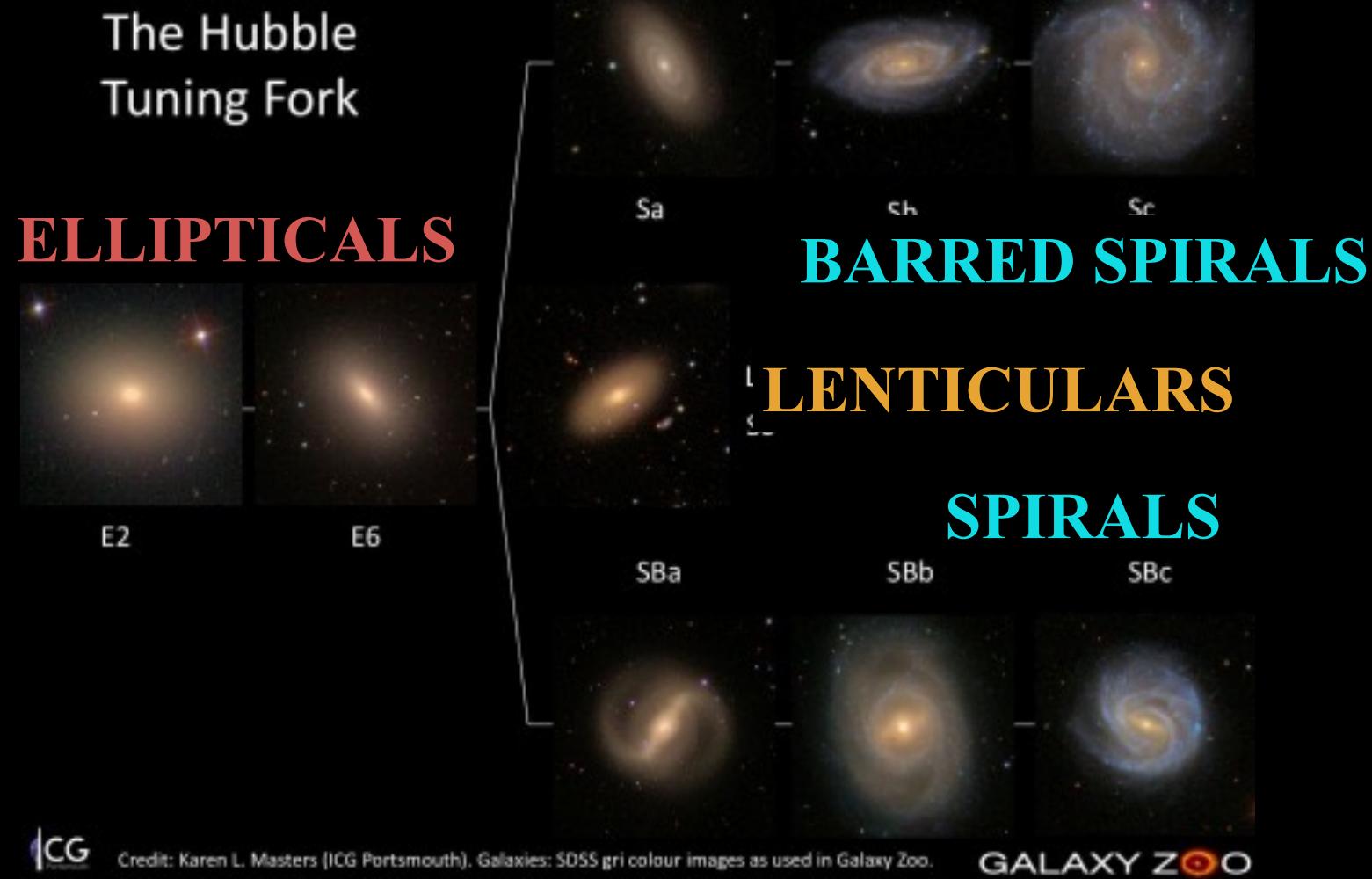
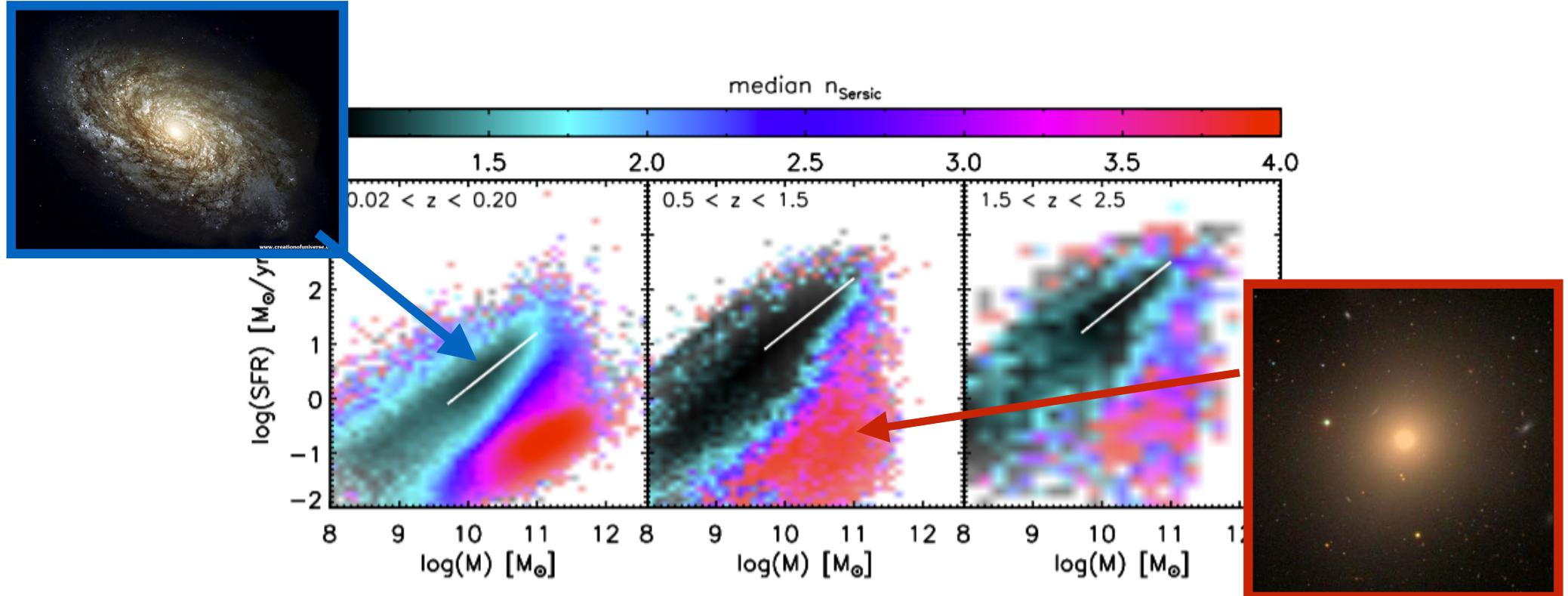


A TECHNICAL PAUSE...

Morphologies...



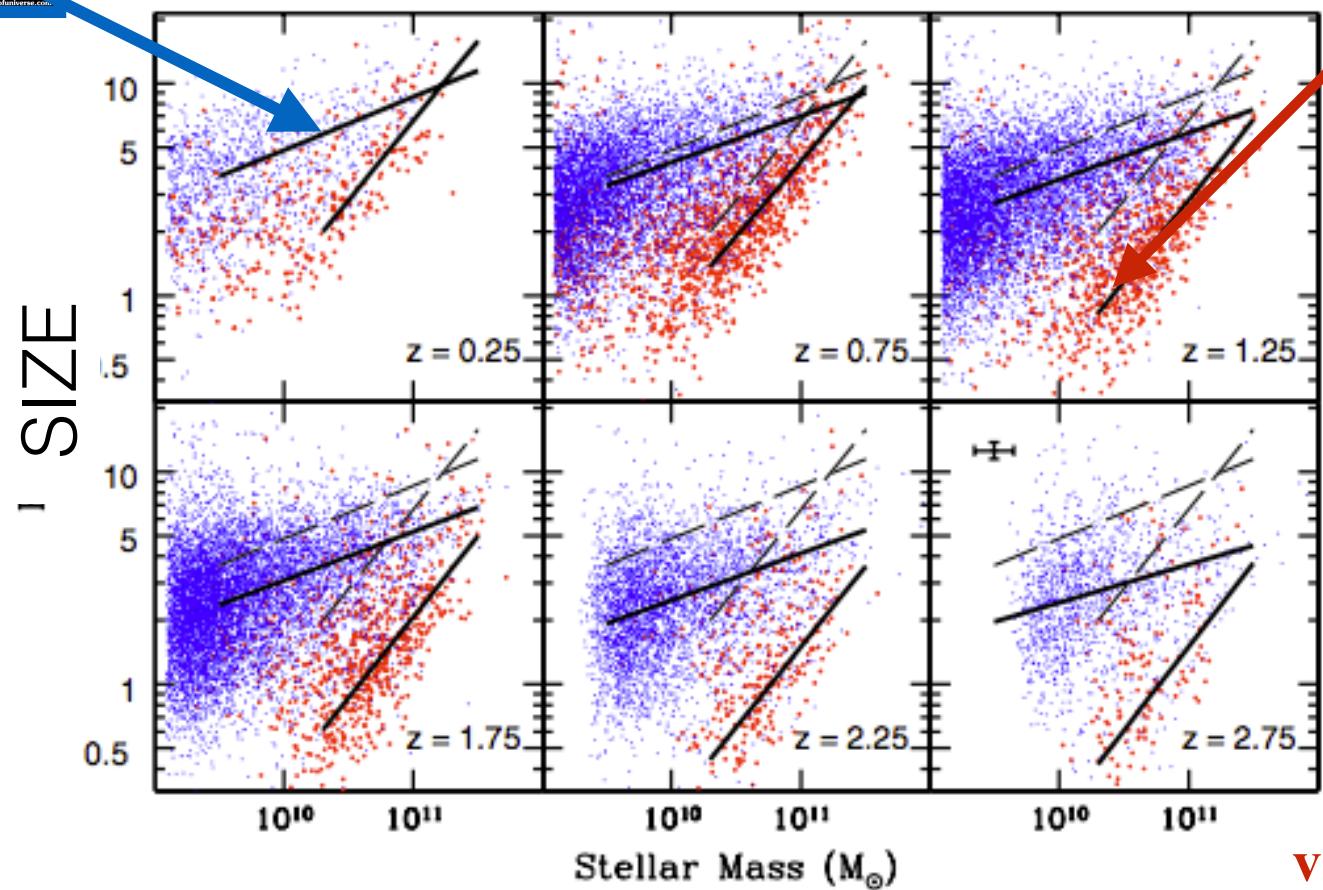
Structure/morphology as tracer of past history...

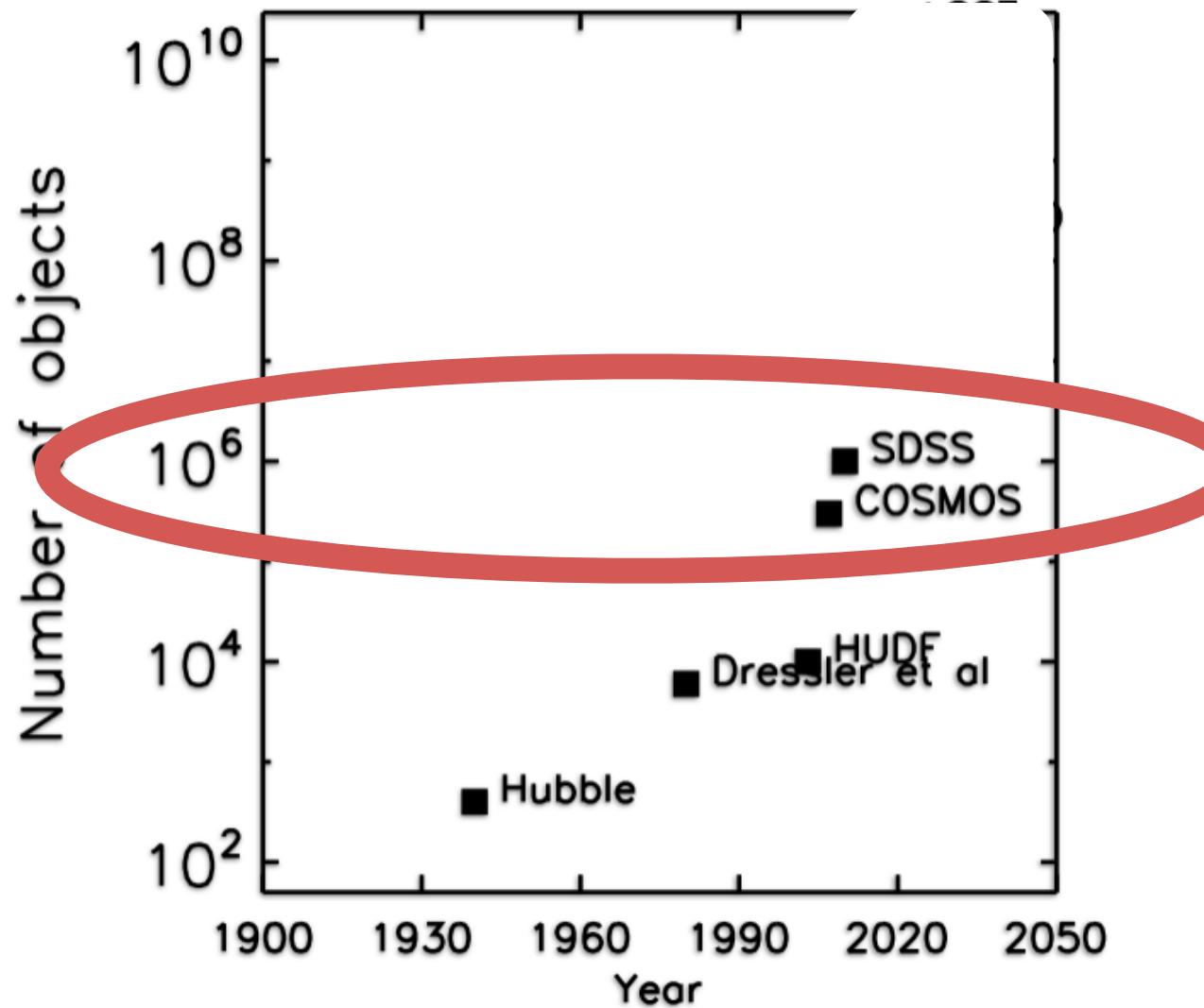


Wuyts+11

Quenched/SF galaxies have different morphologies

Structure/morphology as tracer of past history...





GALAXY ZOO

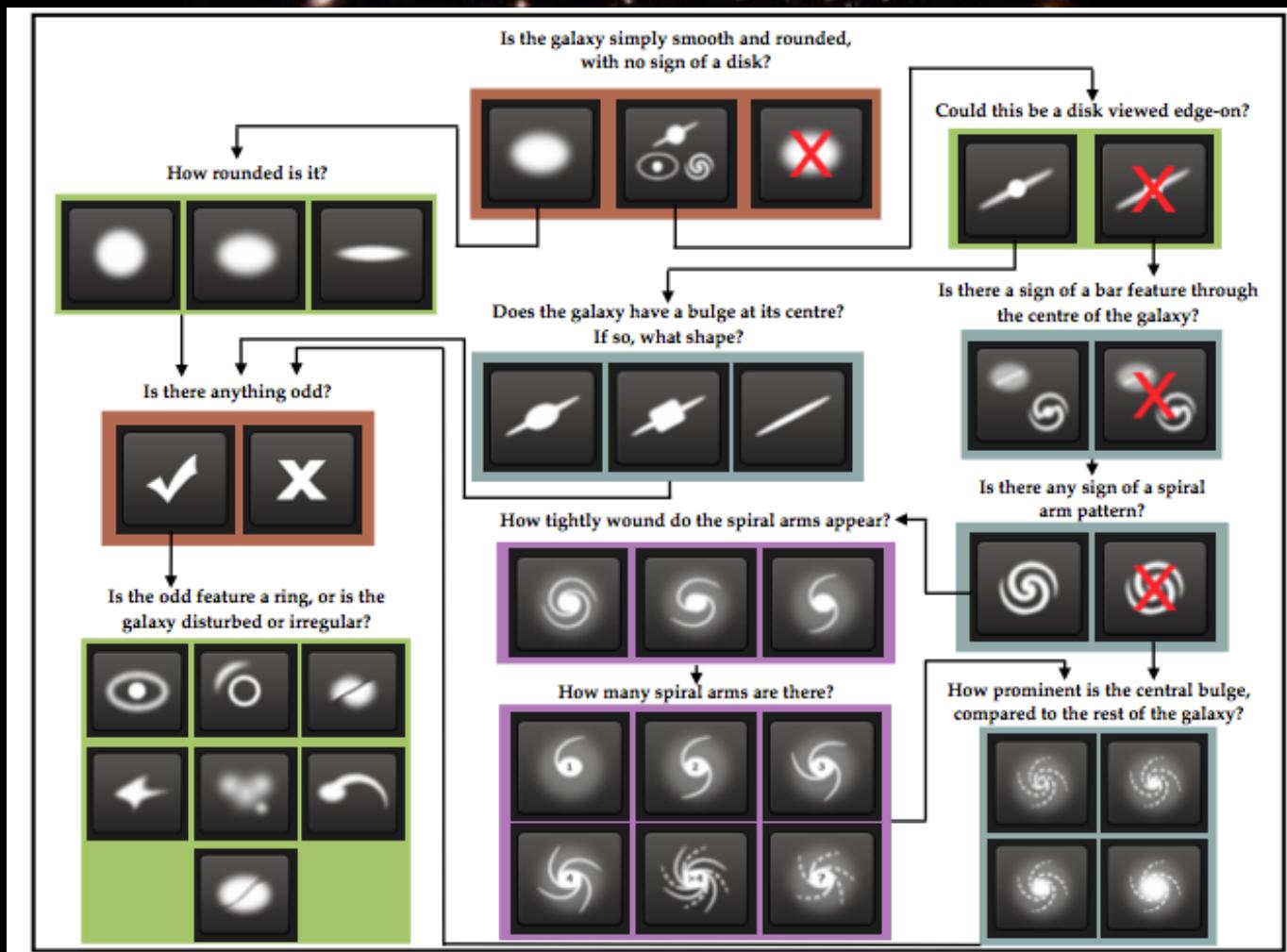
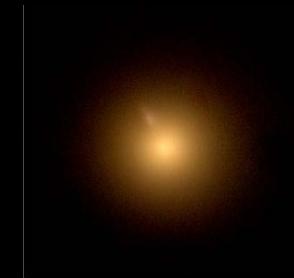


Figure 1. Flowchart of the classification tasks for GZ2, beginning at the top centre. Tasks are colour-coded by their relative depths in the decision tree. Tasks outlined in brown are asked of every galaxy. Tasks outlined in green, blue, and purple are (respectively) one, two or three steps below branching points in the decision tree. Table 2 describes the responses that correspond to the icons in this diagram.

HOWEVER...



GALAXY ZOO FOR EUCLID WOULD TAKE 70 YRS!!



