AN INTRODUCTION TO DEEP LEARNING FOR ASTRONOMY

Marc Huertas-Company

Jerusalem, March 2nd, 3rd 2019

REFERENCES

SEVERAL SLIDES / INFOS SHOWN HERE ARE INSPIRED/ TAKEN FROM OTHER WORKS / COURSES FOUND ONLINE

- Deep Learning: Do-It-Yourself! [Bursuc, Krzakala, Lelarge]
- DEEPLEARNING.AI [COURSERA, Ng, Bensouda, Katanforoosh]
- MACHINE LEARNING LECTURES [Keck]
- EPFL DEEP LEARNING COURSE [Fleuret]

Thanks to all of them!

SOME PRELIMINARY NOTES

I AM NOT A MACHINE LEARNING RESEARCHER

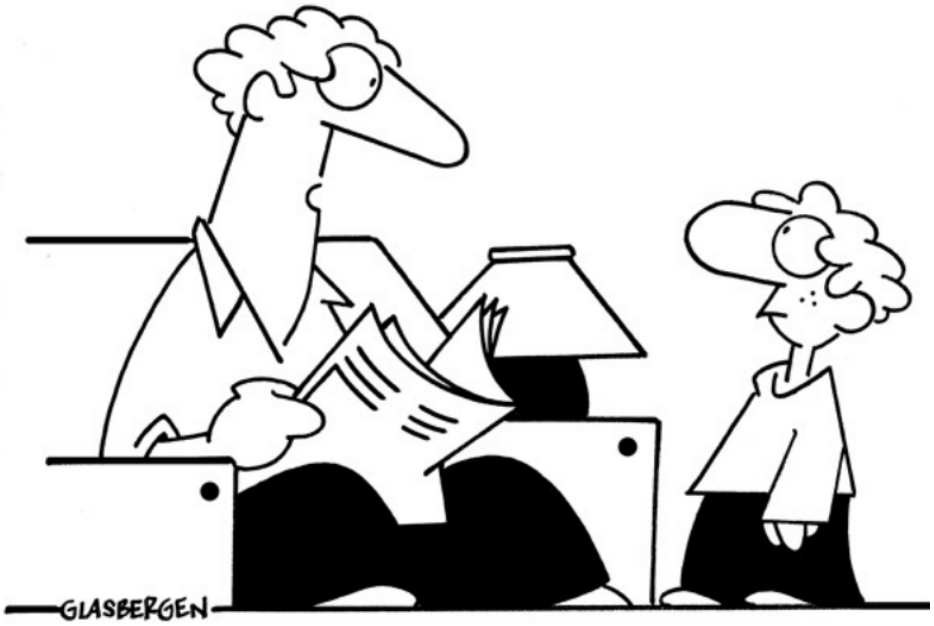
SOME PRELIMINARY NOTES

I AM NOT A MACHINE LEARNING RESEARCHER

ONLY AN ASTRONOMER WHO HAS BEEN USING MACHINE LEARNING FOR THE LAST ~14 YEARS FOR MY RESEARCH

THESE LECTURES ARE INTENDED TO PROVIDE A **GLOBAL** UNDERSTANDING OF HOW AI TECHNIQUES WORK AND ESPECIALLY **HOW TO USE THEM FOR YOUR RESEARCH**

WHAT ARE WE GOING TO LEARN?

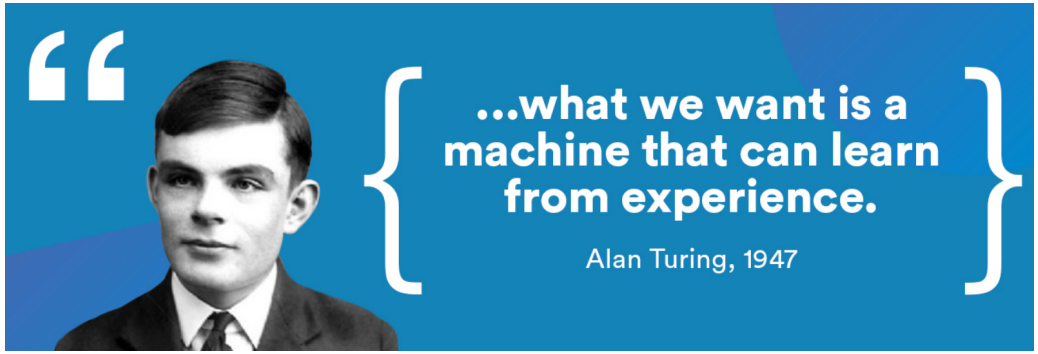


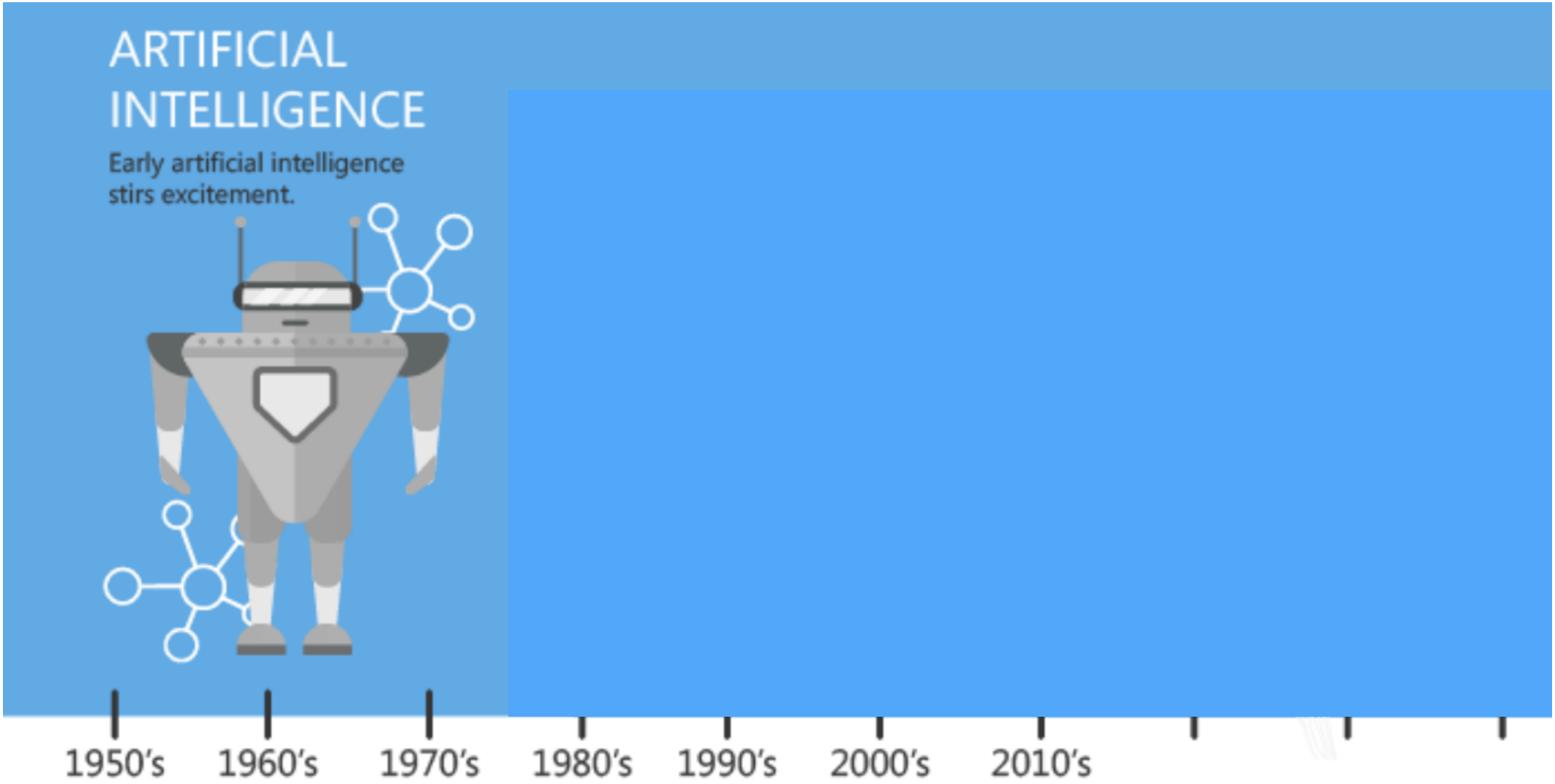
"*Artificial intelligence* is when you get a college degree, but you're still stupid when you graduate."

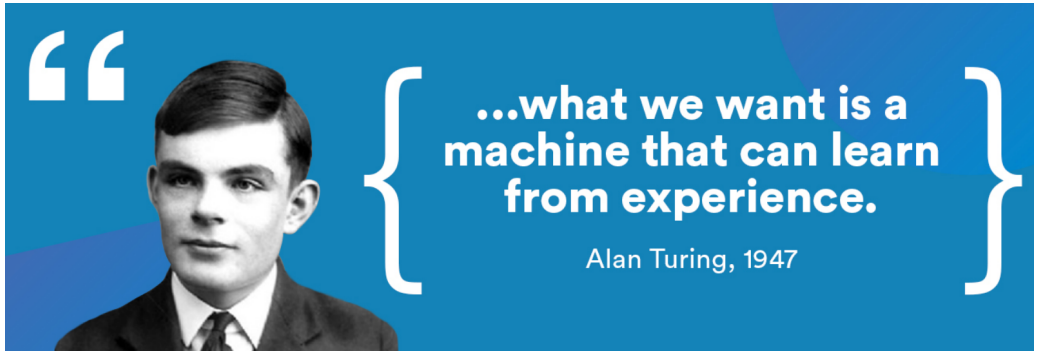
WHAT ARE WE GOING TO LEARN?

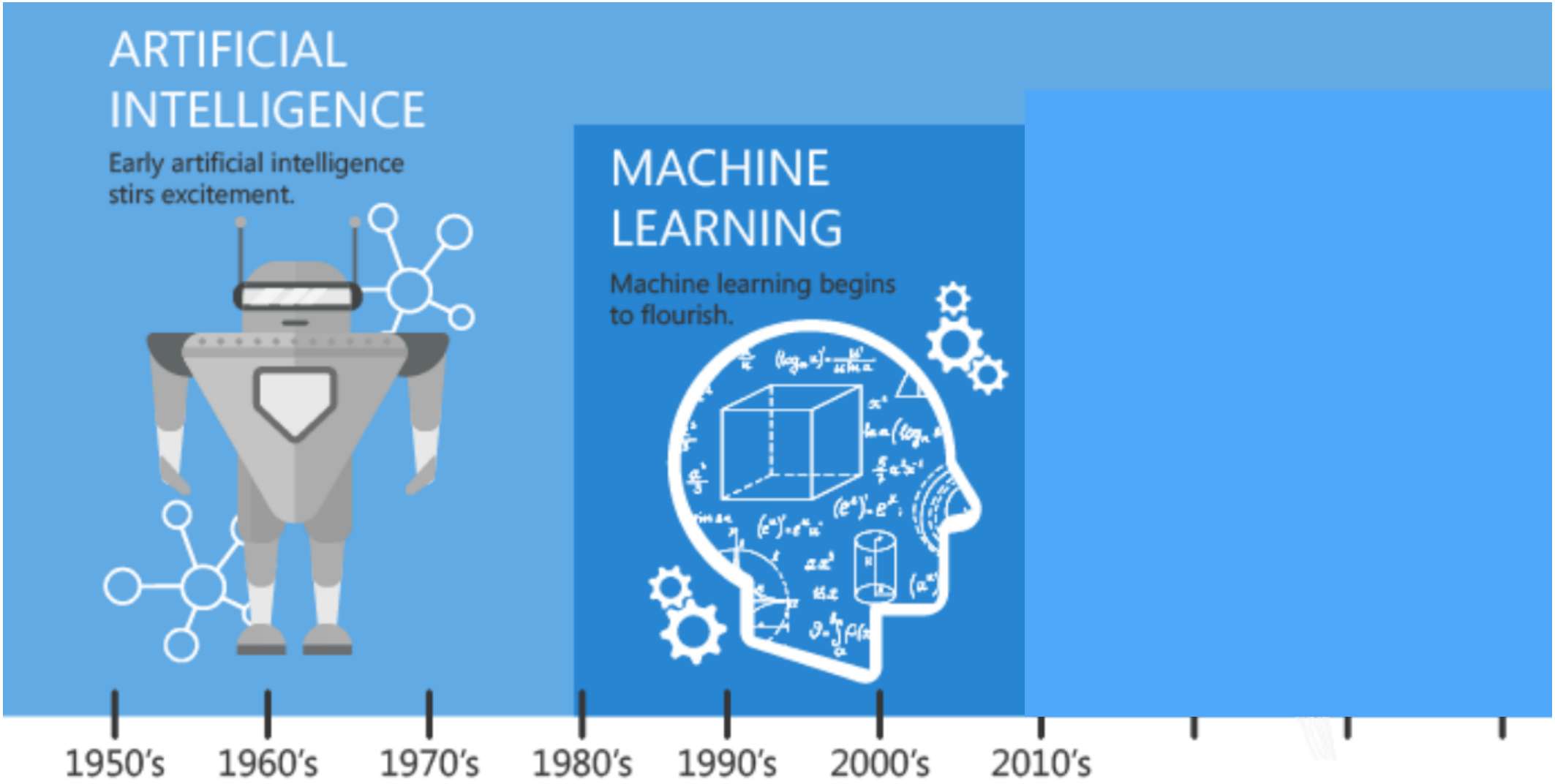
A BUNCH OF SOMETIMES CONFUSING TERMS…

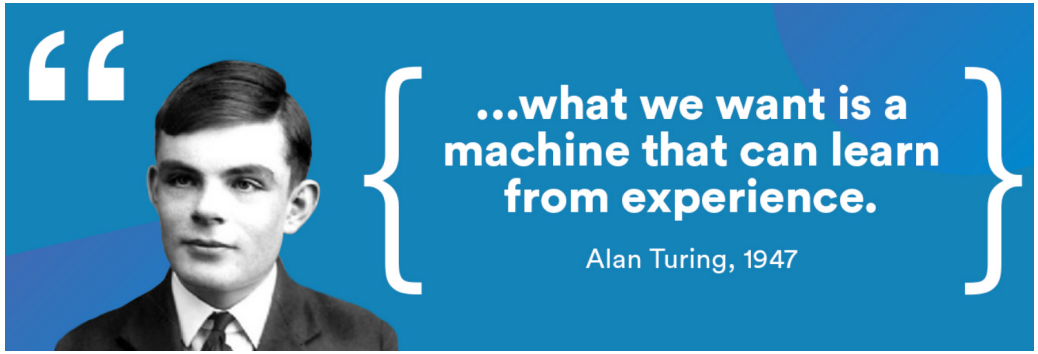
"Artificial intelligence is when you get a college degree, but you're still stupid when you graduate."

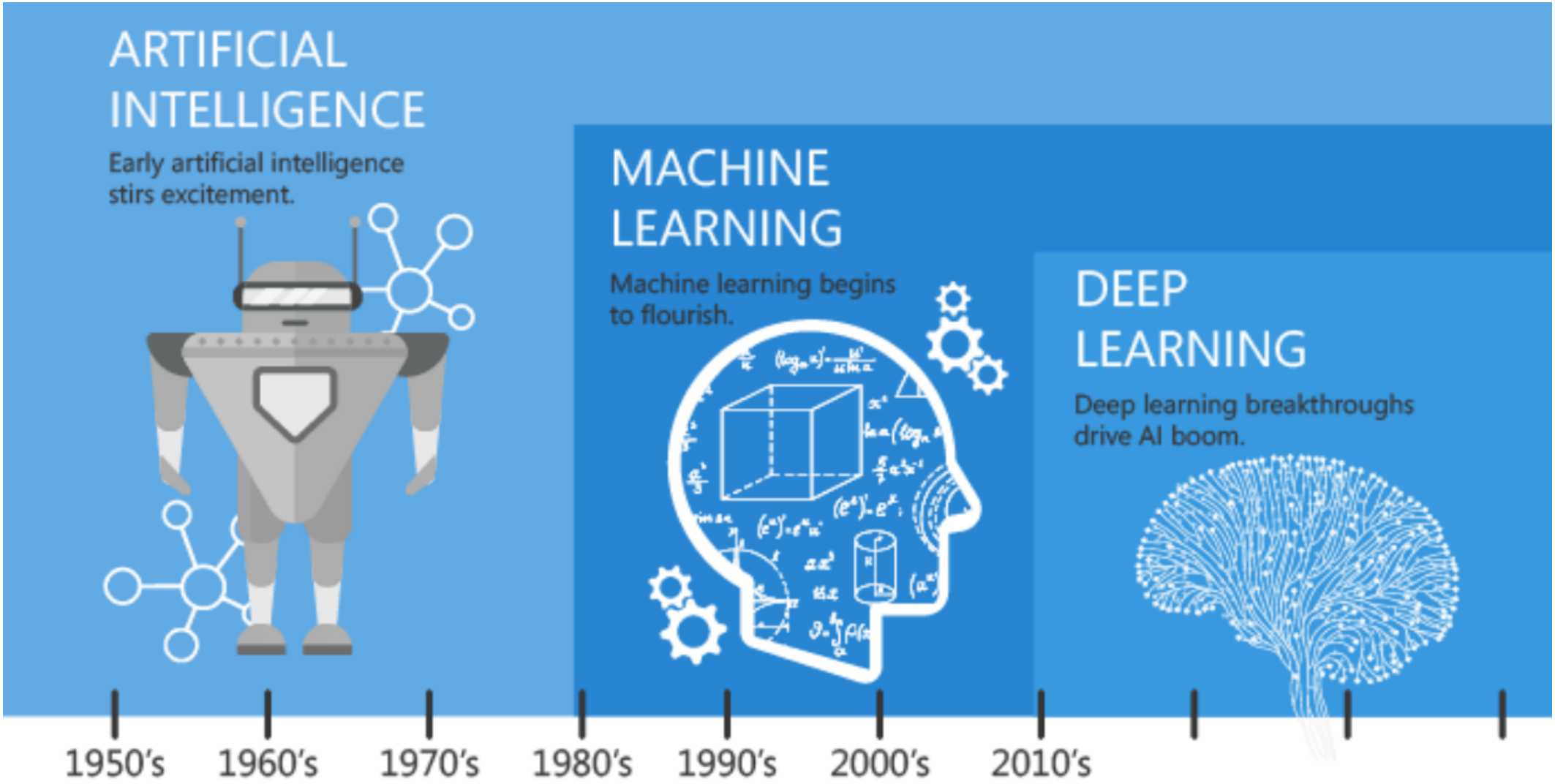

...what we want is a machine that can learn from experience.
Alan Turing, 1947


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
1950's
1960's
1970's
1980's
1990's
2000's
2010's


