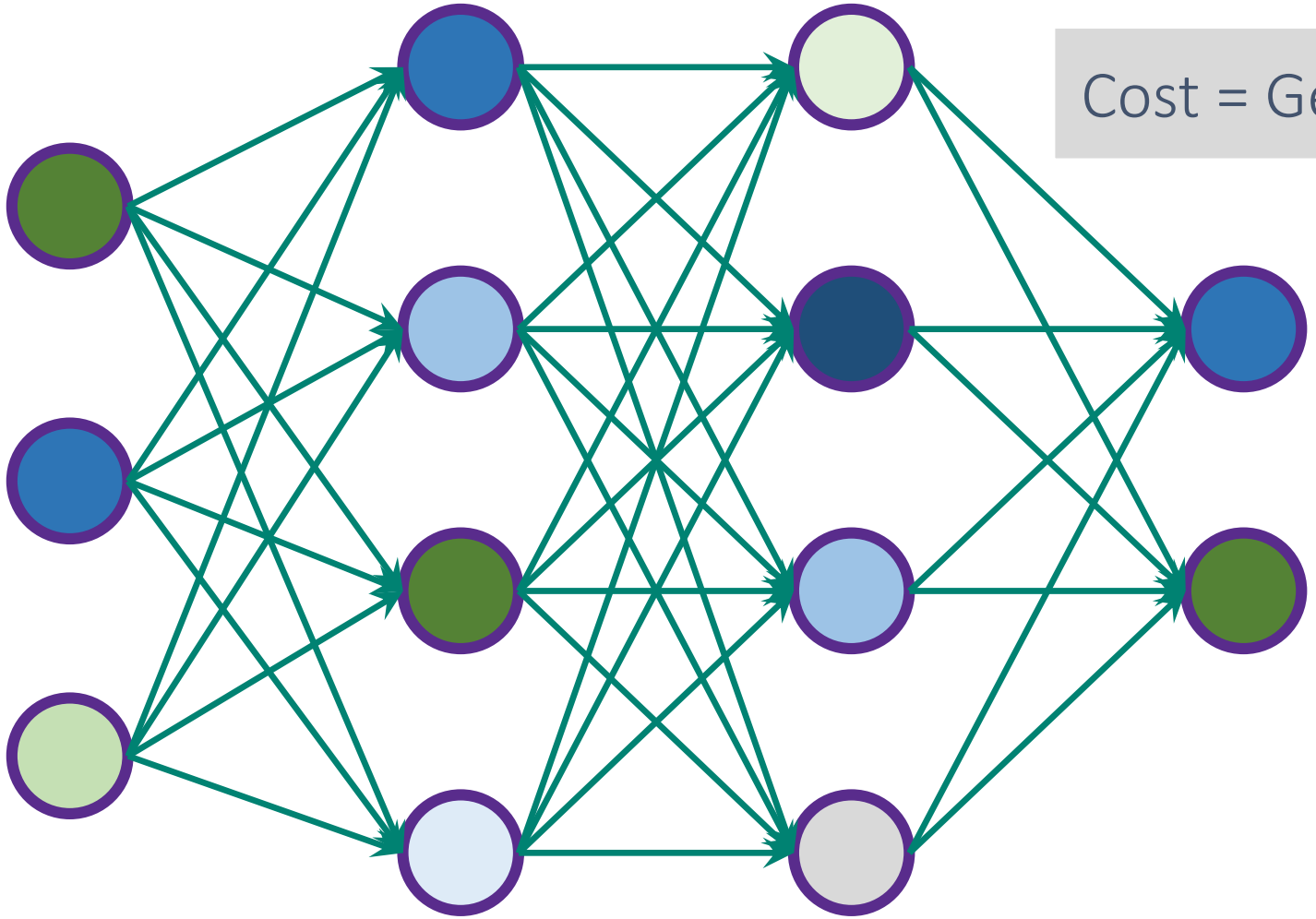
Input Layer Hidden Layers Output Layer

WEIGHTS AND BIAS



Input Layer Hidden Layers Output Layer

TRAINING



Cost = Generated Value – Ground Truth

Category 1

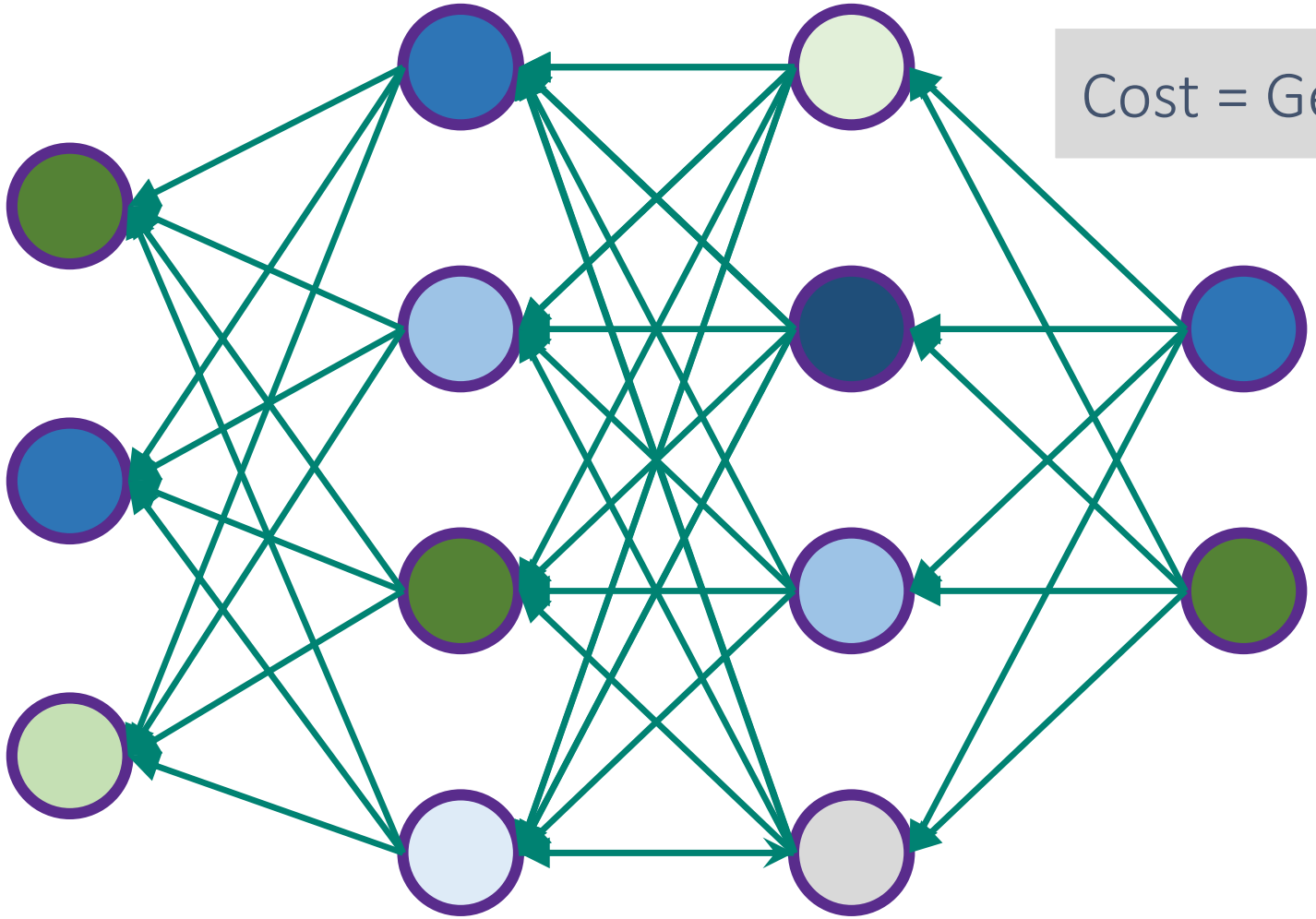
Category 2

Input Layer

Hidden Layers

Output Layer

BACKPROPAGATION



Cost = Generated Value – Ground Truth

Category 1

Error = X

Category 2

Input Layer

Hidden Layers

Output Layer

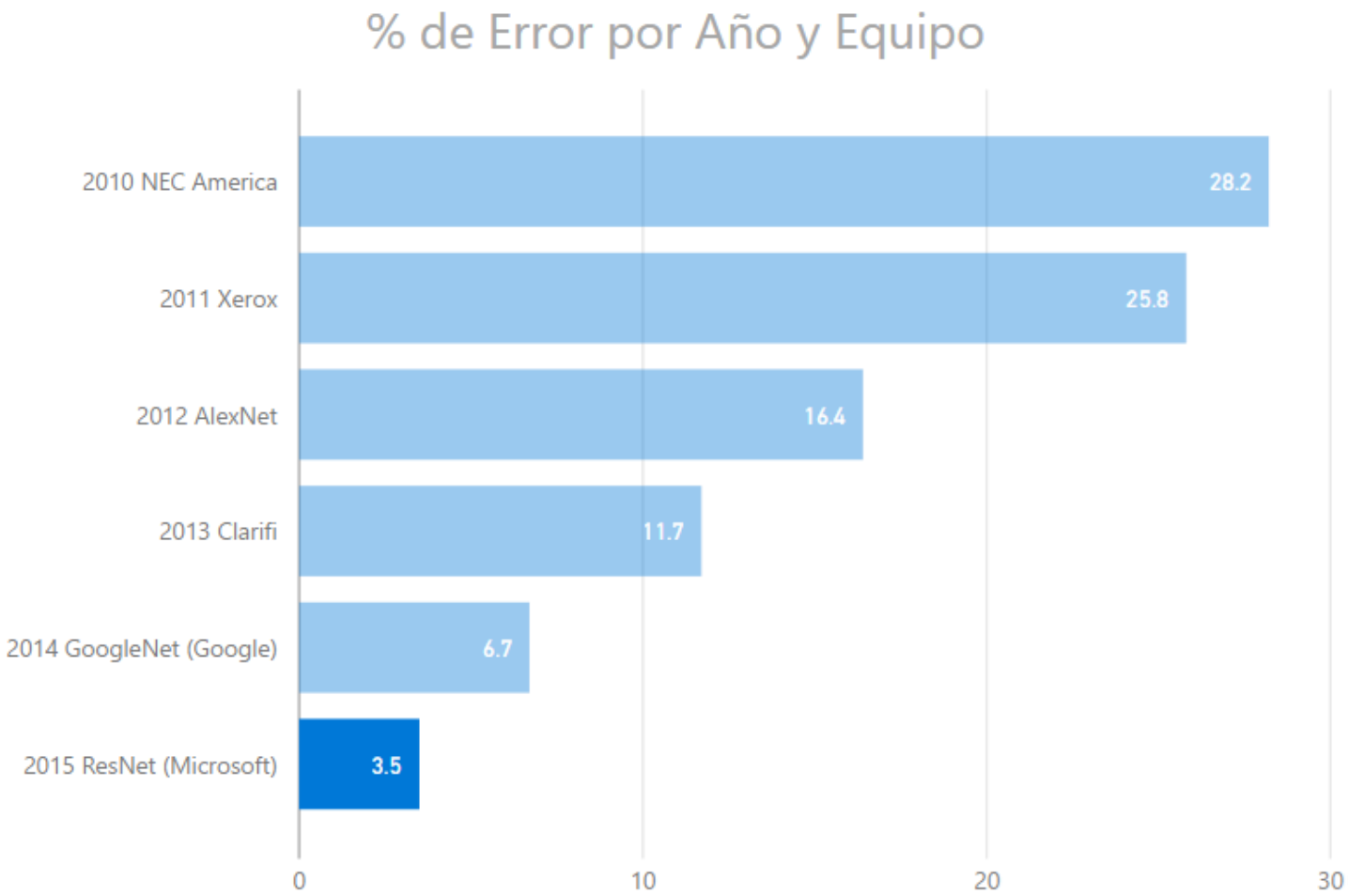
LAB 2: MULTI-LAYER PERCEPTRON

plain concepts

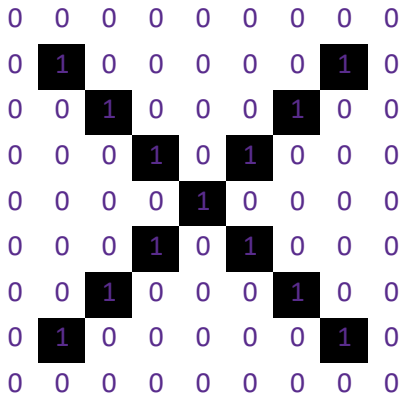
DL: CONVOLUTIONAL NETWORKS

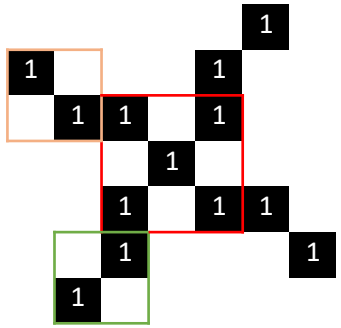
plain concepts

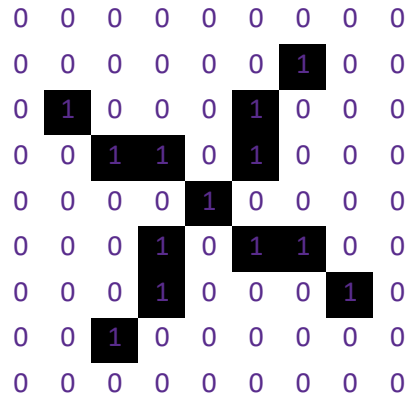
IMAGENET CHALLENGE



FILTERING


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

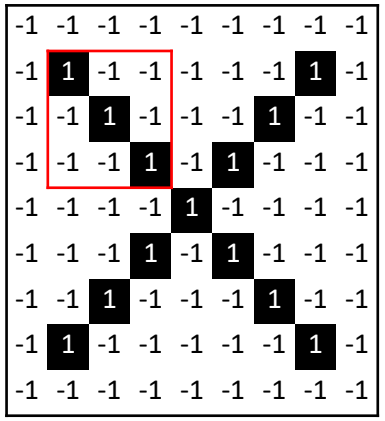

1
1 1
1 1 1
1
1 1 1
1 1
1

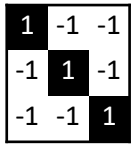

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0
0 1 0 0 0 1 0 0 0
0 0 1 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 1 1 0 0
0 0 0 1 0 0 0 1 0
0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