AUTOMATED TECHNIQUES ARE UNAVOIDABLE

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**

CAT



CAT

SUPERVISED LEARNING

DOG



HUMAN LABELLING

DOG



the machine is told what to look for

CAT



CAT



DOG



DOG



SUPERVISED LEARNING

the machine is told what to look for

HUMAN LABELLING

TRAINING SET
OF LABELED
EXAMPLES

CAT



CAT



DOG



DOG



SUPERVISED LEARNING



ML



CAT

CAT



CAT



DOG



DOG



SUPERVISED LEARNING

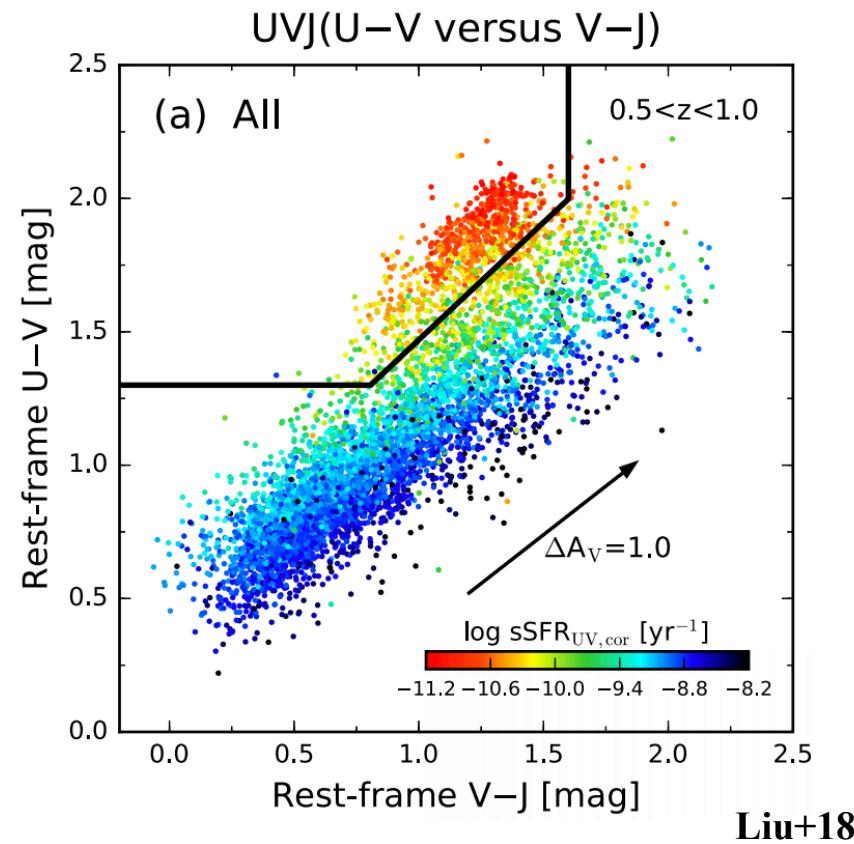


ML

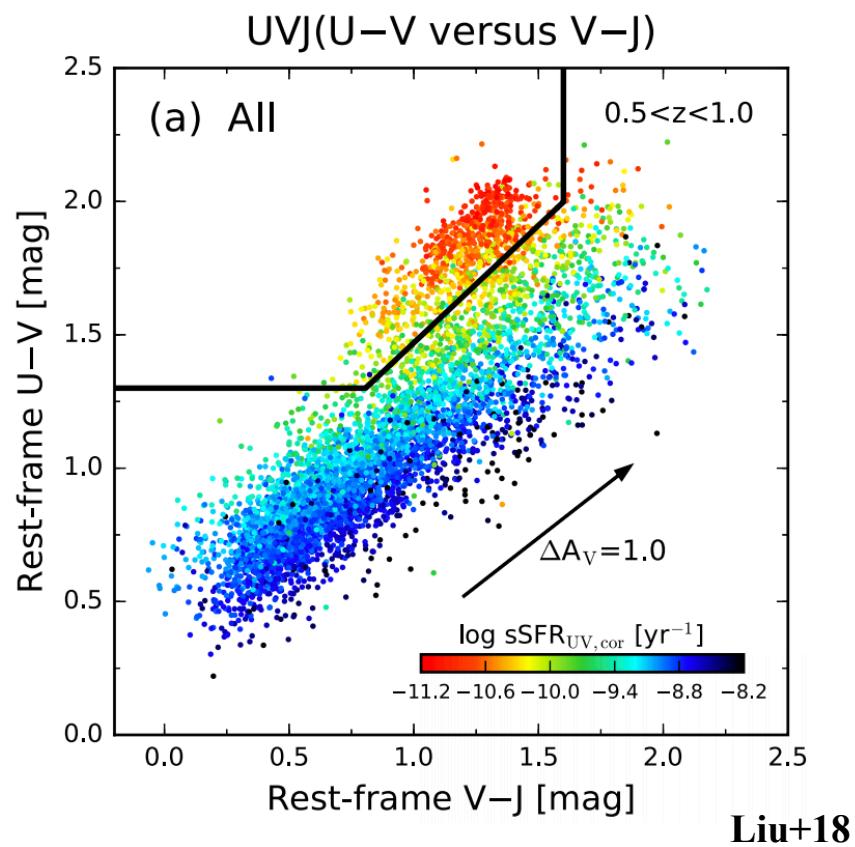


DOG

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

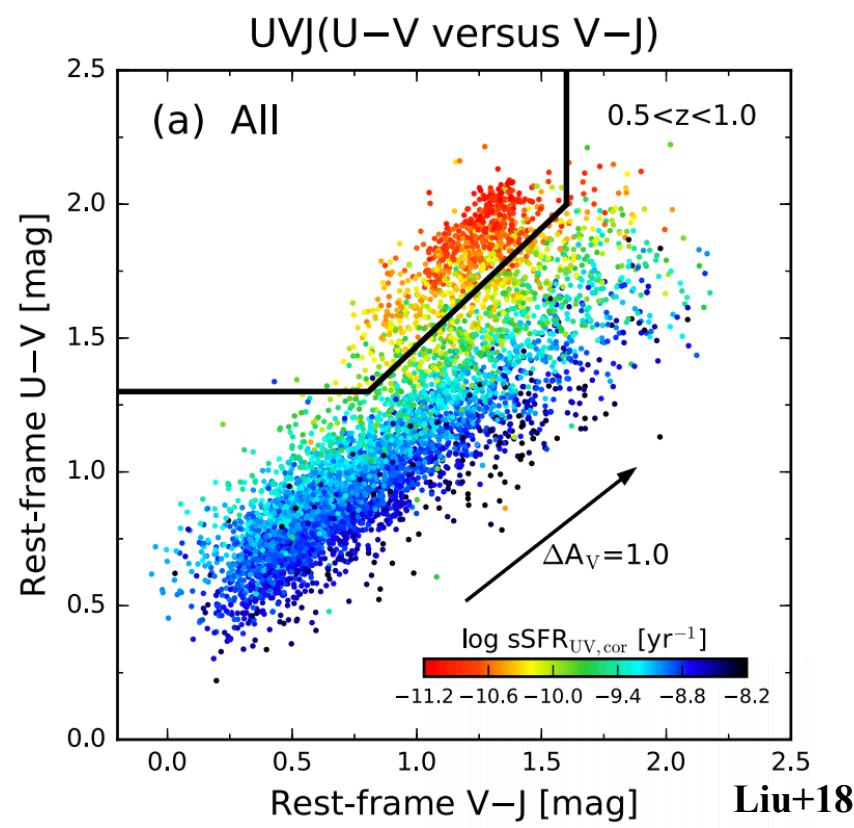


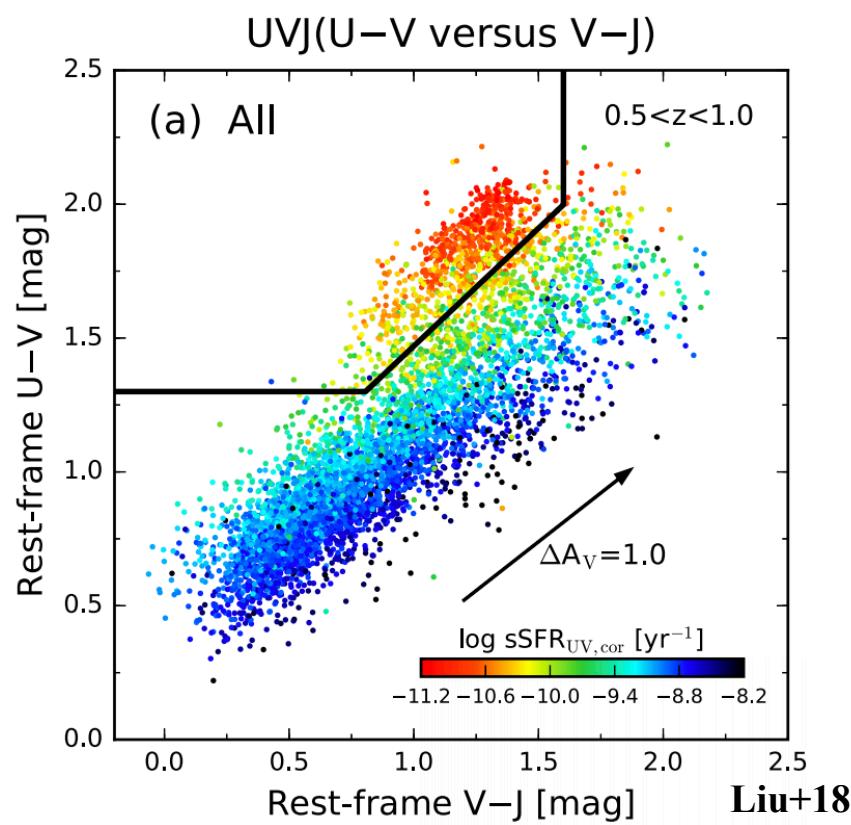
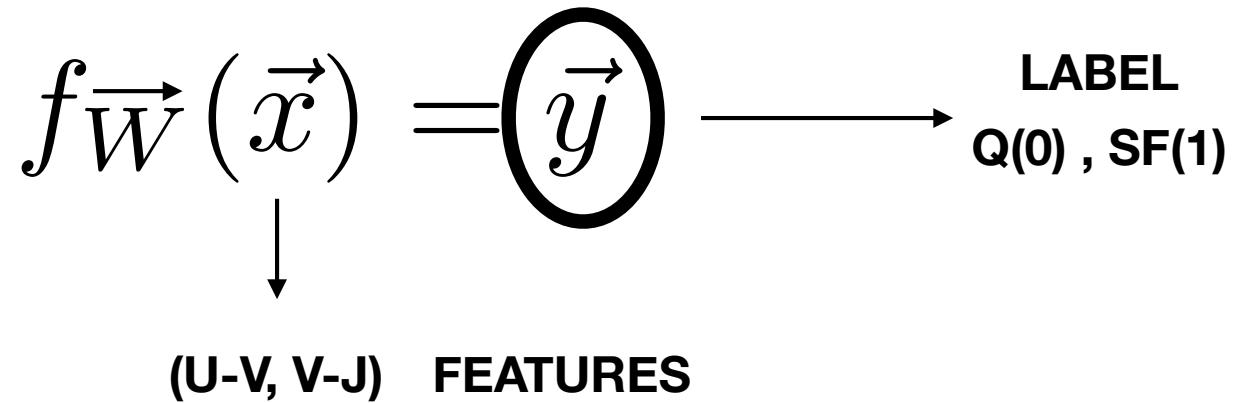
$$f_W(\vec{x}) = \vec{y}$$



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

LABEL
Q , SF





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

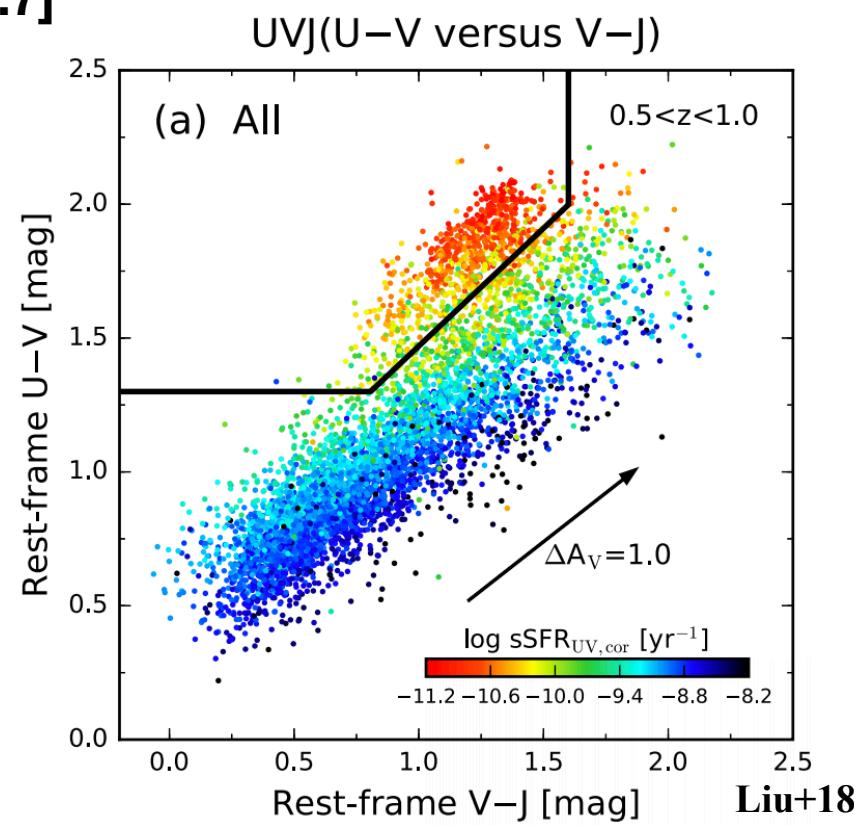
Q(0) , SF(1)

NETWORK FUNCTION

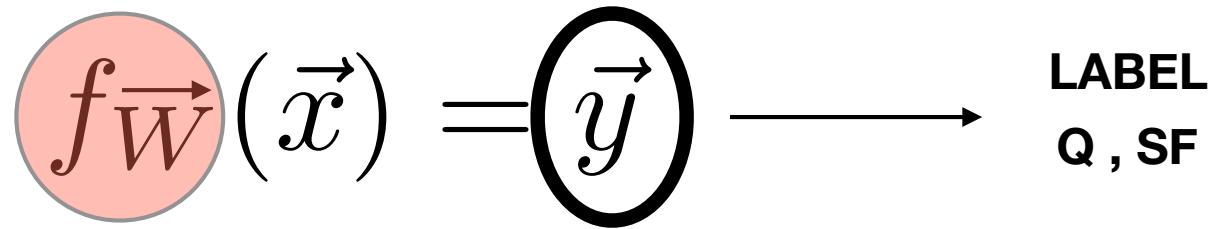
(U-V, V-J) FEATURES

$$\text{sgn}[(u-v)-0.8*(v-j)-0.7]$$

WEIGHTS



**“CLASSICAL”
MACHINE LEARNING**



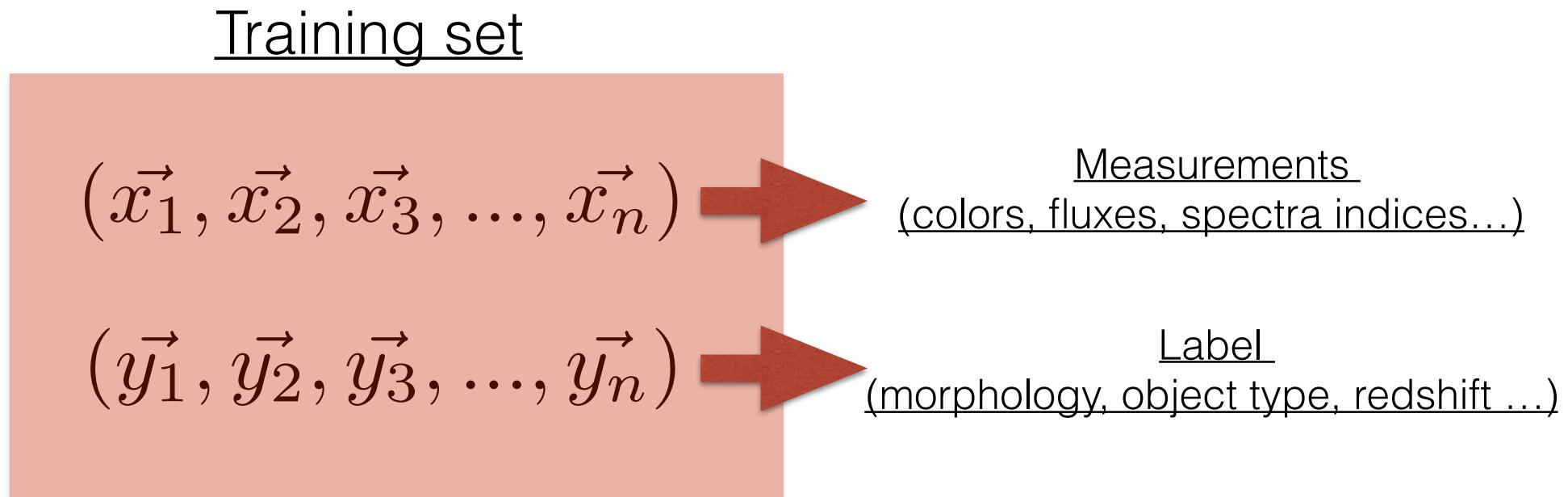
$$\text{sgn}[(u-v)-W_1*(v-j)-W_2]$$



**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



SUPERVISED LEARNING

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

Training set

$$(\vec{x}_1, \vec{x}_2, \vec{x}_3, \dots, \vec{x}_n)$$

$$(\vec{y}_1, \vec{y}_2, \vec{y}_3, \dots, \vec{y}_n)$$

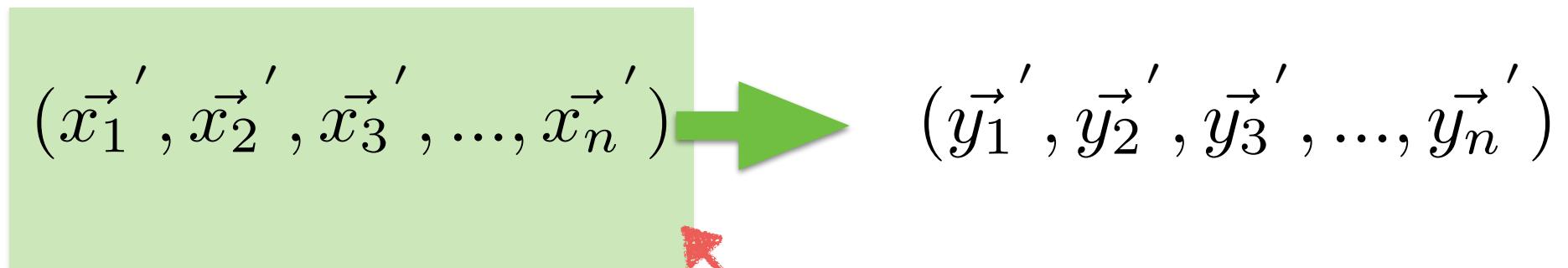


$$f_W(\vec{x}) = \vec{y}$$

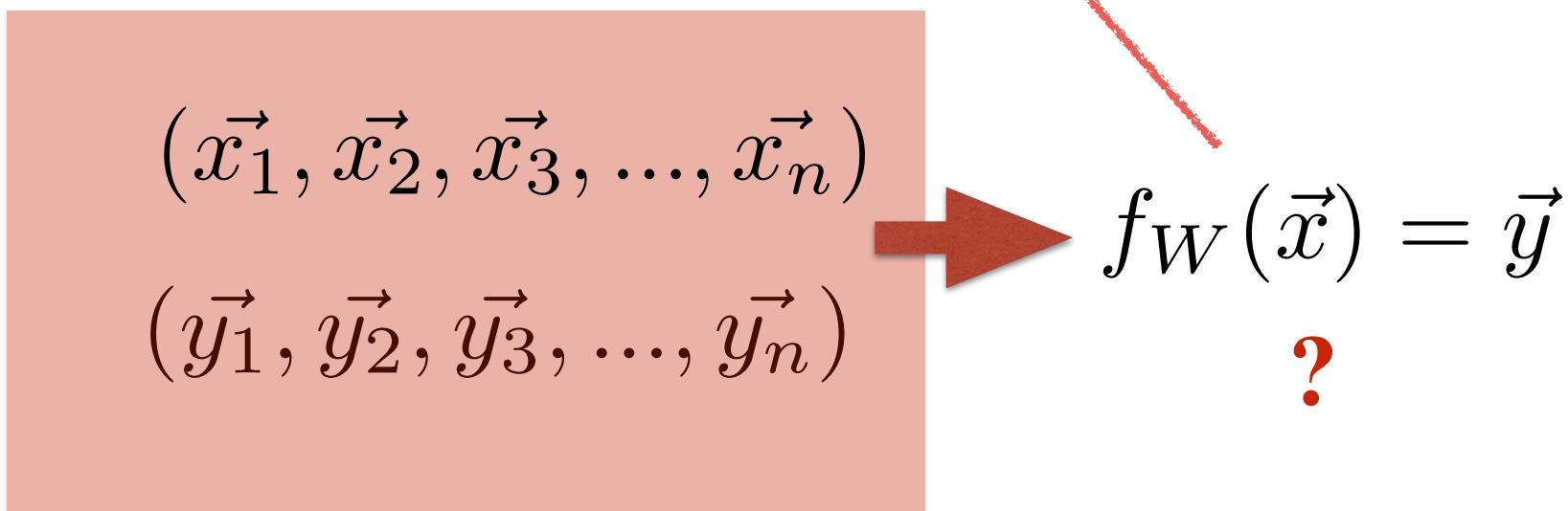
?