...what we want is a machine that can learn from experience.
Alan Turing, 1947


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
1950's
1960's
1970's
1980's
1990's
2000's
2010's


...what we want is a machine that can learn from experience.
Alan Turing, 1947


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

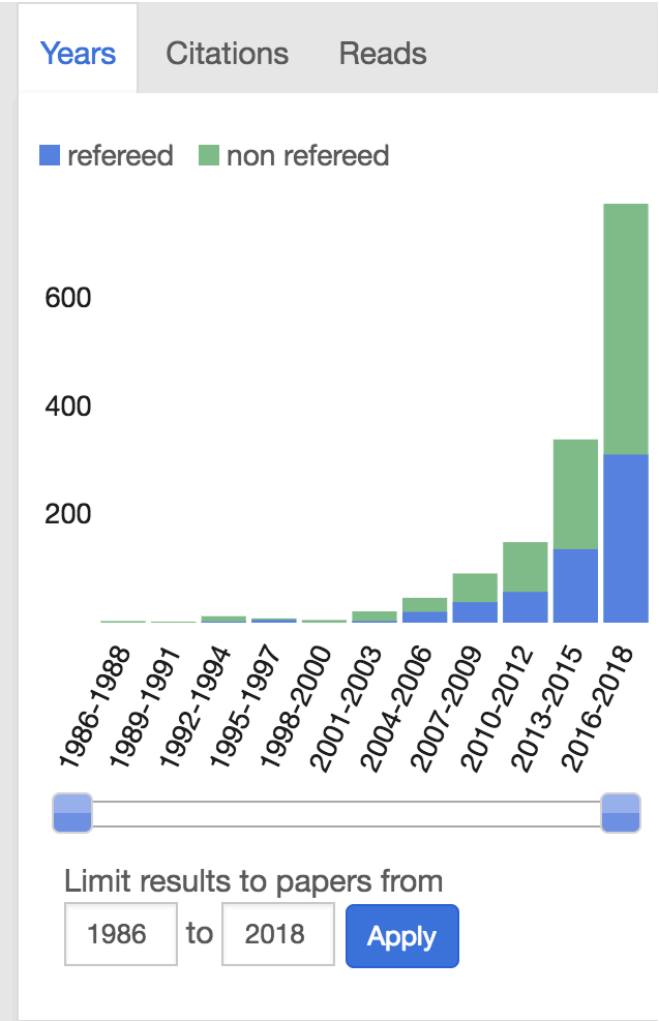
AI FEVER?

AN AMAZING MEDIA ATTENTION

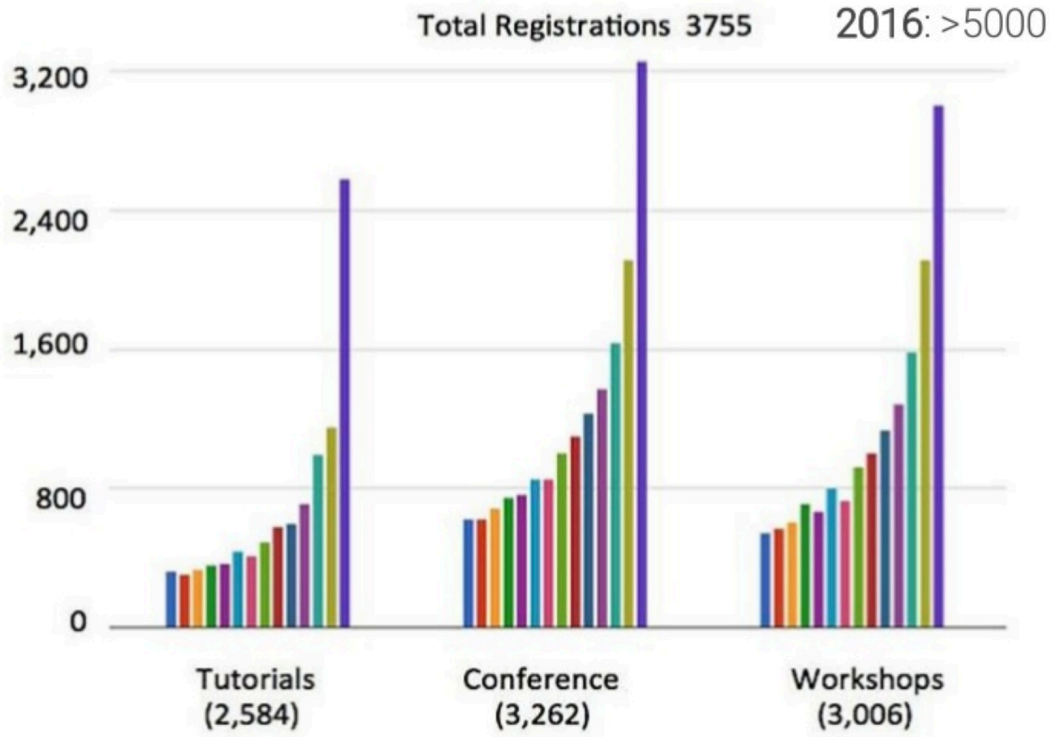
Le CNRS, Inria, l'université PSL et les entreprises Amazon, Criteo, Facebook, Faurecia, Google, Microsoft, NAVER LABS, Nokia Bell Labs, le Groupe PSA, SUEZ et Valeo font converger intérêts académiques et industriels et s'unissent pour créer, à Paris, l'Institut PRAIRIE dont l'objectif est de devenir une référence internationale de l'intelligence artificielle.

AI FEVER?

PUBLICATIONS (ADS)

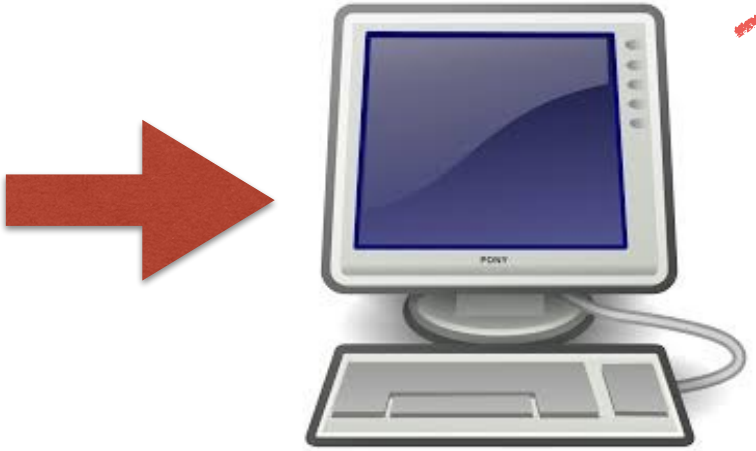


CONFERENCES



Source

BEFORE 2012….

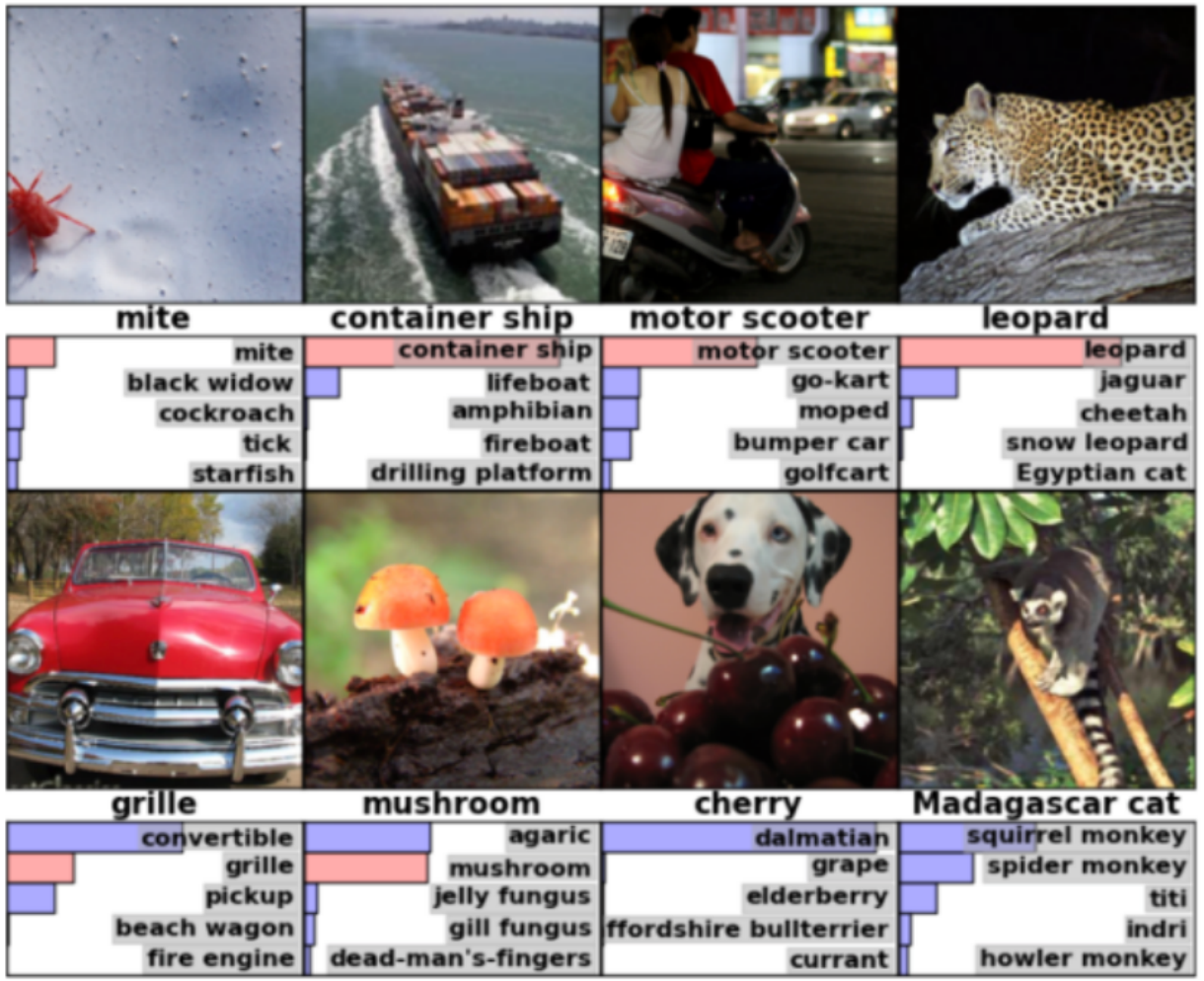


CAT?

DOG?

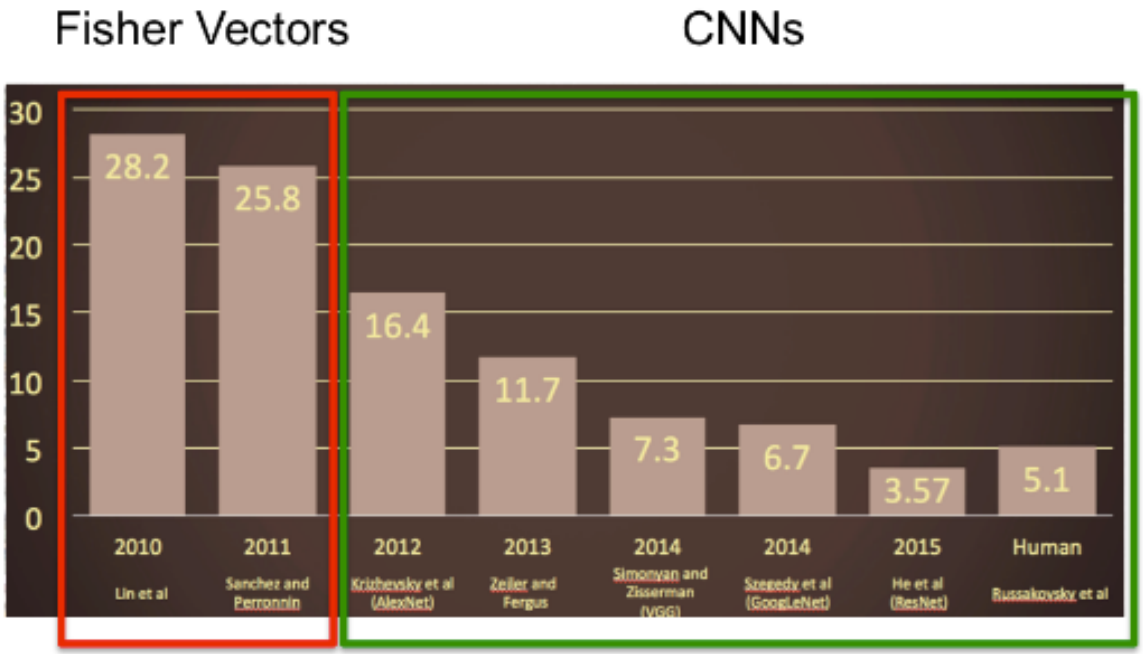
TRIVIAL HUMAN TASKS REMAINED CHALLENGING FOR COMPUTERS

AFTER 2012



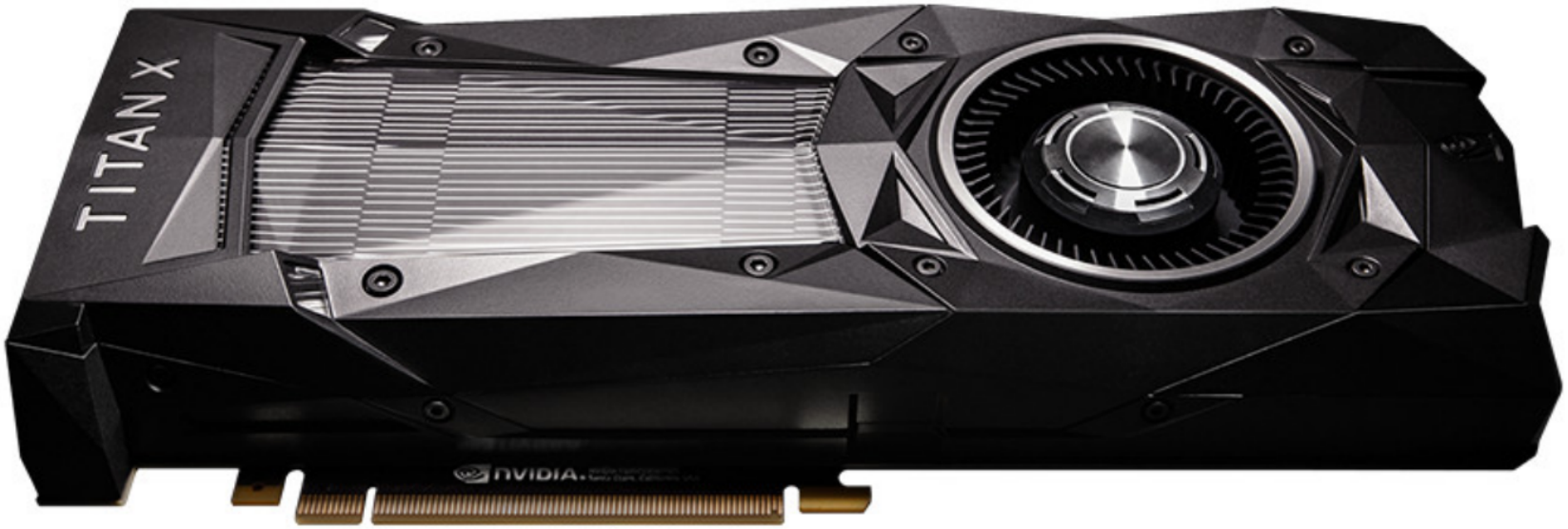
IT HAS BECOME TRIVIAL….

THIS IS A CHANGE OF PARADIGM!