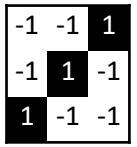
FILTERING

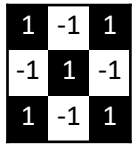
…looking for common characteristics

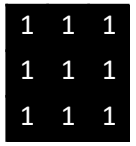
- Filter 1 Filter 2 Filter 3

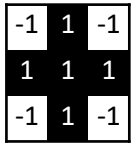


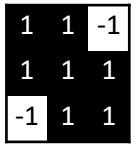






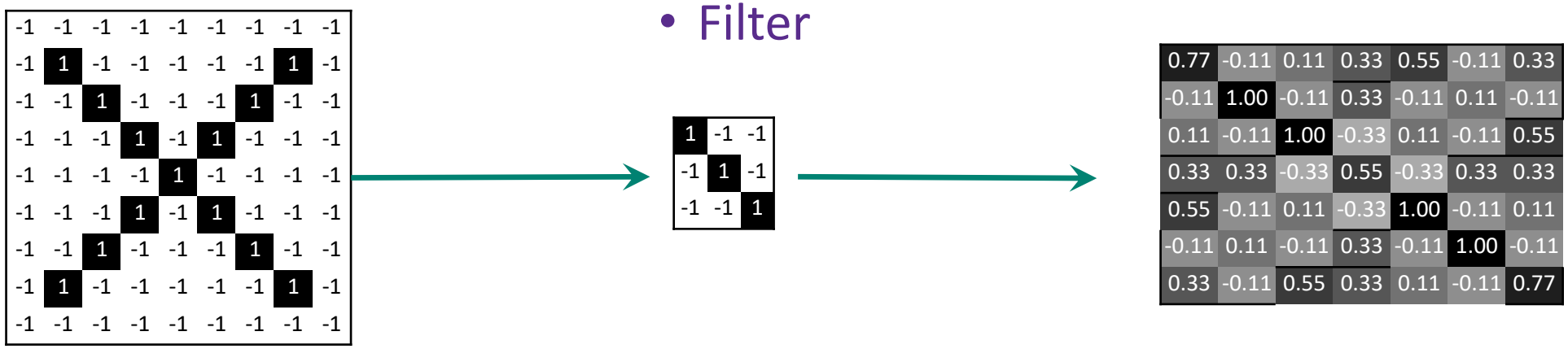




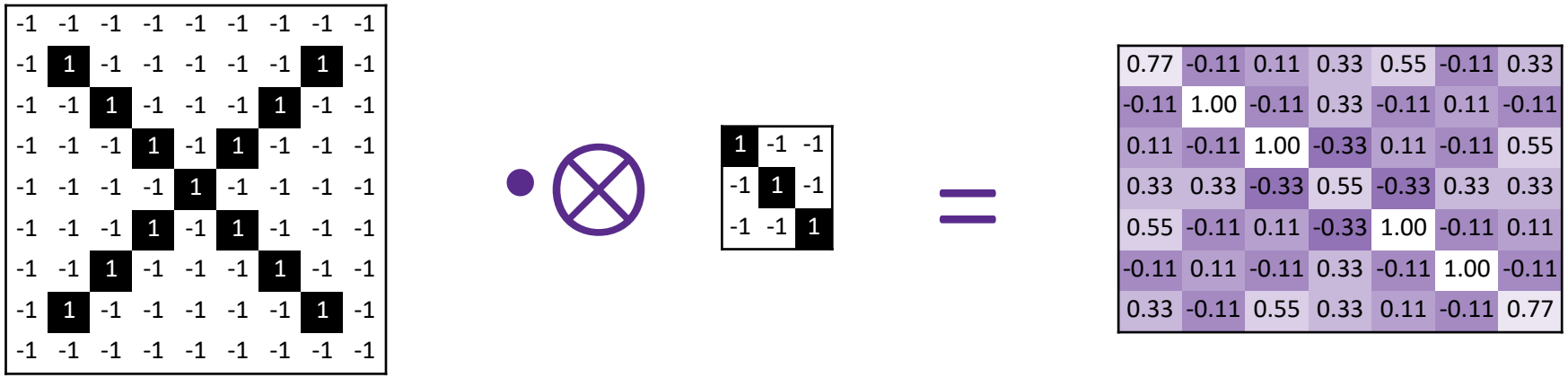


1.0 0.11 0.55

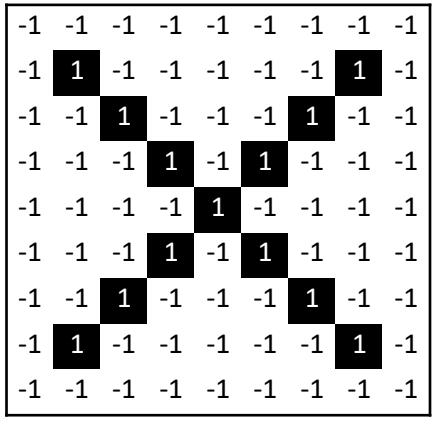
CONVOLUTION



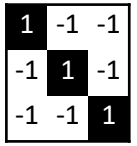
CONVOLUTION



CONVOLUTION

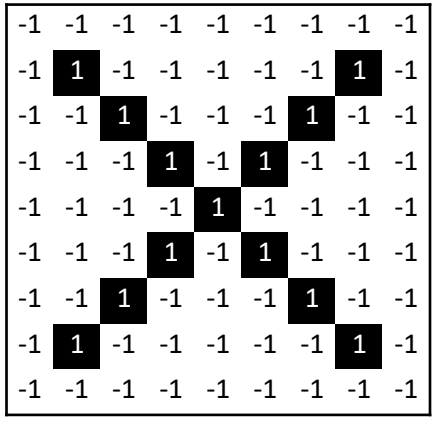


•⊗

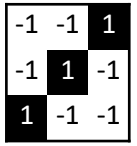


=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

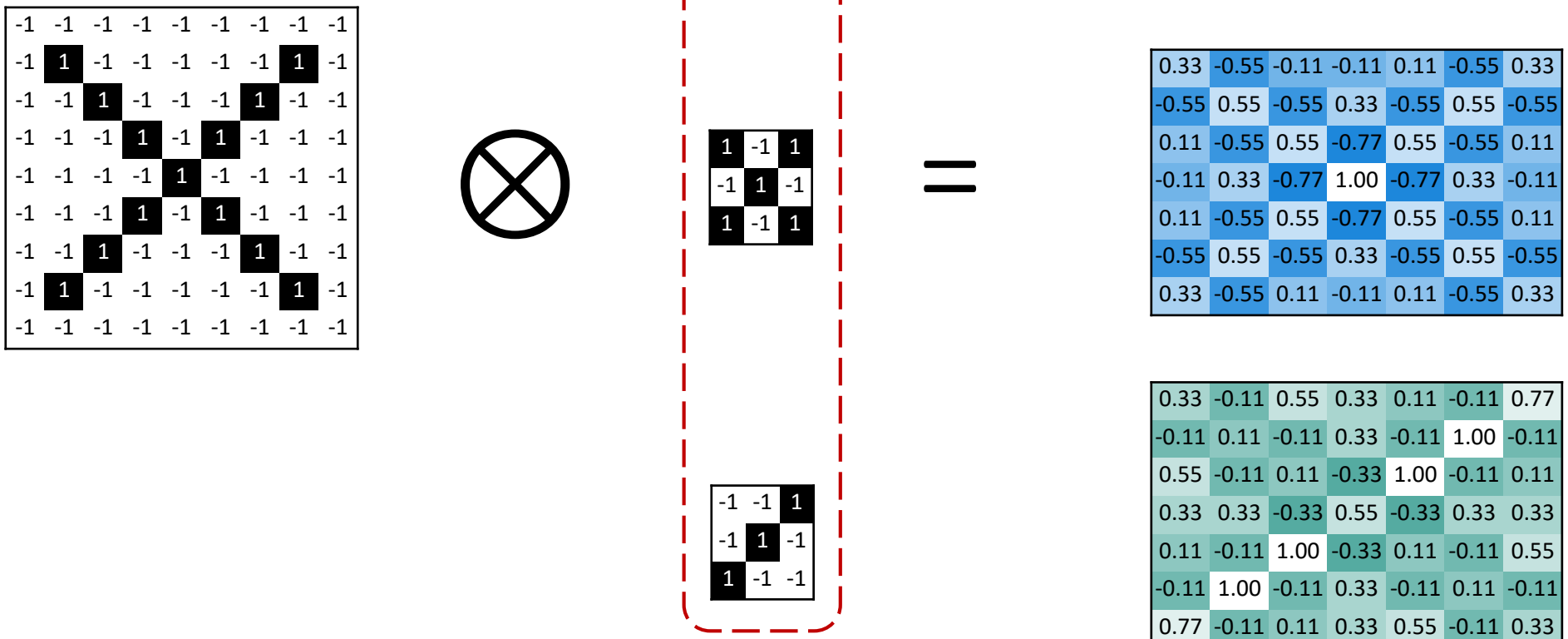


⊗

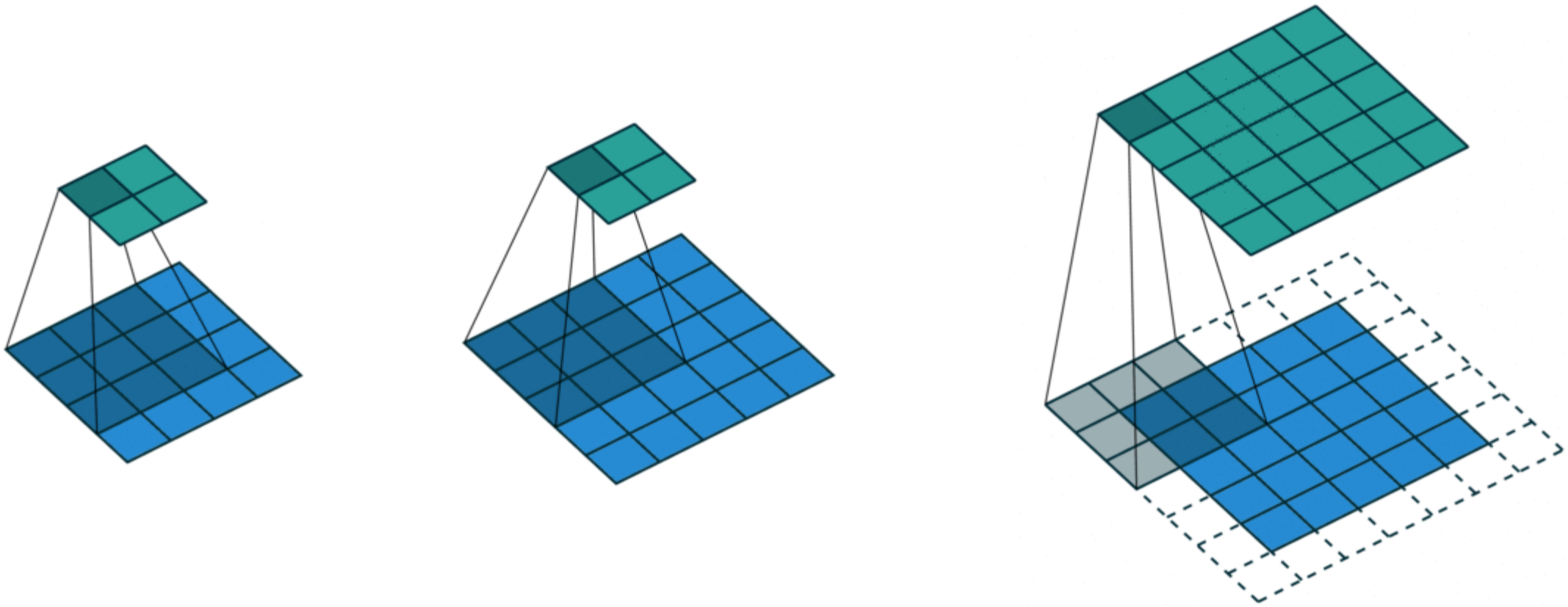


=

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

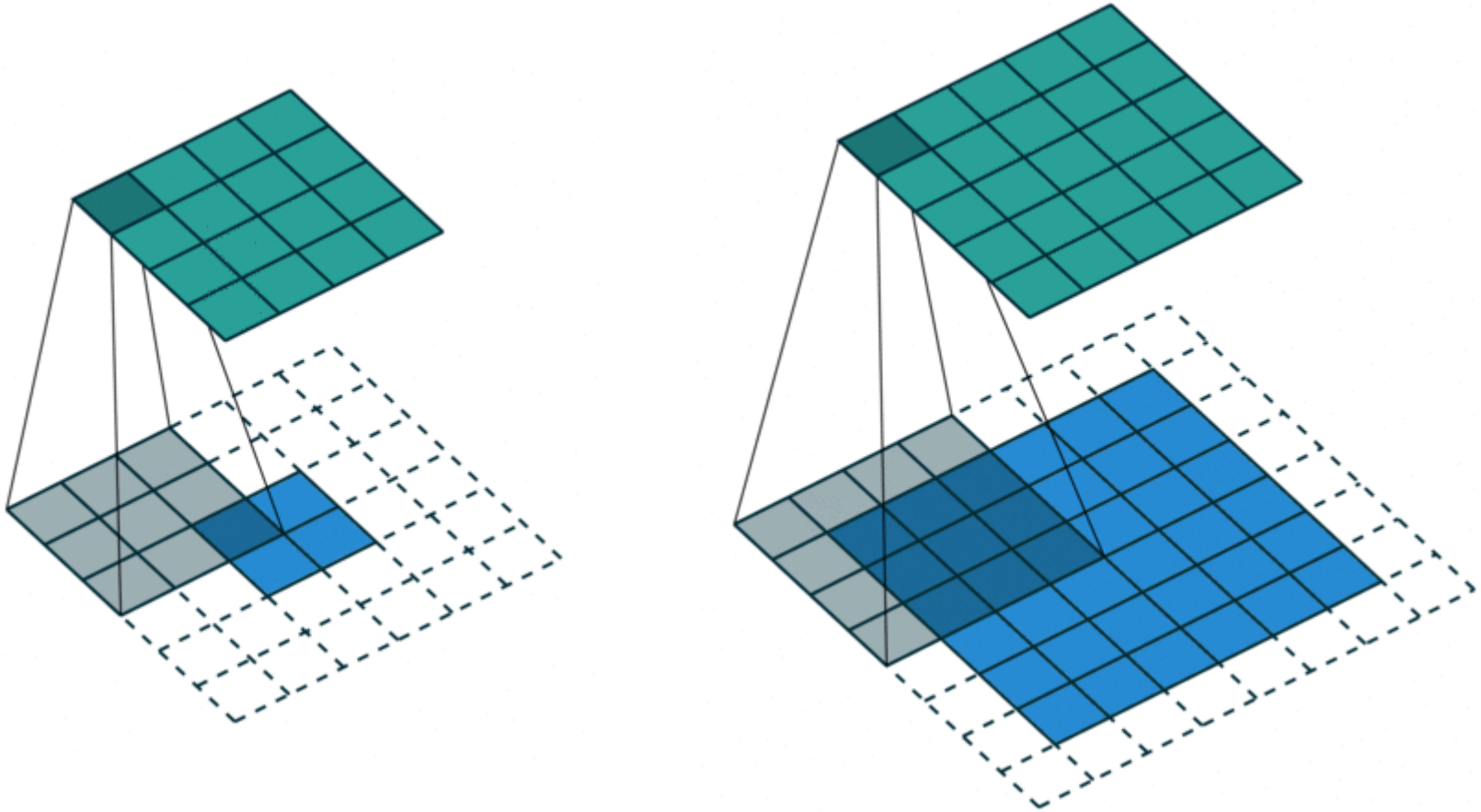

-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
0.33 -0.55 -0.11 -0.11 0.11 -0.55 0.33
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33

STRIDES AND PADDING



source: https://github.com/vdumoulin/conv_arithmetic

DECONVOLUTION – TRANSPOSED CONVOLUTION

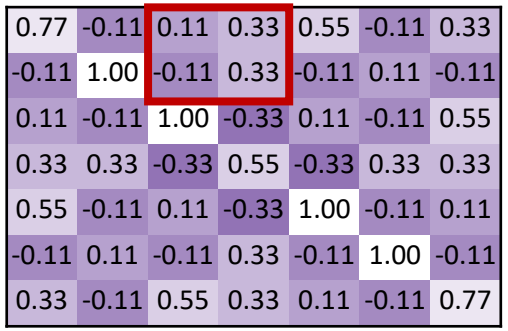


source: https://github.com/vdumoulin/conv_arithmetic

POOLING

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

POOLING


0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77

AVERAGE POOLING – STRIDE 1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.39	0.22	0.17	0.28	0.11	0.06
0.22	0.45	0.22	0.00	0.00	0.11
0.17	0.22	0.22	0.00	0.00	0.28
0.28	0.00	0.00	0.22	0.22	0.17
0.11	0.00	0.00	0.22	0.45	0.22
0.06	0.11	0.28	0.17	0.22	0.39

MAX POOLING – STRIDE 1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	1.00	0.33	0.55	0.55	0.33
1.00	1.00	1.00	0.33	0.11	0.55
0.33	1.00	1.00	0.55	0.33	0.55
0.55	0.33	0.55	1.00	1.00	0.33
0.55	0.11	0.33	1.00	1.00	1.00
0.33	0.55	0.55	0.33	1.00	1.00

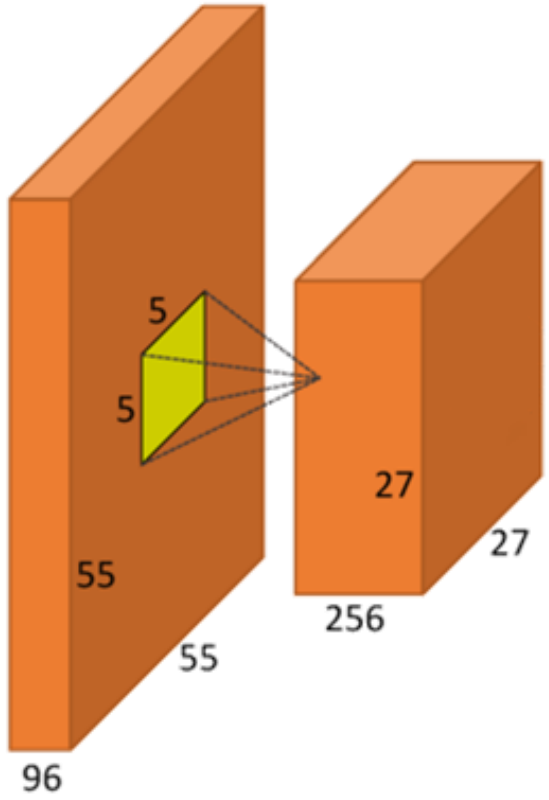
AVERAGE POOLING – STRIDE 2

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

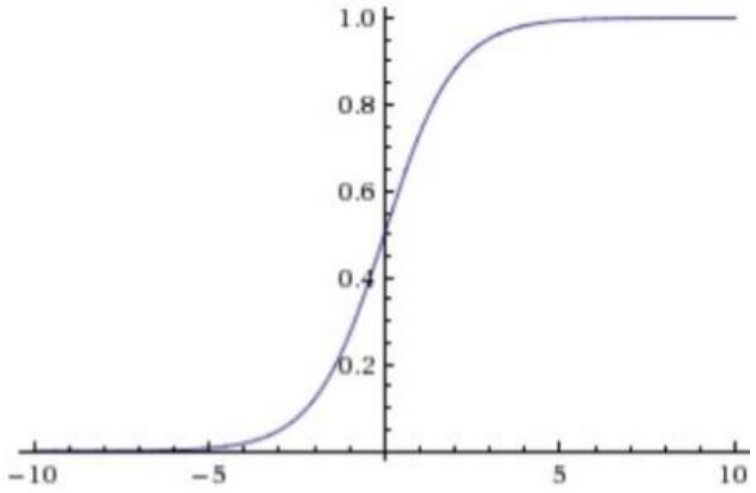
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

WHY POOLING?

ACTIVATION FUNCTIONS?



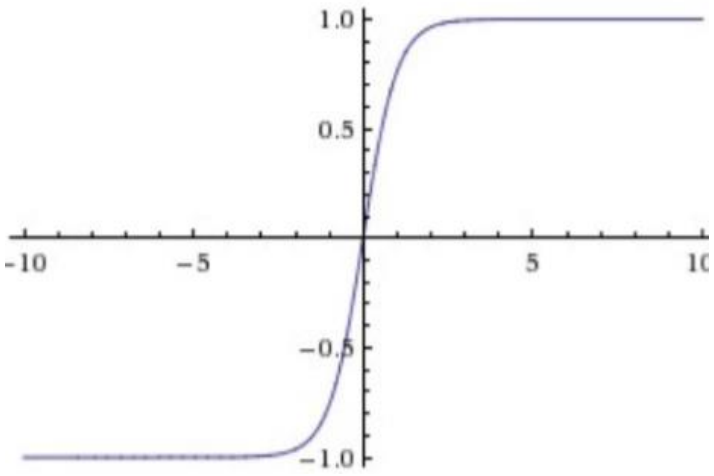
ACTIVATION FUNCTIONS?



LOGISTIC FUNCTION (“*sigmoid*”):

- Simple
- Drawbacks:
  - Saturates
  - Non-centered in zero
- We don’t use it anymore, but we thank her for the services provided to the cause :)

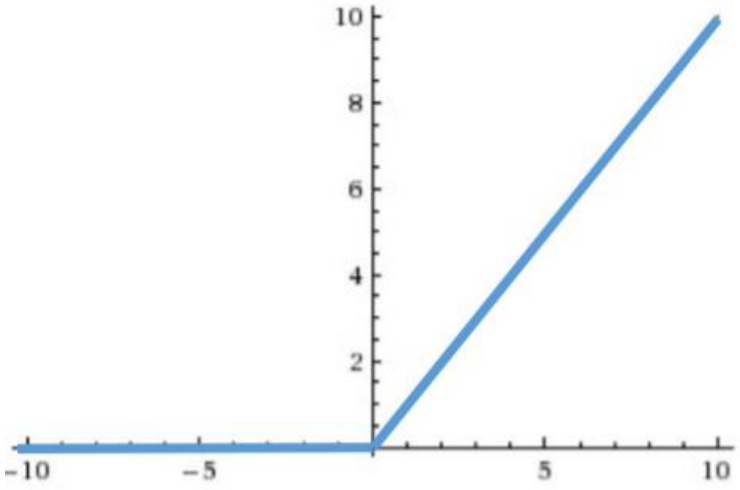
ACTIVATION FUNCTIONS?



HYPERBOLIC TANGENT (*tanh*):

- Solves the issue with not being centered in zero
- Still ‘killing gradiets’ due to saturating behaviouir
- It is always better than the logistic function

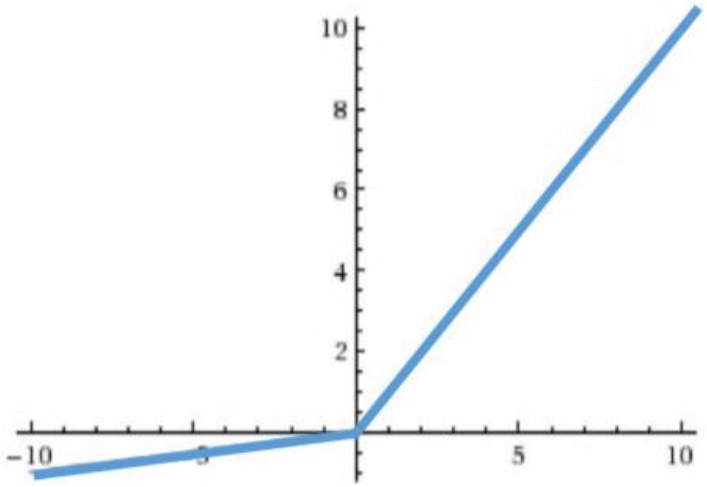
ACTIVATION FUNCTIONS?



RECTIFIED LINEAR UNITS (ReLU):

- Optimization for the SGD convergence
- More simple from a computational point of view
- Fragile

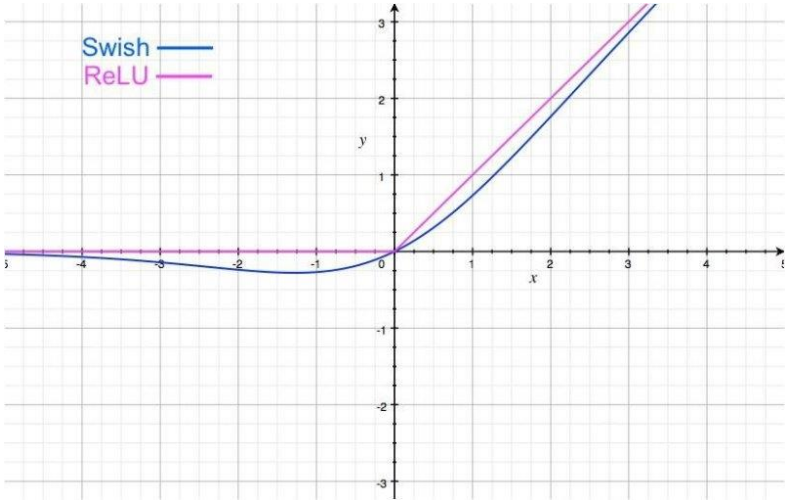
ACTIVATION FUNCTIONS?