SUPERVISED LEARNING

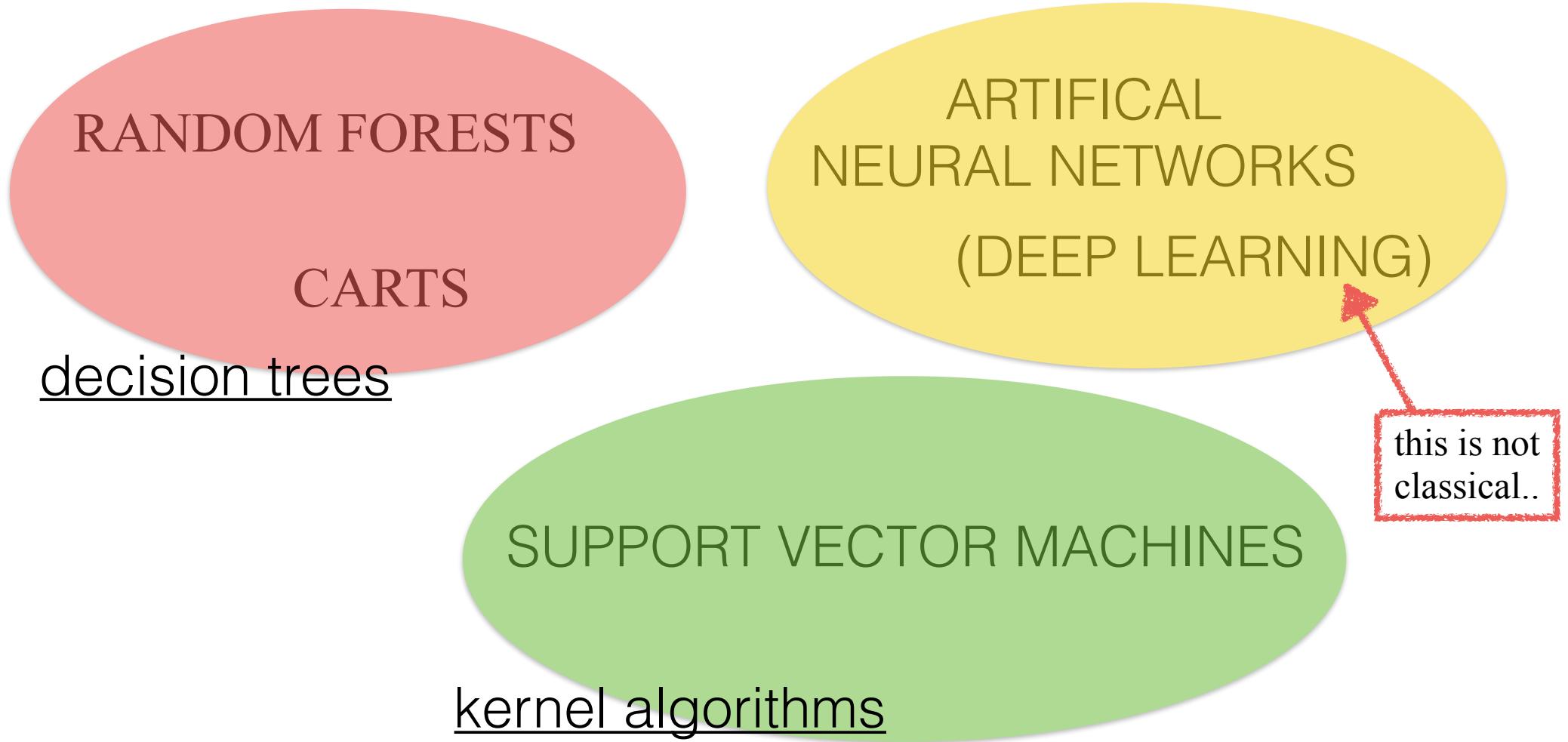
Unlabeled set



Training set



Different “classical” supervised machine learning methods



DECISION TREES ALGORITHMS

DECISION TREES

CARTS

Classification Trees

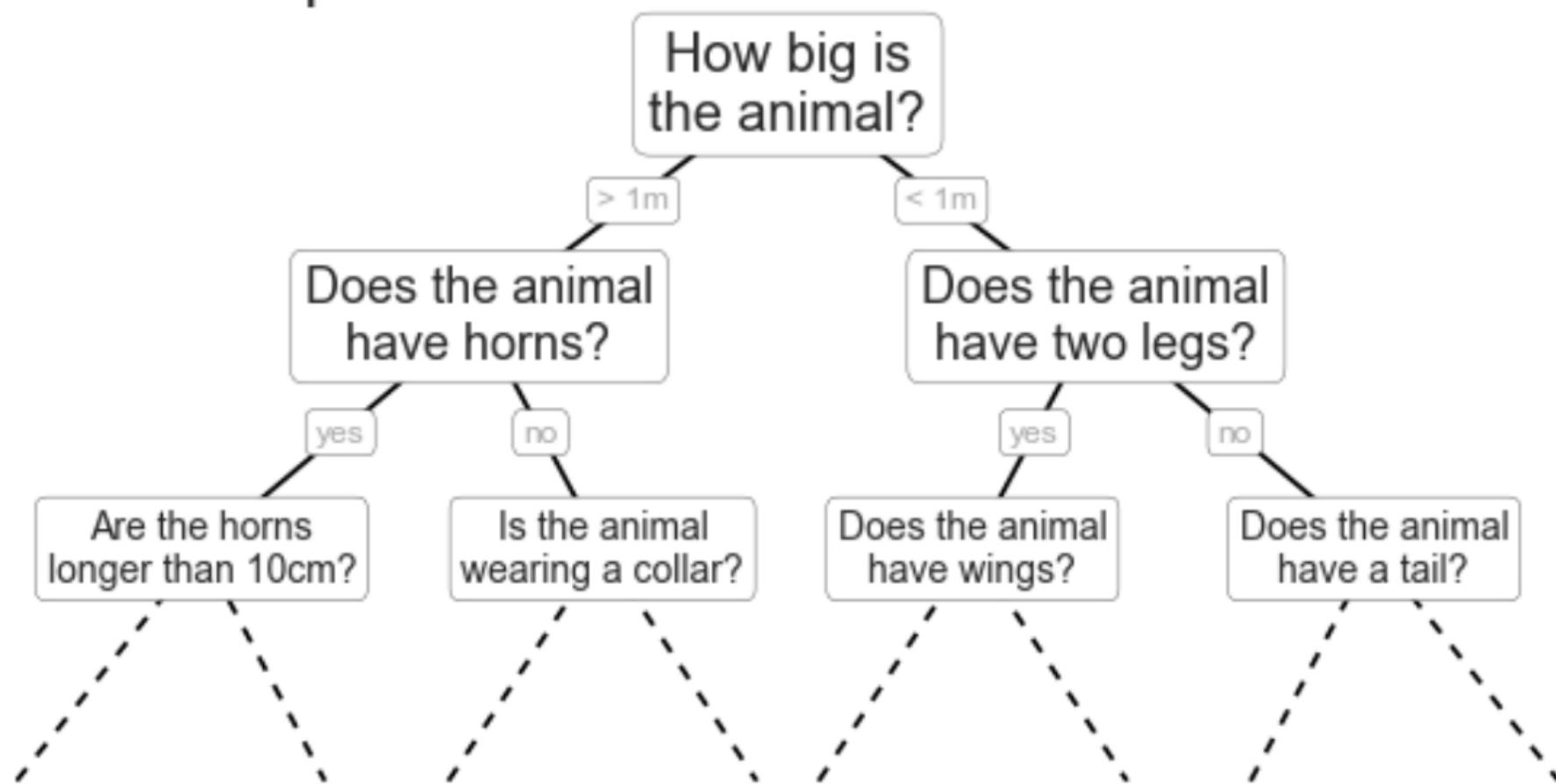
Regression Trees

BOOSTED TREES

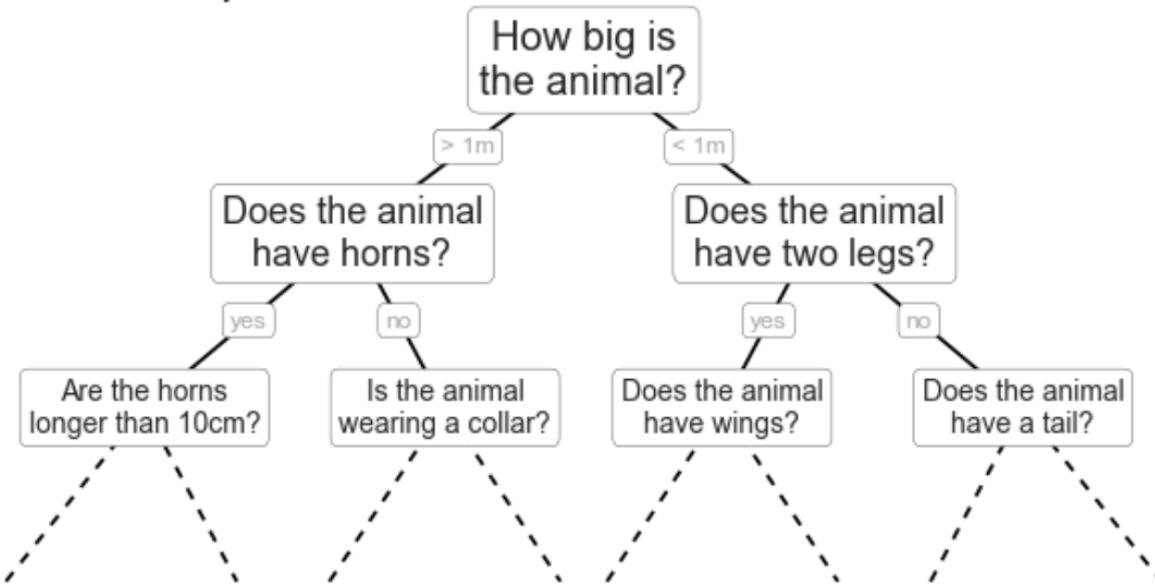
Random Forests

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 LOSS FUNCTION USED:

GINI IMPURITY

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

DECISION TREES (CARTS)

TYPICAL LOSS FUNCTION 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 LOSS FUNCTION USED:

GINI IMPURITY

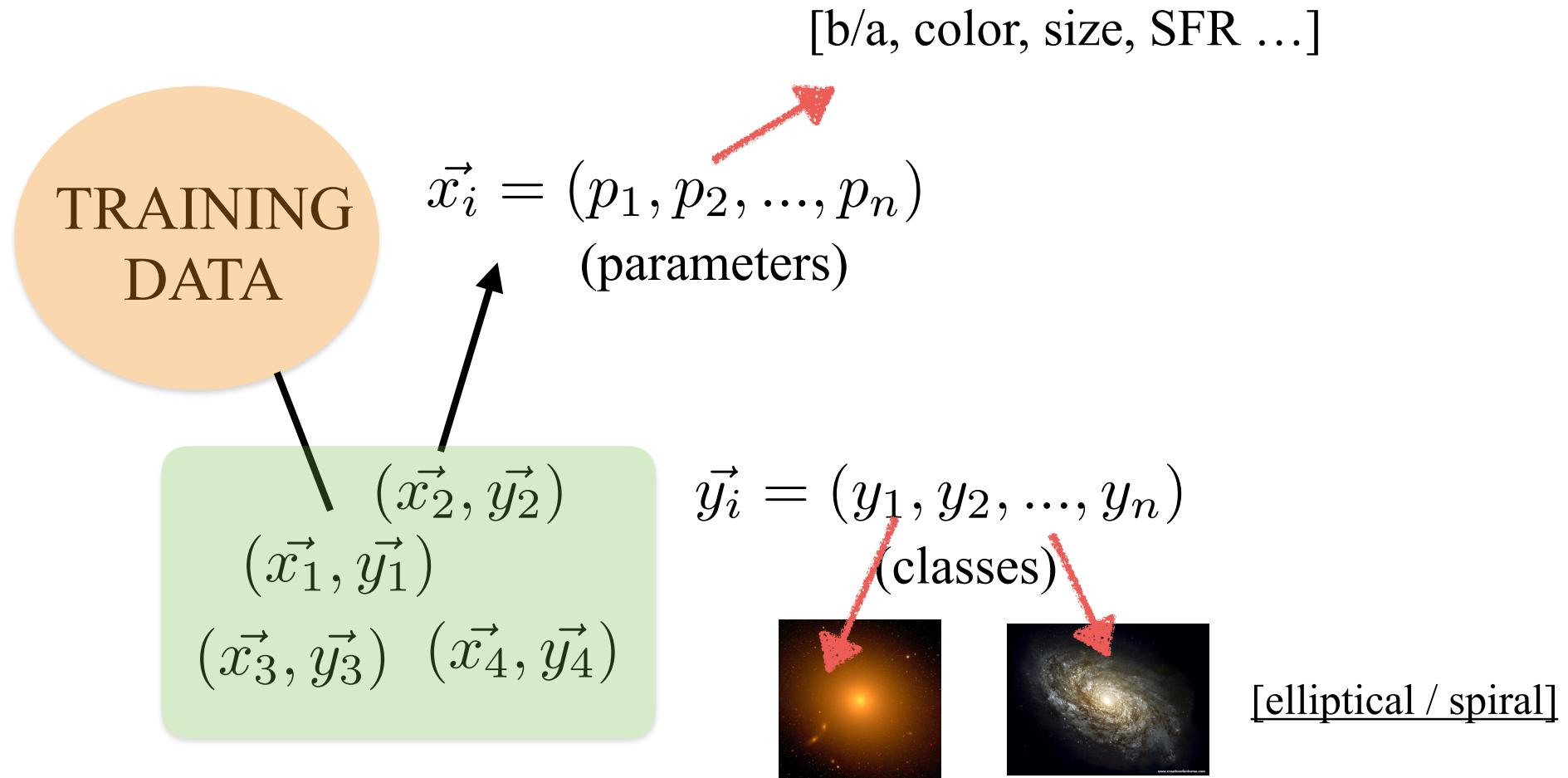
IF THERE IS ONLY ONE CLASS
THE GINI IMPURITY GOES TO 0

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

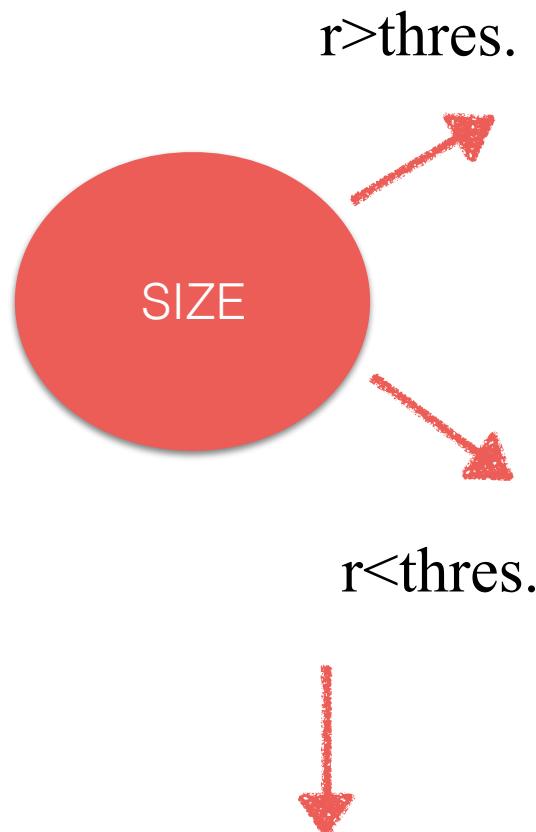


fraction of
objects in each class given
a split value

DECISION TREES (CARTS)



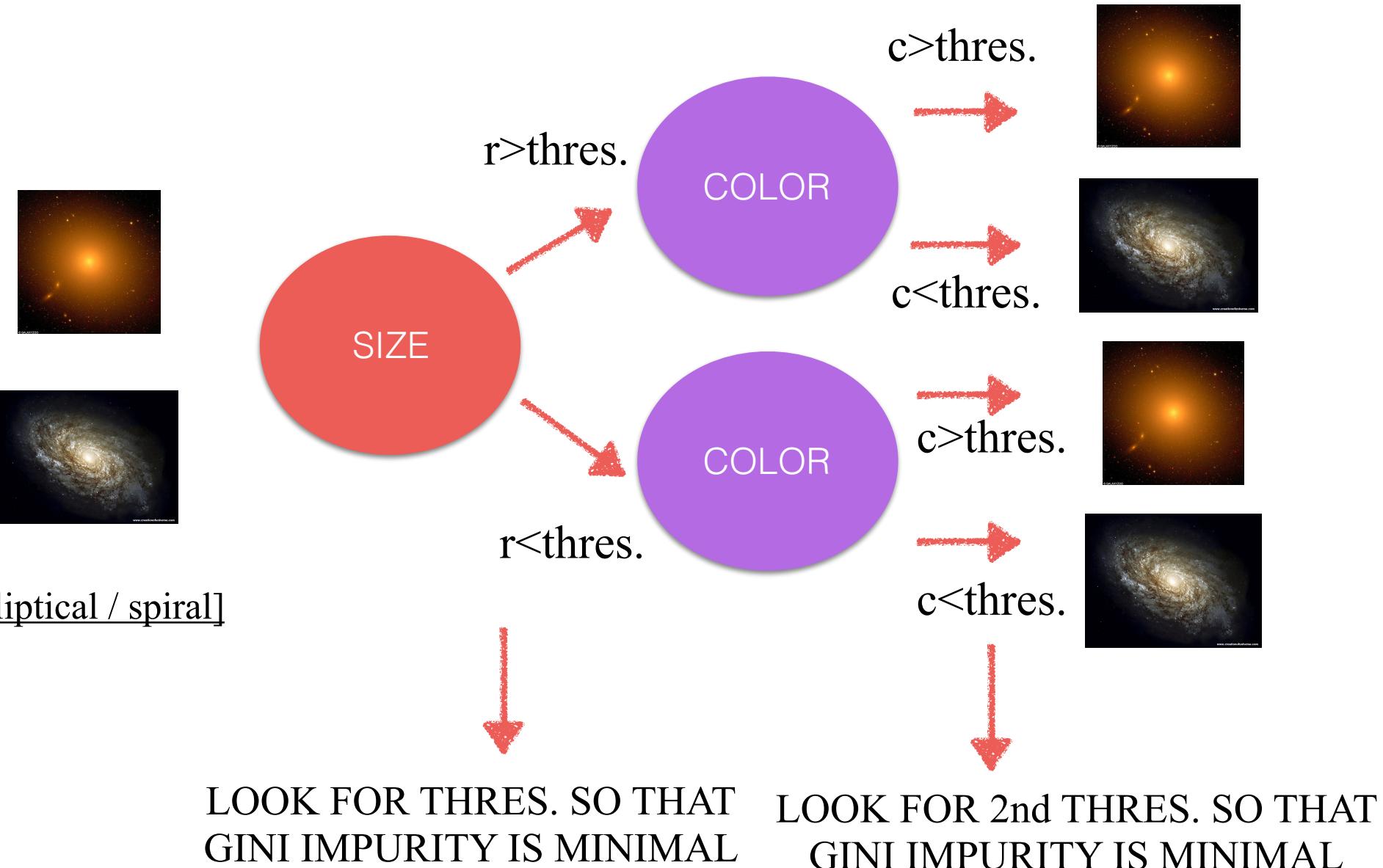
DECISION TREES (CARTS)



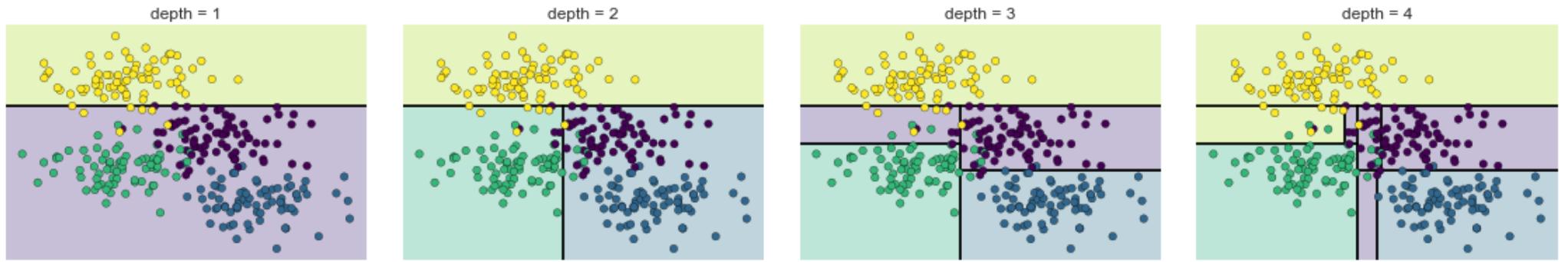
[elliptical / spiral]

LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

DECISION TREES (CARTS)



DECISION TREES (CARTS)



IT IS SIMPLY A PARTITION OF THE PARAMETER SPACE WITH CONSTANT BOUNDARIES - THE NUMBER OF BOUNDARIES DEPENDS ON THE DEPTH OF THE ALGORITHM (HYPER-PARAMETER)

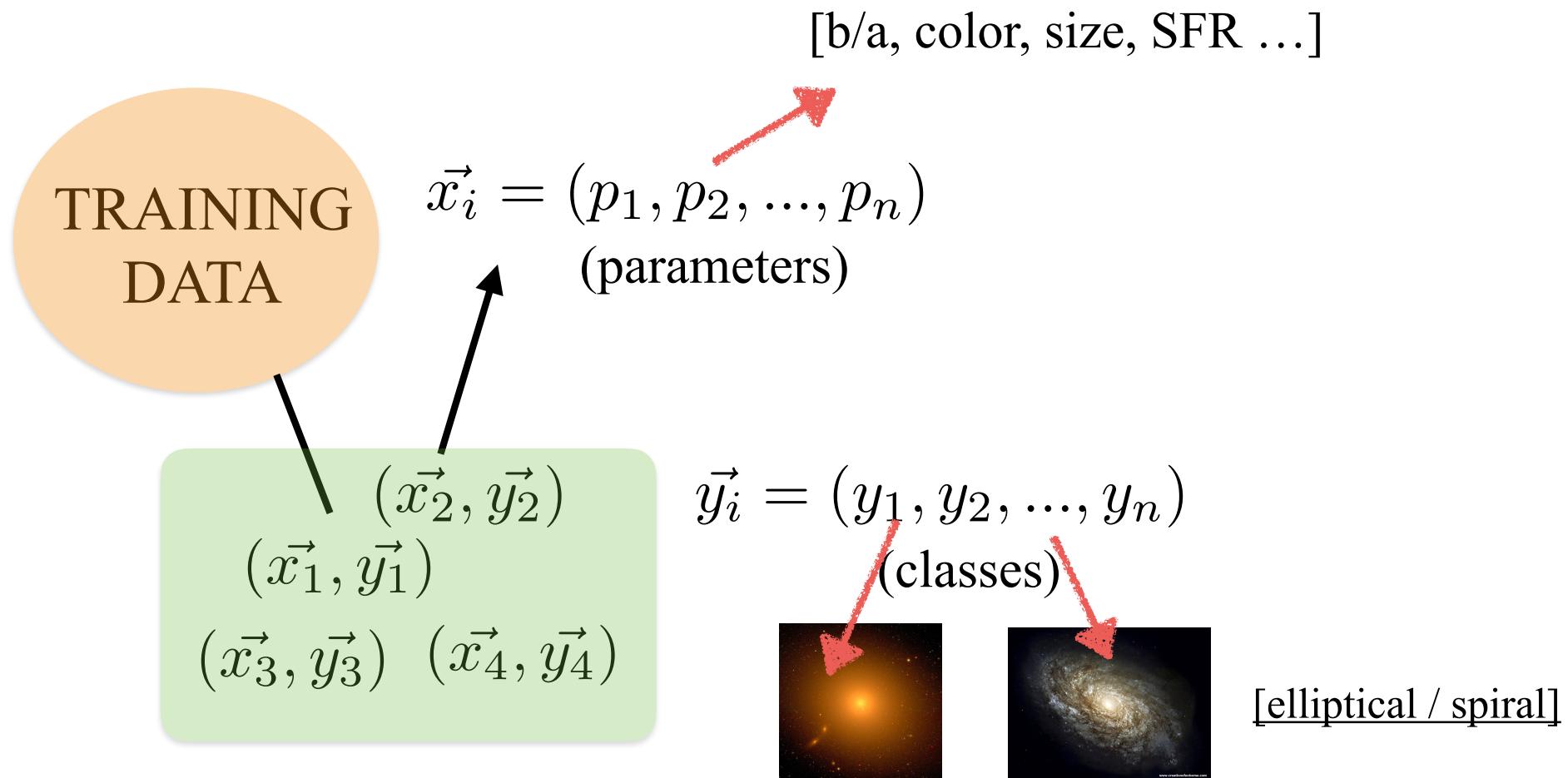
RANDOM FORESTS

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

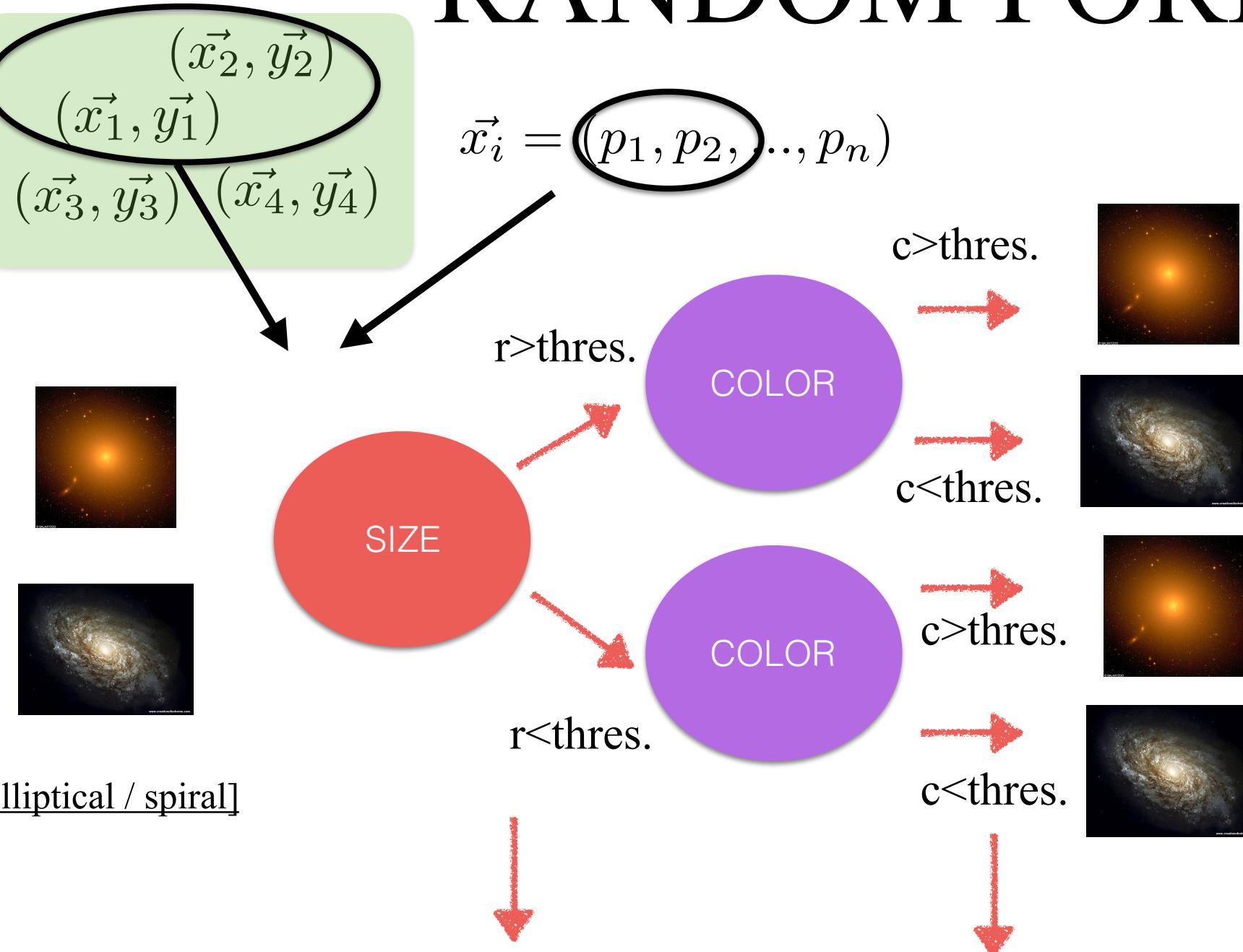
(\vec{x}_2, \vec{y}_2)

(\vec{x}_1, \vec{y}_1)

(\vec{x}_3, \vec{y}_3) (\vec{x}_4, \vec{y}_4)

$\vec{x}_i = (p_1, p_2, \dots, p_n)$

RANDOM FORESTS

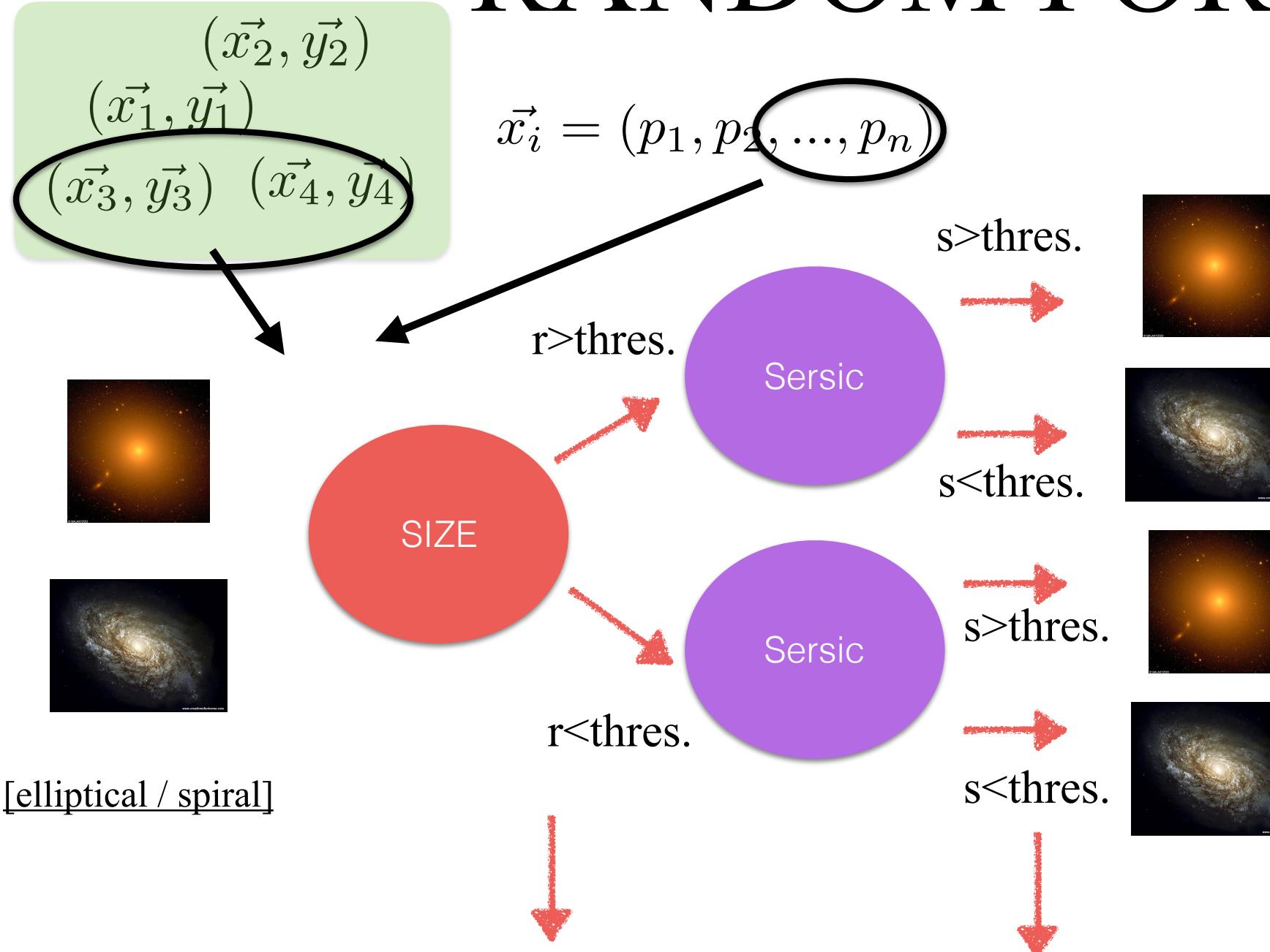


[elliptical / spiral]

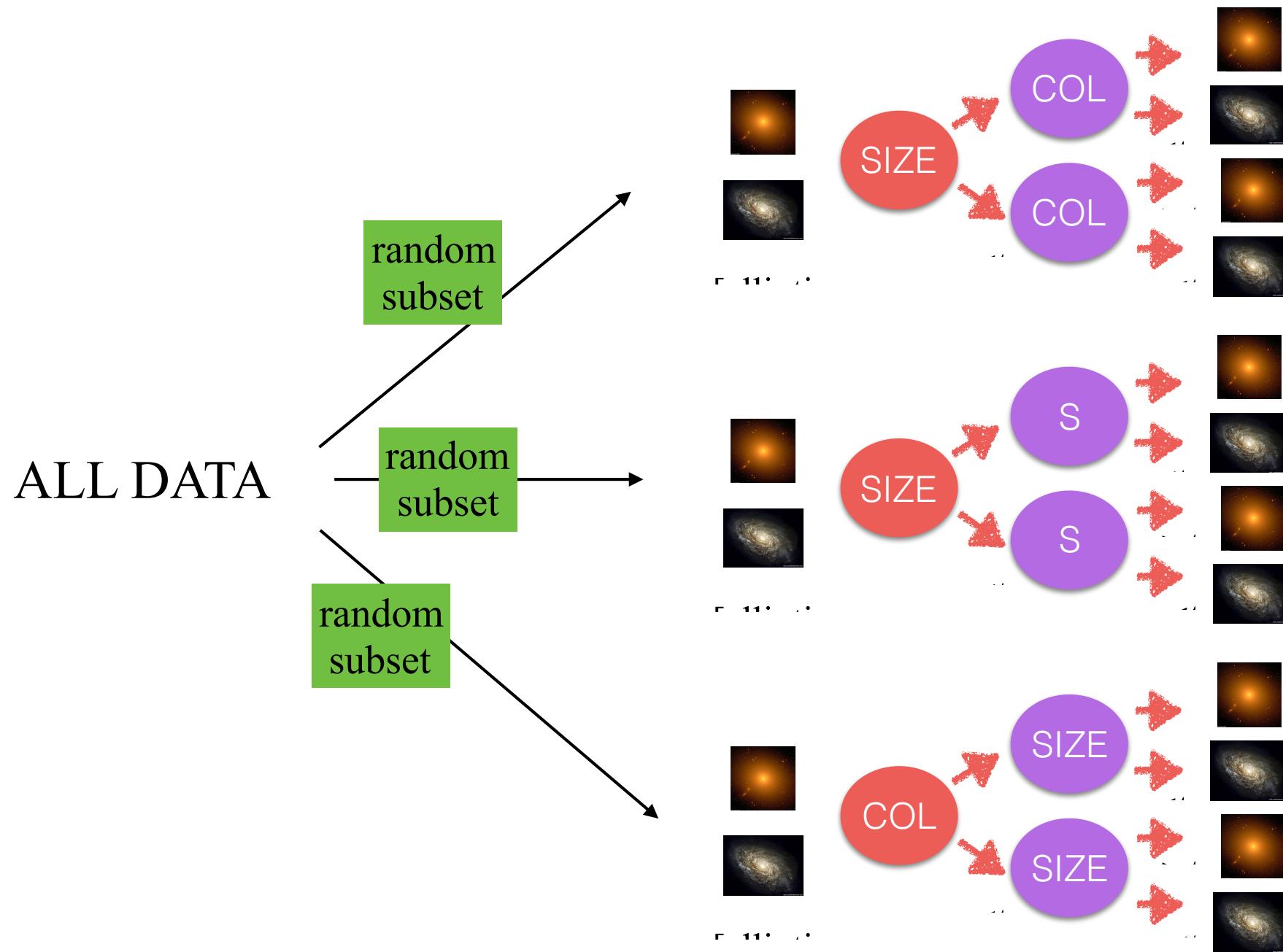
LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

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

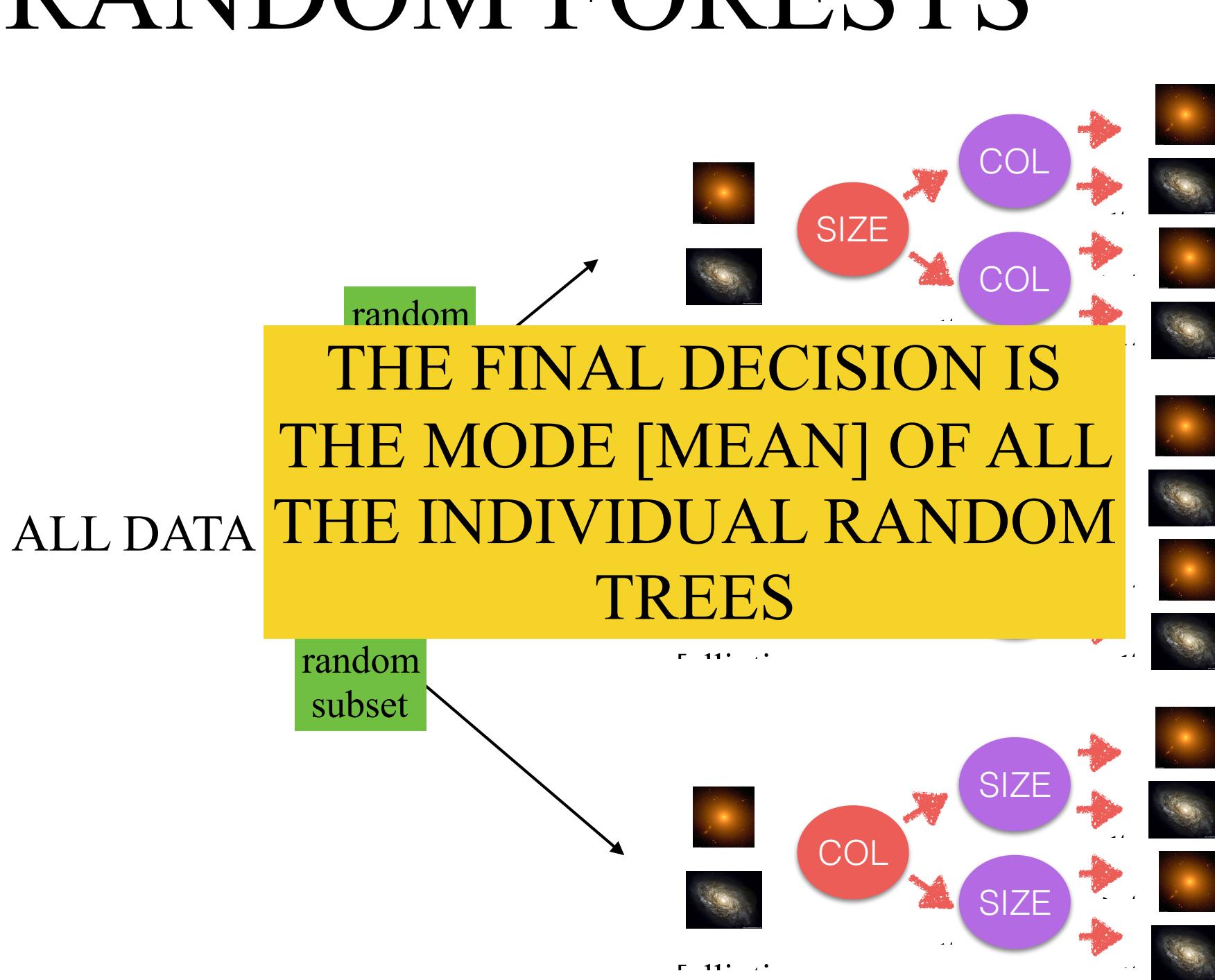
RANDOM FORESTS



RANDOM FORESTS



RANDOM FORESTS



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

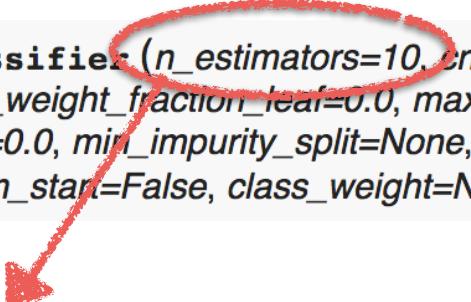
Rank	Property	AUC	Success label ^a
1	ALL	0.9074 ± 0.0106	Outstanding
2	CVD	0.8559 ± 0.0039	Excellent
3	M_{bulge}	0.8335 ± 0.0060	Excellent
4	B/T	0.8267 ± 0.0028	Excellent
5	M_{halo}	0.7983 ± 0.0045	Acceptable
6	M_*	0.7819 ± 0.0025	Acceptable
7	M_{disc}	0.7124 ± 0.0016	Acceptable
8	δ_5	0.5894 ± 0.0015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