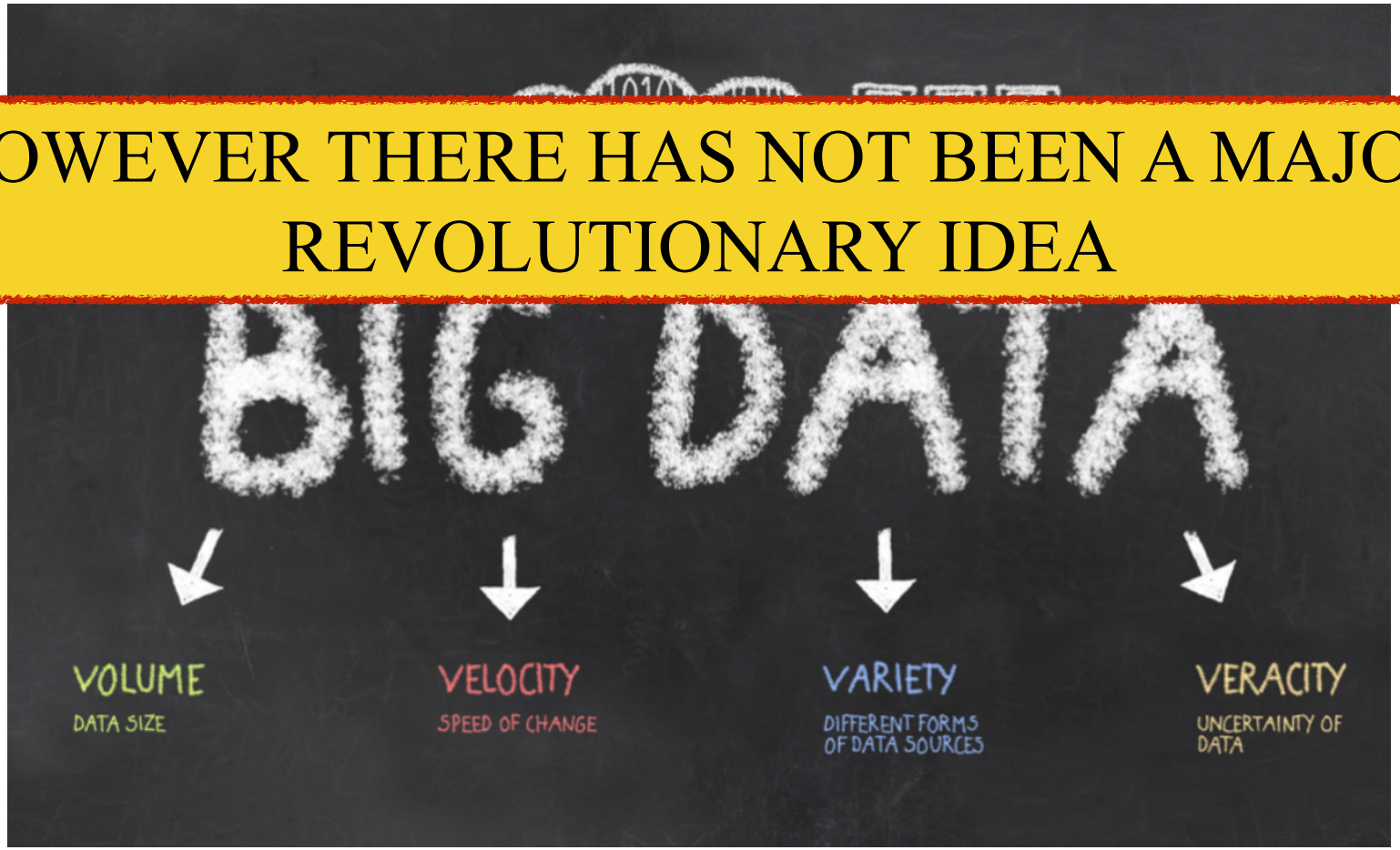
ONE OF THE MAIN REASONS OF THIS BREAKTHROUGH IS THE AVAILABILITY OF VERY LARGE DATASETS TO LEARN

COMBINED WITH THE TECHNOLOGY TO PROCESS ALL THIS DATA



ONE OF THE MAIN REASONS OF THIS BREAKTHROUGH IS THE AVAILABILITY OF VERY LARGE DATASETS TO LEARN

HOWEVER THERE HAS NOT BEEN A MAJOR REVOLUTIONARY IDEA



WHAT ARE WE GOING TO LEARN?

BASICS OF CLASSICAL MACHINE LEARNING

BASICS OF DEEP LEARNING
(BOTH SUPERVISED AND UNSUPERVISED)

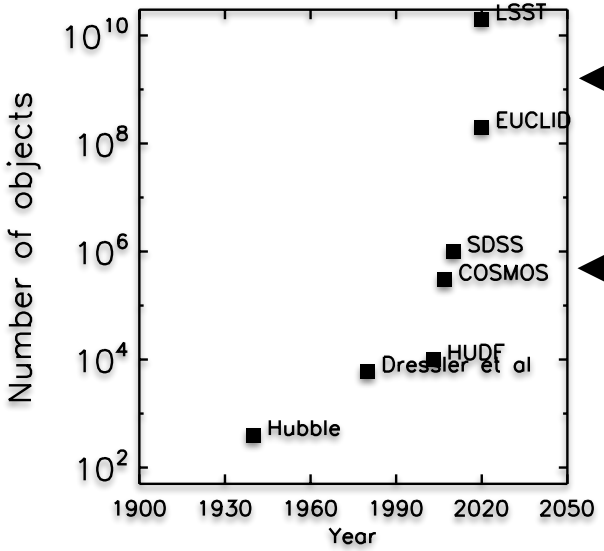
HOPING THAT THIS WOULD BE USEFUL FOR YOUR RESEARCH!

(Apologies in advance for biases on Extra-Galactic Science + imaging)

WHY DO WE NEED THESE TOOLS IN ASTRONOMY?

WHY DO WE NEED THESE TOOLS IN ASTRONOMY?

AS IN MANY OTHER DISCIPLINES THE BIG-DATA REVOLUTION HAS ARRIVED TO ASTRONOMY TOO



BIG-DATA REVOLUTION

we are here

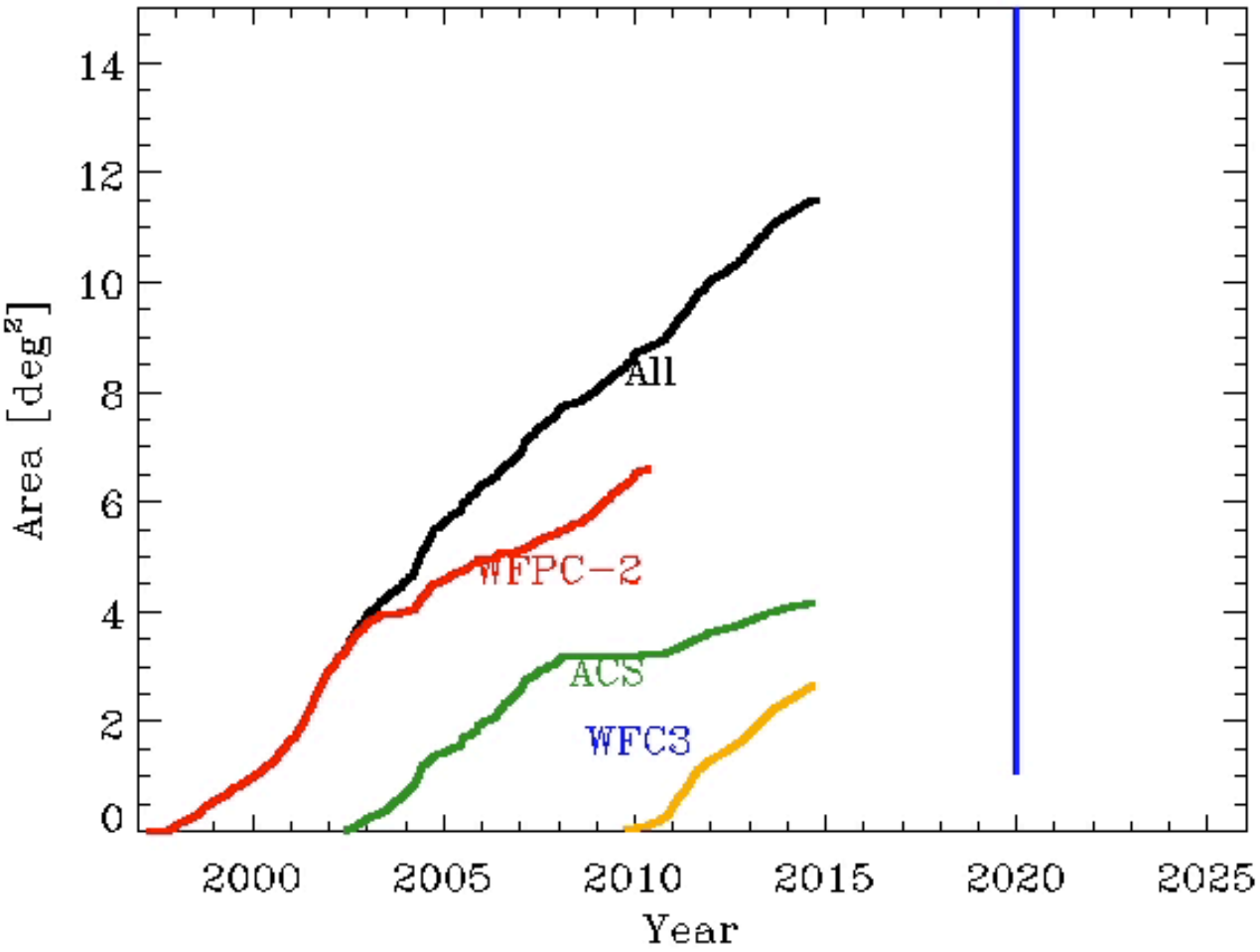
EXTREMELY LARGE IMAGING SURVEYS DELIVERING BILLIONS OF OBJECTS IN 2-5 YEARS

Large Synoptic Survey Telescope

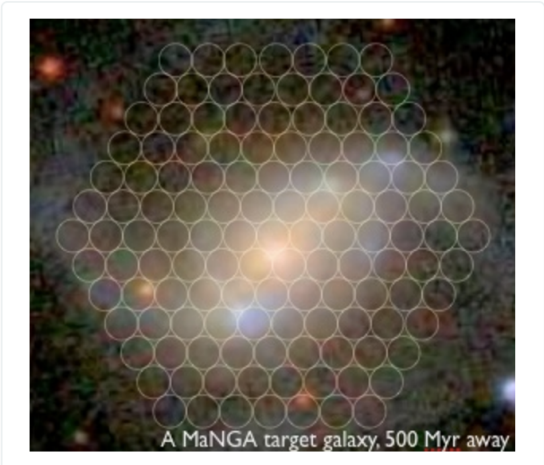
euclid consortium

WIDE-FIELD INFRARED SURVEY TELESCOPE
WFIRST
ASTROPHYSICS • DARK ENERGY • EXOPLANETS

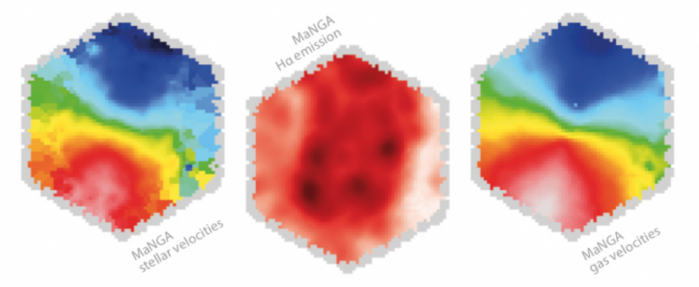
LSST simulation



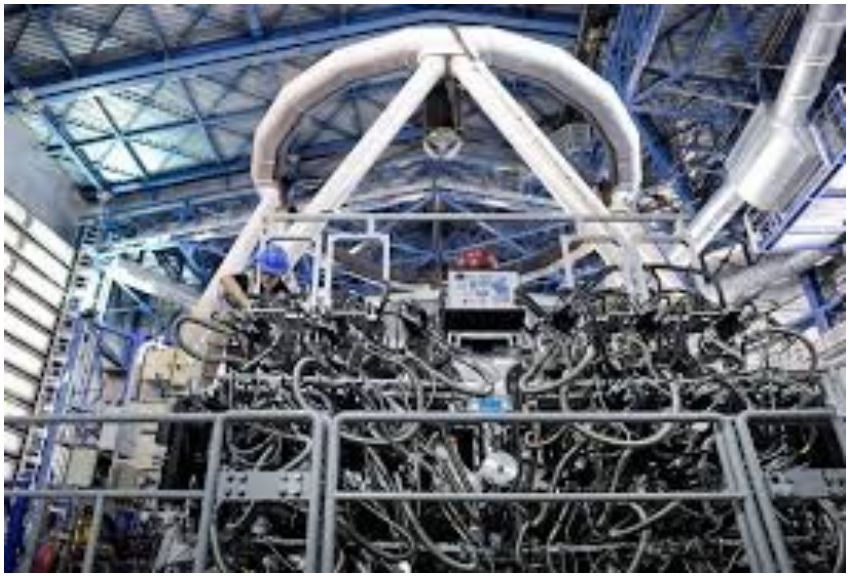
(Thanks to J. Brinchmann)

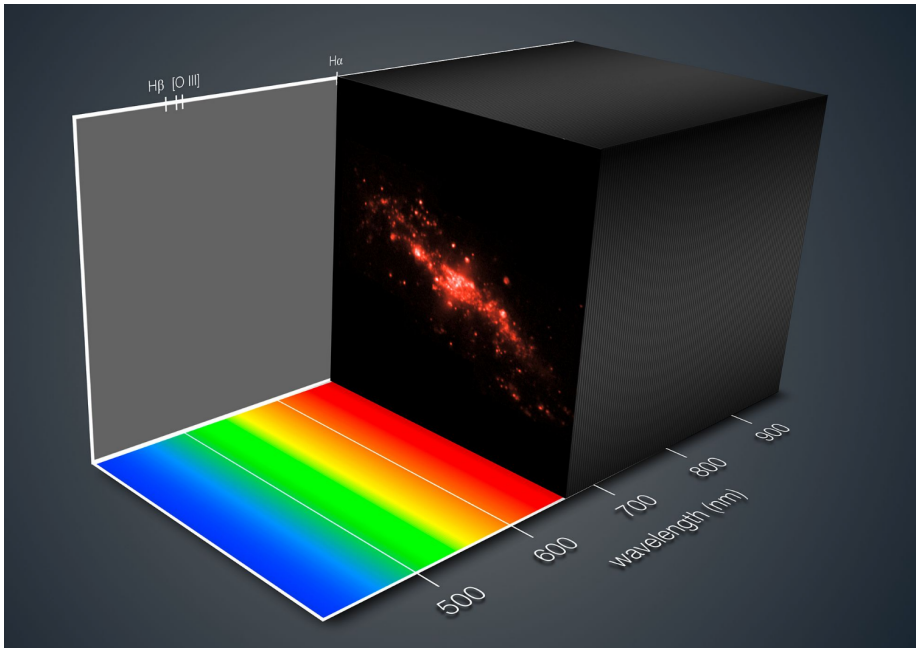


NOT ONLY VOLUME: AN INCREASING COMPLEXITY OF DATA

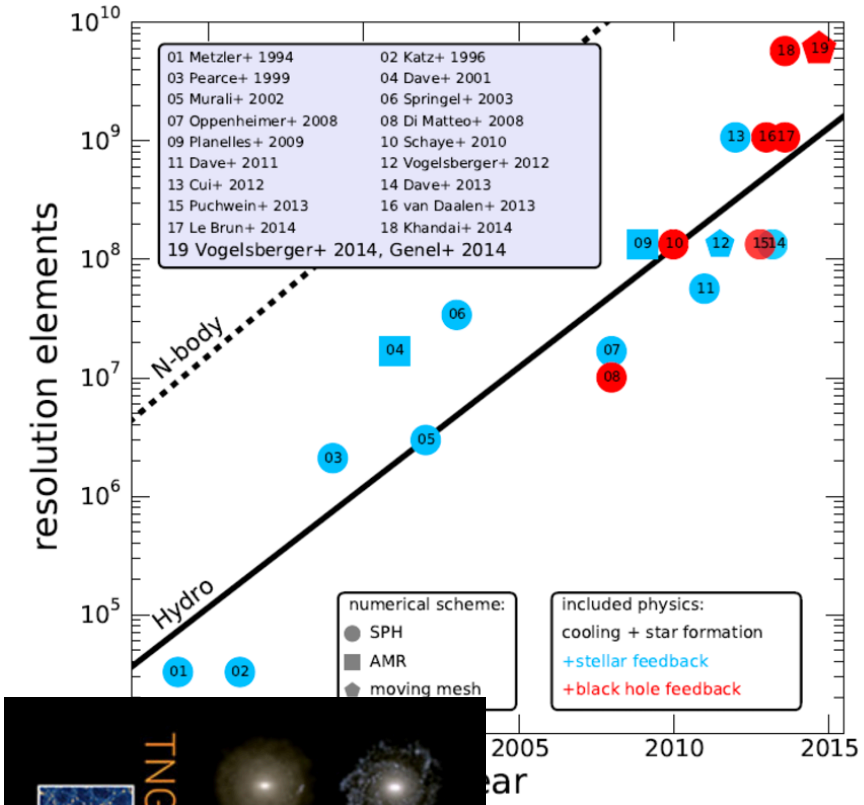


MANGA Survey



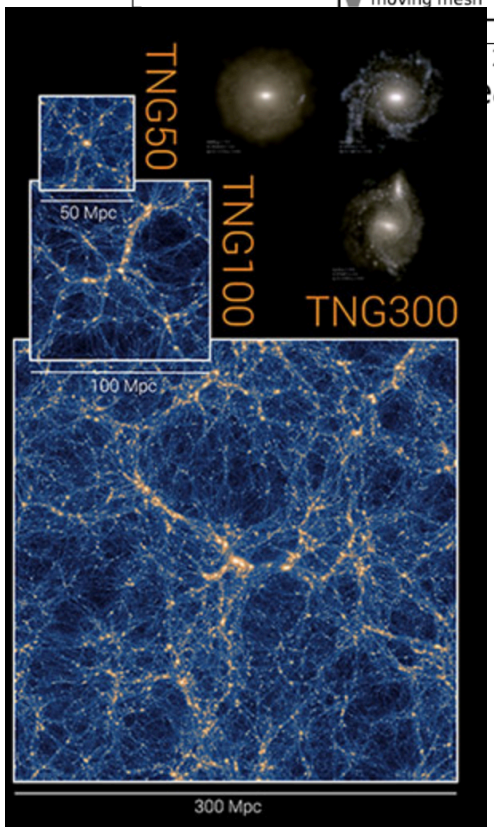


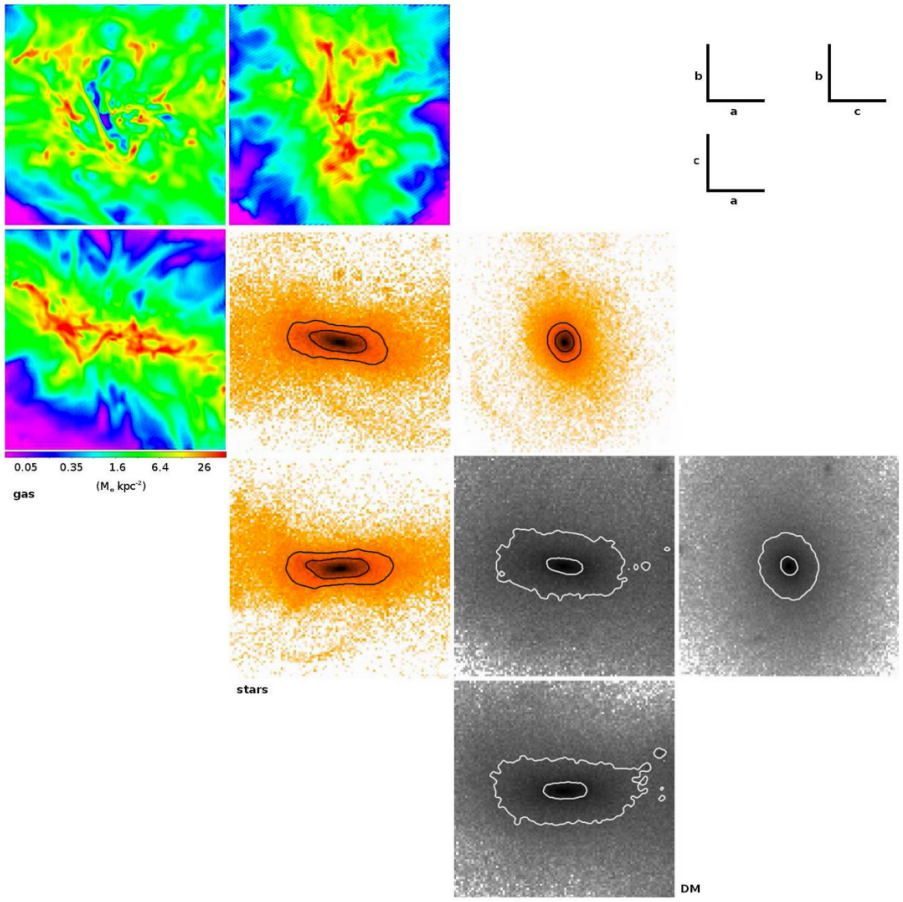
MUSE@VLT



Genel+14

AND ALSO SIMULATIONS!