LEAKY ReLU:

- More stable than traditional ReLu
- Particular case of *MaxOut*

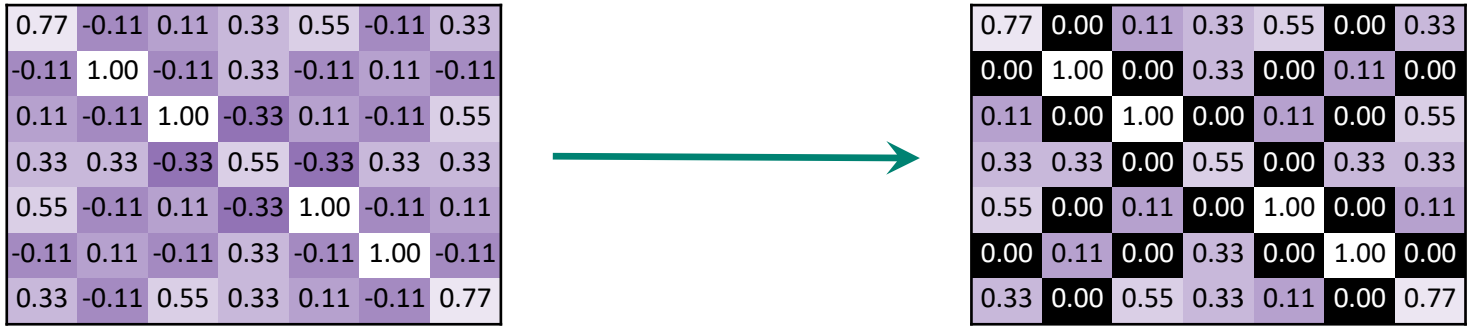
ACTIVATION FUNCTIONS?



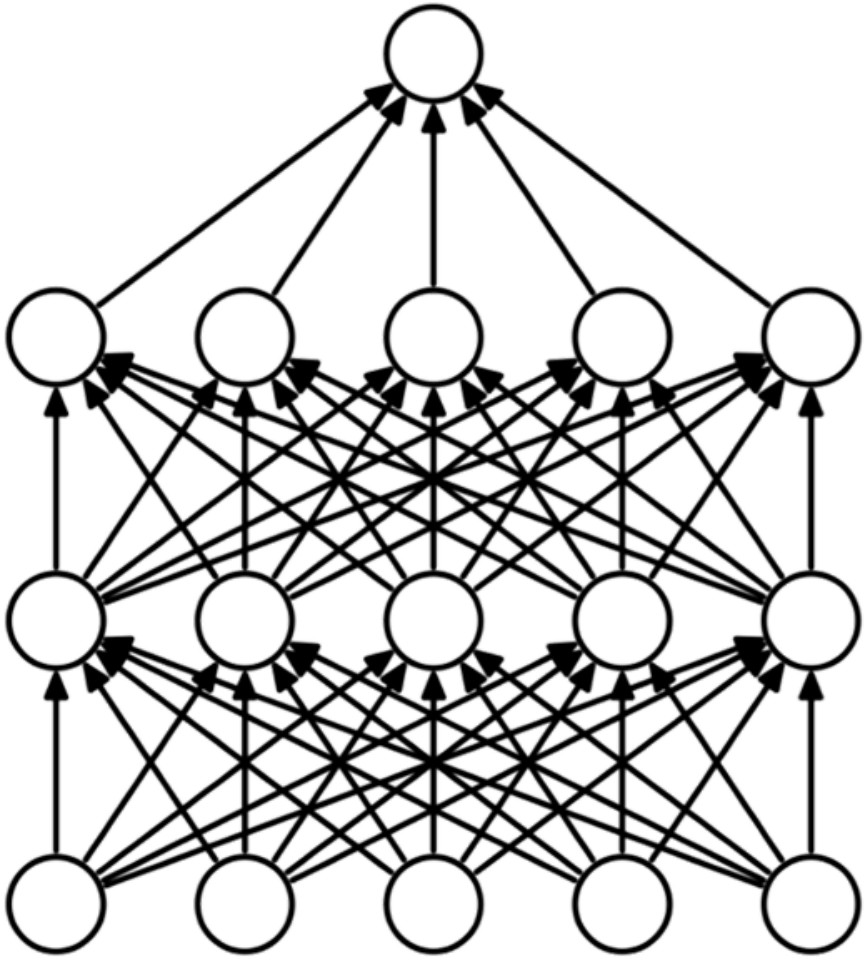
SWISH:

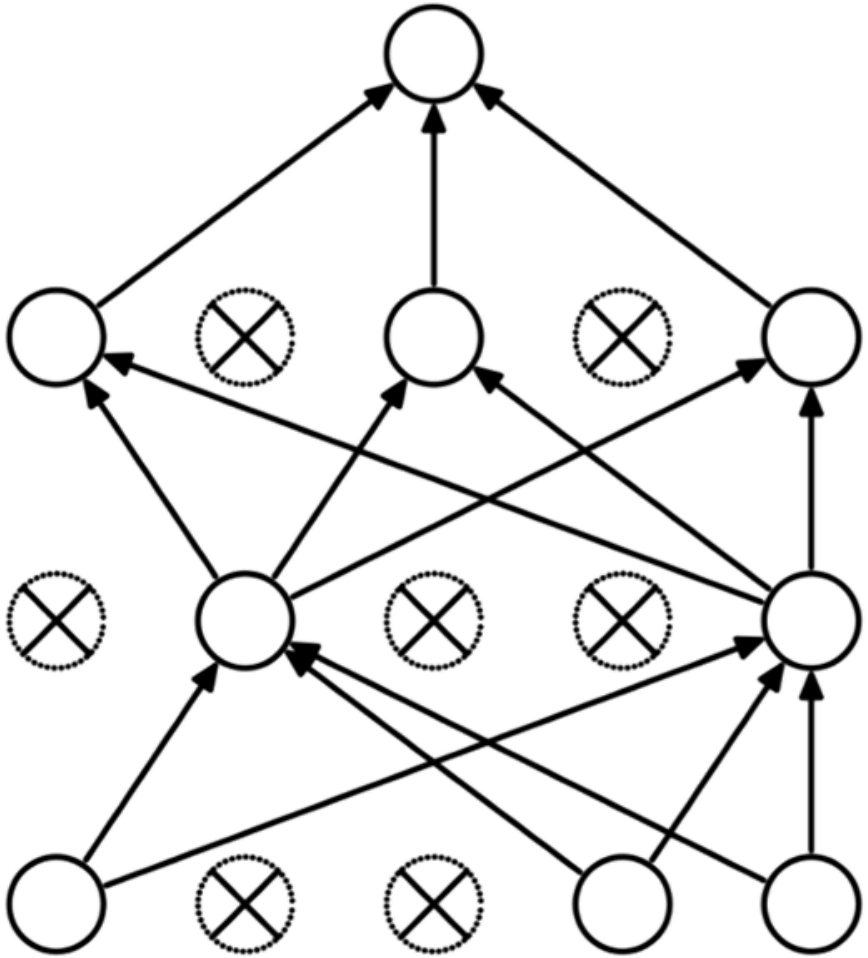
- Self-Gating activation function
- Better performance than ReLu

RECTIFIED LINEAR UNITS (ReLU)

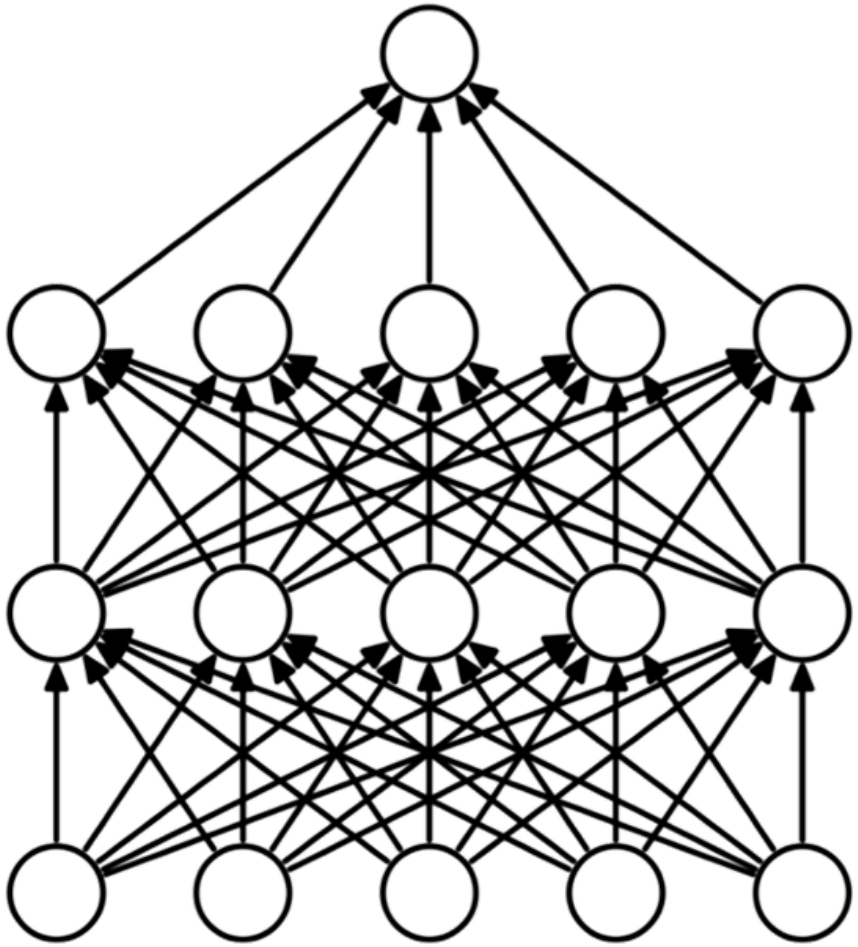


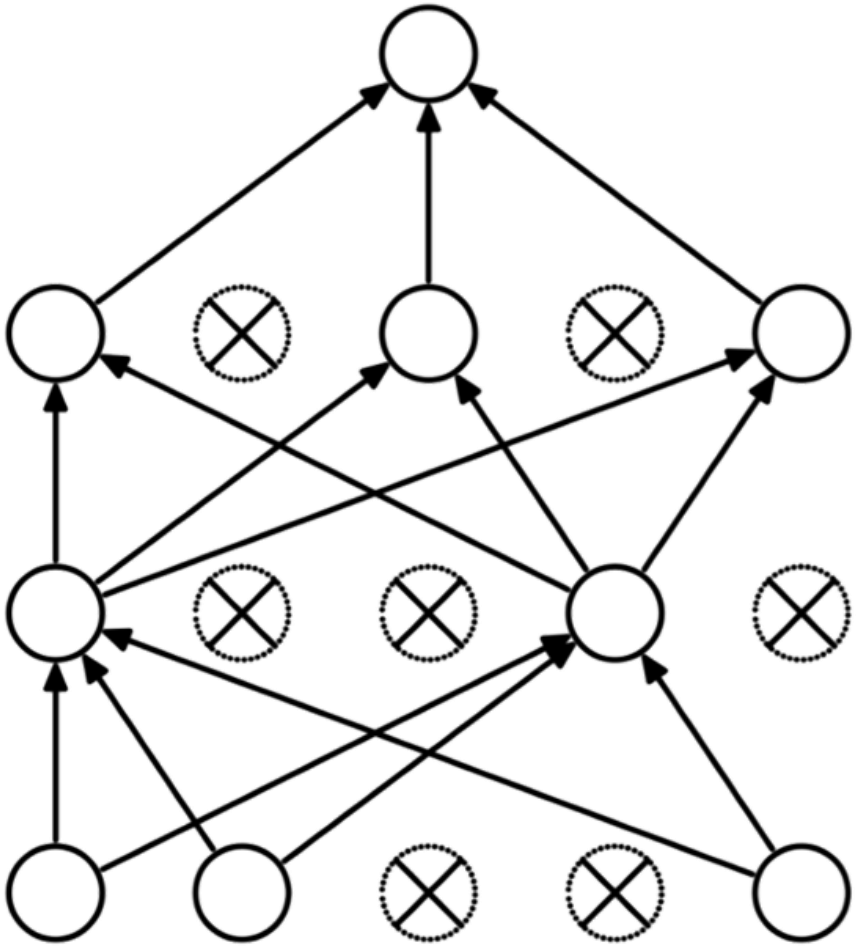
DROPOUT





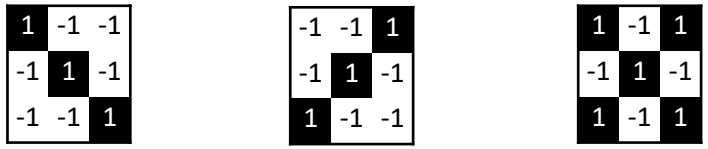
DROPOUT





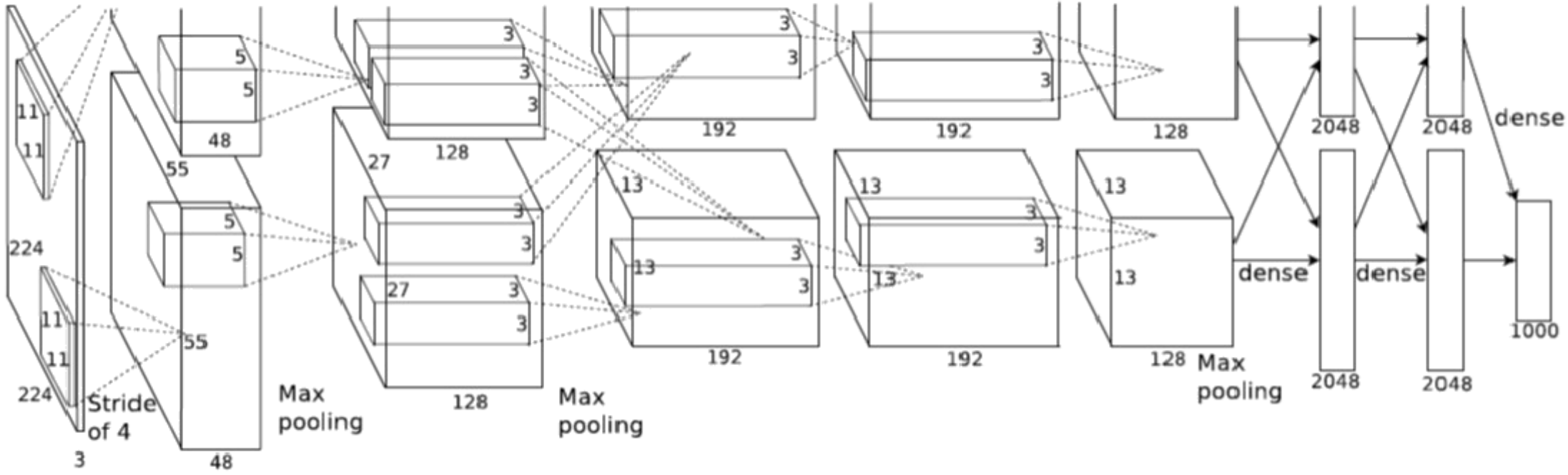
WHAT IS MISSING?

- Filter 1 Filter 2 Filter 3

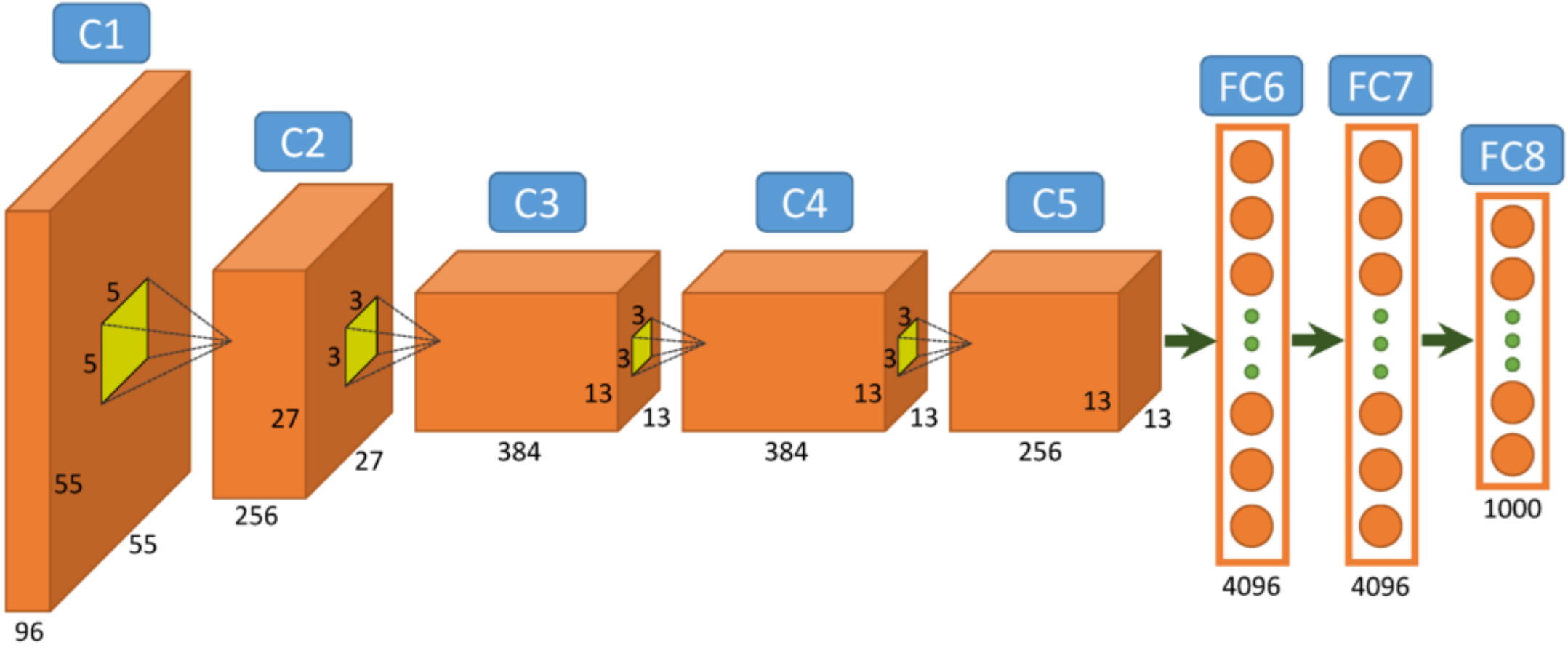


¿De dónde han salido estos?