IMPORTANCE OF
PARAMETERS TO PREDICT IF
A GALAXY IS QUENCHED

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

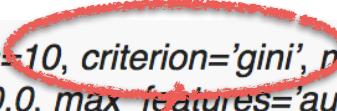
```
class sklearn.ensemble. RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,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

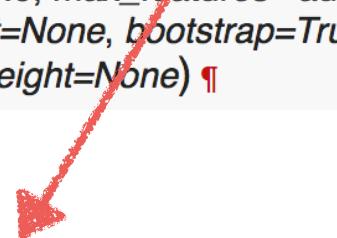
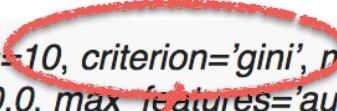
NUMBER OF RANDOM TREES

EVERY ML ALGORITHM HAS A NUMBER OF HYPER PARAMETERS WHICH NEED TO BE TUNED

EXCEPT FOR VERY FEW CASES, THERE ARE NO PRE-DEFINED RULES

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]
```



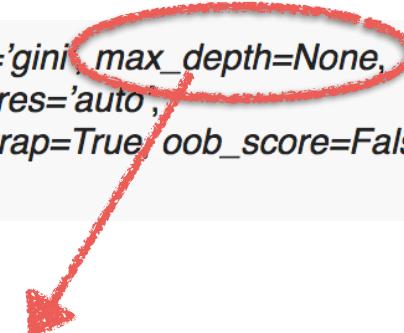
METRIC TO MINIMIZE

EVERY ML ALGORITHM HAS A NUMBER OF HYPER
PARAMETERS WHICH NEED TO BE TUNED

EXCEPT FOR VERY FEW CASES, THERE ARE NO PRE-
DEFINED RULES

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]
```



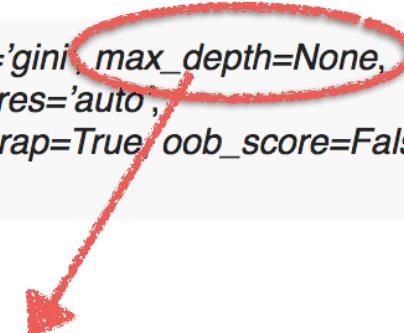
MAXIMUM NUMBER OF SPLITS

EVERY ML ALGORITHM HAS A NUMBER OF HYPER
PARAMETERS WHICH NEED TO BE TUNED

EXCEPT FOR VERY FEW CASES, THERE ARE NO PRE-
DEFINED RULES

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]
```

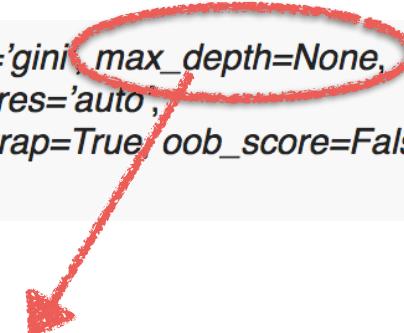


MAXIMUM NUMBER OF SPLITS

```
>>> 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]
```



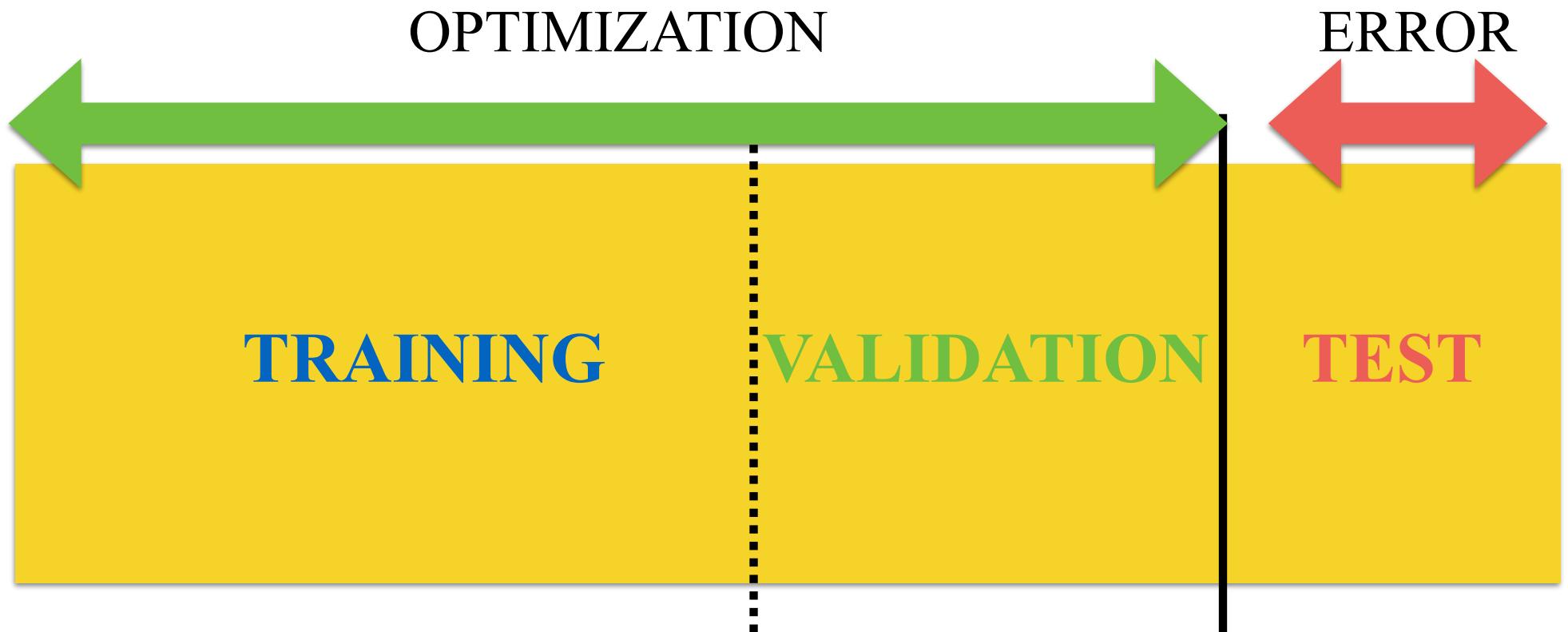
MAXIMUM NUMBER OF SPLITS

```
>>> 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,
```

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

```
>>> print(clf.predict([[0, 0, 0, 0]]))  
[1]
```

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

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

Annotations: "true positives" points to the TP term in the numerator; "true negatives" points to the TN term in the numerator.

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

Evaluation of results [binary class.]

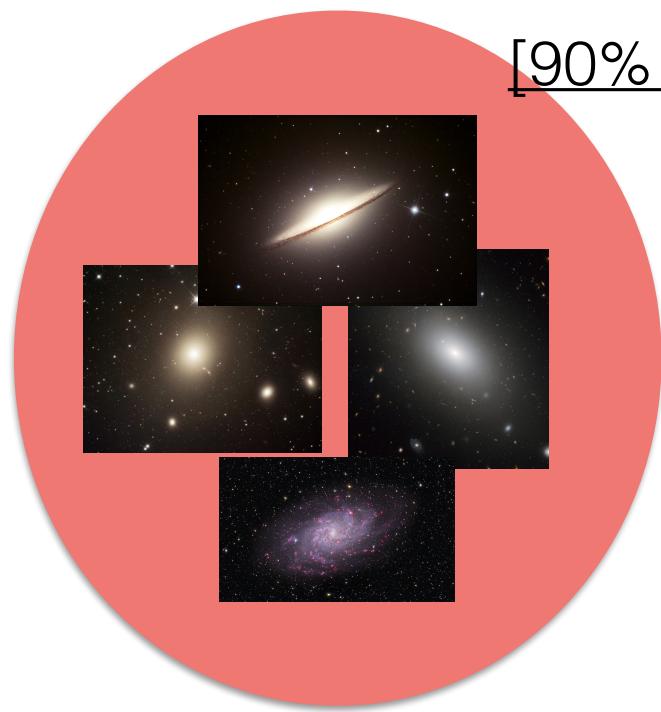
THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE
TOTAL ACCURACY

$$ACC = \frac{\text{true positives} + \text{true negatives}}{TP + TN + FP + FN}$$

NOT VERY
INFORMATIVE IF
UNBALANCED
CLASSES

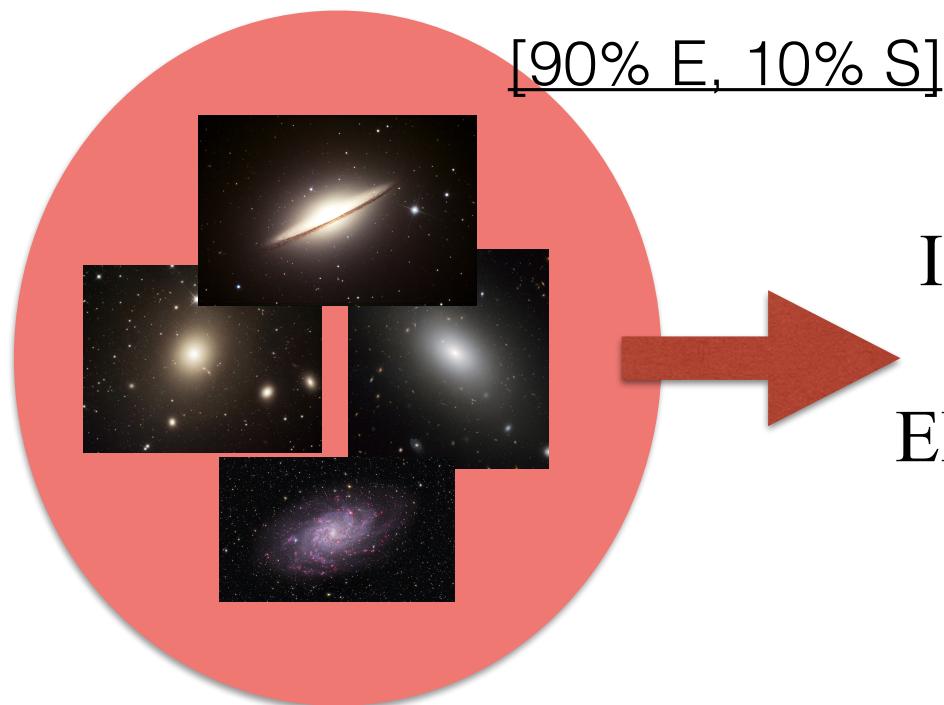
MEASURES HOW MANY OBJECTS ARE CORRECTLY
CLASSIFIED

IMAGINE AN EXTREME CASE WITH VERY UNBALANCED
DATA...



[90% E, 10% S]

IMAGINE AN EXTREME CASE WITH VERY UNBALANCED
DATA...



IF I SAY THAT ALL GALAXIES
IN THE UNIVERSE ARE
ELLIPTICALS I WILL BE RIGHT
90% OF THE TIMES

Evaluation of results [binary class.]

THE ROC CURVE (Receiver Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1.
$$TPR = \frac{TP}{TP + FN}$$
 [Also called Sensitivity, Completeness]

“Fraction of positive examples classified correctly”

Evaluation of results [binary class.]

THE ROC CURVE (Receiver Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1.

$$TPR = \frac{TP}{TP + FN}$$

TRUE POSITIVE RATE
[Also called Sensitivity, Completeness]

“Fraction of positive examples classified correctly”

2.

$$FPR = \frac{FP}{FP + TN}$$

FALSE POSITIVE RATE
[Also called Specificity, Contamination]

“Fraction of negative examples classified as positive”

Evaluation of results [binary class.]

- YOU WANT THIS TO BE AS BIG AS POSSIBLE
- Over Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1.

$$TPR = \frac{TP}{TP + FN}$$

TRUE POSITIVE RATE
[Completeness]

YOU WANT THIS TO BE AS SMALL AS POSSIBLE

“Fraction of positive examples classified correctly”

2.

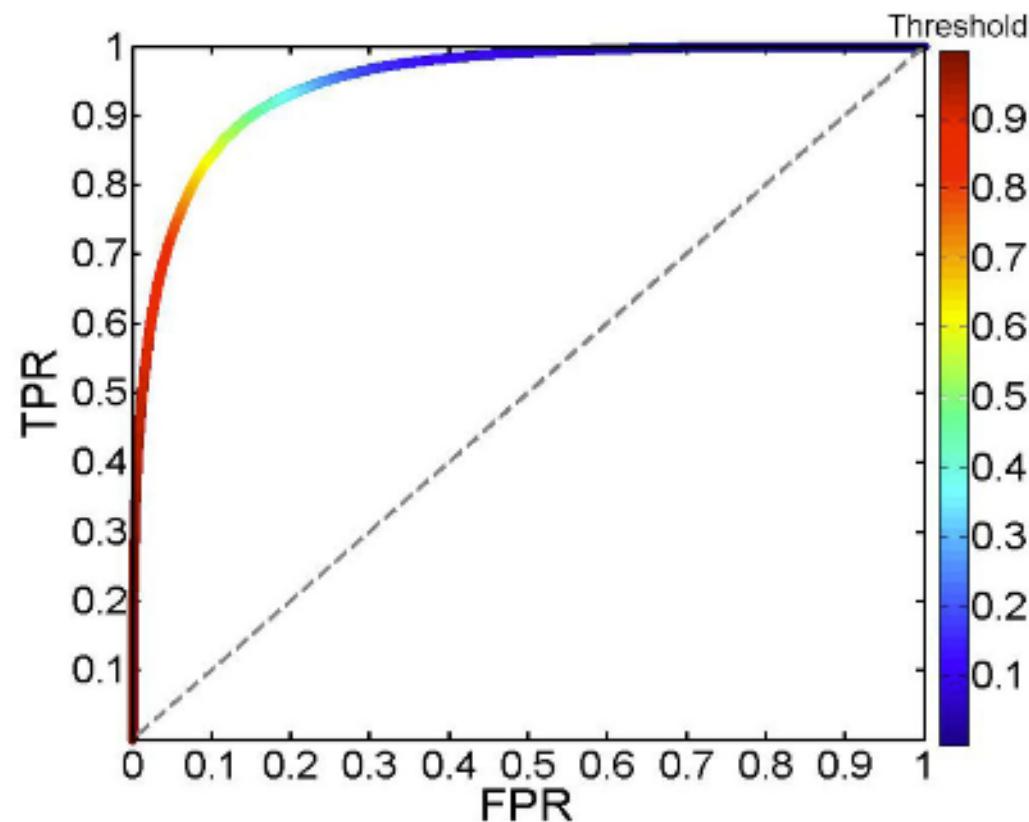
$$FPR = \frac{FP}{FP + TN}$$

FALSE POSITIVE RATE
[Also called Specificity, Contamination]

“Fraction of negative examples classified as positive”

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

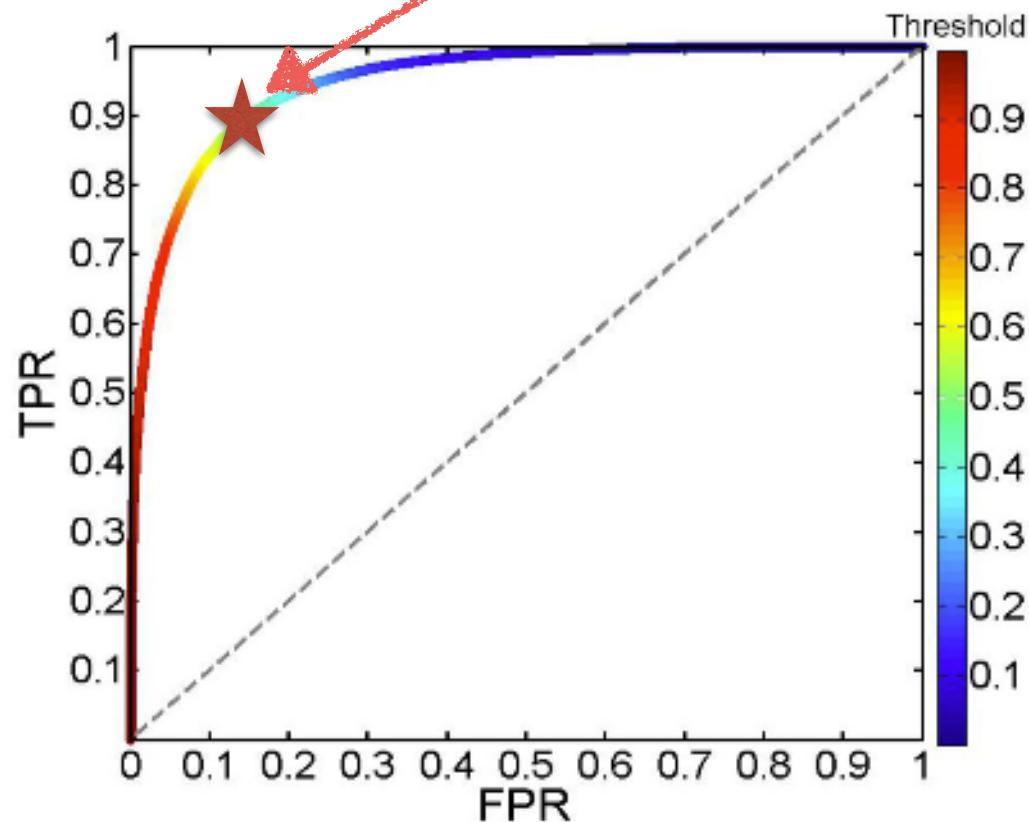
ROC CURVE



IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

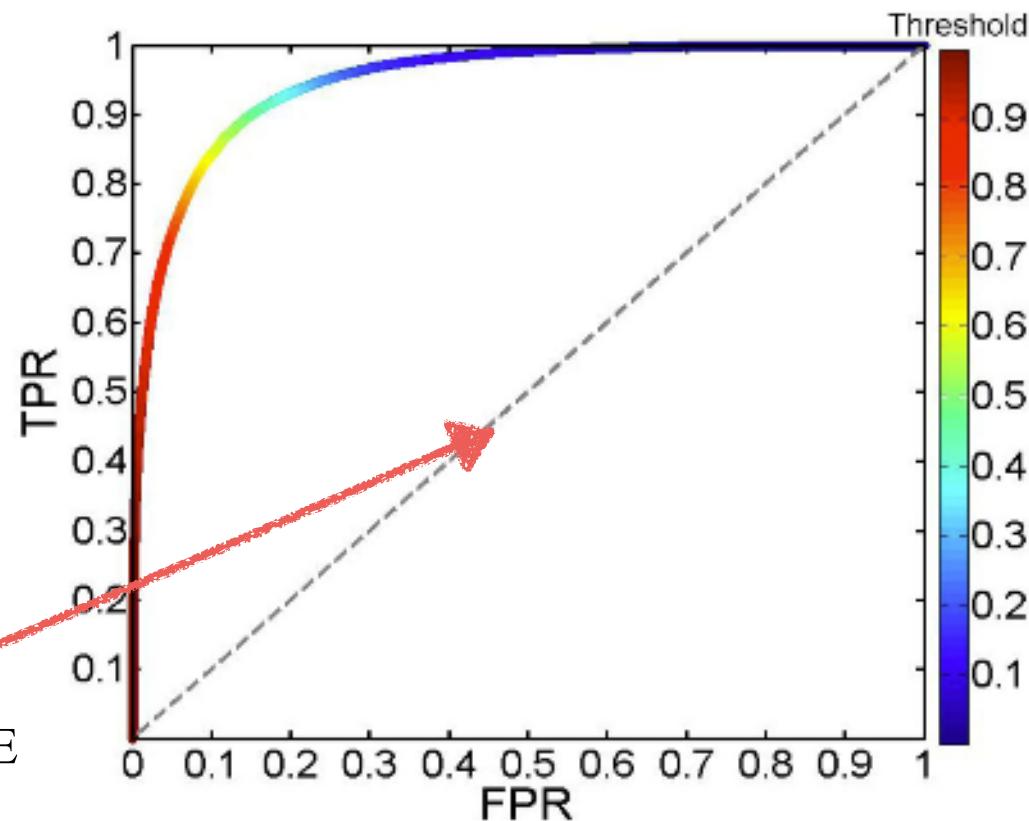
EACH POINT HERE SHOWS THE VALUES OF TPR AND
FPR FOR A GIVEN THRESHOLD