Ceverino+15

PROGRAM FOR THE TWO DAYS

- **PART I: A VERY QUICK INTRODUCTION TO 'CLASSICAL' MACHINE LEARNING**
  - UNSUPERVISED / SUPERVISED
  - GENERAL STEPS TO "TEACH A MACHINE"
  - "CLASSICAL" CLASSIFIERS

PROGRAM FOR THE TWO DAYS

- **PART II: FOCUS ON 'SHALLOW' NEURAL NETWORKS**
  - PERECPTRON, NEURON DEFINITION
  - LAYER OF NEURONS, HIDDEN LAYERS
  - ACTIVATION FUNCTIONS
  - OPTIMIZATION [GRADIENT DESCENT, LEARNING RATES]
  - BACKPROPAGATION

PROGRAM FOR THE TWO DAYS

- **PART III: CONVOLUTIONAL NEURAL NETWORKS**
  - CONVOLUTIONS AS NEURONS
  - CNNs [POOLING, DROPOUT]
  - VANISHING GRADIENT / BATCH NORMALIZATION

PROGRAM FOR THE TWO DAYS

- **PART IV: IMAGE TO IMAGE NETOWRKS + INTRODUCTION TO UNSUPERVISED DEEP LEARNING**
  - NETWORKS FOR IMAGE SEGMENTATION
  - AUTO-ENCODERS
  - GENERATIVE ADVERSARIAL NETOWRKS
  - ANOMALY DETECTION

PROGRAM FOR THE TWO DAYS

- **PART V: SOME PRACTICAL CONSIDERATIONS**
  - HOW DO I SETUP MY CNN?
  - HOW LARGE DO TRAINING SETS NEED TO BE?
  - OPTIMIZING YOUR NET: HYPER PARAMETER SEARCH
  - VISUALIZING CNNs [DECONVNETS, INCEPTIONISM, INTEGRATED GRADIENTS]

HANDS-ON SESSION

WE WILL TRY TO IMPLEMENT SOME OF THE THINGS LEARNED

WE WILL TEST SEVERAL APPROACHES TO CLASSIFY GALAXIES

LET'S TRY TO DISCUSS AS MUCH AS POSSIBLE!

SOFTWARE REQUIREMENTS

- PYTHON 3 OR GREATER
- TENSORFLOW FOR DEEP LEARNING
- KERAS - HIGH LEVEL LIBRARY WHICH MAKES GPU CODING TRANSPARENT - SIMPLIFIES THINGS A LOT AND MOST OF THE TIME ENOUGH FOR OUR APPLICATIONS

PART I: AN INTRODUCTION TO "CLASSICAL" MACHINE LEARNING

WHAT DOES MACHINE LEARNING DO?

the machine is told what to look for

SUPERVISED

the machine is NOT told what to look for

UN-SUPERVISED

TWO BIG TYPES OF MACHINE LEARNING ALGORITHMS

SUPERVISED LEARNING

CAT

CAT

the machine is told what to look for

DOG

HUMAN LABELLING

DOG

SUPERVISED LEARNING

CAT

CAT

the machine is told what to look for

DOG

HUMAN LABELLING

TRAINING SET OF LABELED EXAMPLES

DOG

CAT

CAT

DOG

DOG

SUPERVISED LEARNING

ML

CAT

SUPERVISED LEARNING

CAT

CAT

DOG

DOG

ML

DOG

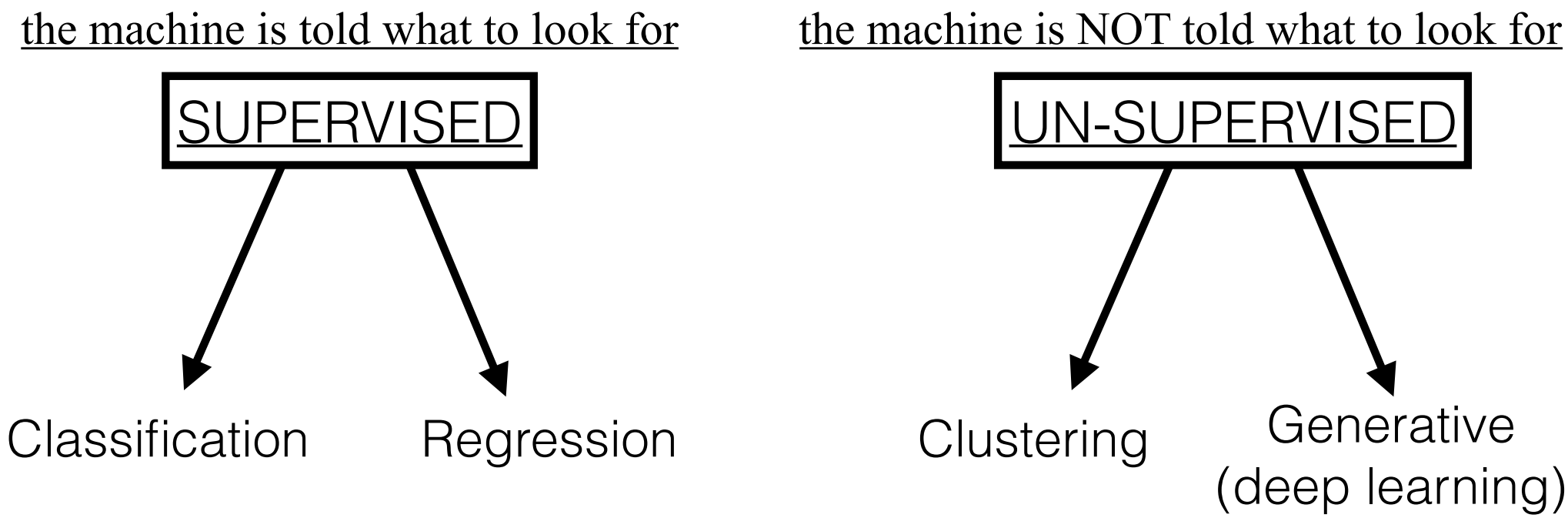
UNSUPERVISED LEARNING

UNSUPERVISED LEARNING

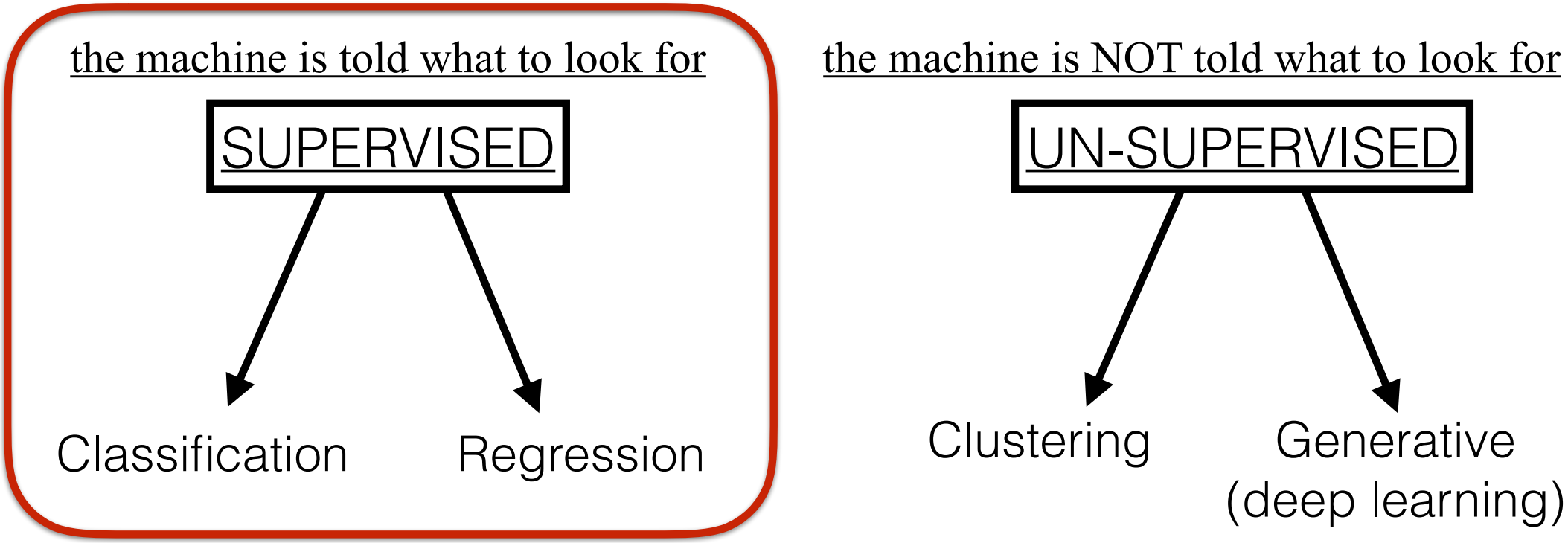
UNSUPERVISED LEARNING

THE DEFINITION OF CLASSES IS SOMETIMES NOT OBVIOUS

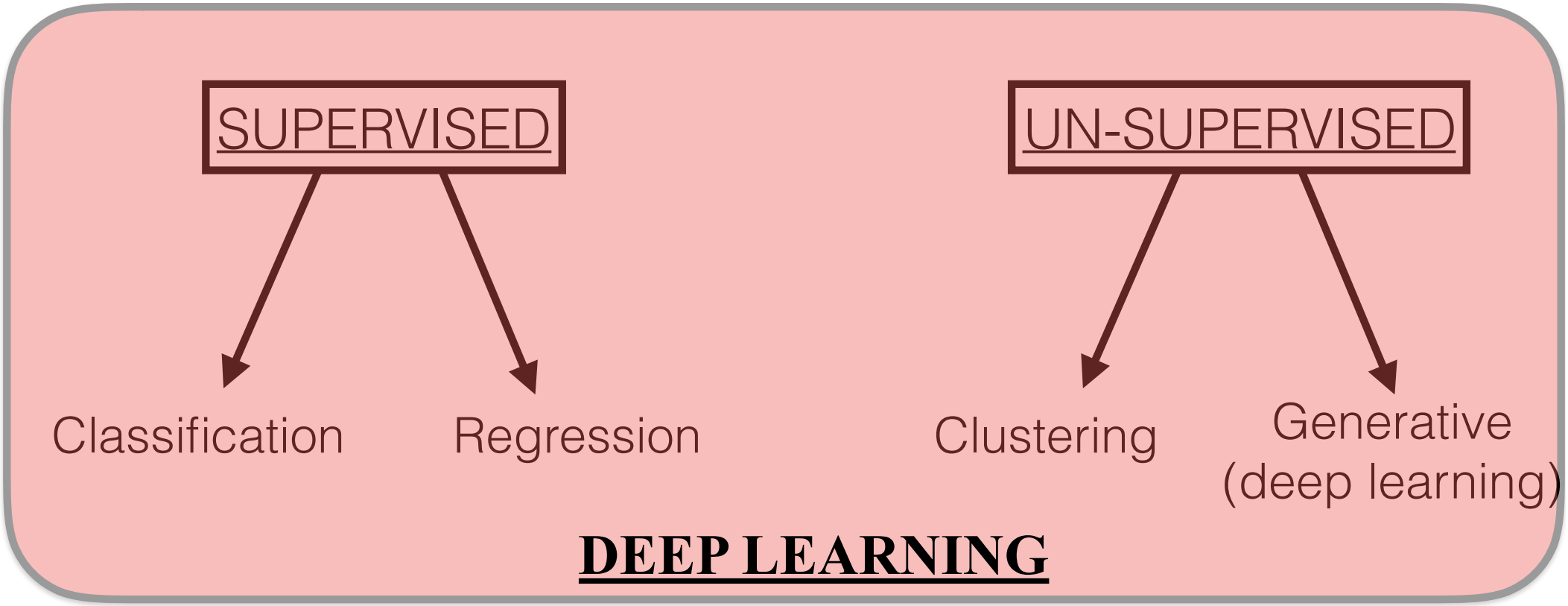
WHAT DOES MACHINE LEARNING DO?



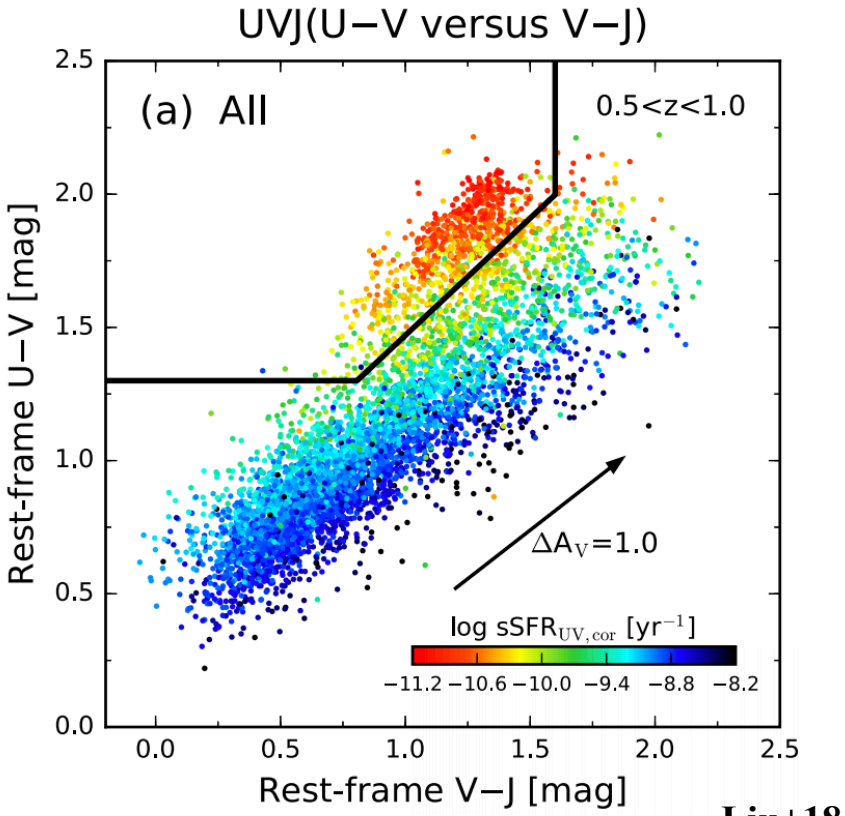
WHAT DOES MACHINE LEARNING DO?



WHAT DOES MACHINE LEARNING DO?