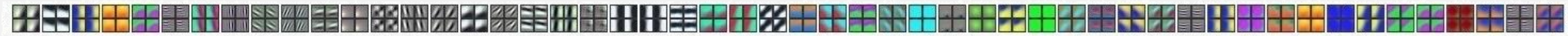
AlexNet


11
11
224
224
3
Stride
of 4
5
5
48
48
55
55
5
5
3
3
3
128
128
27
27
3
3
3
3
3
192
192
128
13
13
13
13
13
13
3
3
3
3
3
3
192
192
128
Max
pooling
Max
pooling
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000

AlexNet

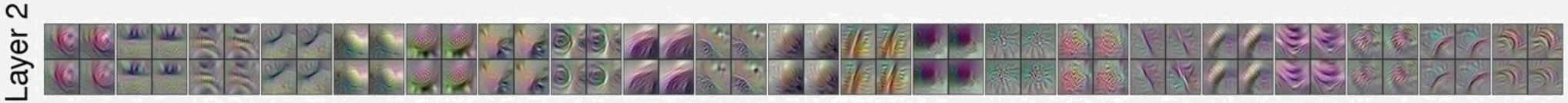

C1
C2
C3
C4
C5
FC6
FC7
FC8
5
5
96
55
55
3
3
256
27
27
3
3
384
13
13
3
3
384
13
13
3
3
256
13
13
4096
4096
1000

HOW A CNN SEES

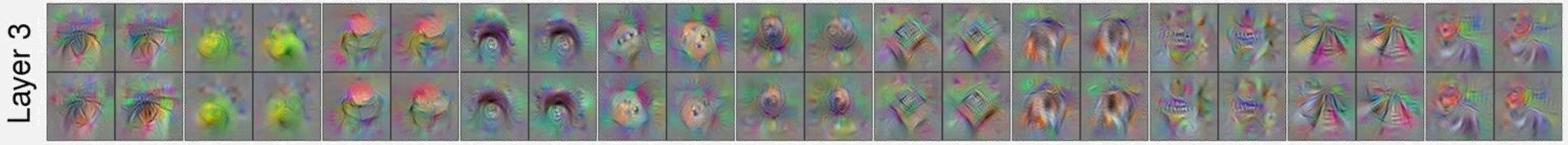


Layer 1

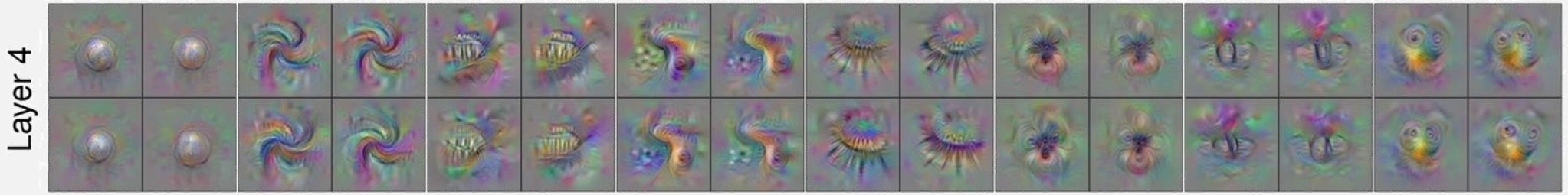
HOW A CNN SEES



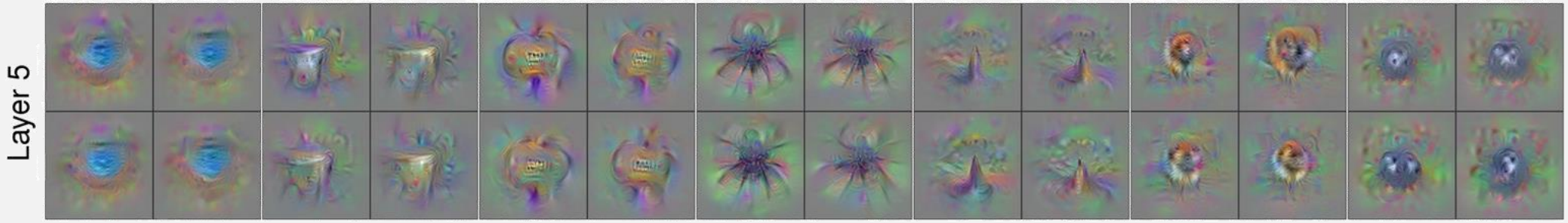
HOW A CNN SEES



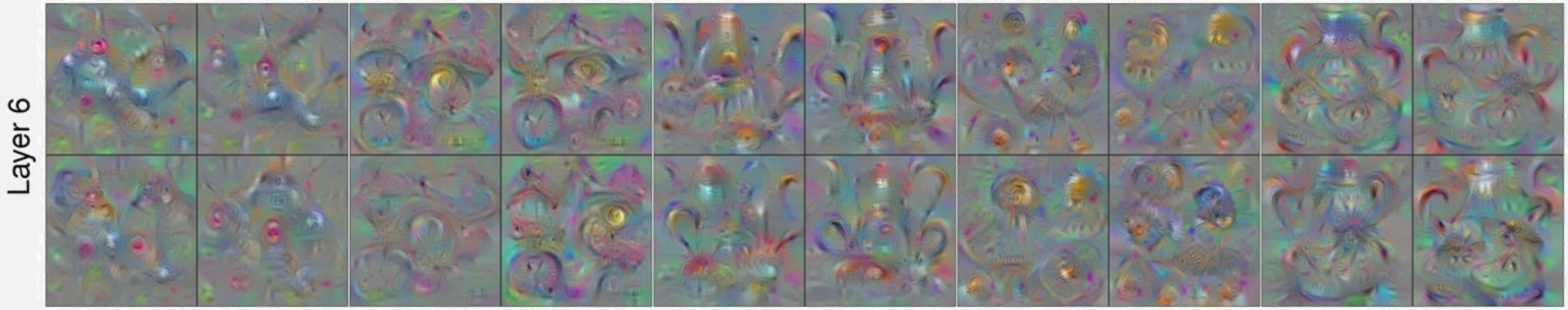
HOW A CNN SEES



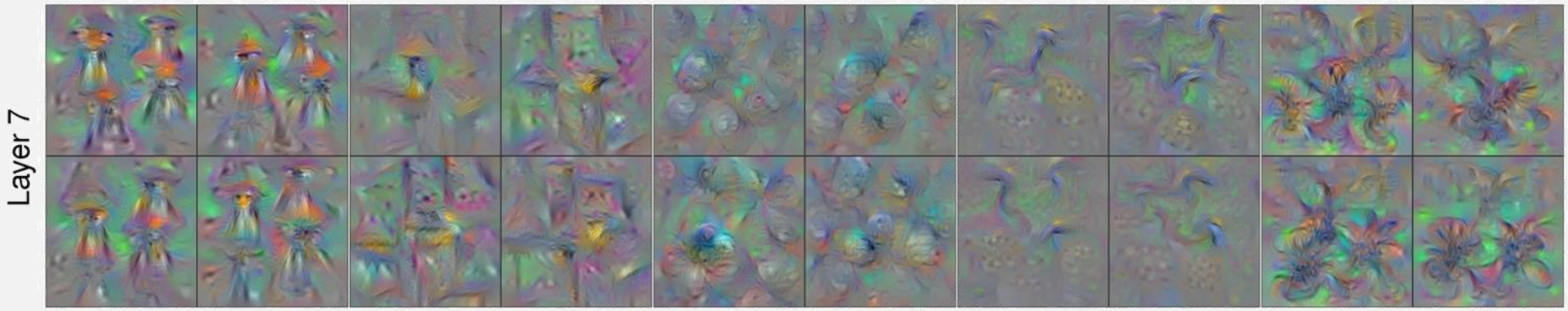
HOW A CNN SEES



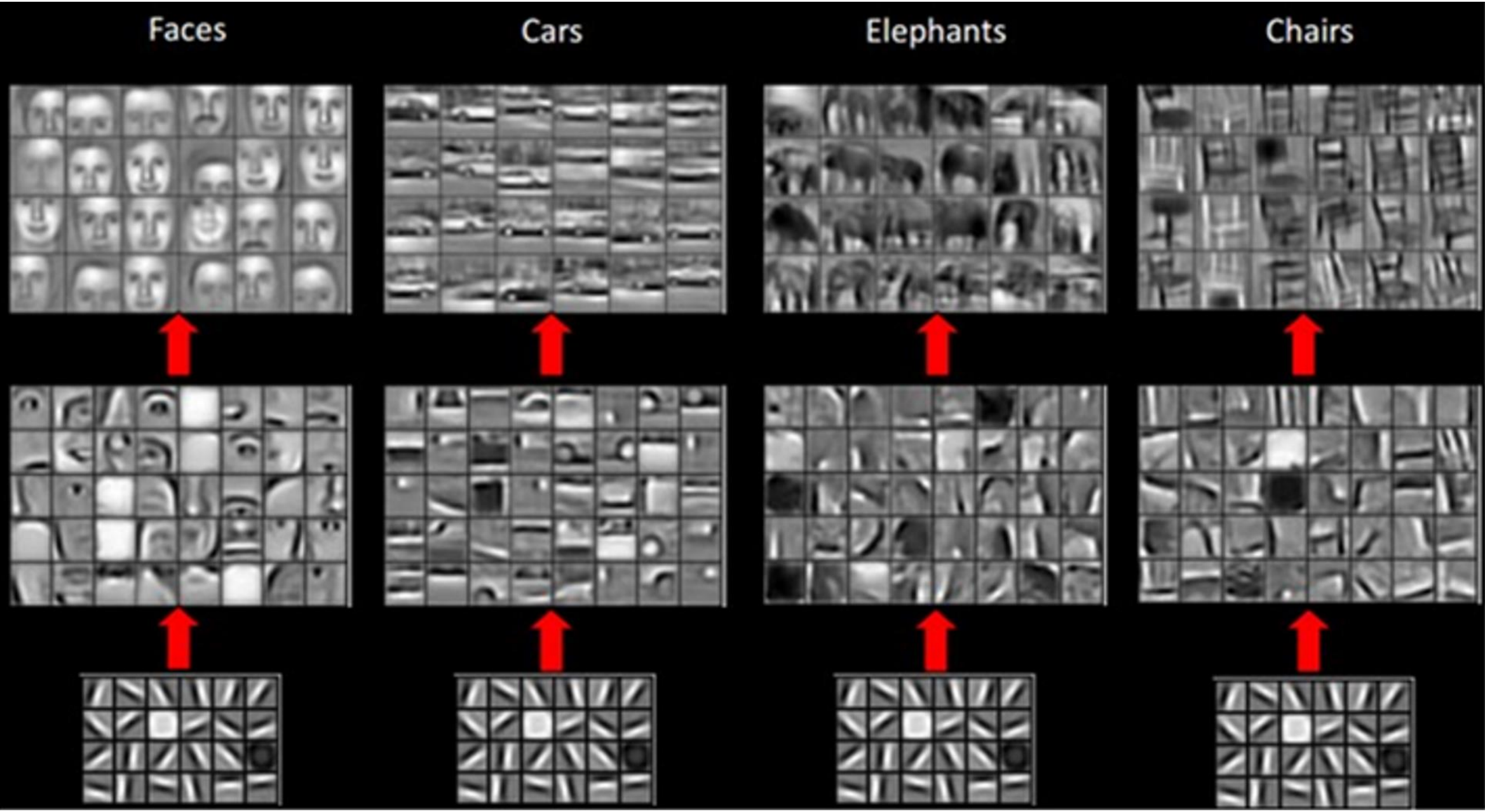
HOW A CNN SEES



HOW A CNN SEES



HOW A CNN SEES


Faces
Cars
Elephants
Chairs

LAB 3: CONVOLUTIONAL NETWORKS

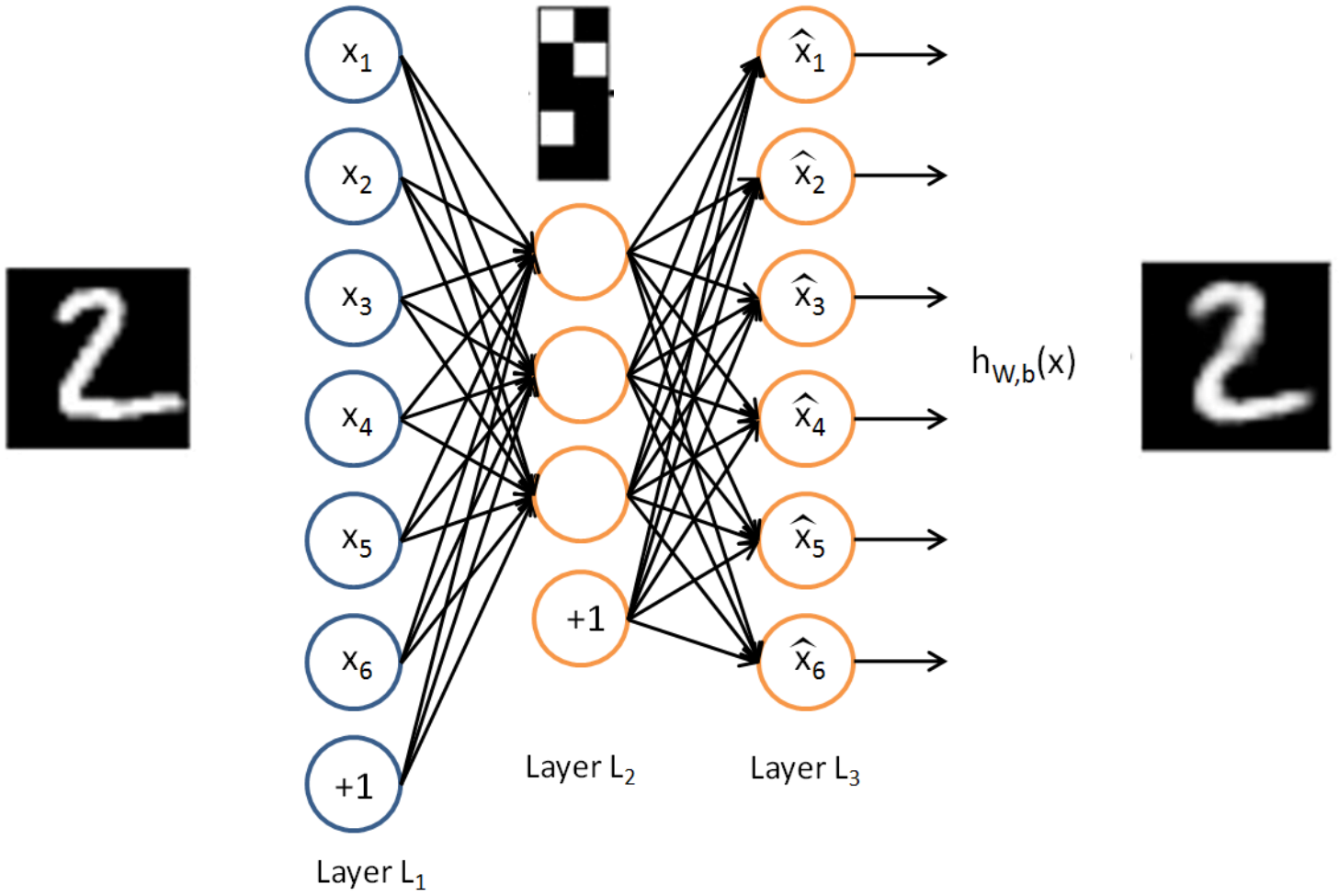
plain concepts


Movidius™
Neural Compute Stick

AUTOENCODERS

plain concepts

BASIC AUTOENCODER

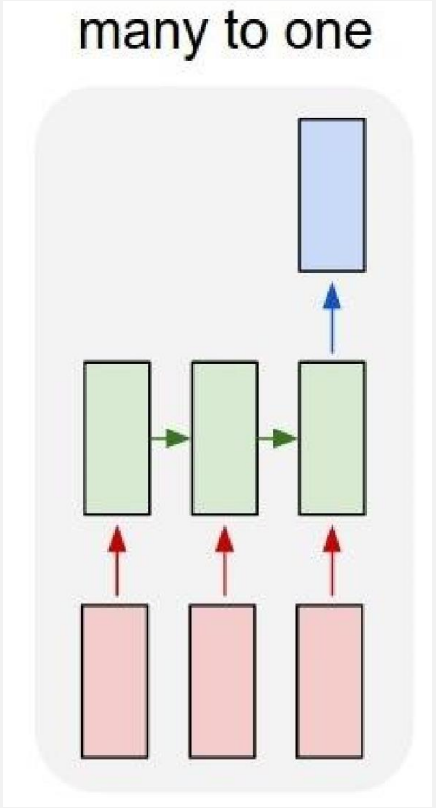


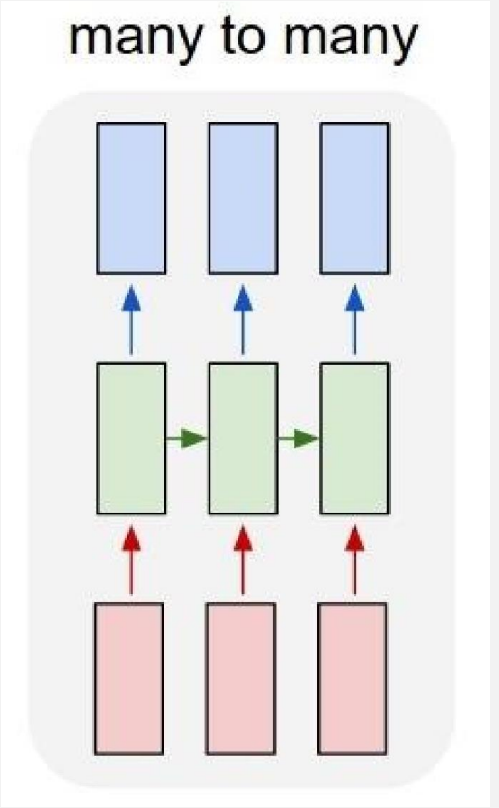
$$\hat{x} = h_{W,b} \approx x$$

RECURRENT NEURAL NETWORKS

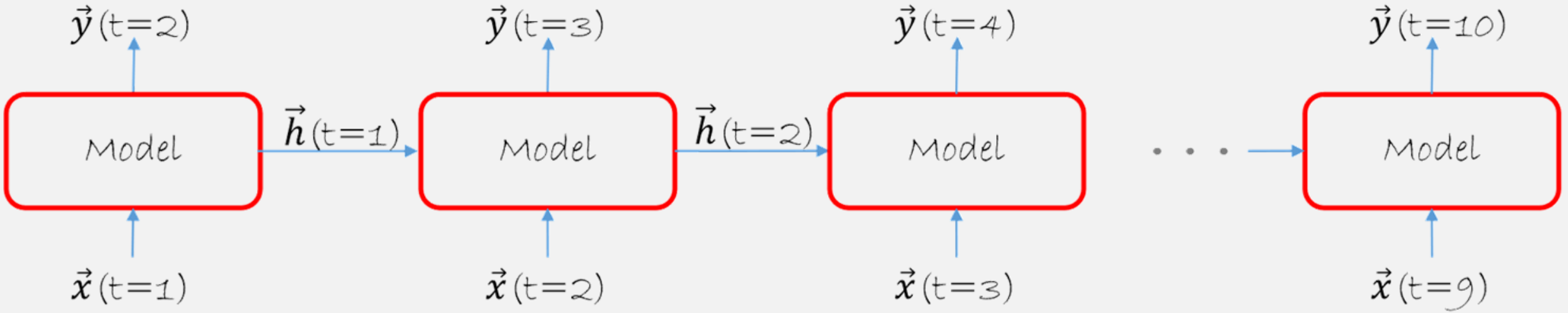
plain concepts

HANDLING SEQUENCES

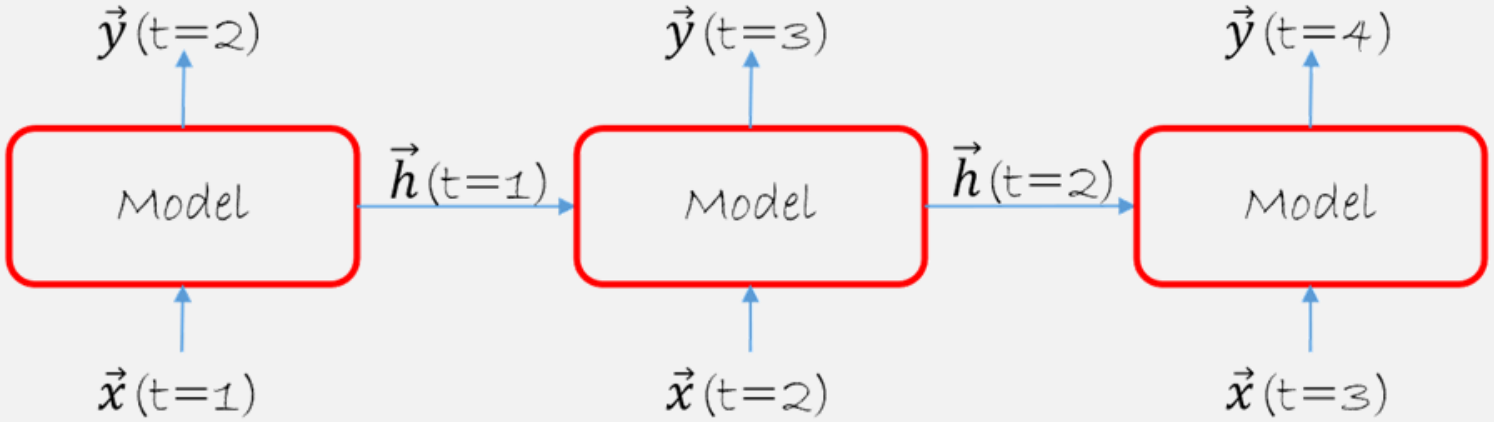


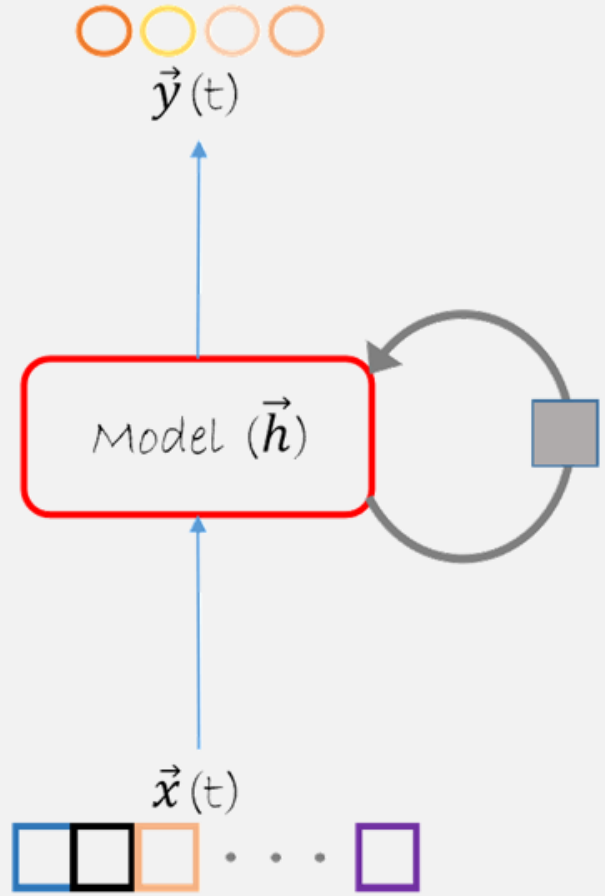


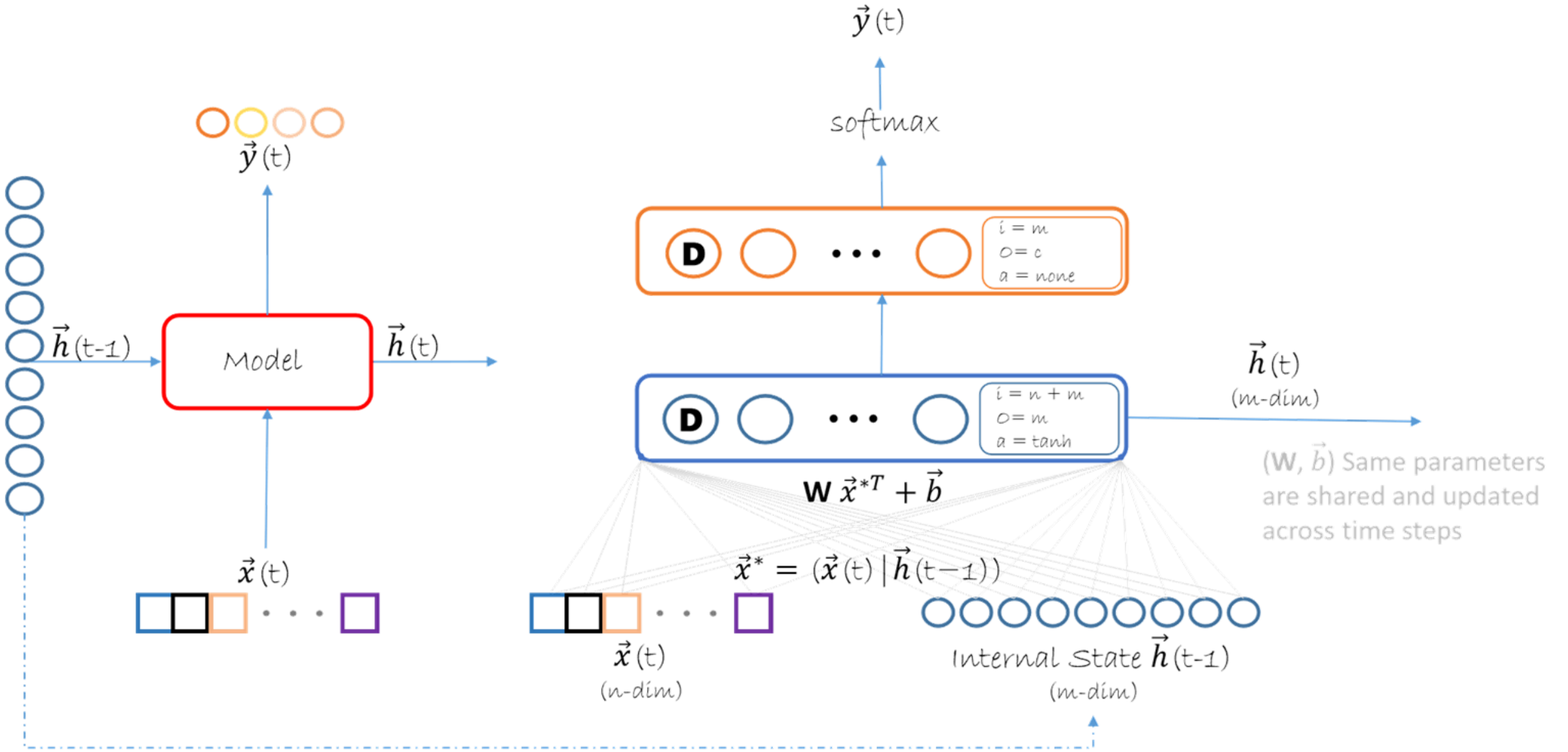
TRACKING THE HISTORY



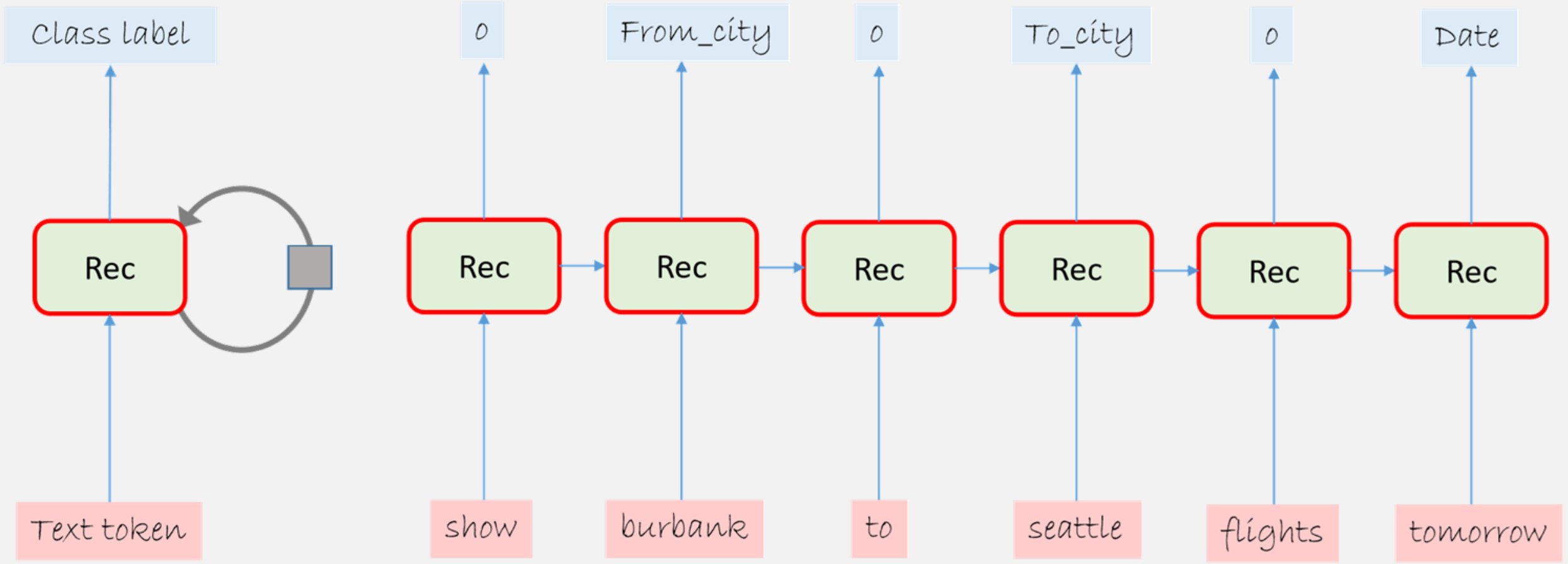
RECURRENCE


$\vec{y}(t=2)$
$\vec{y}(t=3)$
$\vec{y}(t=4)$
Model
$\vec{h}(t=1)$
Model
$\vec{h}(t=2)$
Model
$\vec{x}(t=1)$
$\vec{x}(t=2)$
$\vec{x}(t=3)$

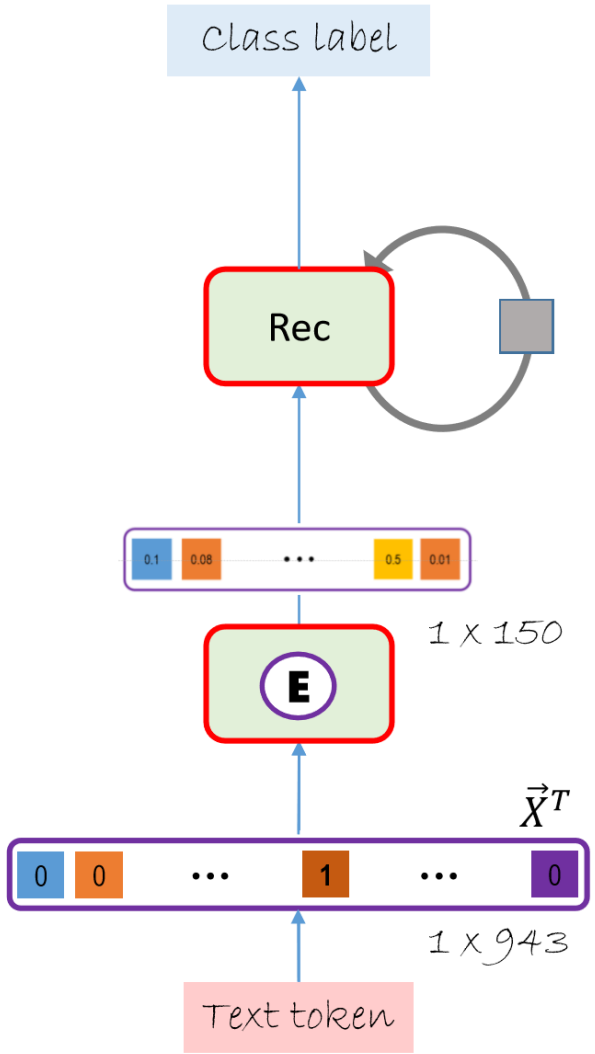

$\vec{y}(t)$
Model $(\vec{h})$
$\vec{x}(t)$

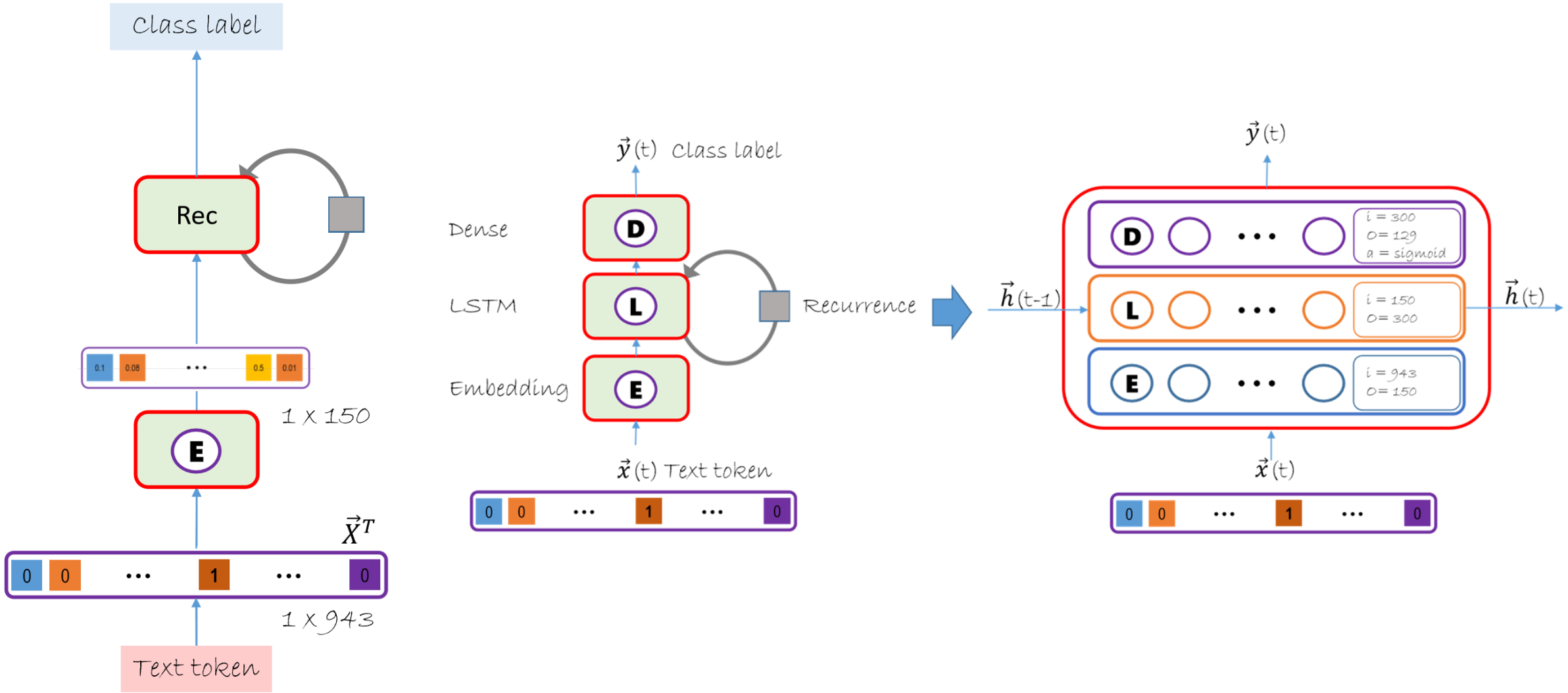

$\vec{y}$(t)
softmax
D
i = m
o = c
a = none
D
i = n + m
o = m
a = tanh
$\vec{h}$(t)
(m-dim)
$\mathbf{W}\,\vec{x}^{*T} + \vec{b}$
(W, $\vec{b}$) Same parameters are shared and updated across time steps