ROC CURVE



IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

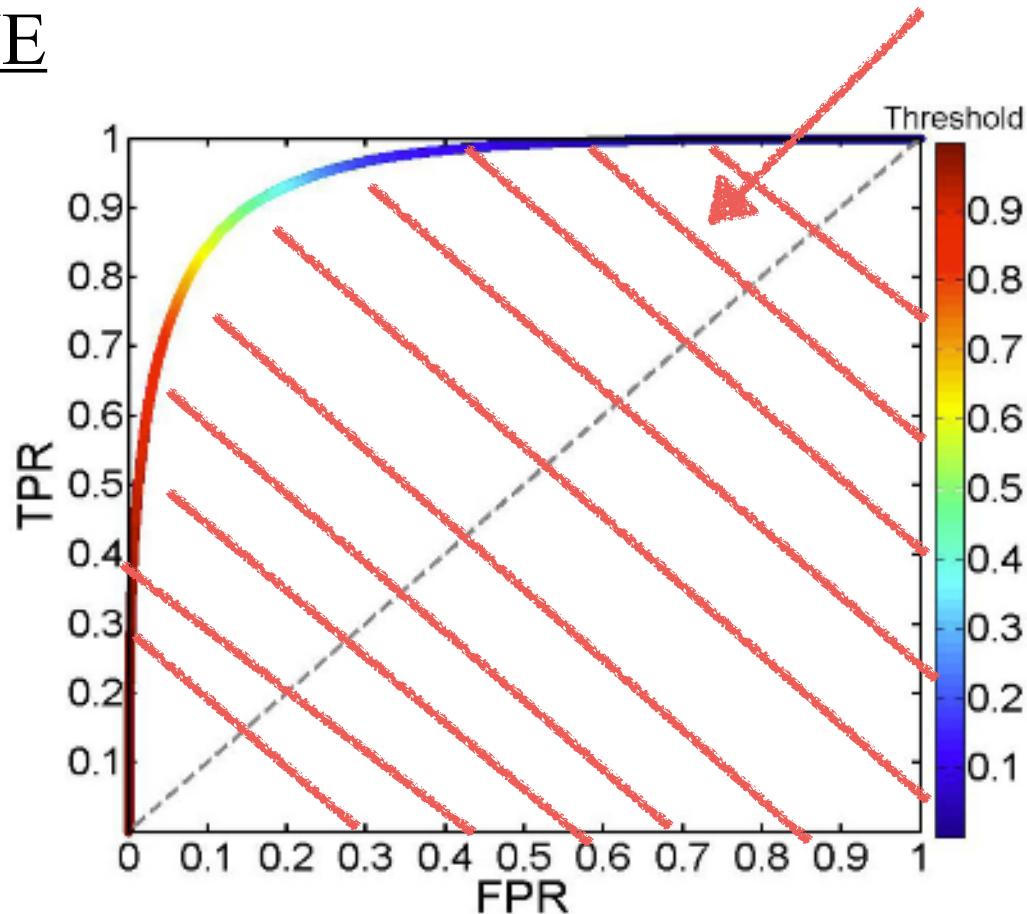
ROC CURVE



THE ONE-TO-ONE
LINE IS A
RANDOM
CLASSIFICATION

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

ROC CURVE



THE AREA UNDER THE
CURVE AUC ALSO
MEASURES THE
GLOBAL ACCURACY

Evaluation of results

THE P-R CURVE (Precision - Recall)

$$Recall = \frac{TP}{TP + FN} = TPR \quad [\text{completeness}]$$

$$Precision = \frac{TP}{TP + FP} \quad [\text{purity}]$$

Evaluation of results

THE P-R CURVE (Precision - Recall)

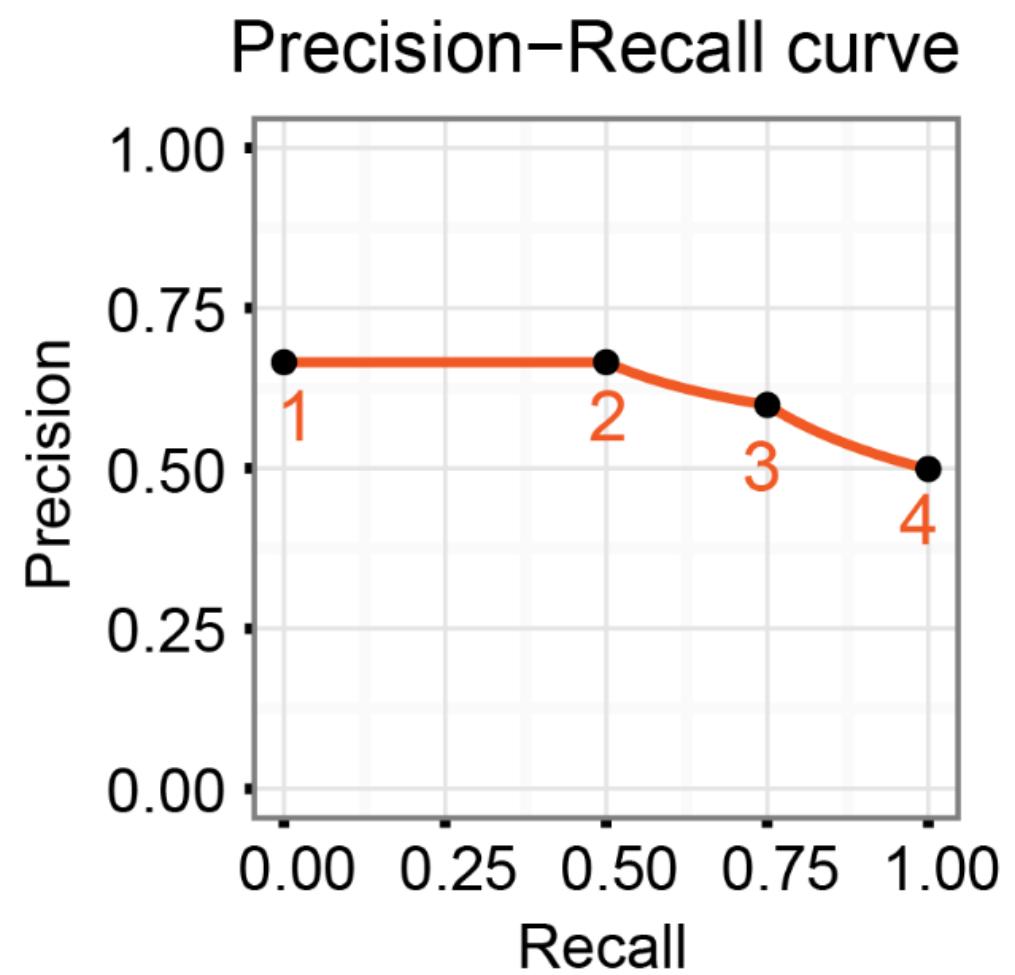
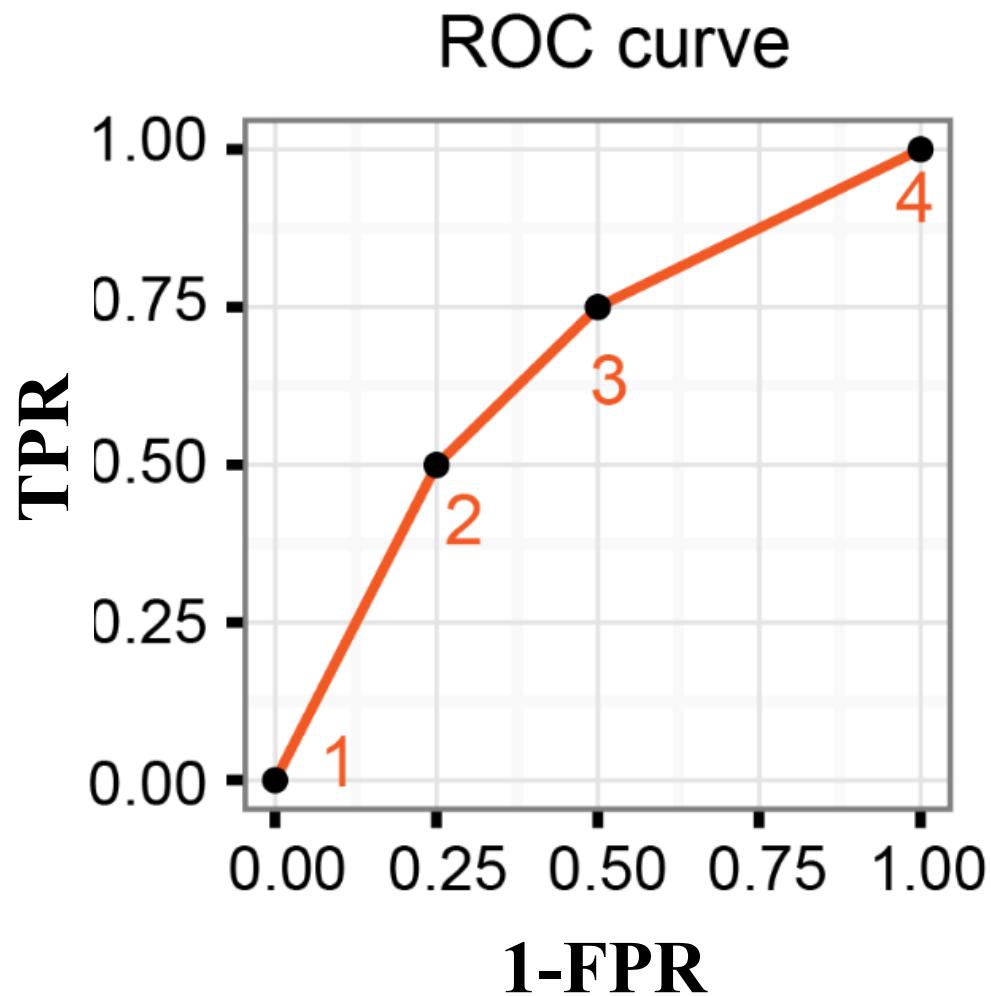
$$Recall = \frac{TP}{TP + FN} = TPR \quad [\text{completeness}]$$

$$Precision = \frac{TP}{TP + FP} \quad [\text{purity}]$$



FOR BALANCED DATA: $Precision \sim 1 - FPR$

Evaluation of results



SUMMARY OF DIFFERENT ACCURACY TRACERS

		True condition			
Total population	Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$	
Predicted condition	Predicted condition positive	True positive, Power	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR-}}$	$F_1 \text{ score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
	<u>False negative rate (FNR)</u> , Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	<u>True negative rate (TNR)</u> , Specificity (SPC) $= \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$		

SOURCE

SUMMARY OF DIFFERENT ACCURACY TRACERS

		True condition			
Total population	Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$	
Predicted condition	Predicted condition positive	True positive, Power	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR-}}$	$F_1 \text{ score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
	False negative rate (FNR), Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$		

THE F1 SCORE
COMBINES BOTH INFORMATIONS IN ONE VALUE

[SOURCE](#)

ALL THESE ARE INCLUDED IN SKLEARN

AND ARE VERY EASY TO USE. NO NEED OF CODING THEM AGAIN!

```
sklearn.metrics. precision_recall_curve (y_true, probas_pred, pos_label=None, sample_weight=None) ¶
```

[\[source\]](#)

Compute precision-recall pairs for different probability thresholds

Note: this implementation is restricted to the binary classification task.

The precision is the ratio $\text{tp} / (\text{tp} + \text{fp})$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

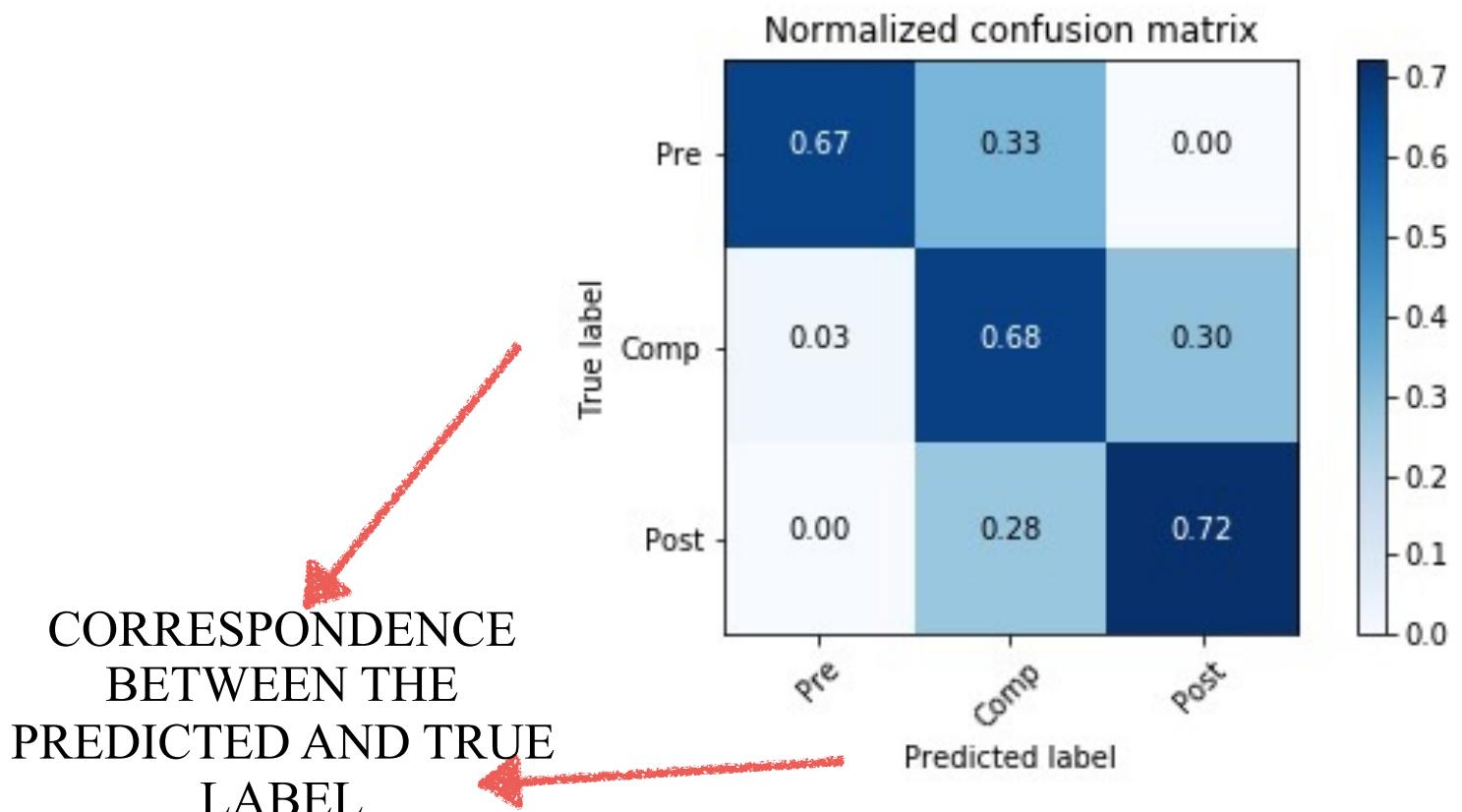
The recall is the ratio $\text{tp} / (\text{tp} + \text{fn})$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The last precision and recall values are 1. and 0. respectively and do not have a corresponding threshold. This ensures that the graph starts on the x axis.

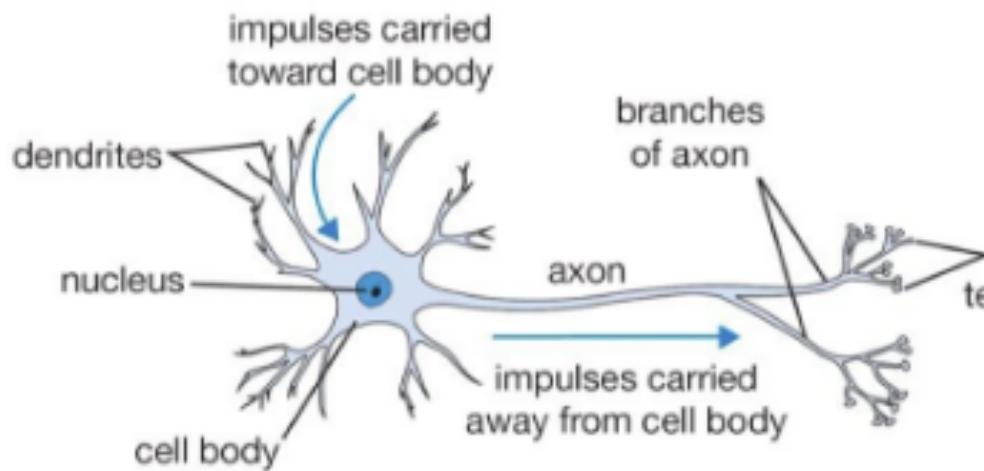
Evaluation of results

[multi-class]

CONFUSION MATRIX



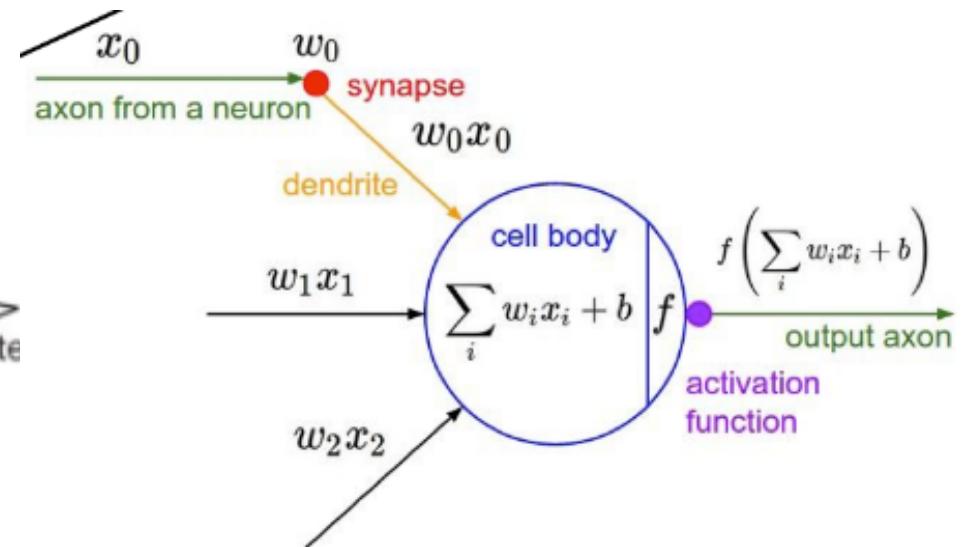
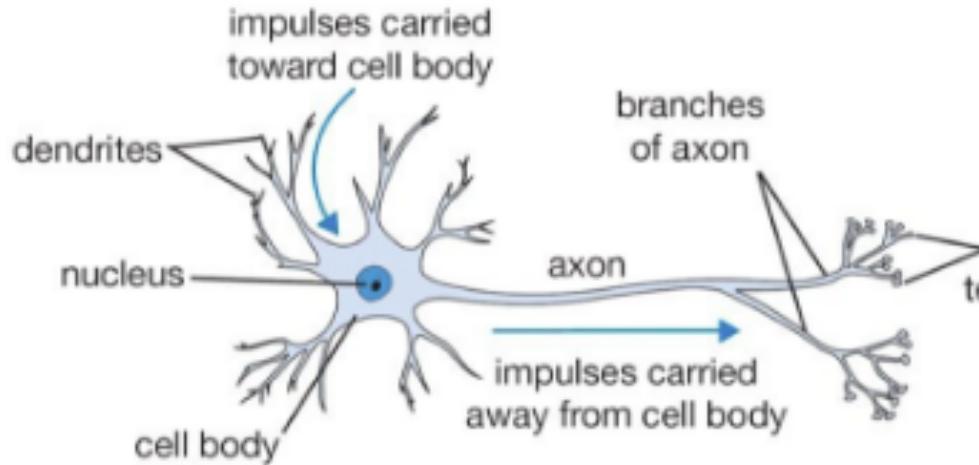
THE NEURON



INSPIRED BY NEURO - SCIENCE?