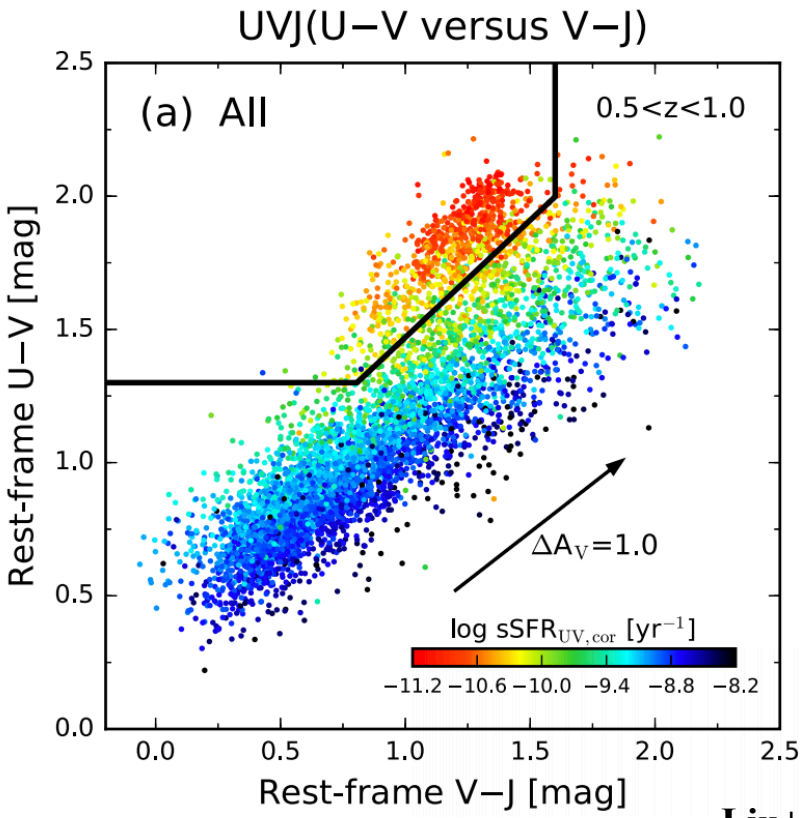


THRE IS NO MAGIC IN MACHINE LEARNING, AND IT IS ACTUALLY PRETTY SIMPLE



Liu+18

$$f_W(\vec{x}) = \vec{y}$$

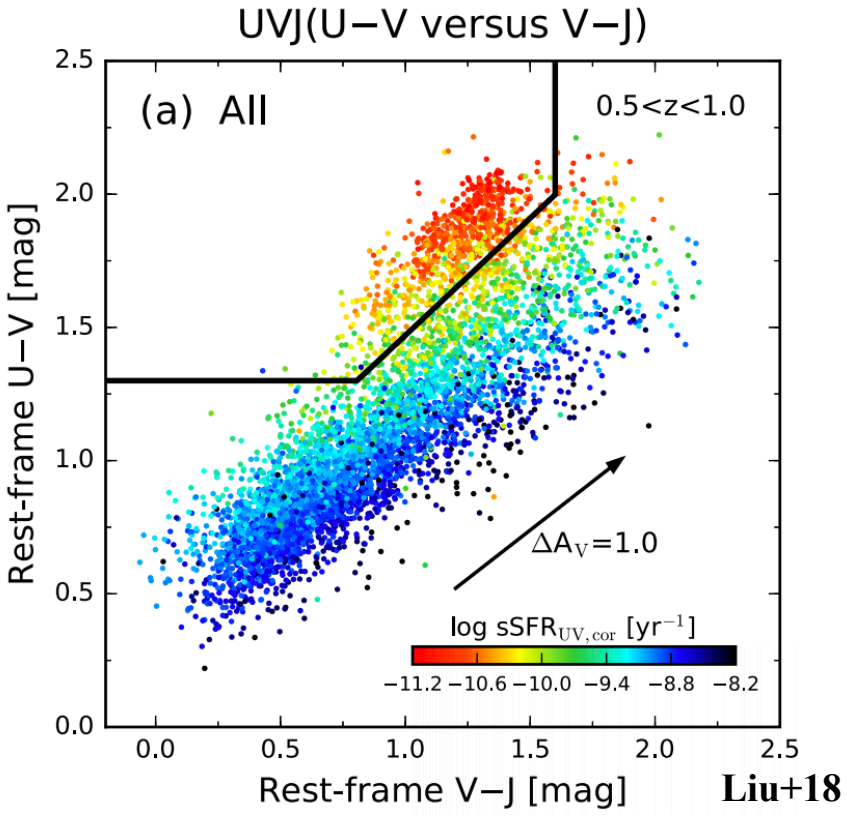


Liu+18

$$f_W(\vec{x}) = \vec{y}$$

LABEL

Q , SF

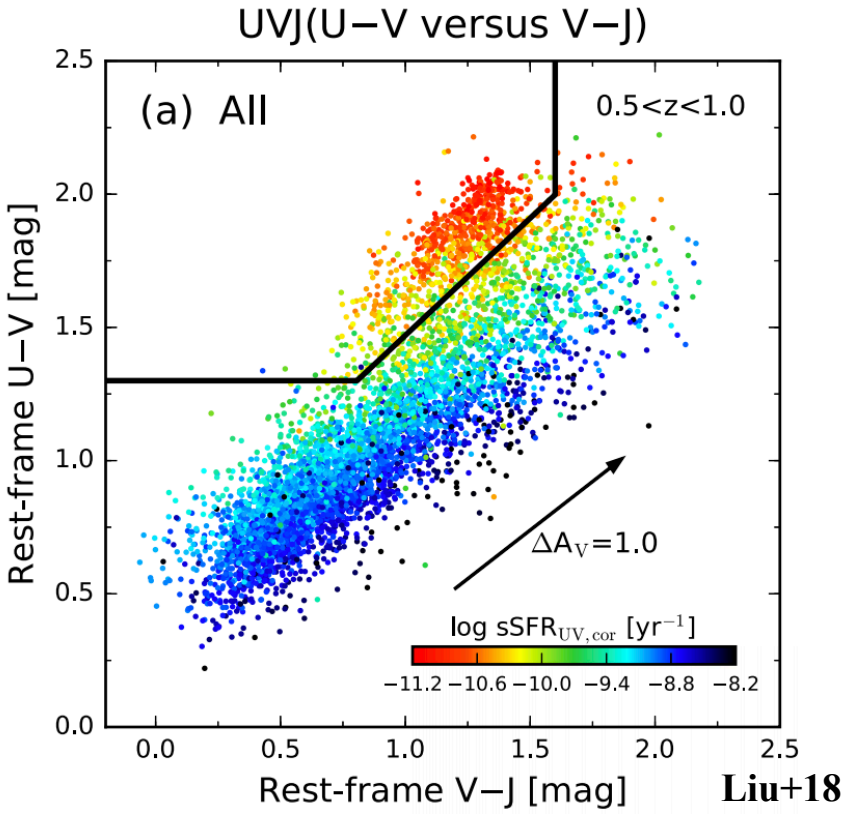


$$f_{\vec{W}}(\vec{x}) = \vec{y}$$

LABEL

Q(0) , SF(1)

(U-V, V-J) FEATURES



Liu+18

$$f_{\vec{W}}(\vec{x}) = \vec{y}$$

LABEL

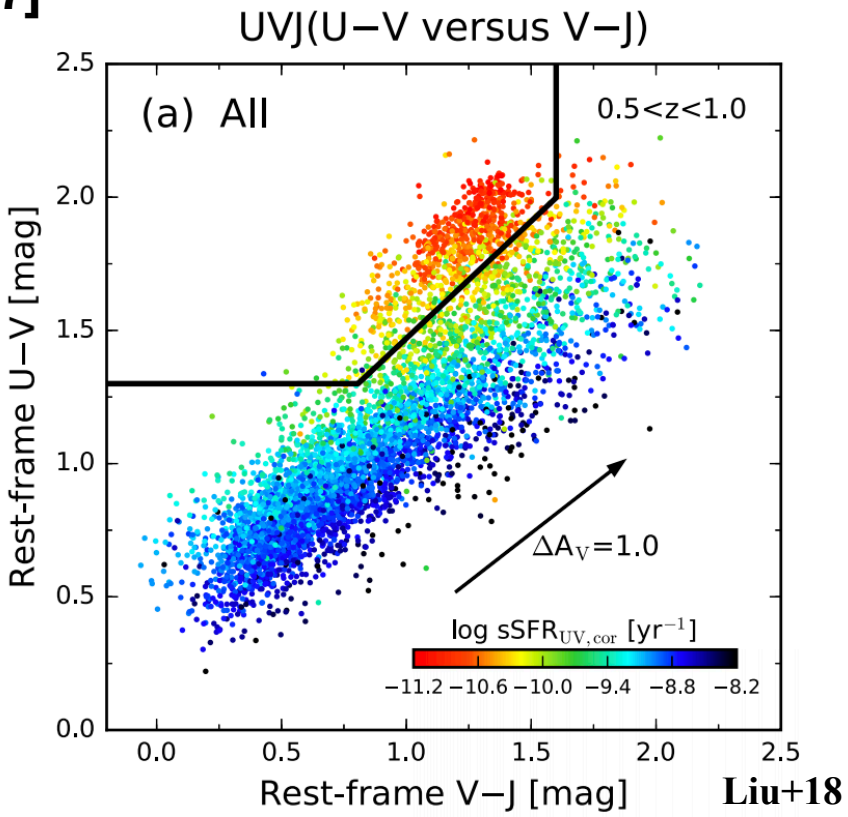
Q(0) , SF(1)

NETWORK FUNCTION

(U-V, V-J) FEATURES

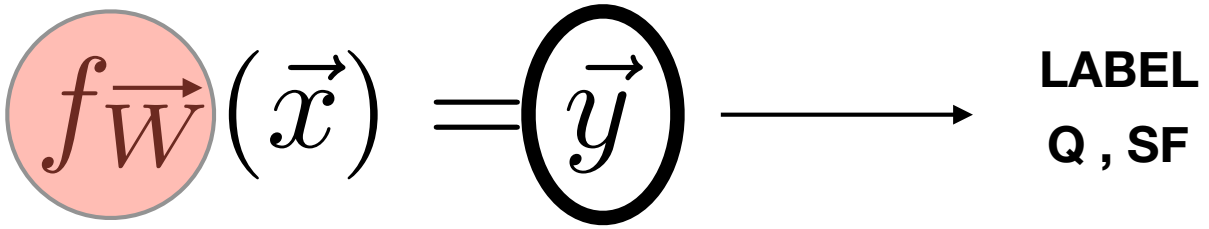
sgn[(u-v)-0.8*(v-j)-0.7]

WEIGHTS



Liu+18

“CLASSICAL” MACHINE LEARNING



sgn[(u-v)-W1*(v-j)-W2]

REPLACE THIS BY A GENERAL NON LINEAR FUNCTION WITH SOME PARAMETERS W

SUPERVISED LEARNING

Given a dataset with known labels (measurements) - find a function that can assign (predict) measurements for an unlabeled dataset

Training set

$(\vec{x_1}, \vec{x_2}, \vec{x_3}, ..., \vec{x_n})$ → Measurements (colors, fluxes, spectra indices…)

$(\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n})$ → Label (morphology, object type, redshift …)

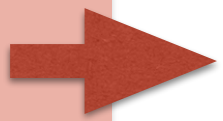
SUPERVISED LEARNING

Given a dataset with known labels (measurements) - find a function that can assign (predict) measurements for an unlabeled dataset

Training set

$(\vec{x_1}, \vec{x_2}, \vec{x_3}, ..., \vec{x_n})$

$(\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n})$



$f_W(\vec{x}) = \vec{y}$

?

SUPERVISED LEARNING

Unlabeled set

$(\vec{x_1}', \vec{x_2}', \vec{x_3}', ..., \vec{x_n}')$

$(\vec{y_1}', \vec{y_2}', \vec{y_3}', ..., \vec{y_n}')$

Training set

$(\vec{x_1}, \vec{x_2}, \vec{x_3}, ..., \vec{x_n})$

$(\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n})$

$f_W(\vec{x}) = \vec{y}$

?

$$(\vec{x_1}, \vec{x_2}, \vec{x_3}, ..., \vec{x_n}) \qquad \vec{x} \in \mathbb{R}^d$$

$$(\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n}) \qquad \vec{y} \in \mathbb{R} \qquad \vec{y} \in \mathbb{N}$$

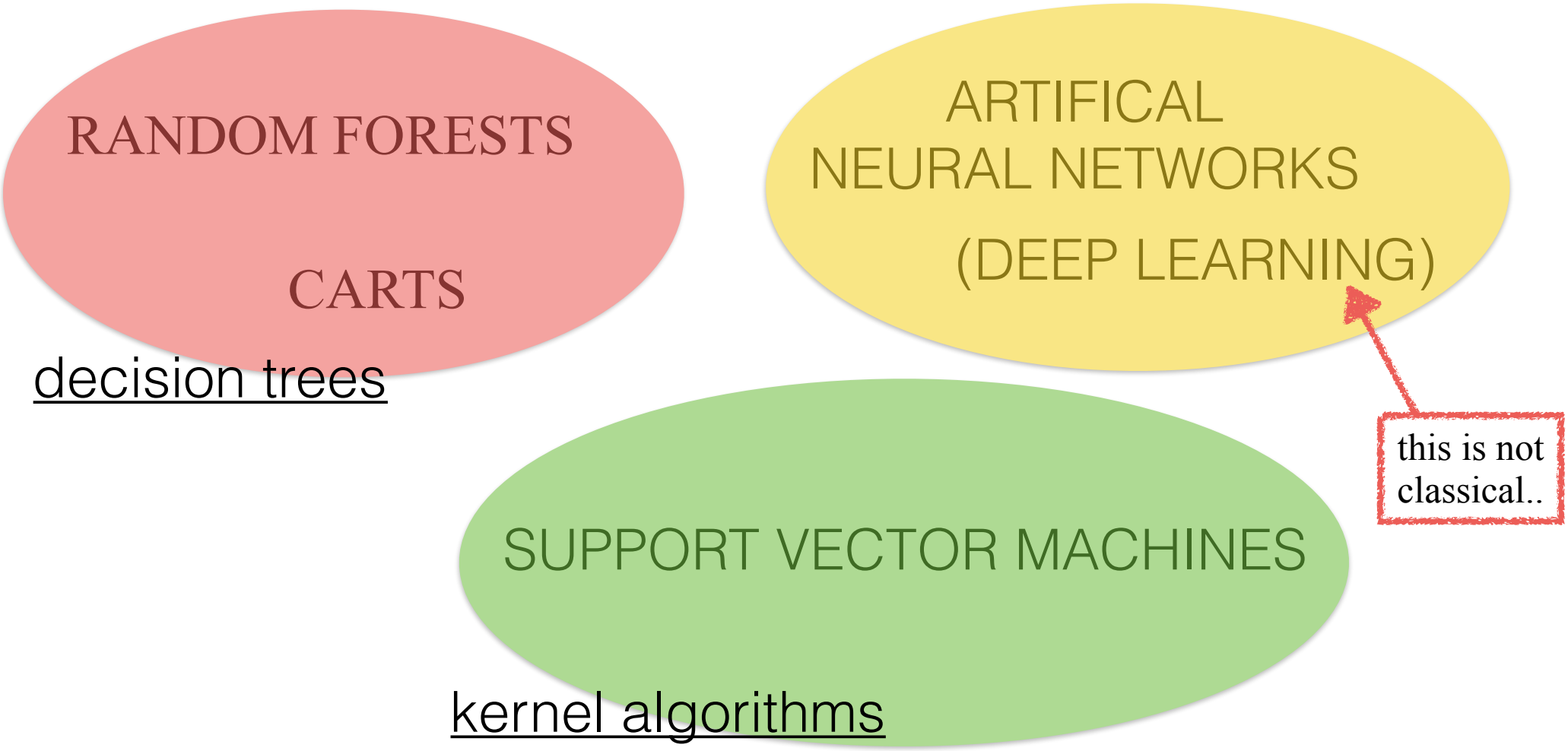
GENERAL GOAL: Find a (non-linear) function that outputs the correct class / measurement for a given input object:

$$f_W(\vec{x})$$

Number of parameters - can be large

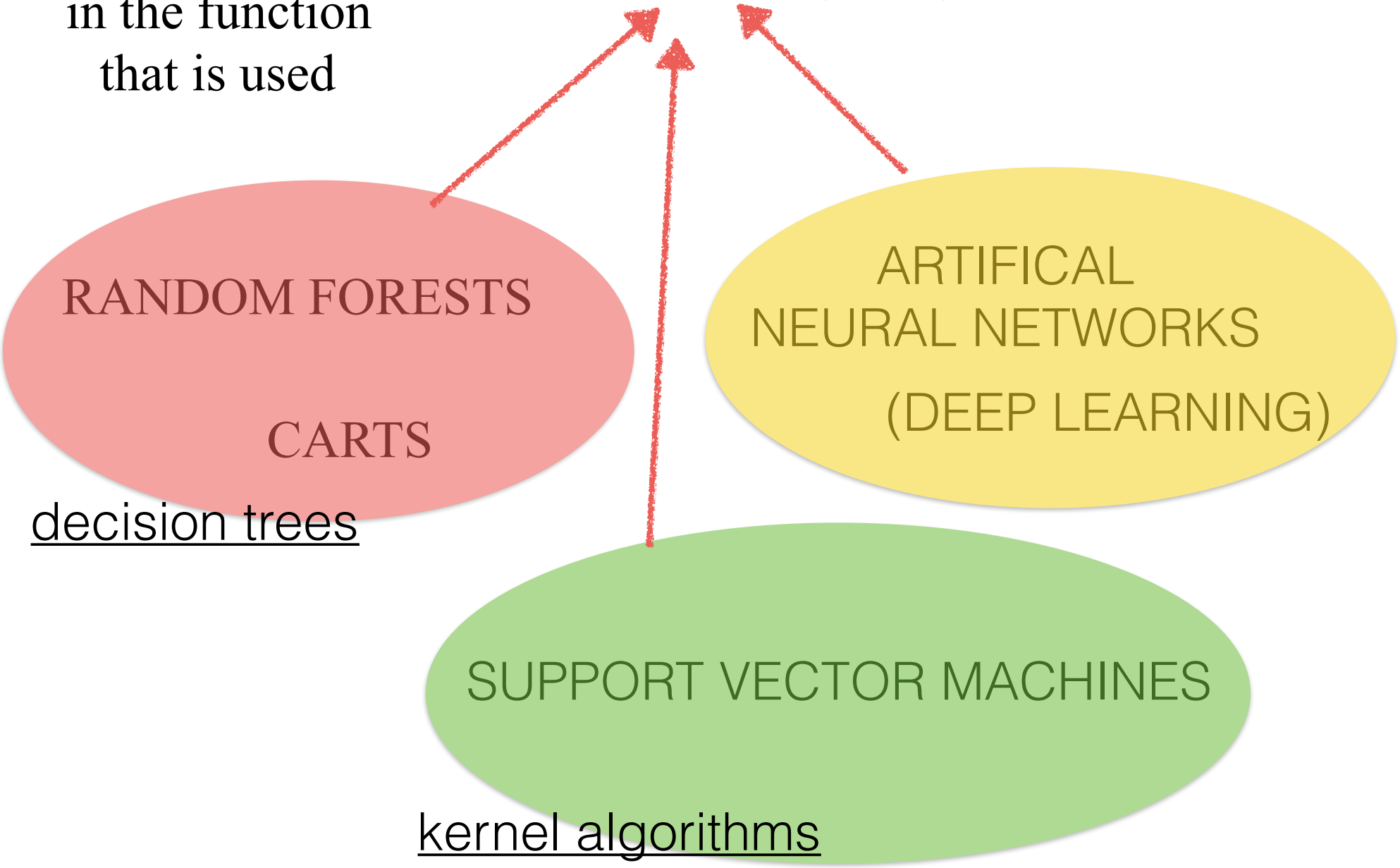
It is translated into a minimization problem : find W such as the prediction error is minimal <u>over all unseen vectors</u>

Different "classical" supervised machine learning methods



The differences are in the function that is used

$$f_W(\vec{x})$$



We need two key elements

1. A LOSS FUNCTION

2. A MINIMIZATION OR OPTIMIZATION ALGORITHM

We need two key elements

1. A LOSS FUNCTION

2. A MINIMIZATION OR OPTIMIZATION ALGORITHM

THIS IS COMMON TO ALL MACHINE LEARNING ALGORITHMS

1. DEFINE A LOSS FUNCTION

$$loss(F_W(.), \vec{x_i}, \vec{y_i})$$

For example: $(F_W(\vec{x_i}) - \vec{y_i})^2$ Quadratic loss function

2. MINIMIZE THE EMPIRICAL RISK

$$\Re_{empirical}(W) = \frac{1}{N} \sum_{i}^{N} [loss(W, \vec{x}, \vec{y})]$$

MINIMIZE THE RISK

EMPIRICAL RISK?

$$\Re_{empirical}(W) = \frac{1}{N}\sum_{i}^{N}[loss(W, \vec{x}, \vec{y})]$$

WE ARE MINIMIZING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

EMPIRICAL RISK?

$$\Re_{empirical}(W) = \frac{1}{N}\sum_{i}^{N}[loss(W, \vec{x}, \vec{y})]$$

WE ARE MINIMIZING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

OBSERVED DATASET

EMPIRICAL RISK?