$\vec{x}^{*} = (\vec{x}(t) \mid \vec{h}(t-1))$
$\vec{y}$(t)
$\vec{h}$(t-1)
Model
$\vec{h}$(t)
$\vec{x}$(t)
$\vec{x}$(t)
(n-dim)
Internal State $\vec{h}$(t-1)
(m-dim)

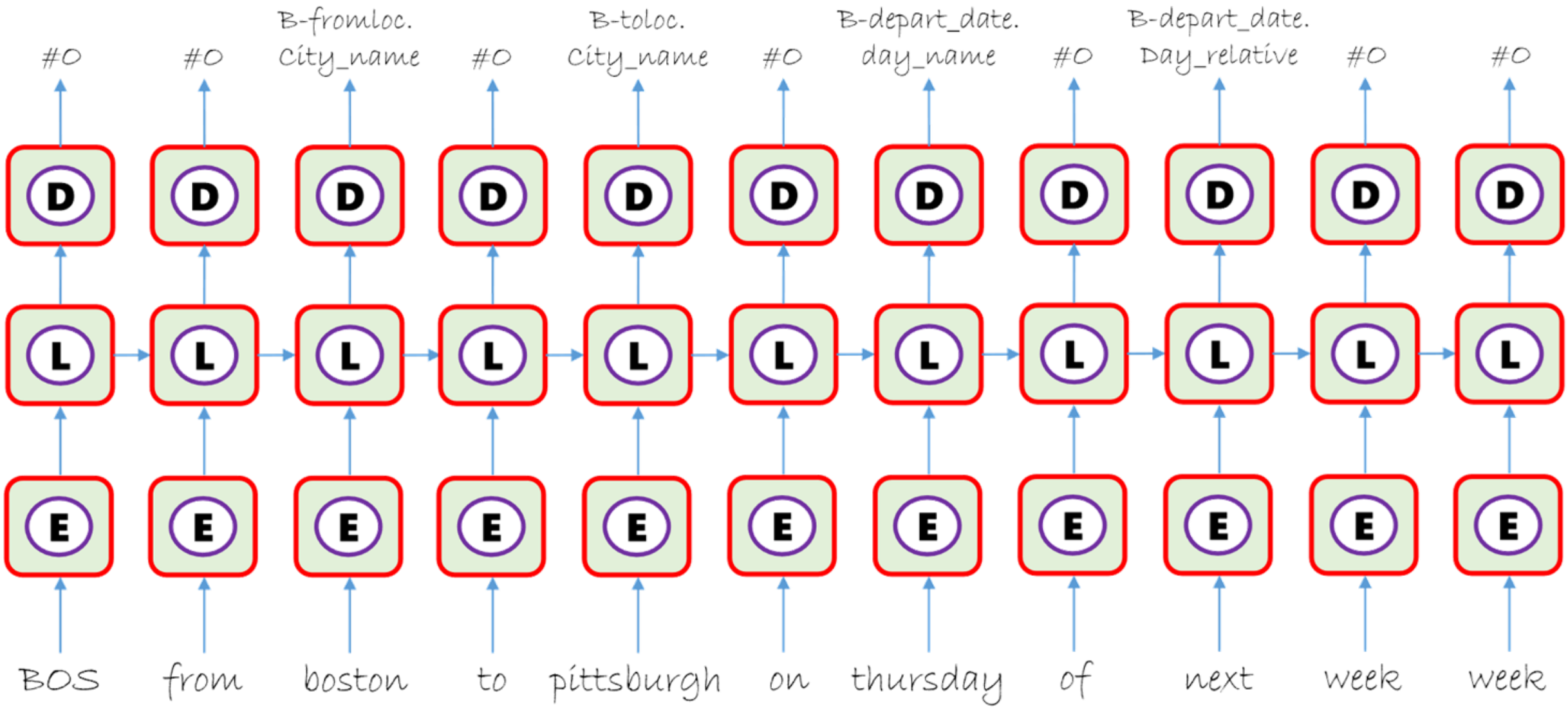
LONG SHORT TERM MEMORY



Sequence Id	Input Word (sample)	Word Index (in vocabulary) S0	Word Label	Label Index (S2)
19	# BOS	178:1	# O	128:1
19	# please	688:1	# O	128:1
19	# give	449:1	# O	128:1
19	# me	581:1	# O	128:1
19	# the	827:1	# O	128:1
19	# flights	429:1	# O	128:1
19	# from	444:1	# O	128:1
19	# boston	266:1	# B-fromloc.city_name	48:1
19	# to	851:1	# O	128:1
19	# pittsburgh	682:1	# B-toloc.city_name	78:1
19	# on	654:1	# O	128:1
19	# thursday	845:1	# B-depart_date.day_name	26:1
19	# of	646:1	# O	128:1
19	# next	621:1	# B-depart_date.date_relative	25:1
19	# week	910:1	# O	128:1
19	# EOS	179:1	# O	128:1


Class label
Rec
0.1
0.08
0.5
0.01
1 x 150
E
$\vec{X}^T$
0
0
1
0
1 x 943
Text token


Class label
Rec
0.1
0.08
0.5
0.01
1 X 150
E
$\vec{X}^T$
0
0
1
0
1 X 943
Text token
$\vec{y}$(t) Class label
Dense
D
LSTM
L
Recurrence
Embedding
E
$\vec{x}$(t) Text token
0
0
1
0
$\vec{y}$(t)
$\vec{h}$(t-1)
D
i = 300
o = 129
a = sigmoid
L
i = 150
o = 300
$\vec{h}$(t)
E
i = 943
o = 150
$\vec{x}$(t)
0
0
1
0


#O
#O
B-fromloc.
City_name
#O
B-toloc.
City_name
#O
B-depart_date.
day_name
#O
B-depart_date.
Day_relative
#O
#O
D
L
E
BOS
from
boston
to
pittsburgh
on
thursday
of
next
week
week

EMBEDINGS

WHAT IS WORD2VEC?

Two distinct models

- CBoW
- Skip-Gram

Various training methods

- Negative Sampling
- Hierarchical Softmax

A rich preprocessing pipeline

- Dynamic Context Windows
- Subsampling
- Deleting Rare Words

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

“word2vec Explained…”
Goldberg & Levy, arXiv 2014

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

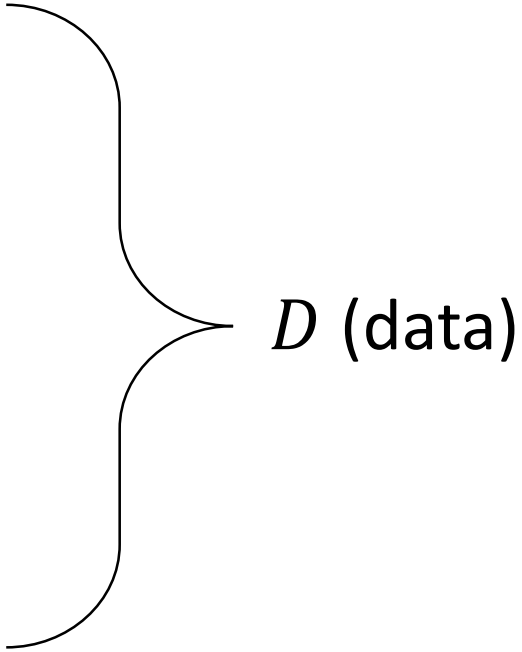
Marco saw a furry little wampimuk hiding in the tree.