Credit: Karpathy

THE NEURON



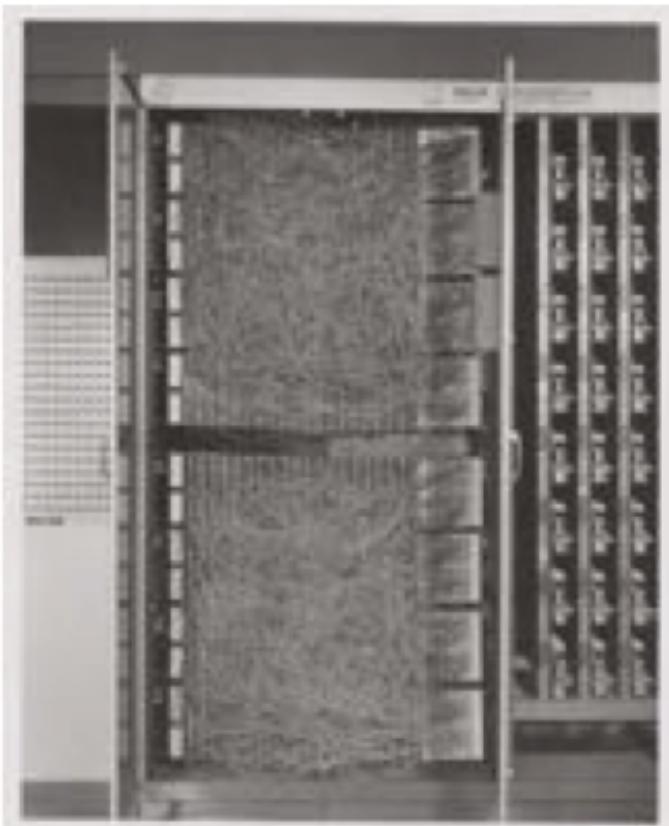
INSPIRED BY NEURO - SCIENCE?

Credit: Karpathy

Rosenblatt Perceptron

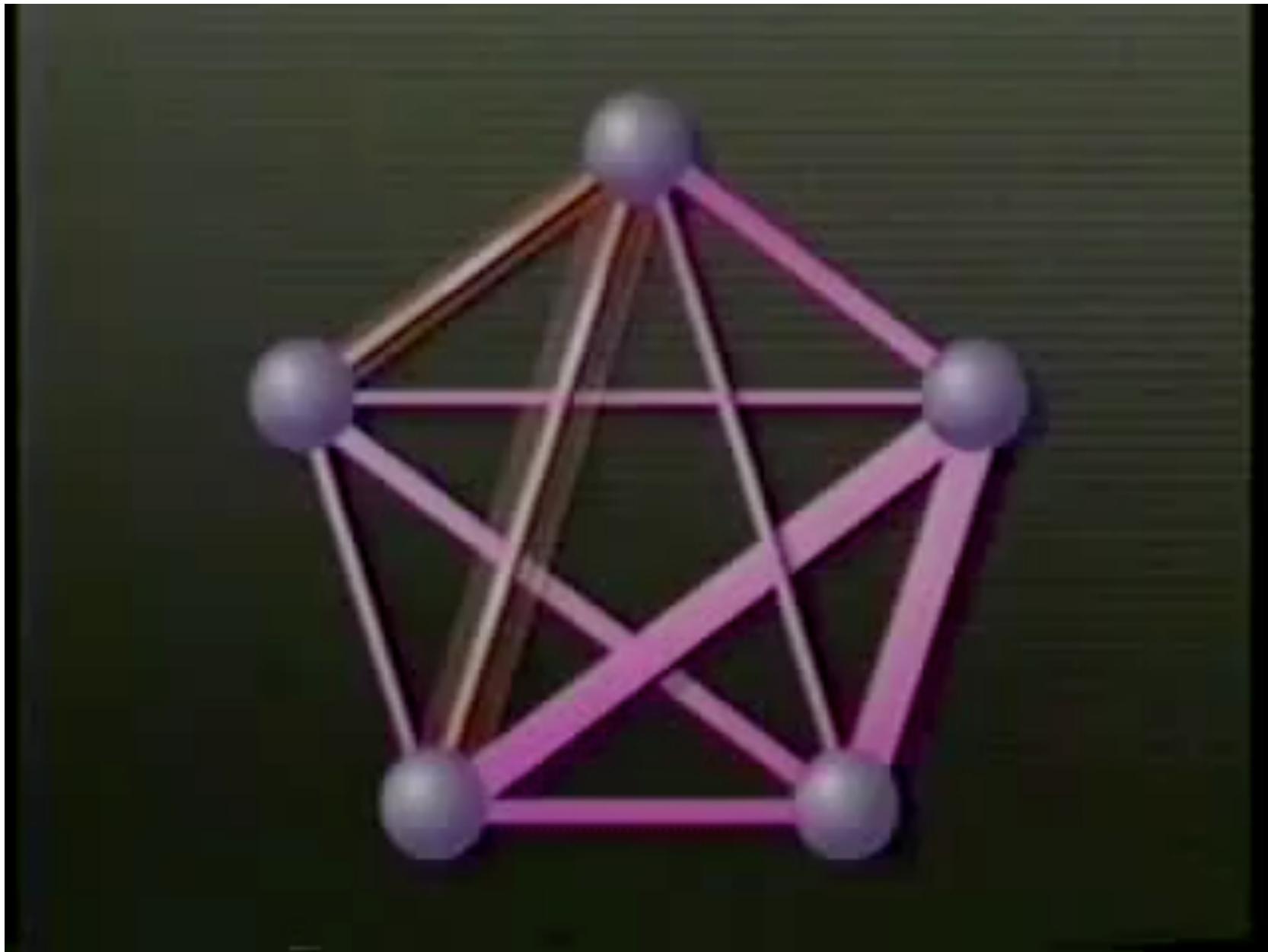
FIRST IMPLEMENTATION OF NEURAL NETWORK [Rosenblatt, 1957!]

INTENDED TO BE A MACHINE (NOT AN ALGORITHM)

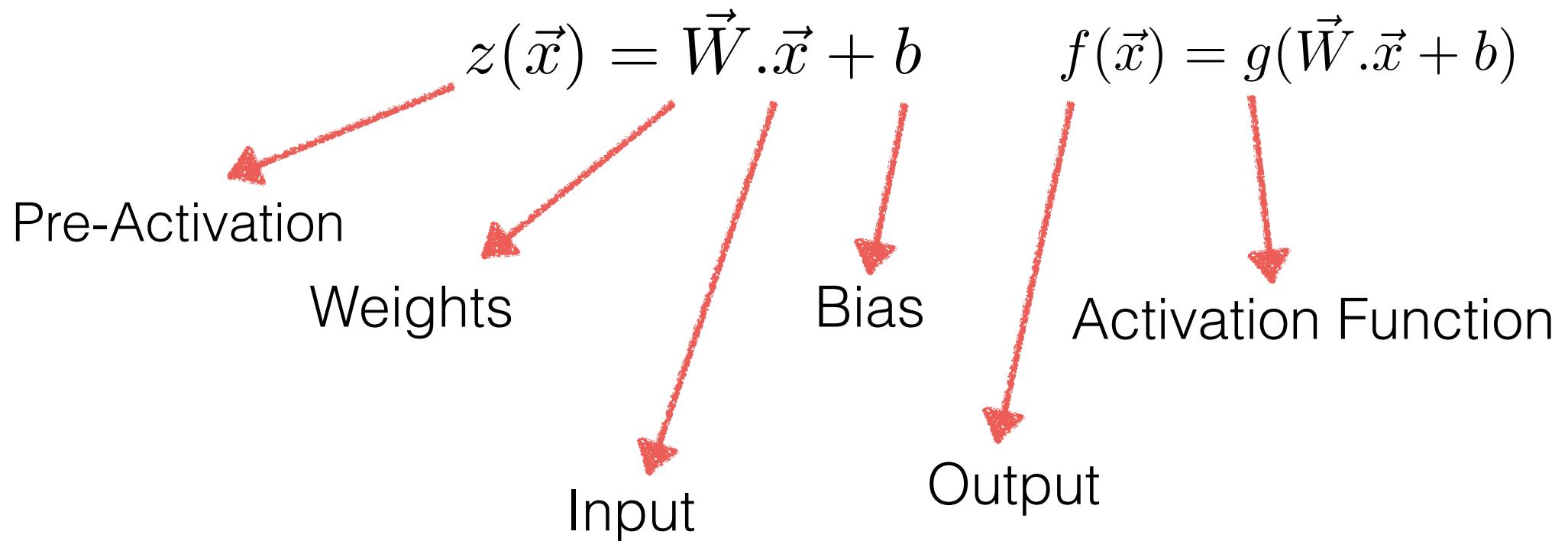
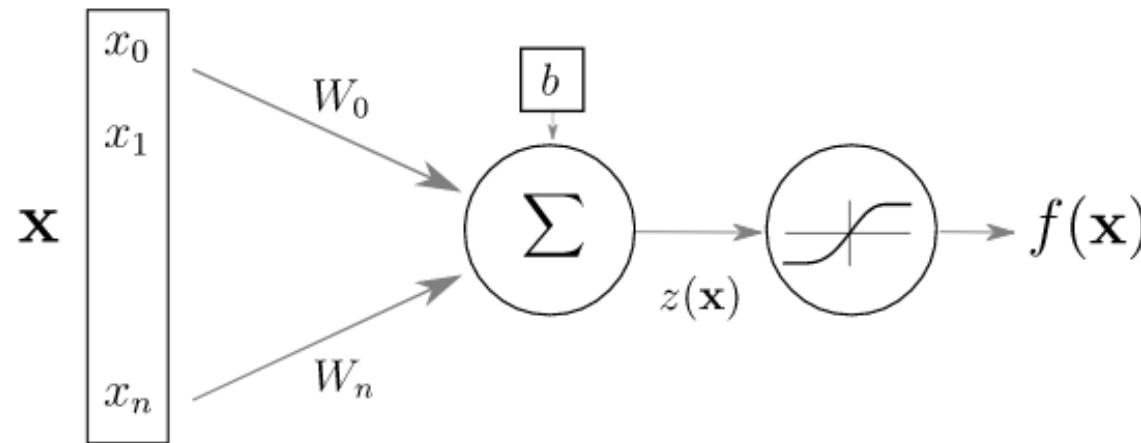


it had an array of 400 photocells,
randomly connected to the "neurons".

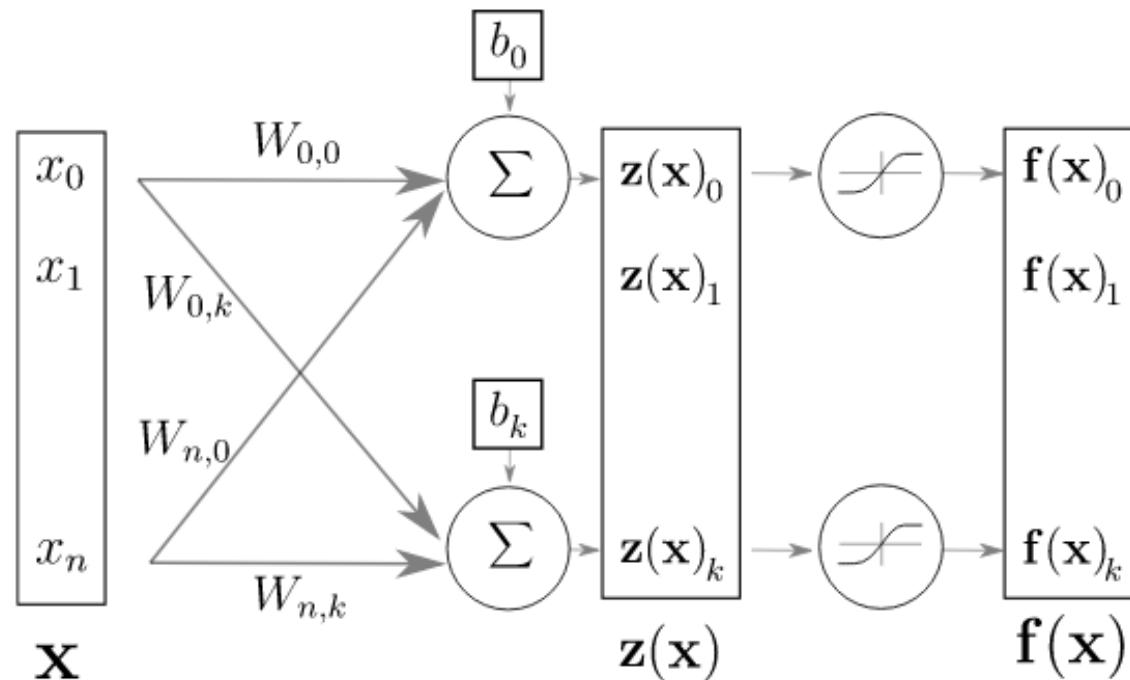
Weights were encoded in
potentiometers, and weight updates
during learning were performed by
electric motors



TODAY'S ARTIFICIAL NEURON



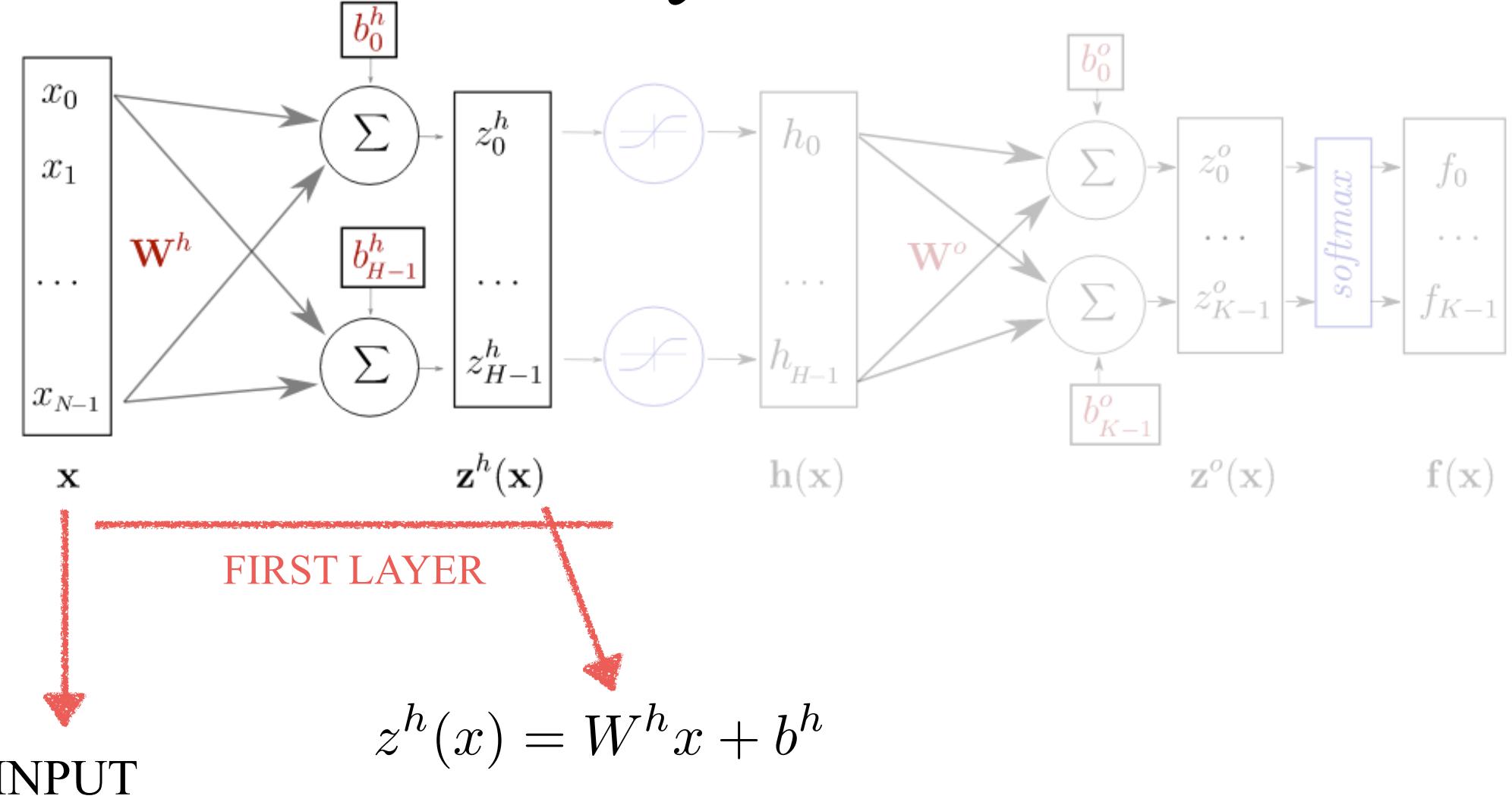
LAYER OF NEURONS



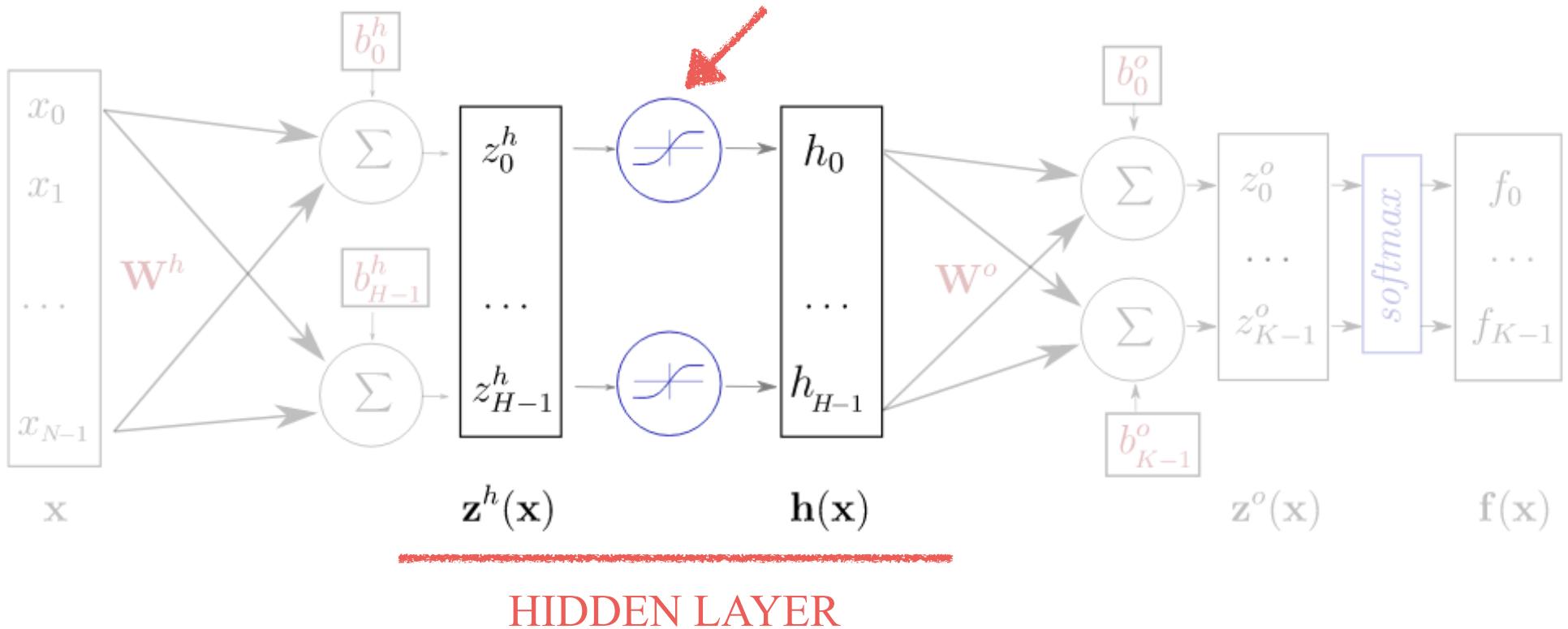
$$f(\vec{x}) = g(\mathbf{W} \cdot \vec{x} + \vec{b})$$