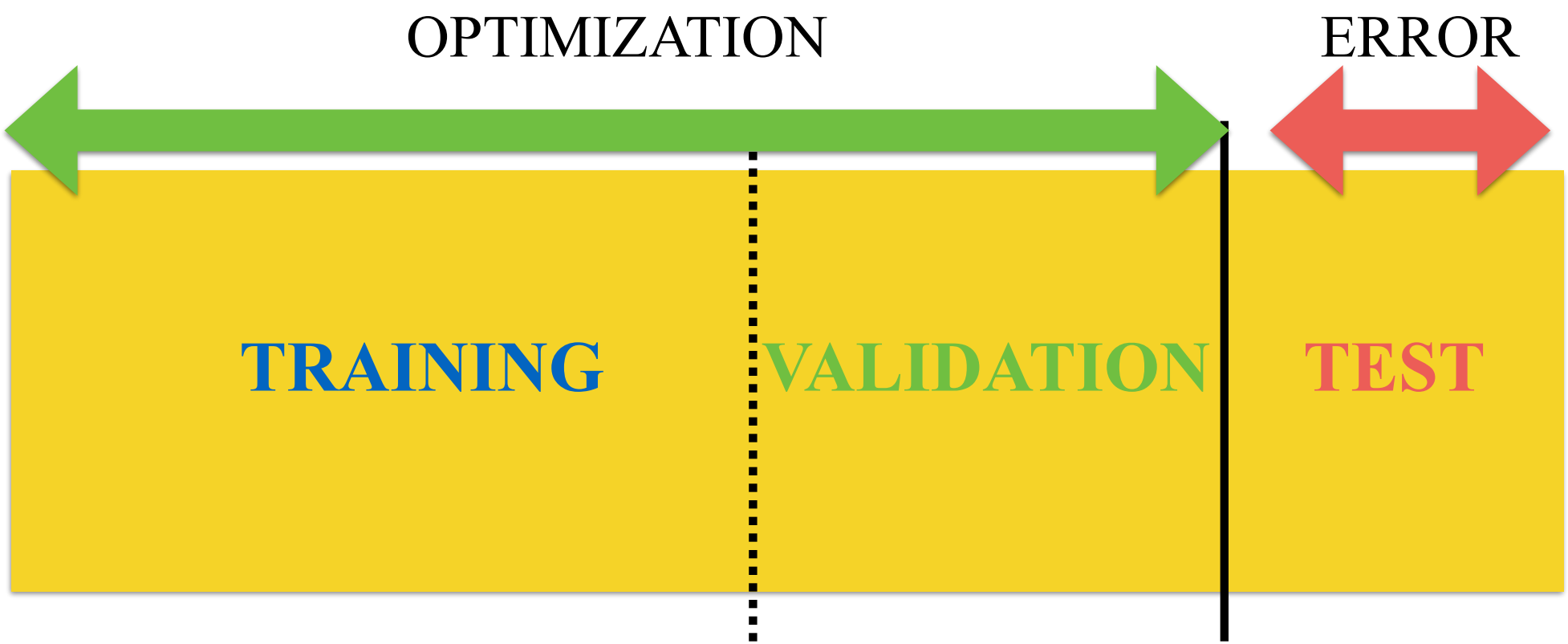
$$\Re_{empirical}(W) = \frac{1}{N}\sum_{i}^{N}[loss(W, \vec{x}, \vec{y})]$$

WE ARE MINIMIZING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

ALL “GALAXIES IN THE UNIVERSE”

OBSERVED DATASET

In practice

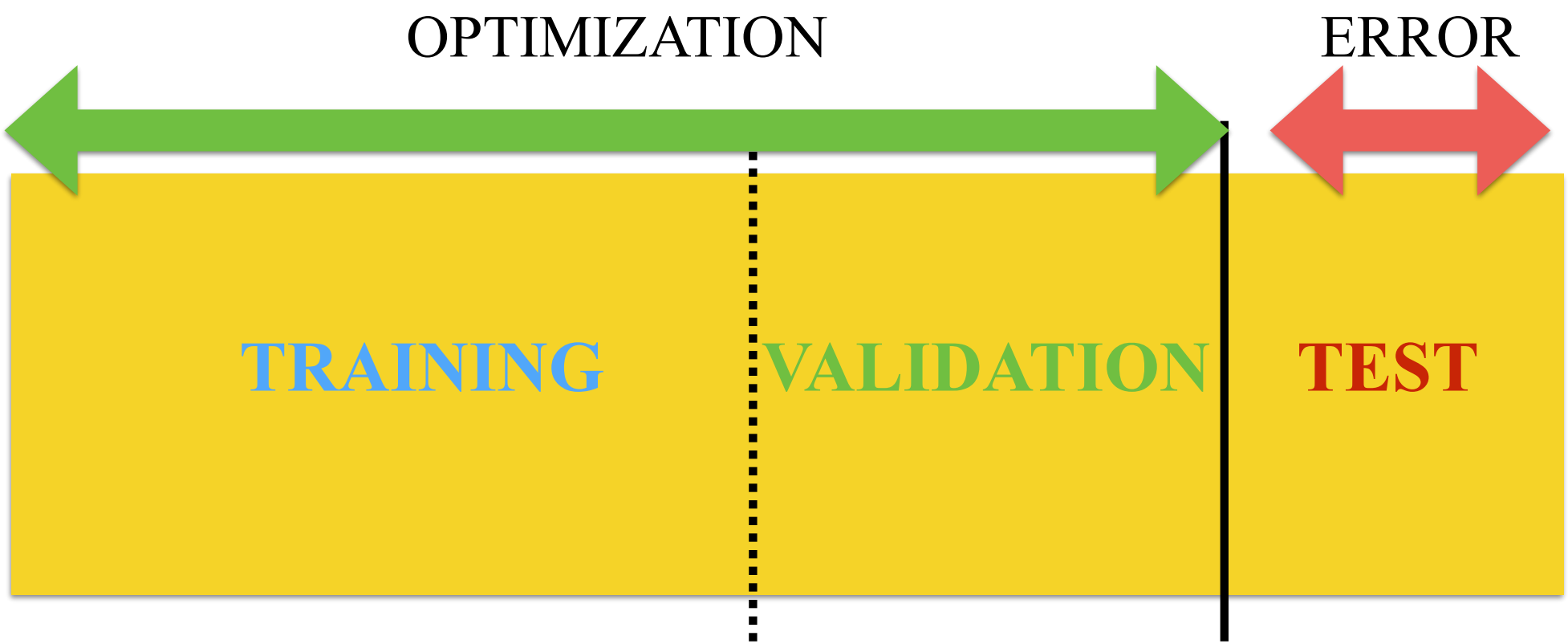


training set: use to train the classifier

validation set: use to monitor performance in real time - check for overfitting

test set: use to train the classifier

In practice



NO CHEATING! NEVER USE TRAINING TO VALIDATE YOUR ALGORITHM!