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

Marco saw a furry little wampimuk hiding in the tree.

words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in
…	…



"word2vec Explained…"
Goldberg & Levy, arXiv 2014

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

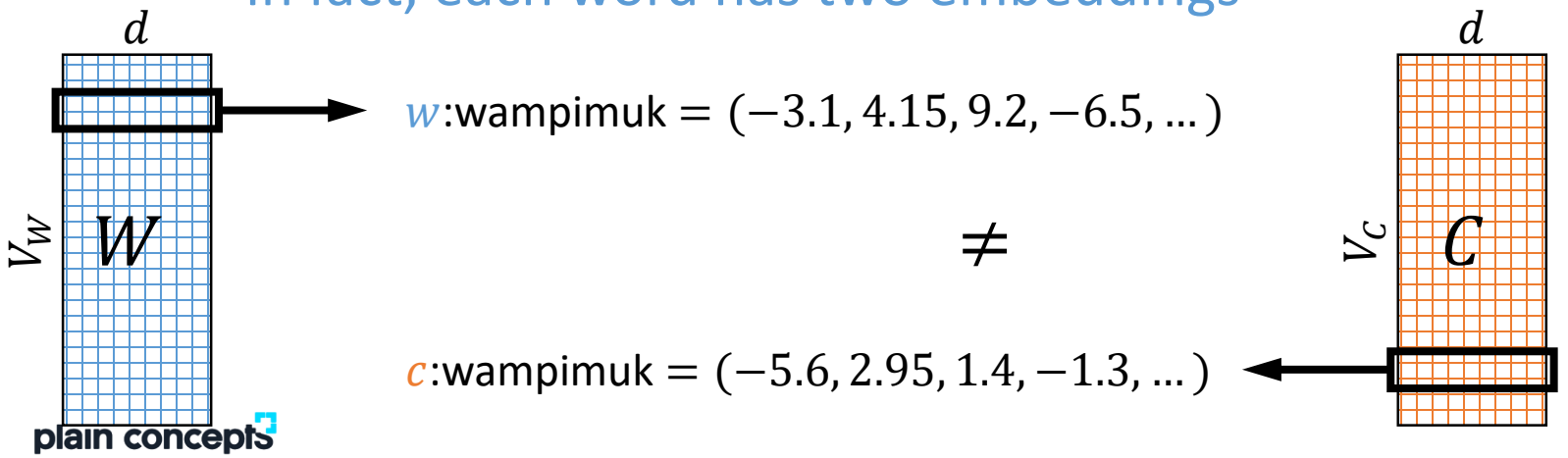
SGNS finds a vector $\vec{w}$ for each word w in our vocabulary V_W

Each such vector has d latent dimensions (e.g. $d = 100$)

Effectively, it learns a matrix W whose rows represent V_W

Key point: it also learns a similar auxiliary matrix C of context vectors

In fact, each word has two embeddings



"word2vec Explained…"
Goldberg & Levy, arXiv 2014

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

- **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$
  - c was **observed** with w

words	**contexts**
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

- **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$
  - c was **observed** with w

words	contexts
wampimuk	furry
wampimuk	little
wampimuk	hiding
wampimuk	in

- **Minimize:** $\sigma(\vec{w} \cdot \vec{c}')$
  - c' was **hallucinated** with w

words	contexts
wampimuk	Australia
wampimuk	cyber
wampimuk	the
wampimuk	1985

"word2vec Explained…"
Goldberg & Levy, arXiv 2014

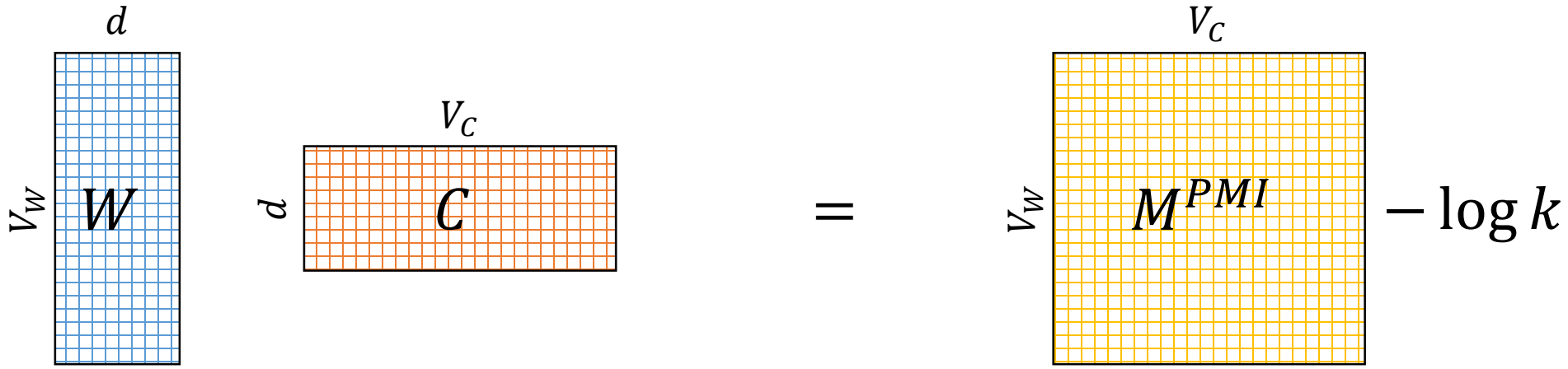
SKIP-GRAMS WITH NEGATIVE SAMPLING (SGNS)

- "Negative Sampling"
- SGNS samples k contexts c' **at random** as **negative examples**
- "Random" = unigram distribution

$$P(c) = \frac{\#c}{|D|}$$

- **Spoiler:** Changing this distribution has a significant effect

WHAT IS SGNS LEARNING?



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

SOME OTHER ADVANCED IDEAS

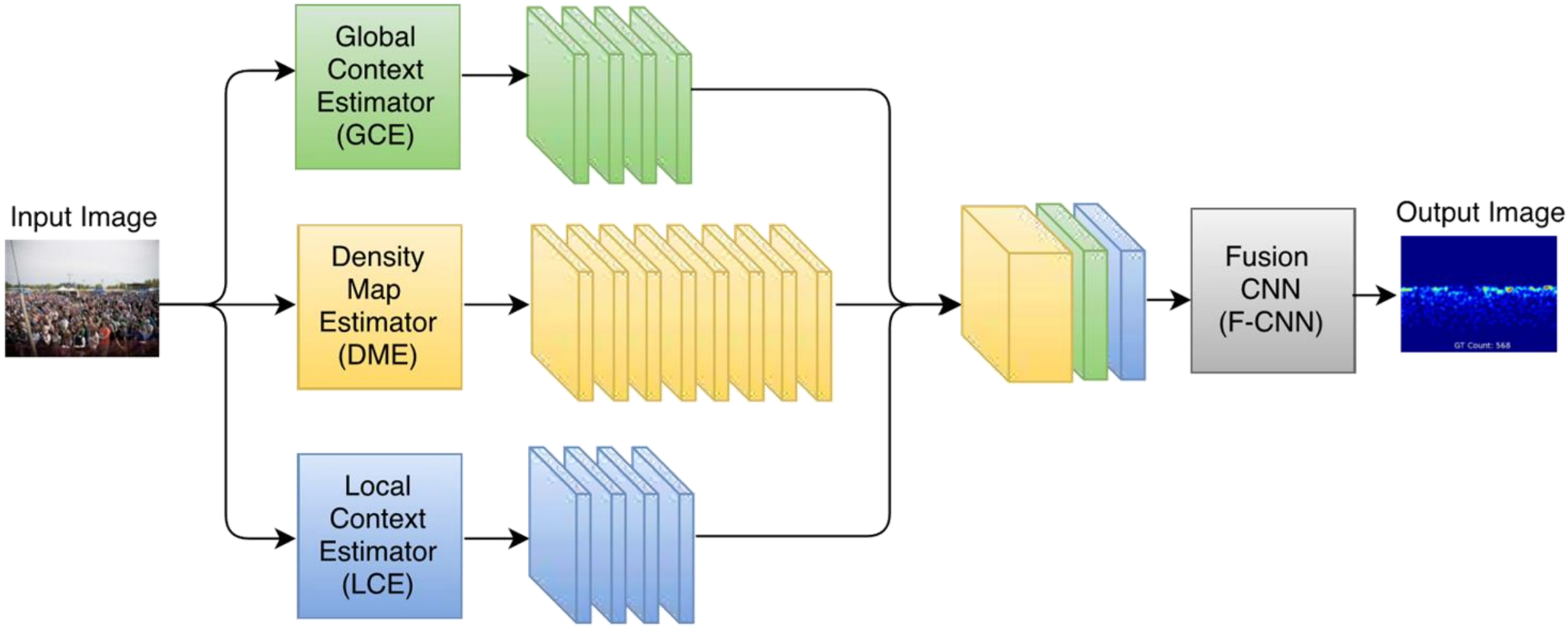
Counting people in a multitude

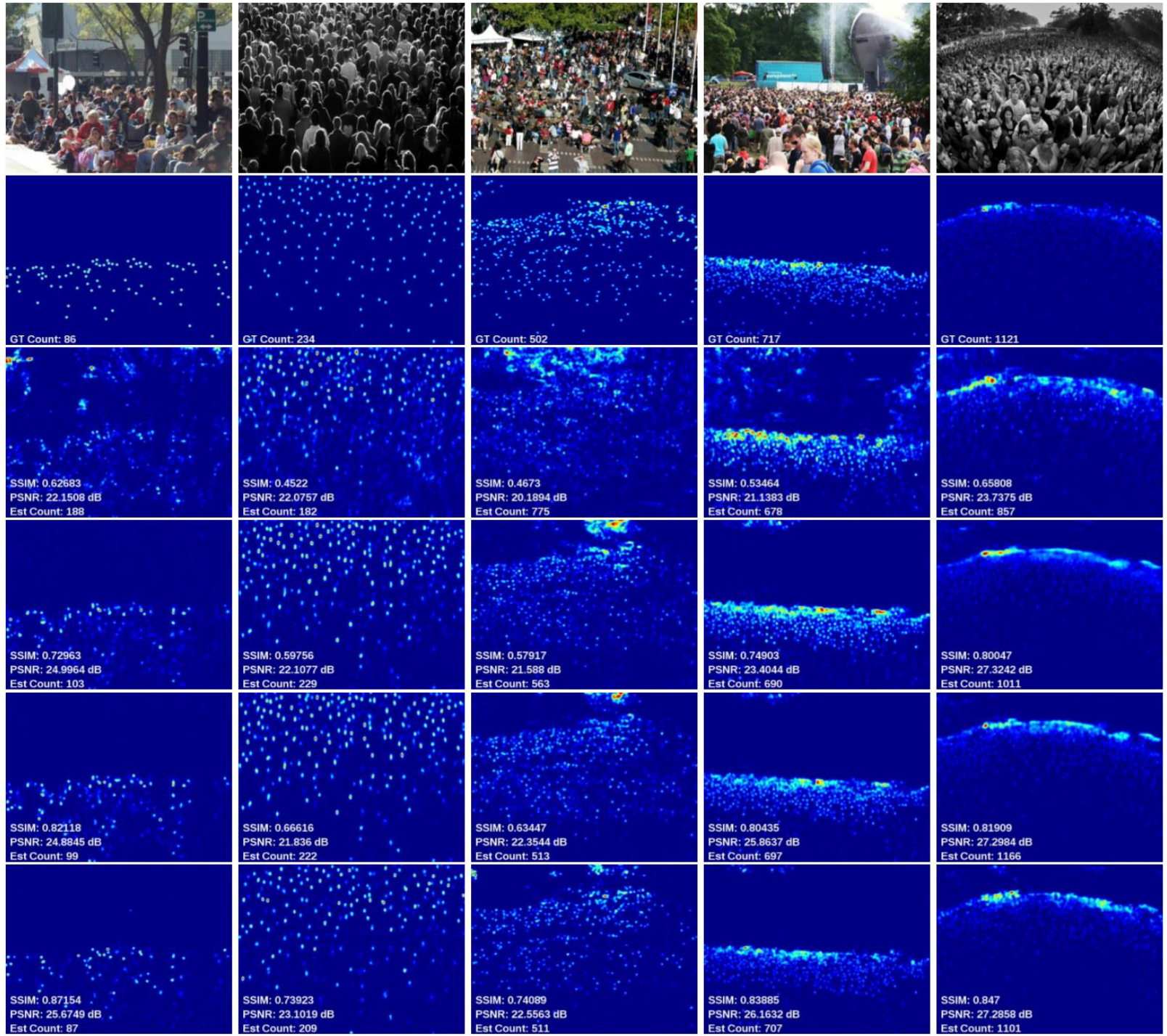
- CONTEXTUAL PYRAMID CNN (CP-CNN)

A new method named Contextual Pyramid CNN (CP-CNN) is proposed here to generate density maps and influx estimations, by explicitely incorporating global and local context information. Composed of four modules: Global Context Estimator (GCE), Local Context Estimator (LCE), Density Map Estimator (DME) and a Fusion-CNN (F-CNN) convolutional network.

Vishwanath A. Sindagi, Vishal M. Patel; The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1861-1870

plain concepts


Input Image
Global Context Estimator (GCE)
Density Map Estimator (DME)
Local Context Estimator (LCE)
Fusion CNN (F-CNN)
Output Image
GT Count: 568

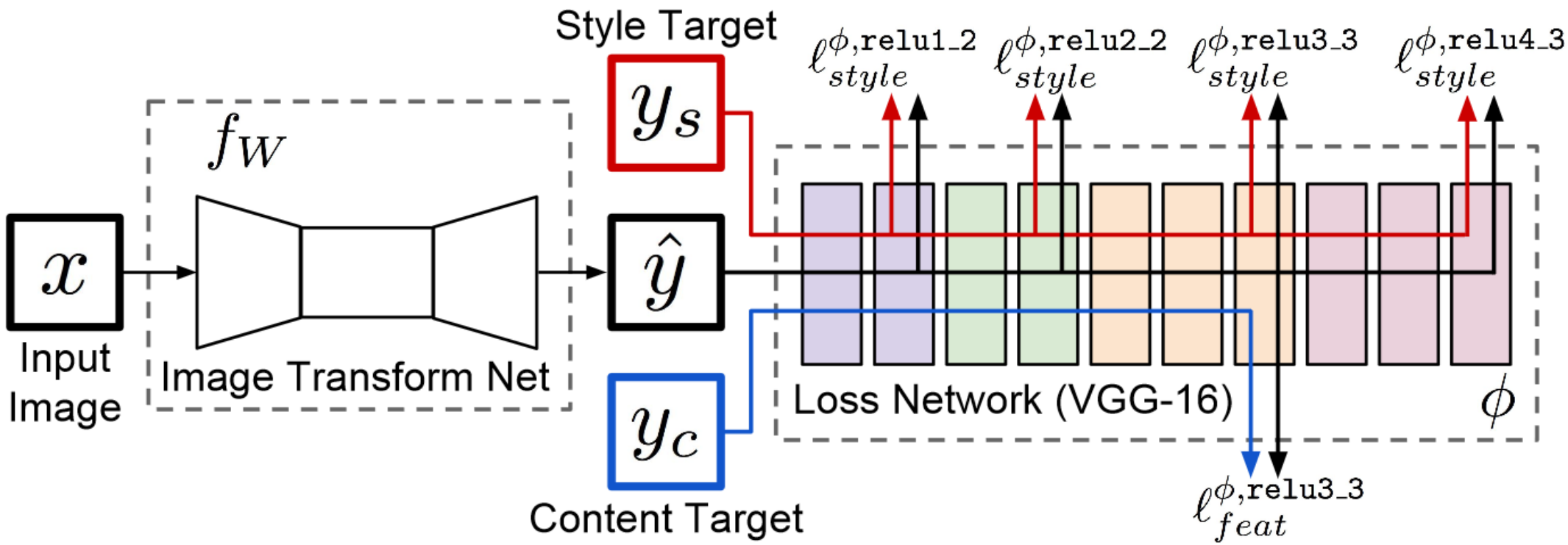

GT Count: 86
GT Count: 234
GT Count: 502
GT Count: 717
GT Count: 1121
SSIM: 0.62683
PSNR: 22.1508 dB
Est Count: 188
SSIM: 0.4522
PSNR: 22.0757 dB
Est Count: 182
SSIM: 0.4673
PSNR: 20.1894 dB
Est Count: 775
SSIM: 0.53464
PSNR: 21.1383 dB
Est Count: 678
SSIM: 0.65808
PSNR: 23.7375 dB
Est Count: 857
SSIM: 0.72963
PSNR: 24.9964 dB
Est Count: 103
SSIM: 0.59756
PSNR: 22.1077 dB
Est Count: 229
SSIM: 0.57917
PSNR: 21.588 dB
Est Count: 563
SSIM: 0.74903
PSNR: 23.4044 dB
Est Count: 690
SSIM: 0.80047
PSNR: 27.3242 dB
Est Count: 1011
SSIM: 0.82118
PSNR: 24.8845 dB
Est Count: 99
SSIM: 0.66616
PSNR: 21.836 dB
Est Count: 222
SSIM: 0.63447
PSNR: 22.3544 dB
Est Count: 513
SSIM: 0.80435
PSNR: 25.8637 dB
Est Count: 697
SSIM: 0.81909
PSNR: 27.2984 dB
Est Count: 1166
SSIM: 0.87154
PSNR: 25.6749 dB
Est Count: 87
SSIM: 0.73923
PSNR: 23.1019 dB
Est Count: 209
SSIM: 0.74089
PSNR: 22.5563 dB
Est Count: 511
SSIM: 0.83885
PSNR: 26.1632 dB
Est Count: 707
SSIM: 0.847
PSNR: 27.2858 dB
Est Count: 1101

Transferring style between images

- CONVOLUTIONAL NEURAL NETWORKS

By using a perceptual loss functions based on high-level features extracted from pretrained networks, networks for image transformation tasks can be trained, and by fine tuning the loss function different features can be kept for the source image and the style image.