The algorithm used to minimize is called OPTIMIZATION

THERE ARE SEVERAL OPTIMIZATION TECHNIQUES

Optimization

THERE ARE SEVERAL OPTIMIZATION TECHNIQUES

THEY DEPEND ON THE MACHINE LEARNING ALGORITHM

Optimization

THERE ARE SEVERAL OPTIMIZATION TECHNIQUES

THEY DEPEND ON THE MACHINE LEARNING ALGORITHM

NEURAL NETWORKS USE THE GRADIENT DESCENT AS WE WILL SEE LATER

$$W_{t+1} = W_t - \lambda_h \bigtriangledown f(W_t)$$

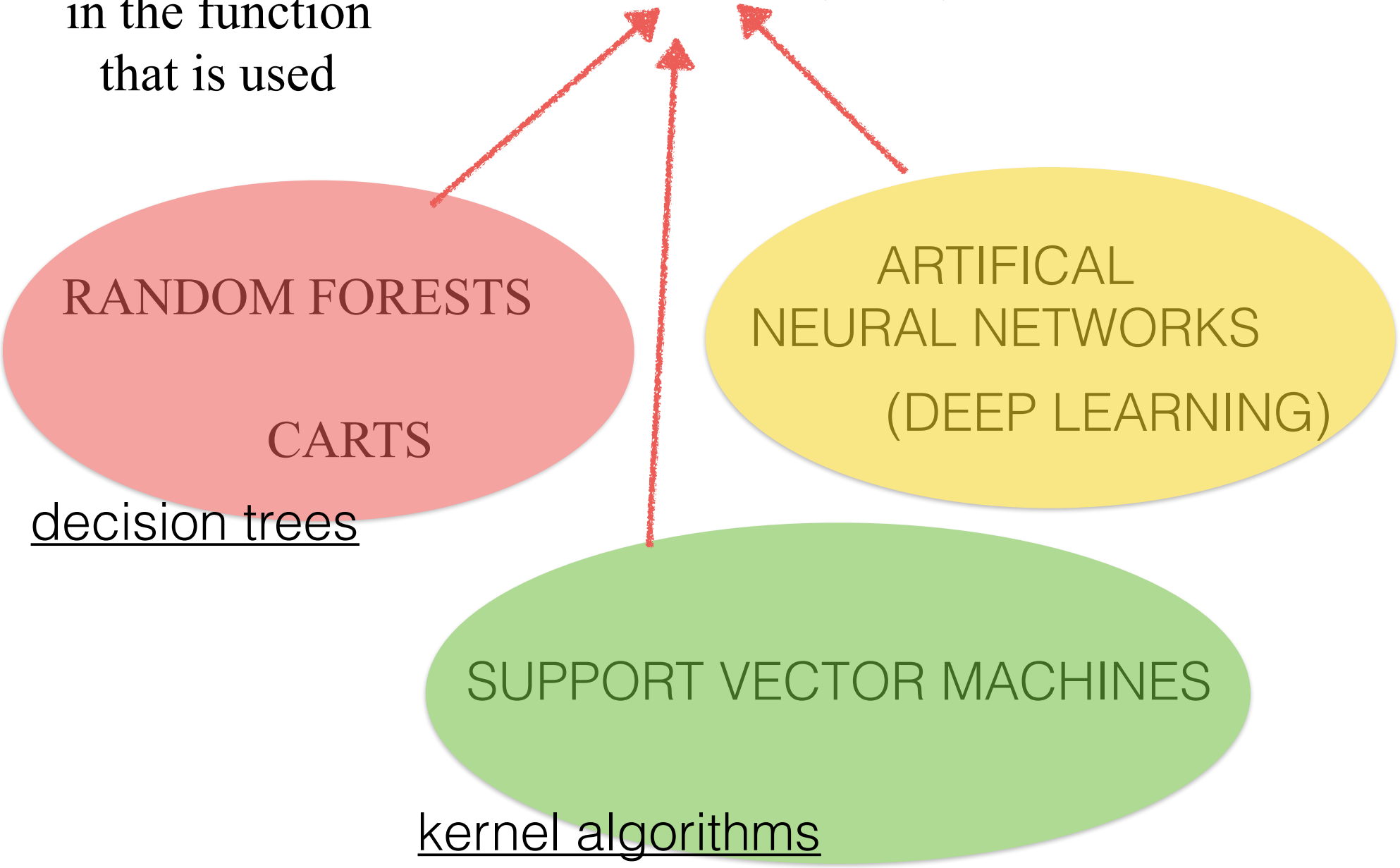
weights to be learned

epoch

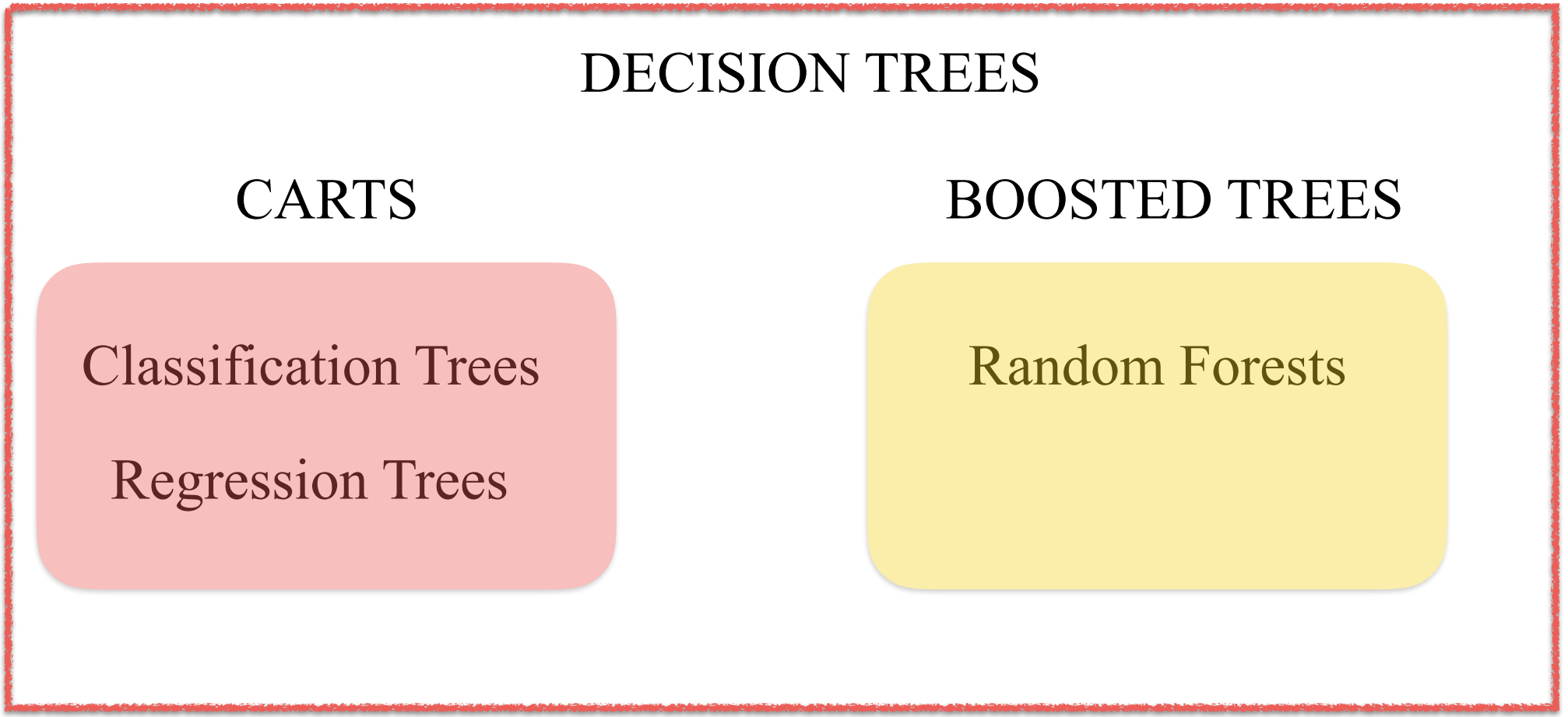
learning rate

The differences are in the function that is used

$$f_W(\vec{x})$$

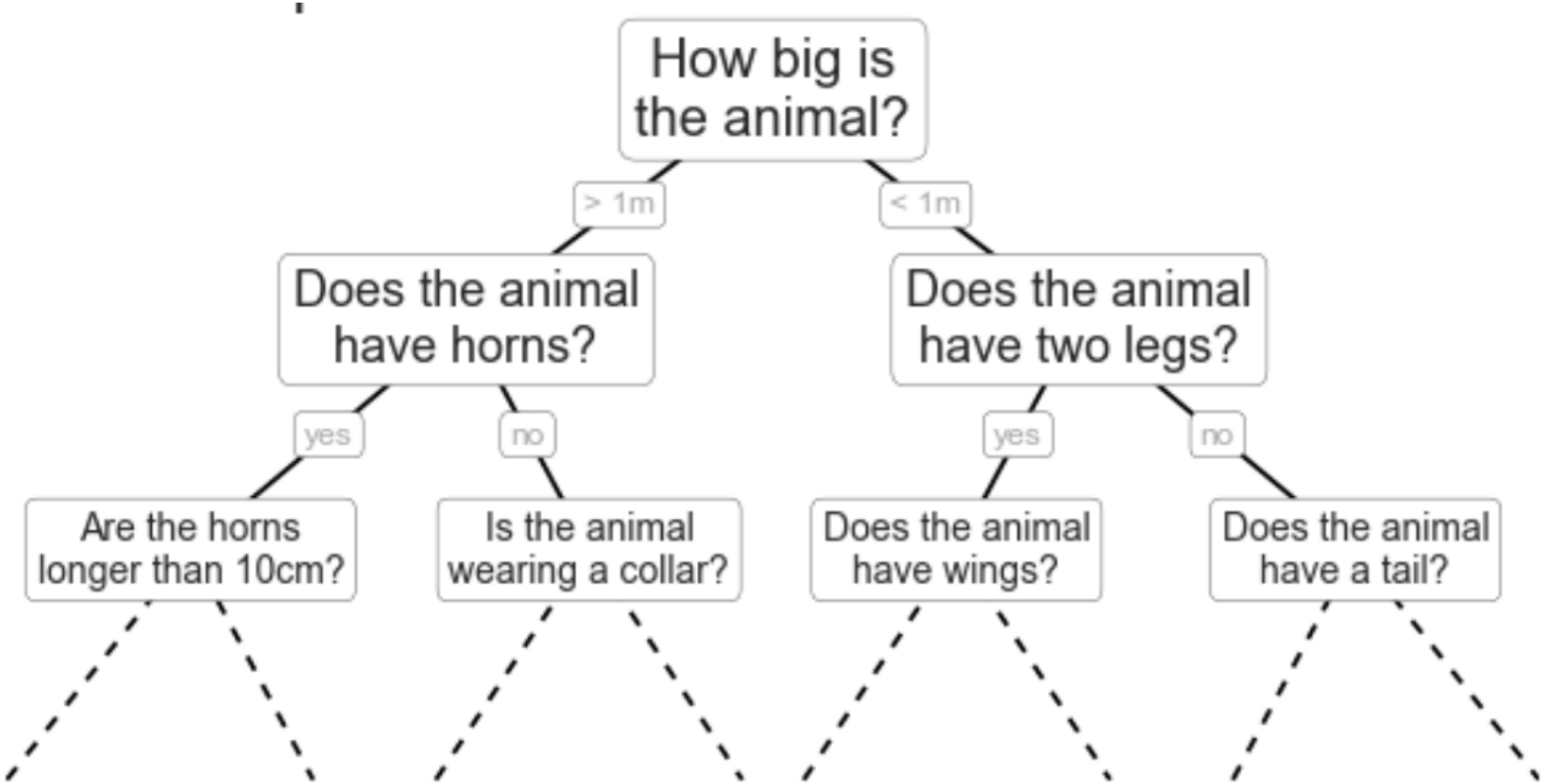


DECISION TREES ALGORITHMS

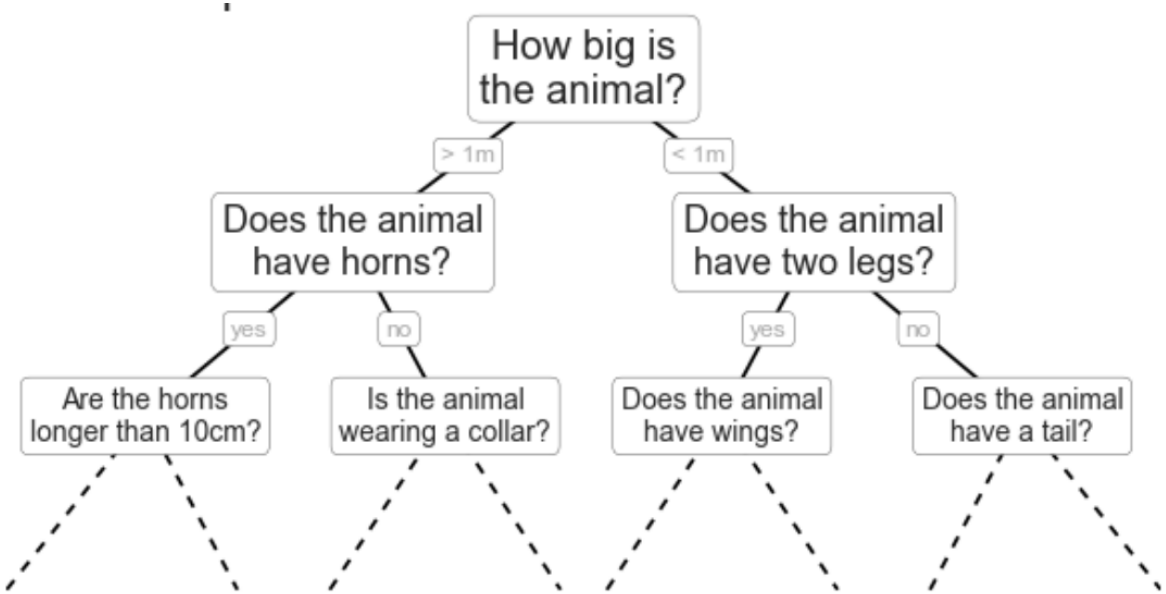


CLASSIFICATION AND REGRESSION TREES (CARTS)

THIS IS THE SIMPLEST AND MORE INTUITIVE MACHINE LEARNING ALGORITHM



DECISION TREES (CARTS)



IT IS BUILT IN AN ITERATIVE WAY

1. FROM THE INPUT PARAMETERS, FIRST FIND THE PROPERTY THAT BEST SPLITS INTO 2 GROUPS **[I.E. MINIMIZES SOME LOSS FUNCTION]**

2. REPEAT STEP 1 WITH ANOTHER PARAMETER

3.AT THE END THERE IS A TREE WHERE, AT EACH POINT, ONE OF TWO DECISIONS CAN BE MADE

DECISION TREES (CARTS)

TYPICAL METRICS USED:

[THE IDEA IS TO FIND THE SPLITTING VALUE THAT PUTS ALL OBJECTS OF A GIVEN CLASS IN ONE LEAF]

DECISION TREES (CARTS)

TYPICAL METRIC USED:

GINI IMPURITY $$G = 1 - \sum_{j=1}^{c} p_j^2$$

DECISION TREES (CARTS)

TYPICAL METRIC USED:

GINI IMPURITY $$G = 1 - \sum_{j=1}^{c} p_j^2$$

fraction of
objects in each class given
a split value

Decision trees (CARTS)

TYPICAL METRIC USED:

GINI IMPURITY

$$G = 1 - \sum_{j=1}^{c} p_j^2$$

IF THERE IS ONLY ONE CLASS
THE GINI IMPURITY GOES TO 0

fraction of
objects in each class given
a split value

DECISION TREES (CARTS)

TRAINING DATA

$$\vec{x_i} = (p_1, p_2, ..., p_n)$$

(parameters)

[b/a, color, size, SFR …]

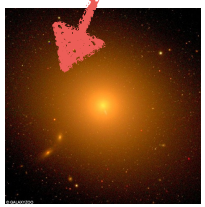
$(\vec{x_2}, \vec{y_2})$

$(\vec{x_1}, \vec{y_1})$

$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$$\vec{y_i} = (y_1, y_2, ..., y_n)$$

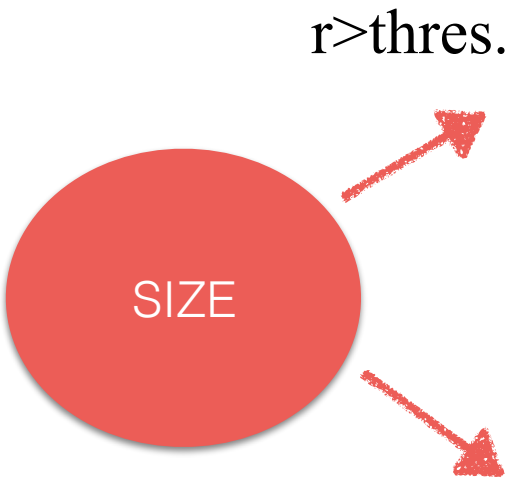
(classes)



[elliptical / spiral]

DECISION TREES (CARTS)

[elliptical / spiral]



LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

DECISION TREES (CARTS)

[elliptical / spiral]

SIZE

r>thres.

COLOR

c>thres.

c<thres.

r<thres.

COLOR

c>thres.

c<thres.

LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

LOOK FOR 2nd THRES. SO THAT GINI IMPURITY IS MINIMAL

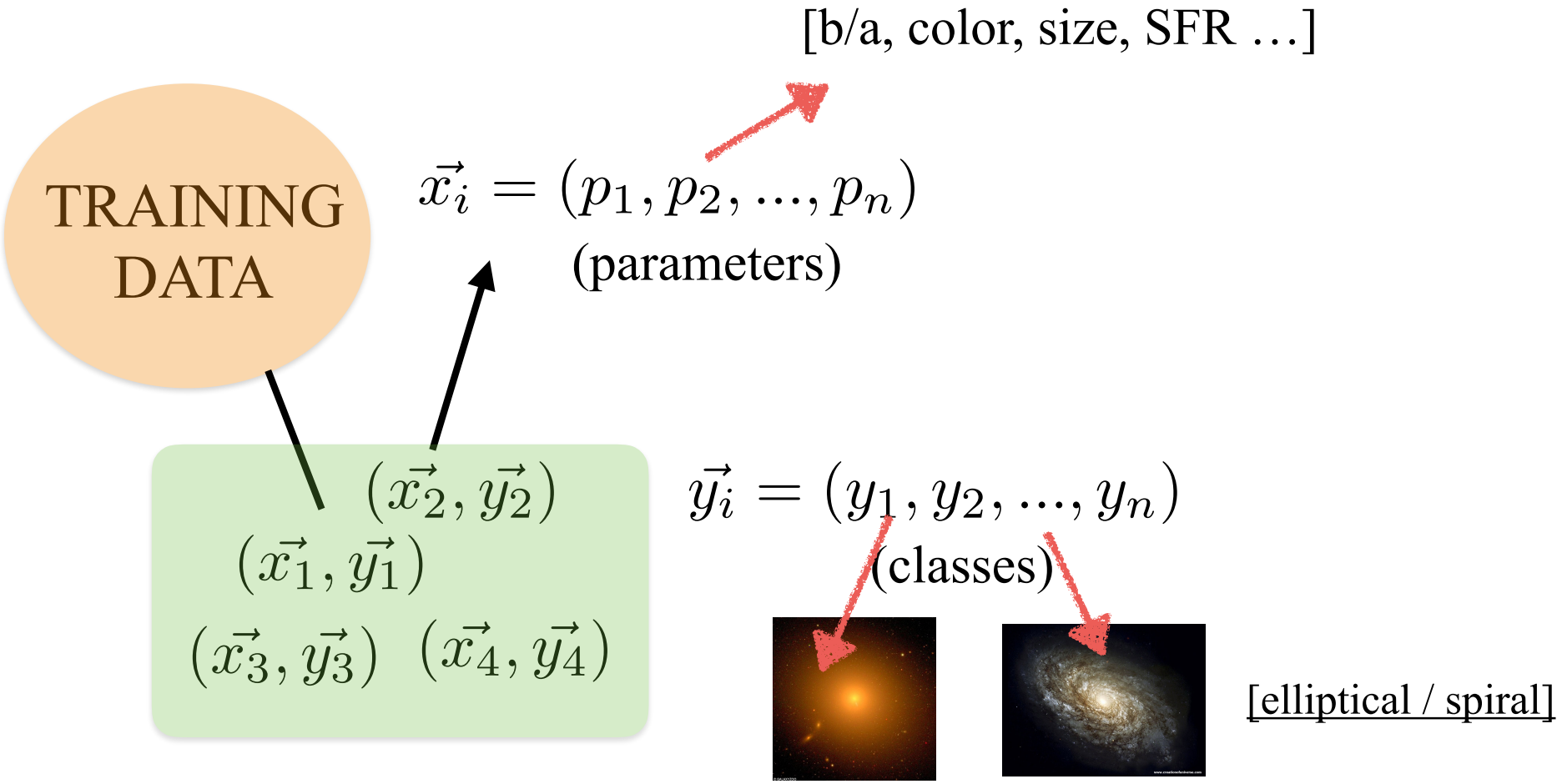
RANDOM FORESTS

ONE PROBLEM WITH CLASSIFICATION TREES IS THAT THEY CAN EASILY OVERFIT

THE DECISIONS ARE VERY SPECIFIC TO THE TRAINING SET AND NOT REPRESENTATIVE OF THE FULL POPULATION

RANDOM FORESTS

RANDOM FORESTS TRY TO SOLVE THIS PROBLEM BY INTRODUCING SOME RANDOM INFORMATION IN THE TRAINING PROCESS



RANDOM FORESTS

$(\vec{x_2}, \vec{y_2})$

$(\vec{x_1}, \vec{y_1})$

$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$\vec{x_i} = (p_1, p_2, ..., p_n)$

RANDOM FORESTS

$(\vec{x_2}, \vec{y_2})$
$(\vec{x_1}, \vec{y_1})$
$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$\vec{x_i} = (p_1, p_2, .., p_n)$

[elliptical / spiral]

SIZE

r>thres.

COLOR

c>thres.

c<thres.

r<thres.

COLOR

c>thres.

c<thres.

LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

LOOK FOR 2nd THRES. SO THAT GINI IMPURITY IS MINIMAL

RANDOM FORESTS

$(\vec{x_2}, \vec{y_2})$

$(\vec{x_1}, \vec{y_1})$

$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$\vec{x_i} = (p_1, p_2, ..., p_n)$

[elliptical / spiral]

SIZE

r>thres.

Sersic

s>thres.

s<thres.

r<thres.

Sersic

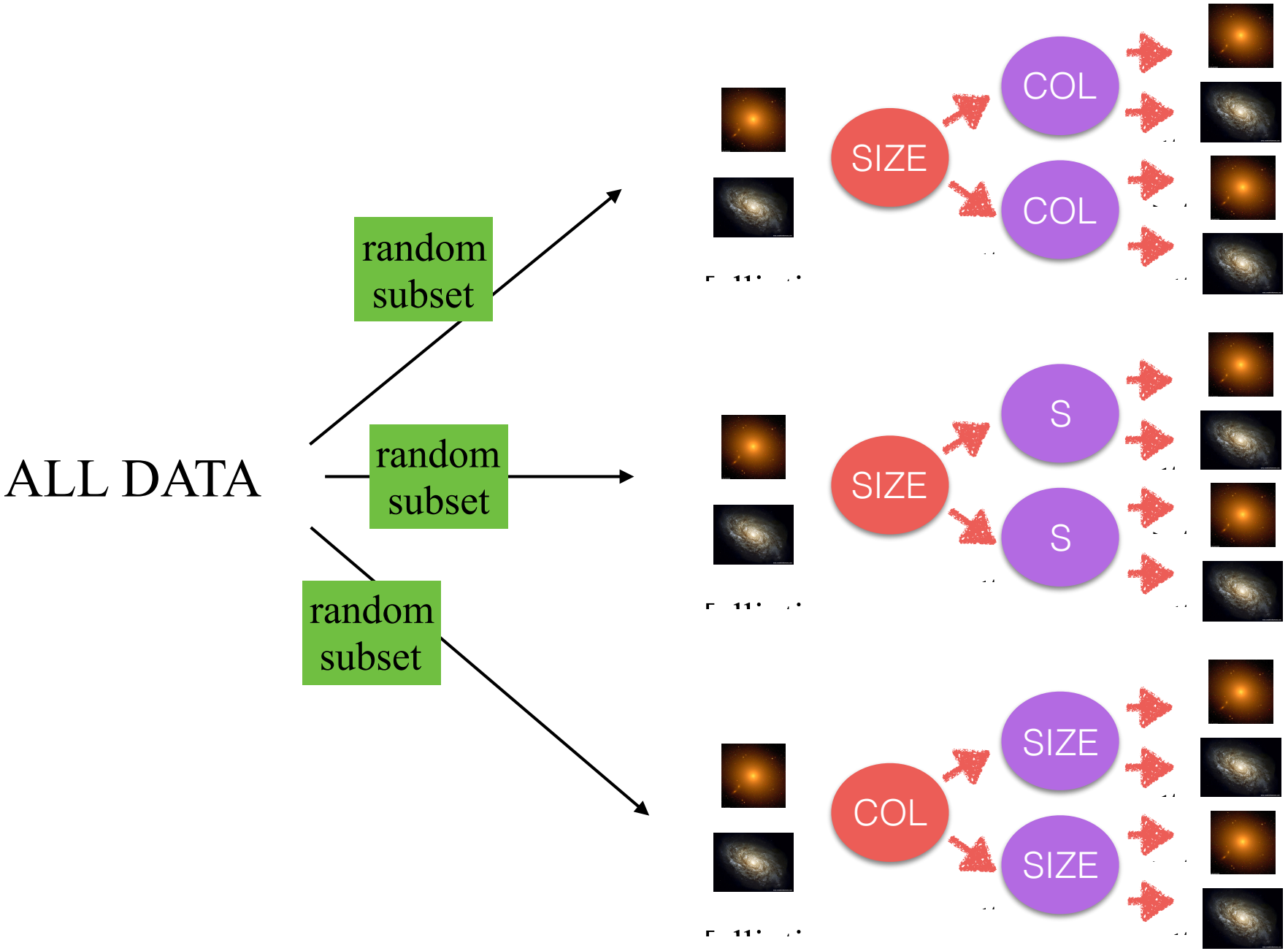
s>thres.

s<thres.

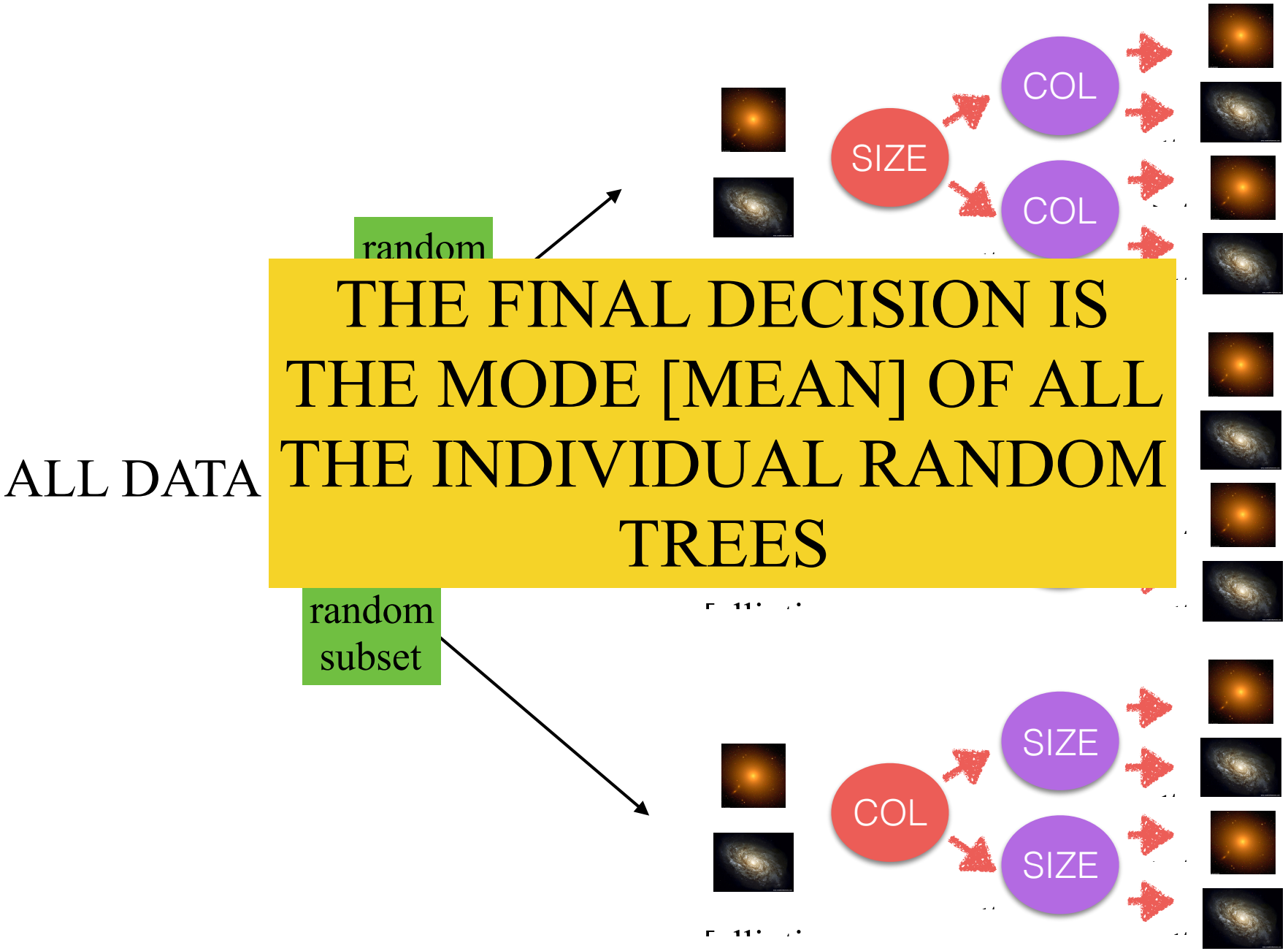
LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

LOOK FOR 2nd THRES. SO THAT GINI IMPURITY IS MINIMAL

RANDOM FORESTS



RANDOM FORESTS



Random Forests

ONE KEY ADVANTAGE OF DECISION TREE ALGORITHMS IS THAT THEY ARE VERY EASY TO INTERPRET

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

RANDOM FORESTS

ONE KEY ADVANTAGE OF DECISION TREE ALGORITHMS IS THAT THEY ARE VERY EASY TO INTERPRET

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

Rank	Property	AUC	Success label[a]
	ALL	0.9074 ± 0.0106	Outstanding
1	CVD	0.8559 ± 0.0039	Excellent
2	M_{bulge}	0.8335 ± 0.0060	Excellent
3	B/T	0.8267 ± 0.0028	Excellent
4	M_{halo}	0.7983 ± 0.0045	Acceptable
5	M_*	0.7819 ± 0.0025	Acceptable
6	M_{disc}	0.7124 ± 0.0016	Acceptable
7	δ_5	0.5894 ± 0.0015	Unacceptable
8	Re	0.5599 ± 0.0013	Unacceptable

IMPORTANCE OF PARAMETERS T TO PREDICT IF A GALAXY IS QUENCHED

Teimiroonia+16

PRACTICAL NOTE: *Python scikit learn*

- All these different methods are standard and available in Python
- Very easy to use
- All info here

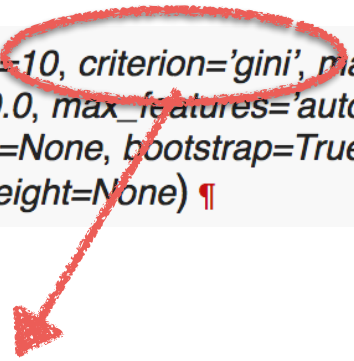
sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

NUMBER OF RANDOM TREES

sklearn.ensemble.RandomForestClassifier

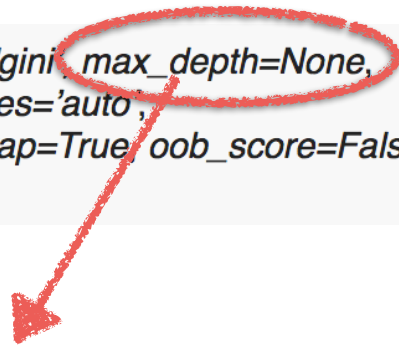
class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]



METRIC TO MINIMIZE

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]



MAXIMUM NUMBER OF SPLITS

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

MAXIMUM NUMBER OF SPLITS

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
>>>
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
>>> print(clf.feature_importances_)
[ 0.17287856  0.80608704  0.01884792  0.00218648]
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

MAXIMUM NUMBER OF SPLITS

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
>>>
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

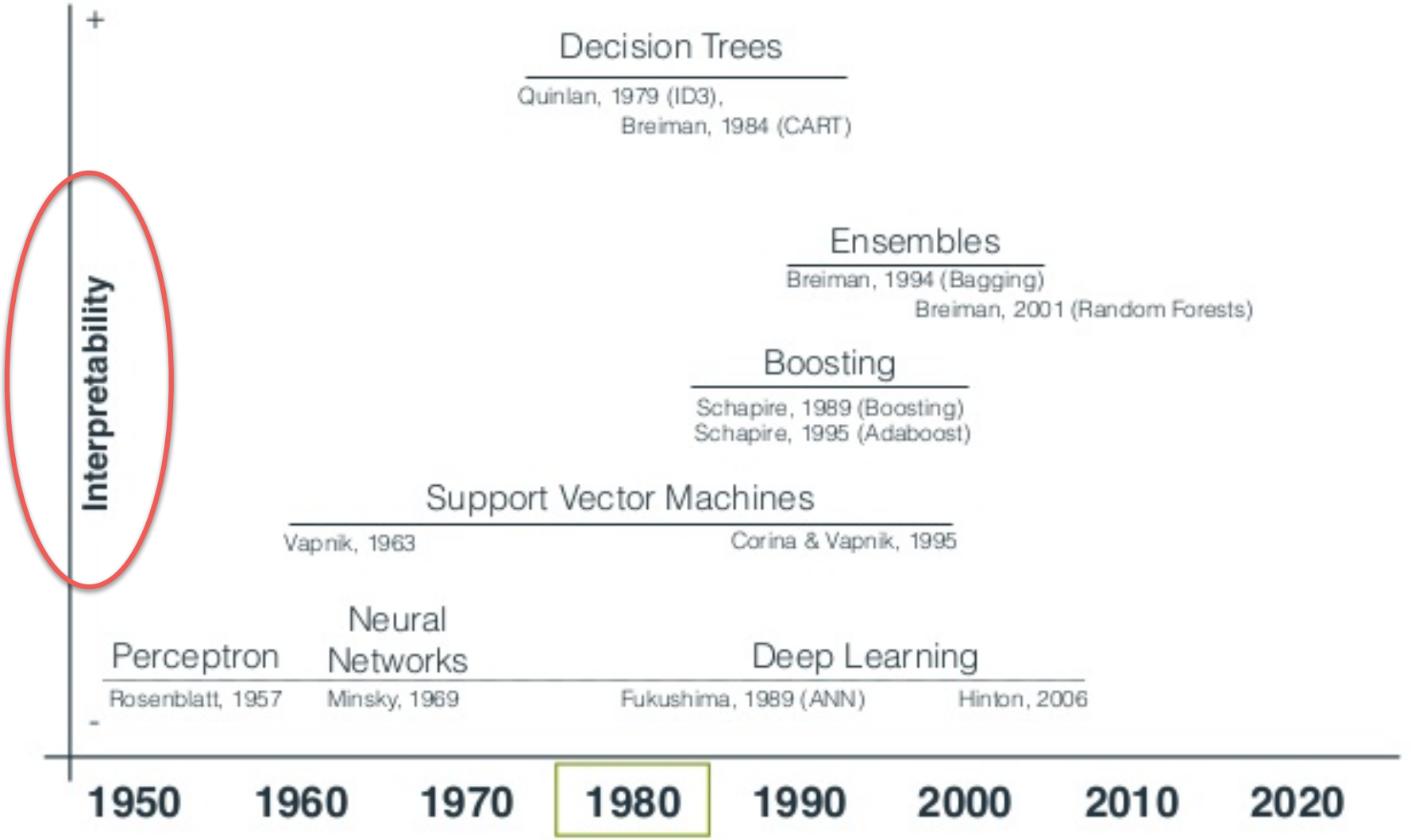
DIFFERENT CLASSIFIERS ARE OBJECTS WITH METHODS TO FIT, PREDICT ETC..

HOW TO CHOOSE YOUR CLASSICAL CLASSIFIER?

NO RULE OF THUMB - REALLY DEPENDS ON APPLICATION

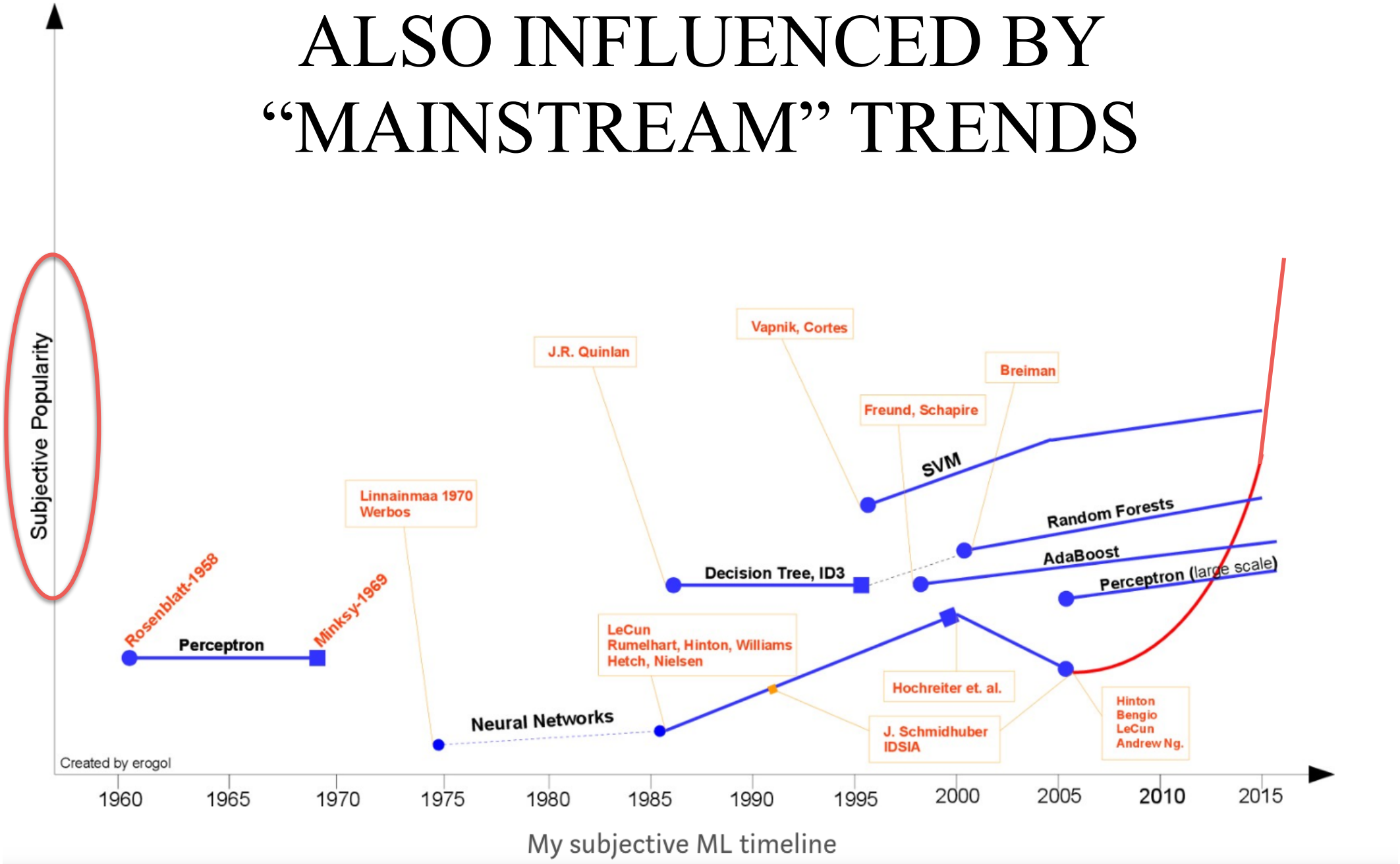
ML METHOD	++	—	Python
CARTS / RANDOM FOREST	Easy to interpret ("White box") Litte data preparation Both numerical + categorical	Over-complex trees Unstable Biased tress if some classes dominate	sklearn.ensemble.RandomForestClassifier sklearn.ensemble.RandomForestRegressor
SVM	Easy to interpret + Fast Kernel trick allows no linear problems	not very well suited to multi-class problems	sklearn.svm sklearn.svc
NN	seed of deep-learning very efficient with large amount of data as we will see	more difficult to interpret computing intensive	sklearn.neural_network.MPL_CLassifier sklearn.neural_network.MPL_Regressor

CAN DEPEND ON YOUR MAIN INTEREST



credit

ALSO INFLUENCED BY "MAINSTREAM" TRENDS

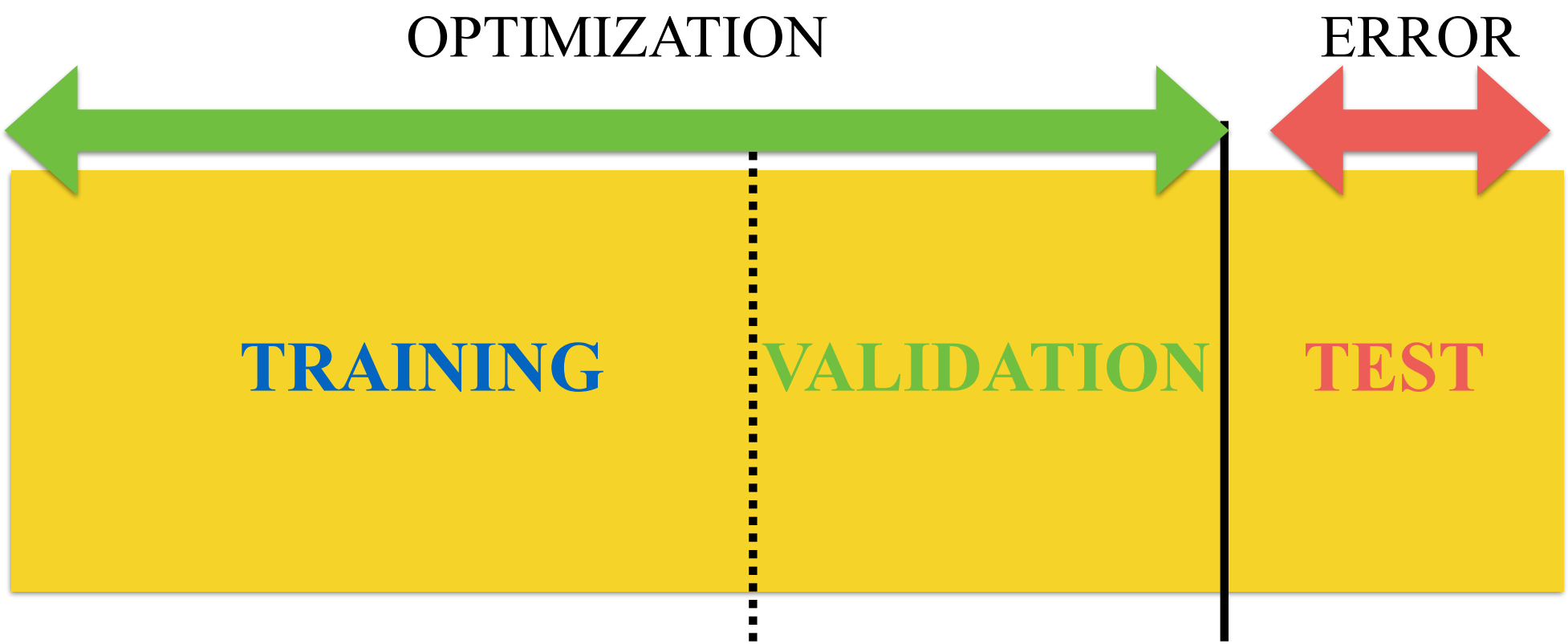


Source

HOW DO I KNOW THAT MY MACHINE LEARNING ALGORITHM IS WORKING?

- The way results are evaluated depends strongly on the type of problem
  - Binary Classification: ACC, ROC, P-C
  - Multi-Class: Confusion Matrix
  - Regression: RMSE, Bias, Scatter etc..

REMEMBER



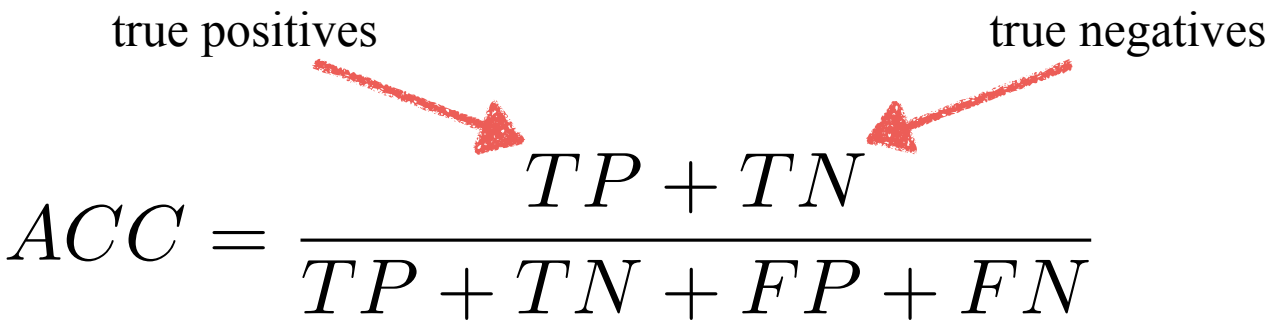
training set: use to train the classifier

validation set: use to monitor performance in real time - check for overfitting

test set: use to train the classifier

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY



MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY

NOT VERY INFORMATIVE IF UNBALANCED CLASSES

true po ue negatives

$$ACC = \frac{}{TP + TN + FP + FN}$$

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

[90% E, 10% S]