Justin Johnson, Alexandre Alahi, Li Fei-Fei; Perceptual Losses for Real-Time Style Transfer and Super-Resolution, 2016


Style Target
y_s
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Content Target
$\ell^{\phi,\texttt{relu1\_2}}_{style}$
$\ell^{\phi,\texttt{relu2\_2}}_{style}$
$\ell^{\phi,\texttt{relu3\_3}}_{style}$
$\ell^{\phi,\texttt{relu4\_3}}_{style}$
Loss Network (VGG-16)
ϕ
$\ell^{\phi,\texttt{relu3\_3}}_{feat}$

Using GANs to drive design decisions

- GENERATIVE ADVERSARIAL NETWORKS

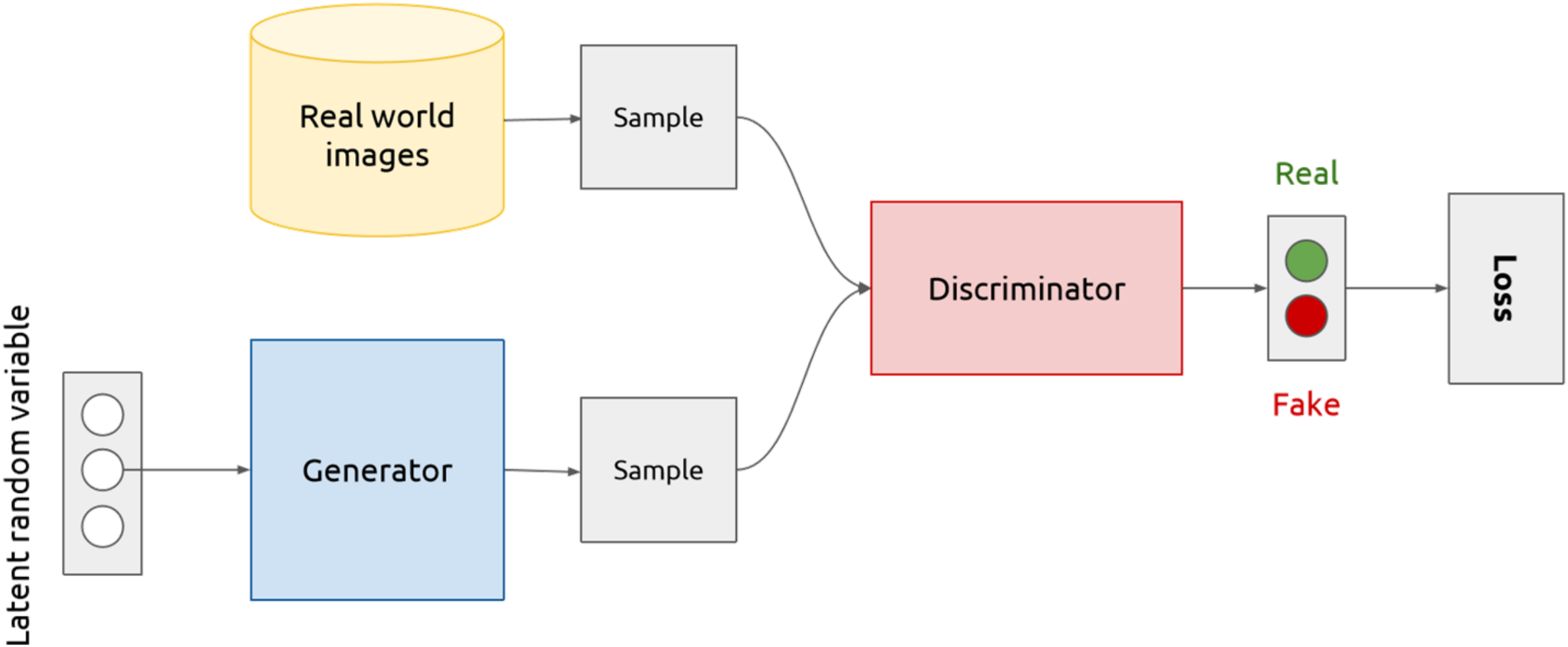
By taking advantage of Generational Adversarial Networks, synthetic images based on the training data can be generated. Including an external array of features, the generated images can be tailored to a specific set of requirements.

Jaime Deverall, Jiwoo Lee, Miguel Ayala; Using Generative Adversarial Networks to Design Shoes

plain concepts


Text description
This bird is blue with white and has a very short beak
This bird has wings that are brown and has a yellow belly
A white bird with a black crown and yellow beak
This bird is white, black, and brown in color, with a brown beak
The bird has small beak, with reddish brown crown and gray belly
This is a small, black bird with a white breast and white on the wingbars.
This bird is white black and yellow in color, with a short black beak
Stage-I images
Stage-II images


Input
Ground truth
Output


Real world images
Sample
Latent random variable
Generator
Sample
Discriminator
Real
Fake
Loss

AZURE MACHINE LEARNING

DATA SCIENCE & AI

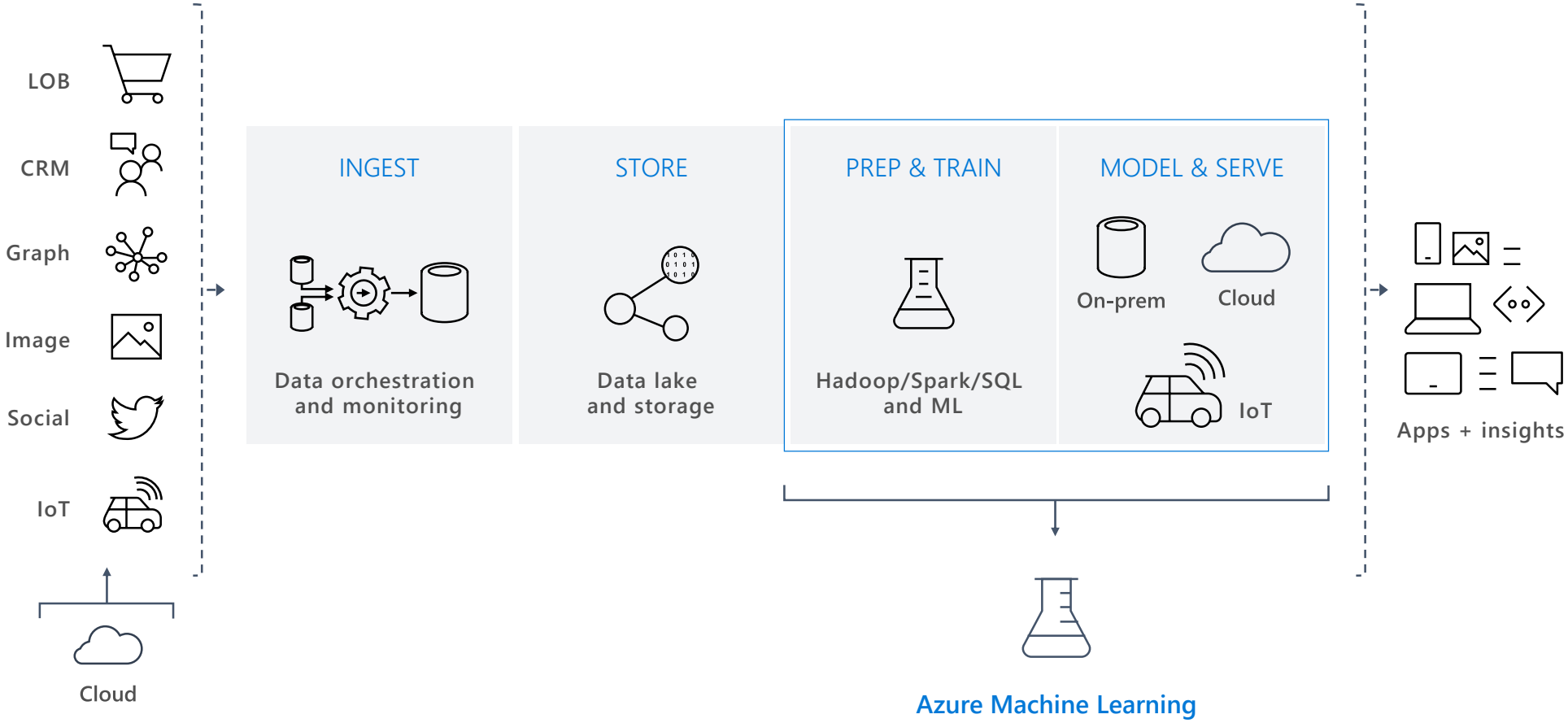
KEY TRENDS

- Accelerating adoption of AI by developers (consuming models)
- Rise of hybrid training and scoring scenarios
- Push scoring/inference to the event (edge, cloud, on-prem)
- Some developers moving into deep learning as non-traditional path to DS / AI dev
- Growth of diverse hardware arms race across all form factors (CPU / GPU / FPGA / ASIC / device)

CHALLENGES

- ⚠ Data prep
- ⚠ Model deployment & management
- ⚠ Model lineage & auditing
- ⚠ Explain-ability

THE AI DEVELOPMENT LIFECYCLE



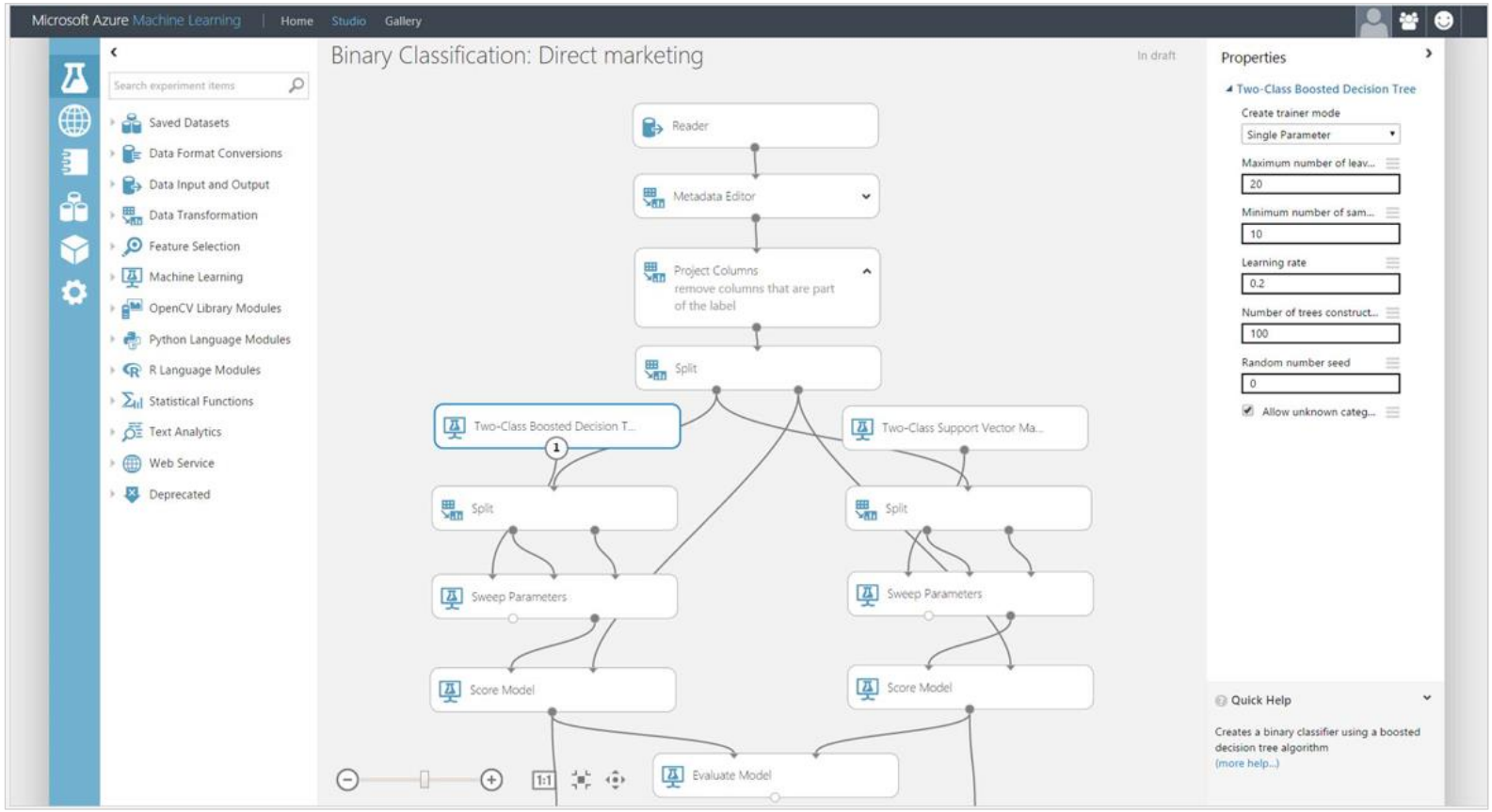
AZURE MACHINE LEARNING STUDIO

Platform for emerging data scientists to graphically build and deploy experiments

- Rapid experiment composition
- > 100 easily configured modules for data prep, training, evaluation
- Extensibility through R & Python
- Serverless training and deployment

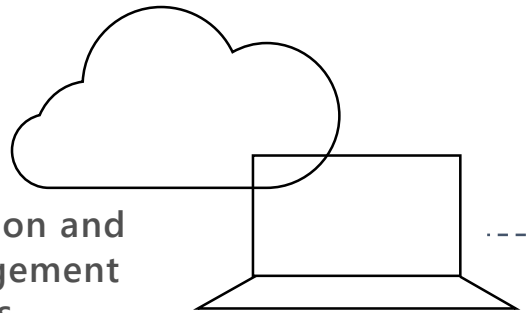
Some numbers:

- 100's of thousands of deployed models serving billions of requests



NEW CAPABILITIES

AZURE MACHINE LEARNING SERVICES



Experimentation and Model Management Services

Notebooks
IDEs
Azure Machine Learning Workbench
VS Code Tools for AI

TRAIN & DEPLOY OPTIONS

AZURE

- Spark
- SQL Server
- Virtual machines
- GPUs
- Container services

ON-PREMISES

- SQL Server
- Machine Learning Server

EDGE

- Azure IoT Edge

New
Dashboard
Data factories
SQL databases
HDInsight clusters
Resource groups
Storage accounts
Data Lake Analytics
Storage accounts (class...
All resources
Recent
Data Lake Store
App Services
Virtual machines
Subscriptions
SQL servers
Virtual machines (classic)
Virtual networks
Cloud services (classic)
More services >

Marketplace

Everything
Compute
Networking
Storage
Web + Mobile
Databases
Data + Analytics
AI + Cognitive Services
Internet of Things
Enterprise Integration
Security + Identity
Developer tools
Monitoring + Management
Add-ons
Containers
Blockchain

AI + Cognitive Services

Filter

Search AI + Cognitive Services

Machine Learning Experimenta...

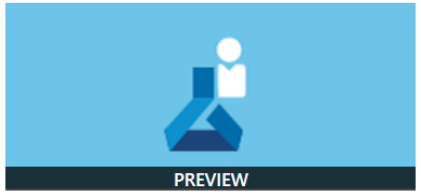
Microsoft

Create an Experimentation Account to start preparing your data, engineer features, author models using existing tools and manage reproducibility through tracking history of runs of all your machine learning models.

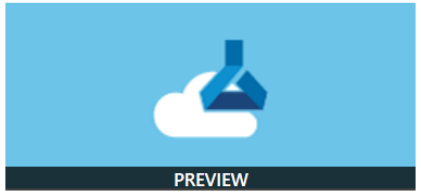
Create

Machine Learning

More



Machine Learning Experimentation (preview)
Microsoft

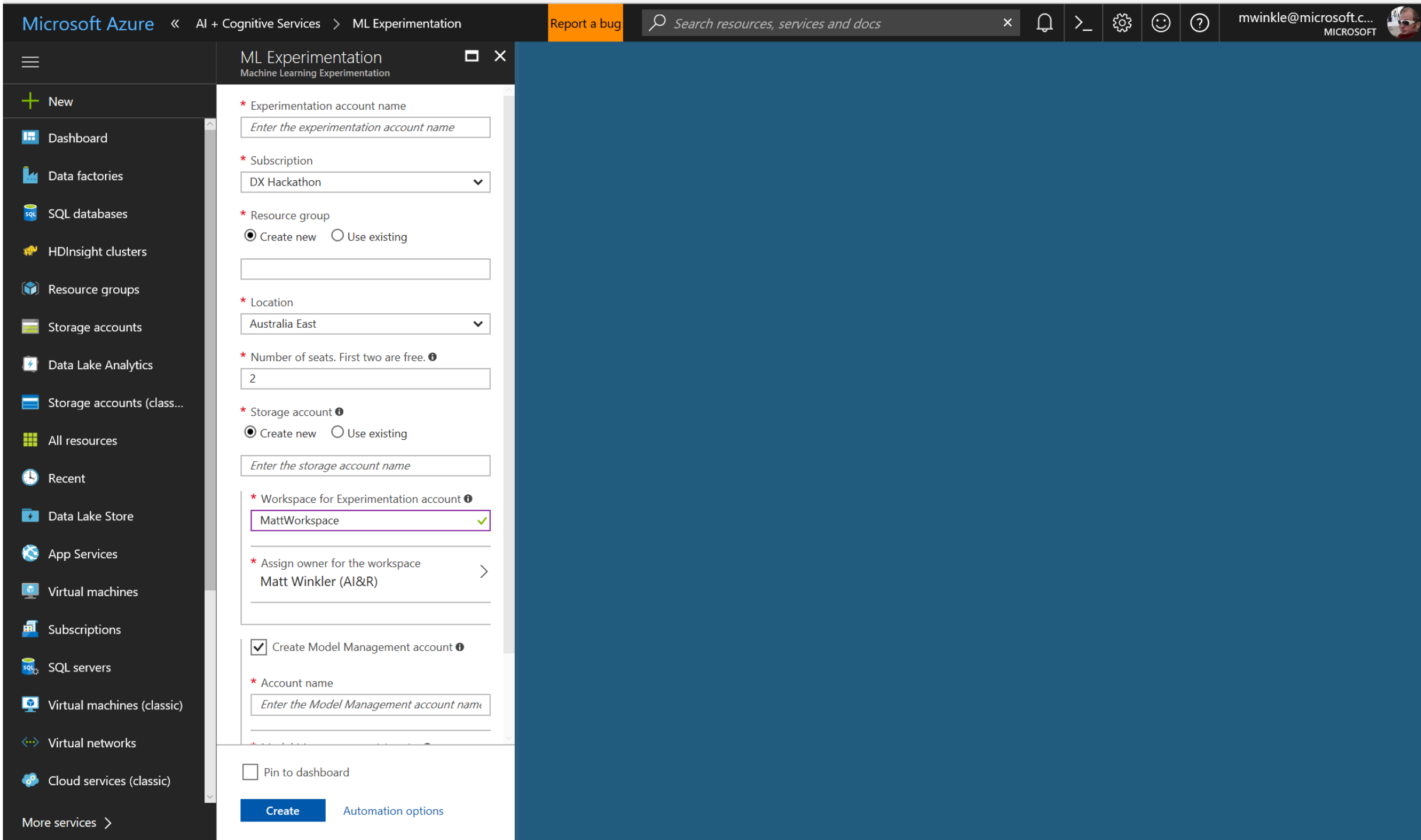


Machine Learning Model Management (preview)
Microsoft

Data Science Virtual Machine - Windows 2016
Microsoft

Bot Service

More


Microsoft Azure
AI + Cognitive Services
ML Experimentation
Report a bug
Search resources, services and docs
mwinkle@microsoft.c...
MICROSOFT
New
Dashboard
Data factories
SQL databases
HDInsight clusters
Resource groups
Storage accounts
Data Lake Analytics
Storage accounts (class...
All resources
Recent
Data Lake Store
App Services
Virtual machines
Subscriptions
SQL servers
Virtual machines (classic)
Virtual networks
Cloud services (classic)
More services
ML Experimentation
Machine Learning Experimentation
* Experimentation account name
Enter the experimentation account name
* Subscription
DX Hackathon
* Resource group
Create new
Use existing
* Location
Australia East
* Number of seats. First two are free.
2
* Storage account
Create new
Use existing
Enter the storage account name
* Workspace for Experimentation account
MattWorkspace
* Assign owner for the workspace
Matt Winkler (AI&R)
Create Model Management account
* Account name
Enter the Model Management account nam
Pin to dashboard
Create
Automation options

DemoExp1
Machine Learning Experimentation - PREVIEW

New
Dashboard
Data factories
SQL databases
HDInsight clusters
Resource groups
Storage accounts
Data Lake Analytics
Storage accounts (class...
All resources
Recent
Data Lake Store
App Services
Virtual machines
Subscriptions
SQL servers
Virtual machines (classic)
Virtual networks
Cloud services (classic)
More services

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

APPLICATION SETTINGS
Update seats
Add workspace
Resync Storage Keys

SETTINGS
Locks
Automation script
Properties

SUPPORT + TROUBLESHOOTING
New support request

Delete | Add workspace | Update seats

Resource group
projectviennaDemos

Status
Enabled

Location
East US 2

Subscription
Project Vienna Demo 1

Subscription ID

Storage
projectviennademostorage

Number of seats
2

Number of workspaces
4

Getting Started

Download Azure Machine Learning Workbench for Windows
Azure Machine Learning Workbench enables you to explore data, build experiments, and compare model performance.

Download Azure Machine Learning Workbench for Mac OS X
Azure Machine Learning Workbench enables you to explore data, build experiments, and compare model performance.

Create an account for Machine Learning Model Management
Machine Learning Model Management accounts allow you to easily deploy your models and manage them in Azure.

Learn how to configure Machine Learning Compute for Model Management
To take advantage of deploying and managing your models in production, you must first configure the environment where the models will be deployed. Learn how to set up your environment to deploy and manage your models in Azure.

WHAT DID I GET?

Experimentation Account

- Keep track of projects
- Local, scale-up, and scale-out training
- Run history tracking

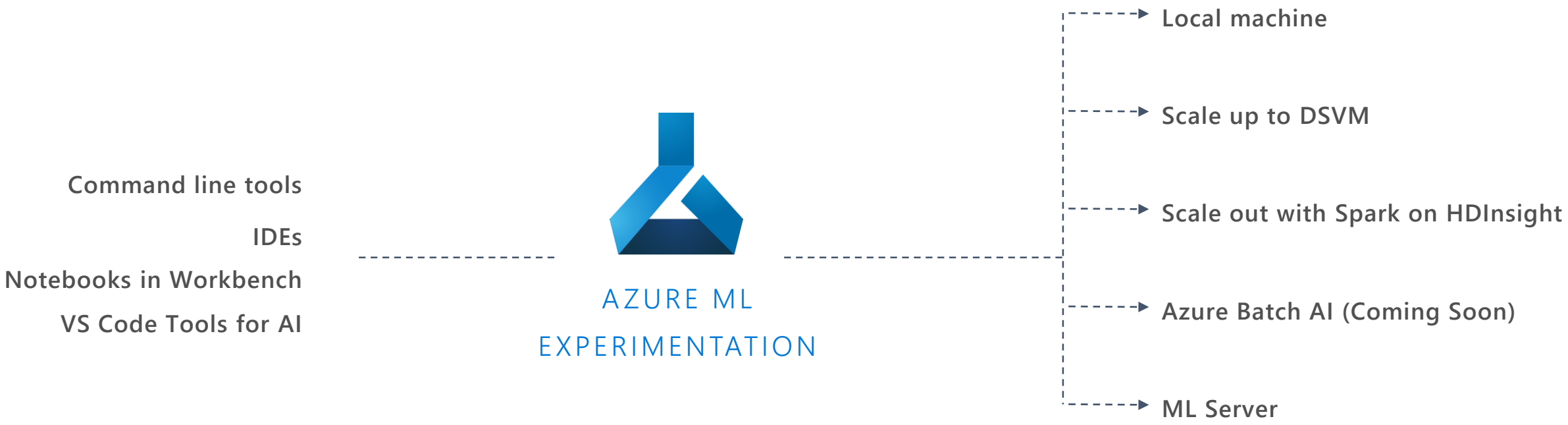
Model Management Account

- Create containers for models
- Manage and monitor deployed models

EXPERIMENTATION SERVICE

plain concepts

Experiment Everywhere



EXPERIMENTATION SERVICE

Manage project dependencies

Manage training jobs locally, scaled-up or scaled-out

Git based checkpointing and version control

Service side capture of run metrics, output logs and models

Use your favorite IDE, and any framework

USE ANY FRAMEWORK OR LIBRARY

USE ANY TOOL

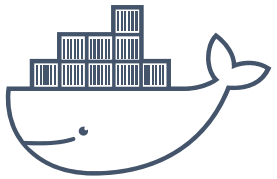
USE THE MOST POPULAR INNOVATIONS

MODEL MANAGEMENT SERVICE

plain concepts

DEPLOY EVERYWHERE

AZURE ML
MODEL MANAGEMENT



DOCKER

- Single node deployment (cloud/on-prem)
- Azure Container Service
- Azure IoT Edge
- Microsoft ML Server
- Spark clusters
- SQL Server

MANAGE MODELS

Deployment and management of models as HTTP services

Container-based hosting of real time and batch processing

Management and monitoring through Azure Application Insights

First class support for SparkML, Python, Cognitive Toolkit, TF, R, extensible to support others (Caffe, MXnet)

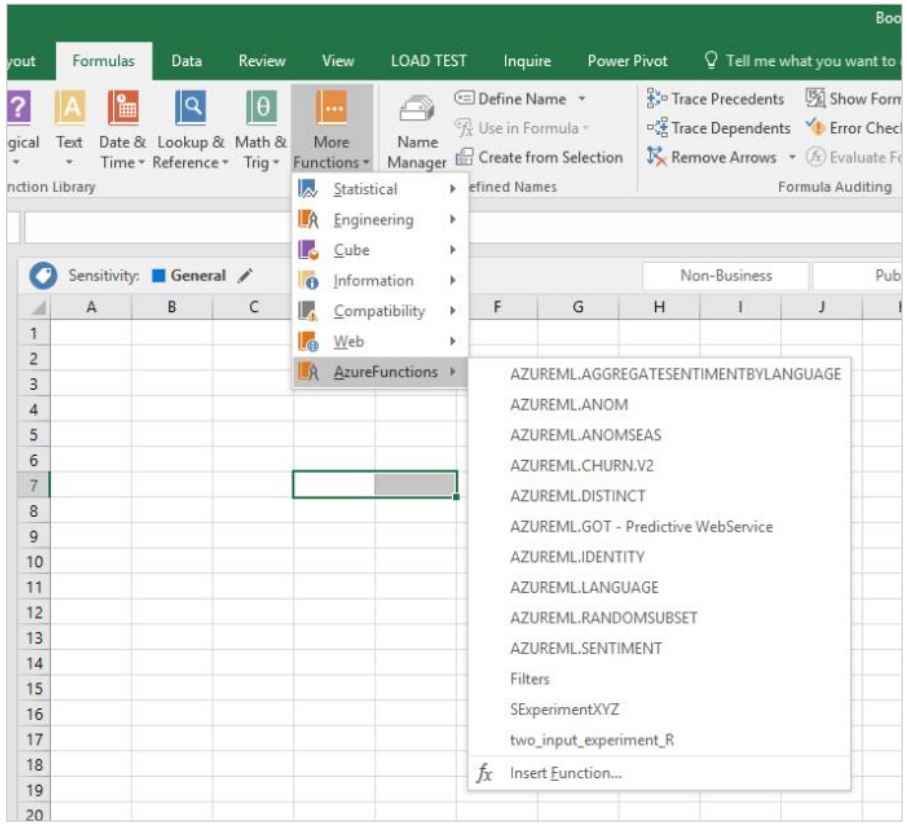
Service authoring in Python

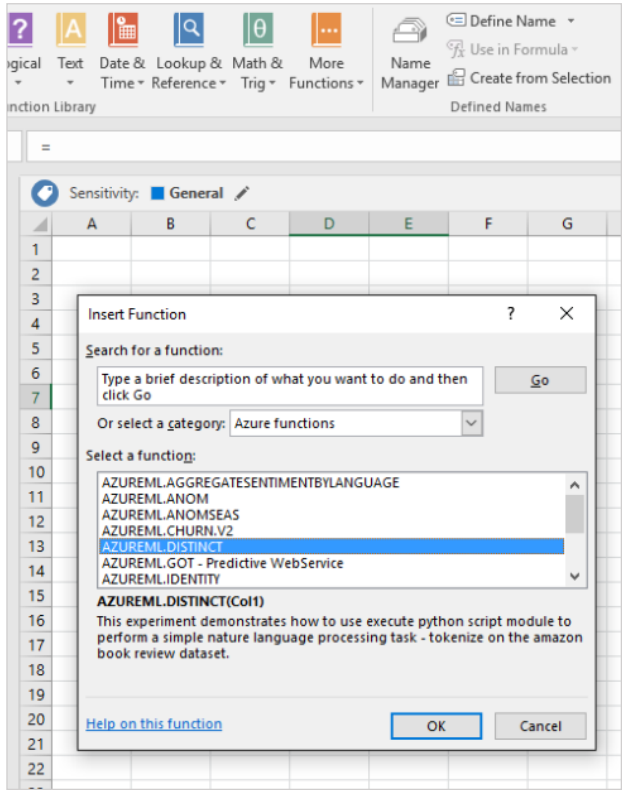
```
# -n app name
# -f scoring script file name
# -m dependencies, in this case it is the pickled model file
# -r type of model, in this case it is the scikit-learn model
$ az ml service create realtime -n irisapp -f iris_score.py -m model.pkl -r scikit-py
```

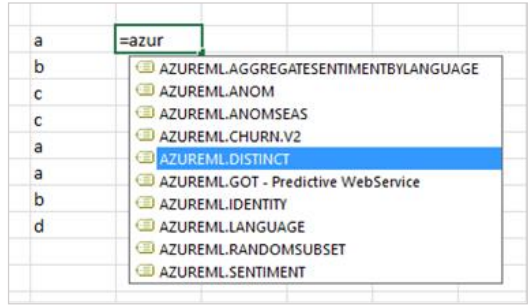
```
$ az ml service run realtime -n irisapp -d '{"input": "[[1.0, 2.2, 2.3, 2.7]]"}'
2
```

AI POWERED SPREADSHEETS

Any model published through Model Management can be called from Excel







plain concepts

VS CODE TOOLS FOR AI

plain concepts

VS CODE TOOLS FOR AI

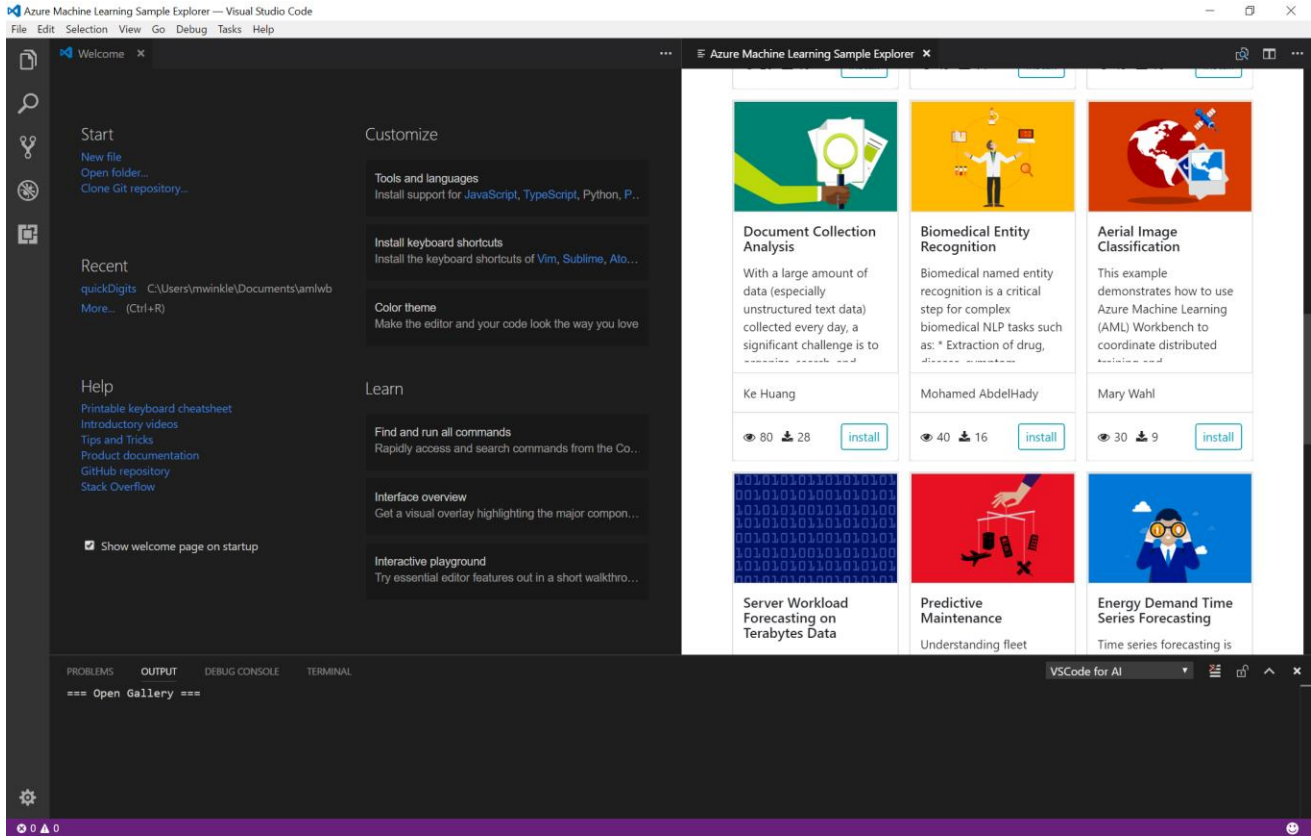
VS Code extension with deep integration to Azure ML

End to end development environment, from new project through training

Support for remote training

Job management

On top of all of the goodness of VS Code (Python, Jupyter, Git, etc)



plain concepts

WORKBENCH

plain concepts

WHAT IS IT?

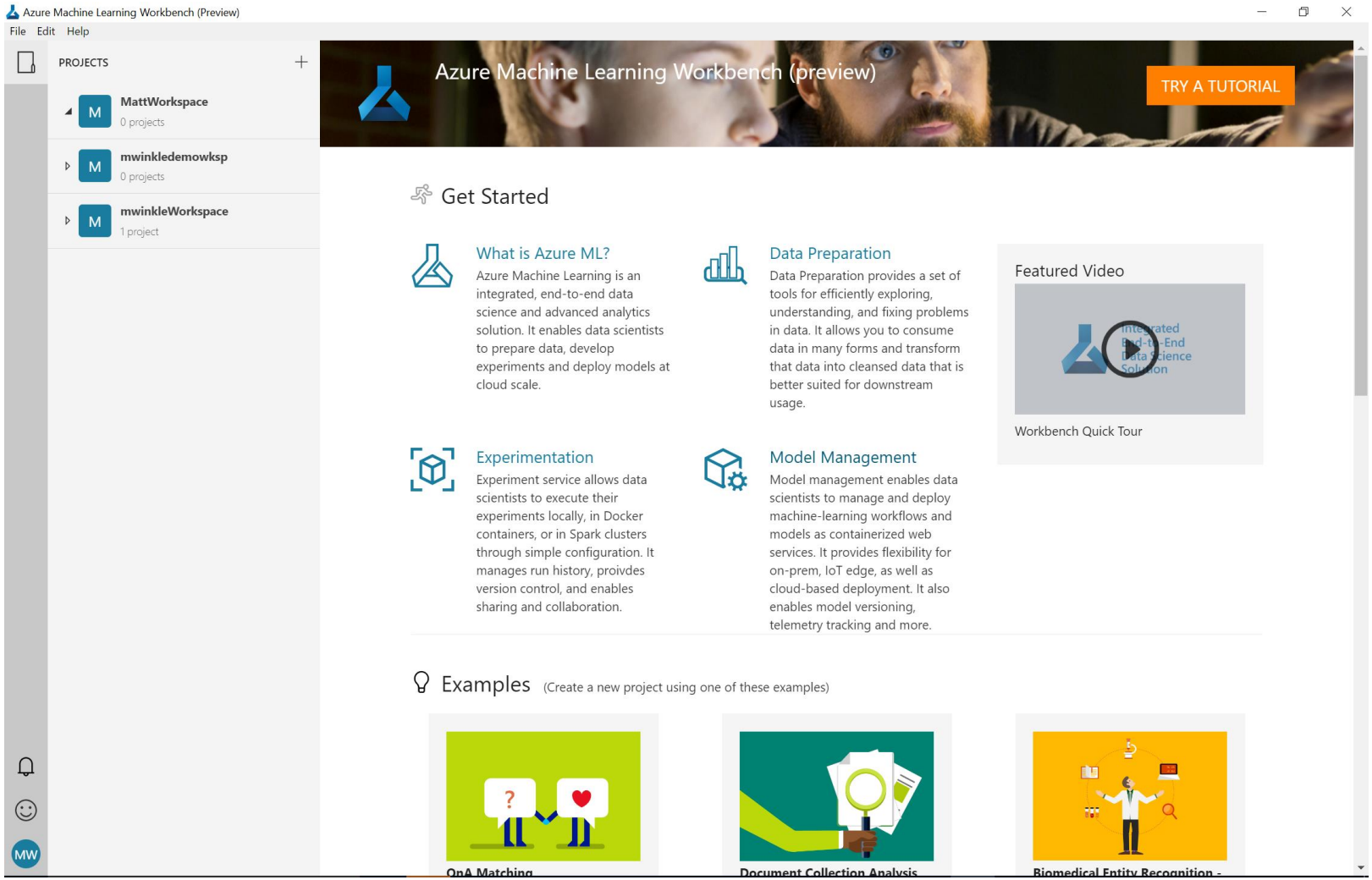
Windows and Mac based companion for AI development

Full environment set up (Python, Jupyter, etc)

Embedded notebooks

Run History and Comparison experience

New data wrangling tools



plain concepts

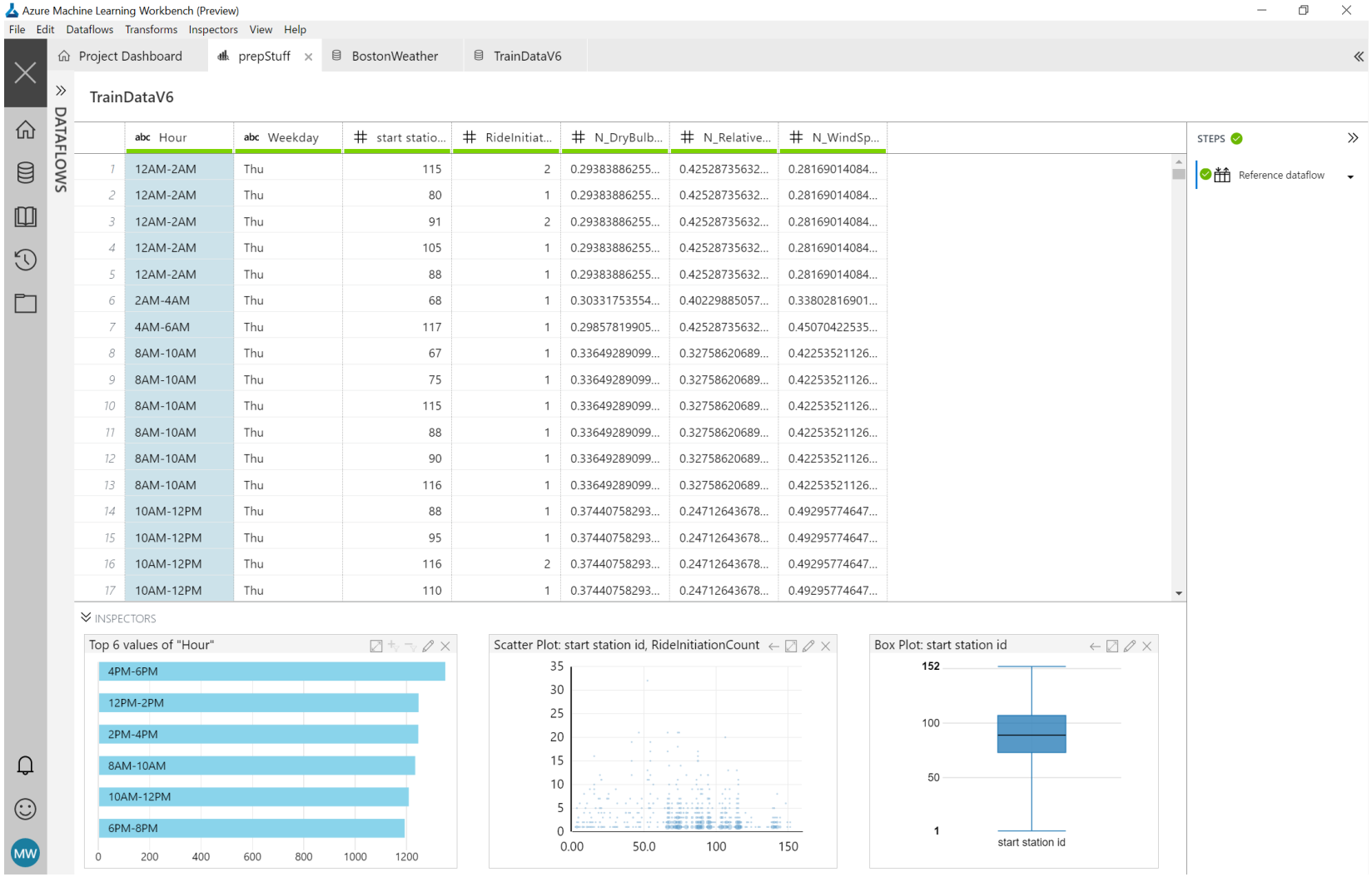
AI POWERED DATA WRANGLING

Rapidly sample, understand, and prep data

Leverage PROSE and more for intelligent, data prep by example

Extend/customize transforms and featurization through Python

Generate Python and Pyspark for execution at scale



plain concepts

MACHINE LEARNING & AI PORTFOLIO

WHEN TO USE WHAT?

Build your own or consume pre-trained models?

Which experience do you want?

Deployment target

What engine(s) do you want to use?