SAME IDEA. NOW **W** becomes a matrix and **b** a vector

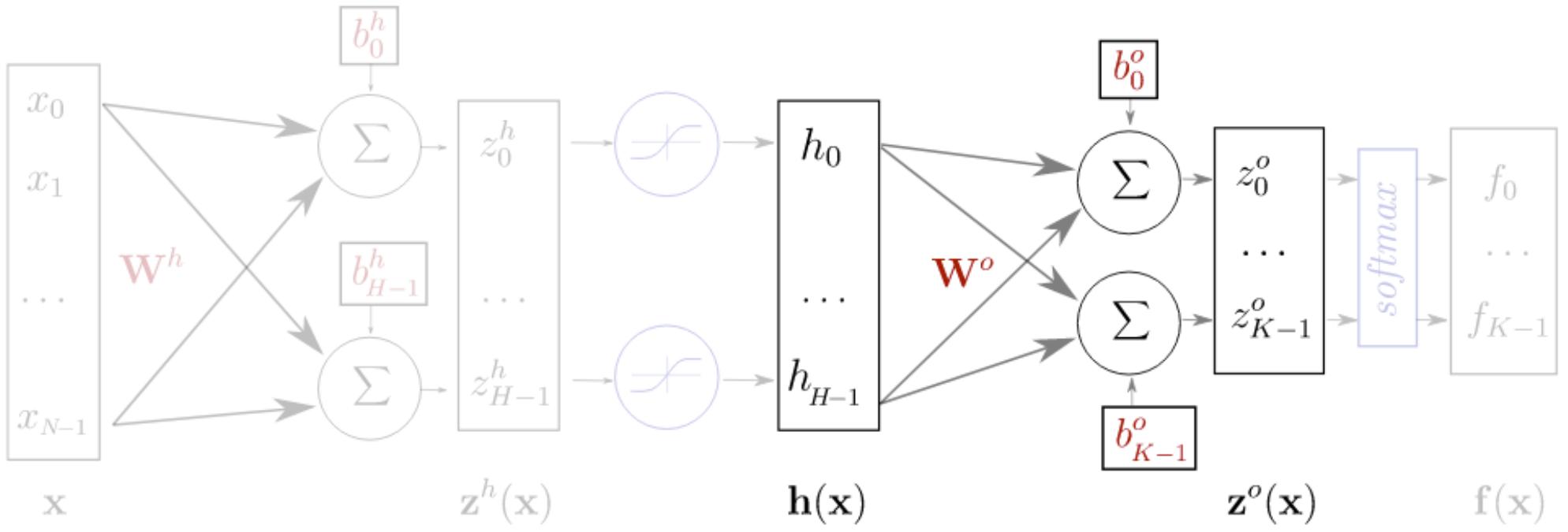
Hidden Layers of Neurons



ACTIVATION FUNCTION

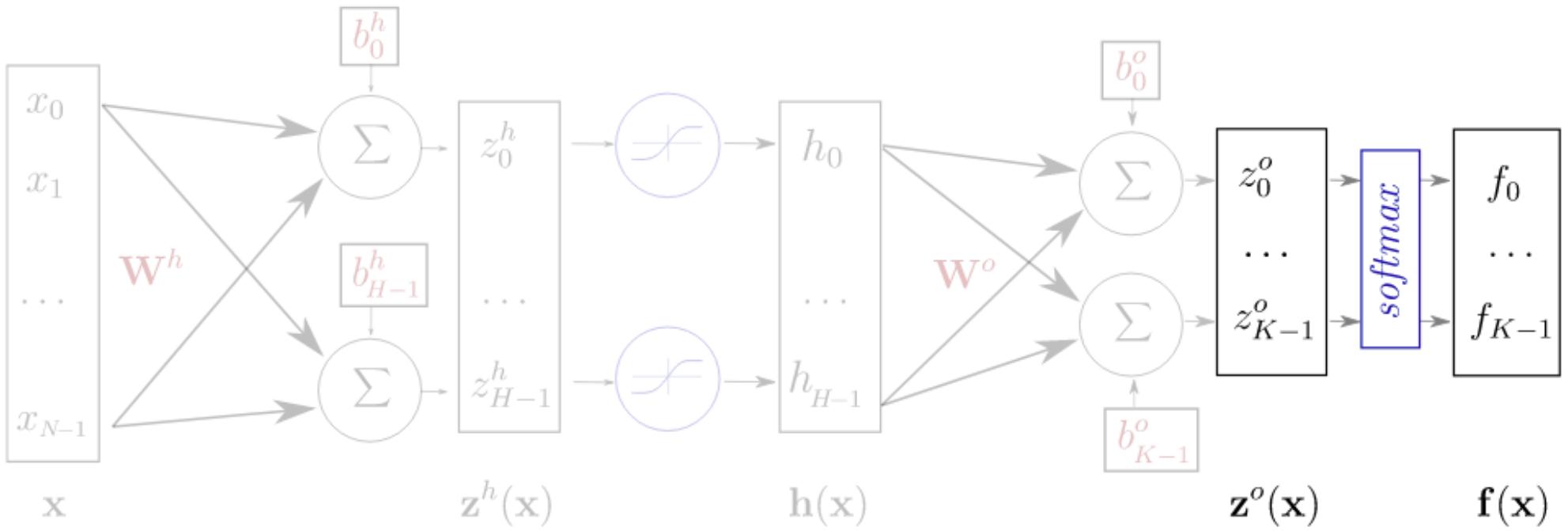


$$h(x) = g(z^h(x)) = g(W^h x + b^h)$$



OUTPUT LAYER

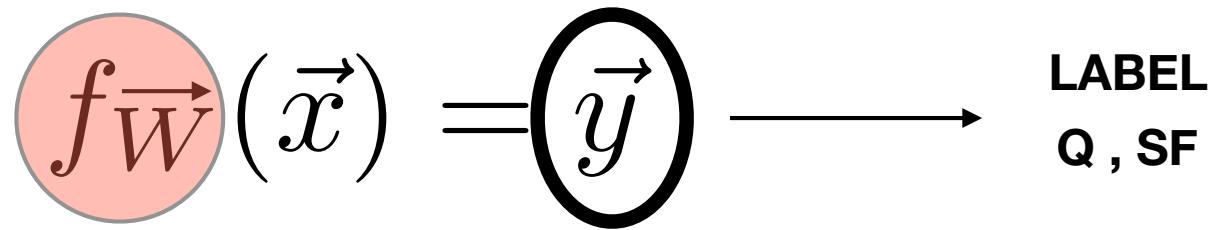
$$z^0(\mathbf{x}) = W^0 h(\mathbf{x}) + b^0$$



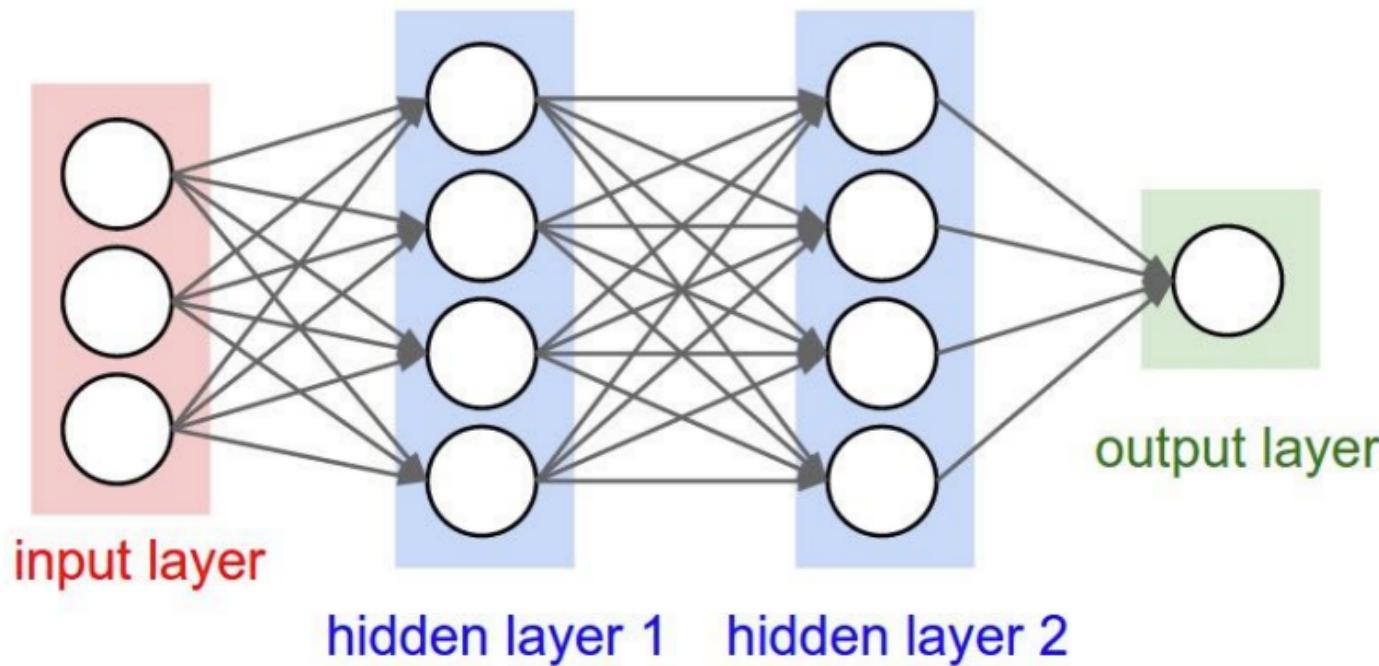
PREDICTION LAYER

$$f(\mathbf{x}) = \text{softmax}(\mathbf{z}^o)$$

**“CLASSICAL”
MACHINE LEARNING**



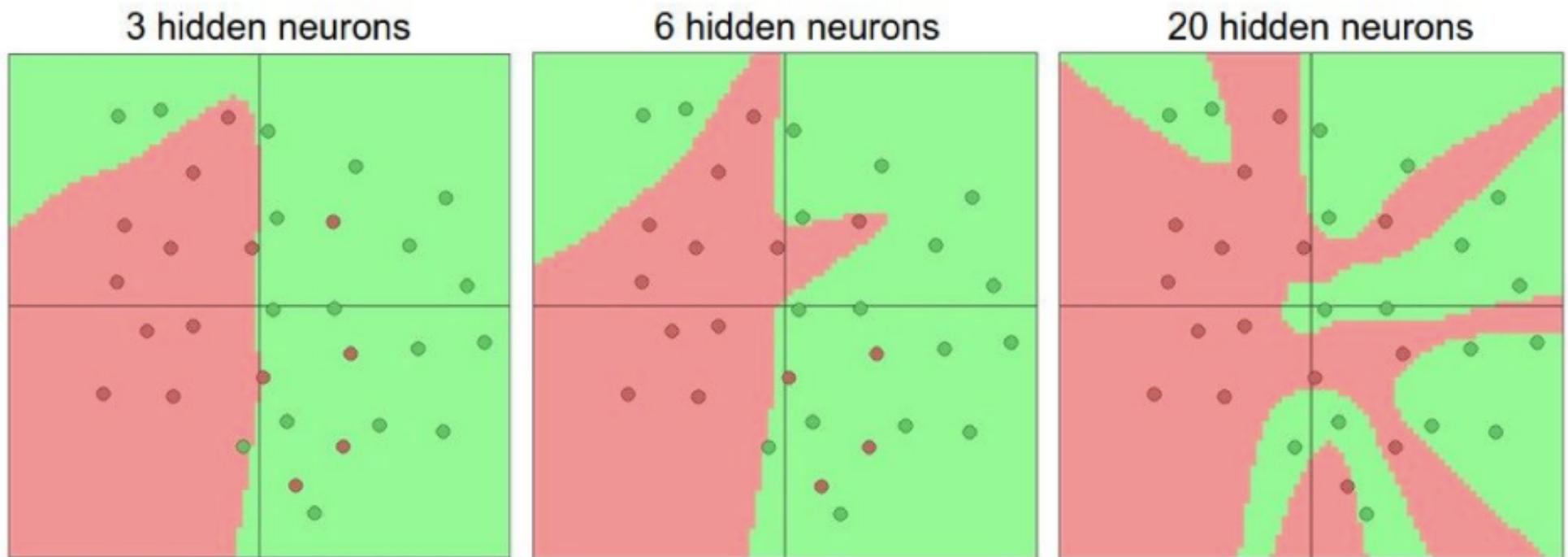
**REPLACE THIS BY A GENERAL
NON LINEAR FUNCTION WITH SOME PARAMETERS W**



$$p = g_3(W_3g_2(W_2g_1(W_1\vec{x}_0)))$$

NETWORK
FUNCTION

WHY HIDDEN LAYERS?



More complex functions allow increasing complexity

Credit: Karpathy

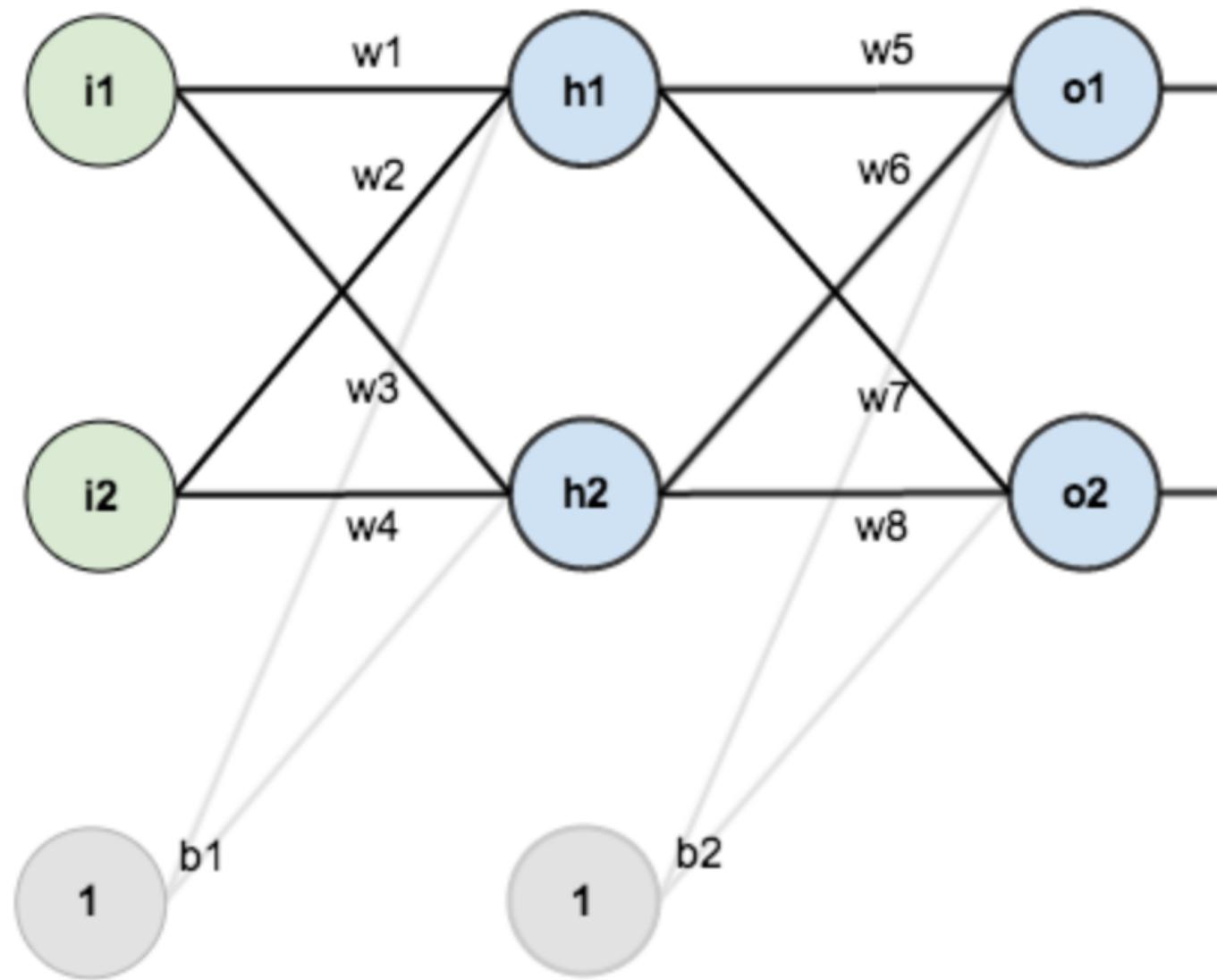
SO LET'S GO DEEPER AND DEEPER!

SO LET'S GO DEEPER AND DEEPER!

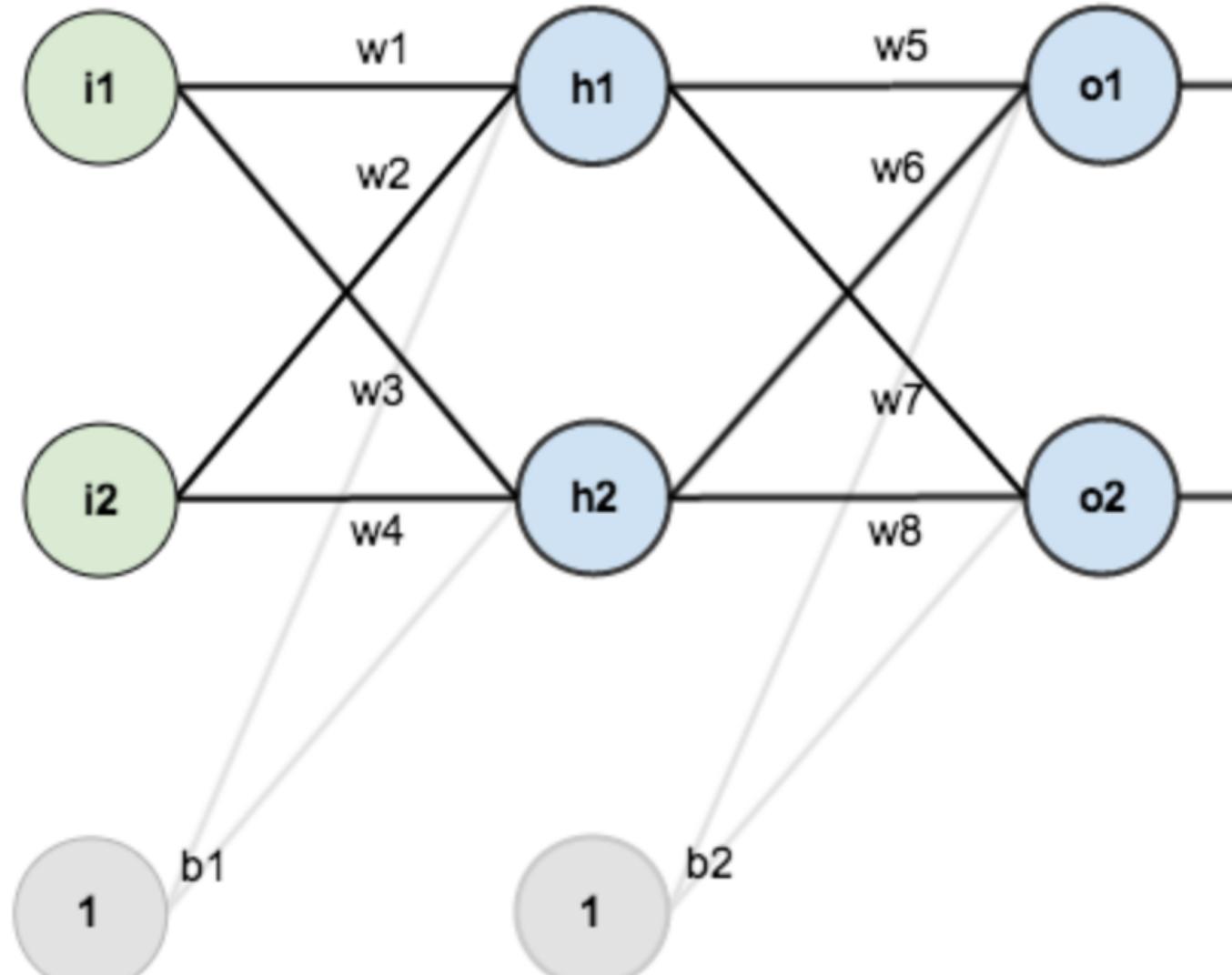
YES BUT...

NOT SO STRAIGHTFORWARD, DEEPER MEANS MORE
WEIGHTS, MORE DIFFICULT OPTIMIZATION, RISK OF
OVERFITTING...

LET'S FOLLOW A NETWORK
WHILE IT LEARNS...

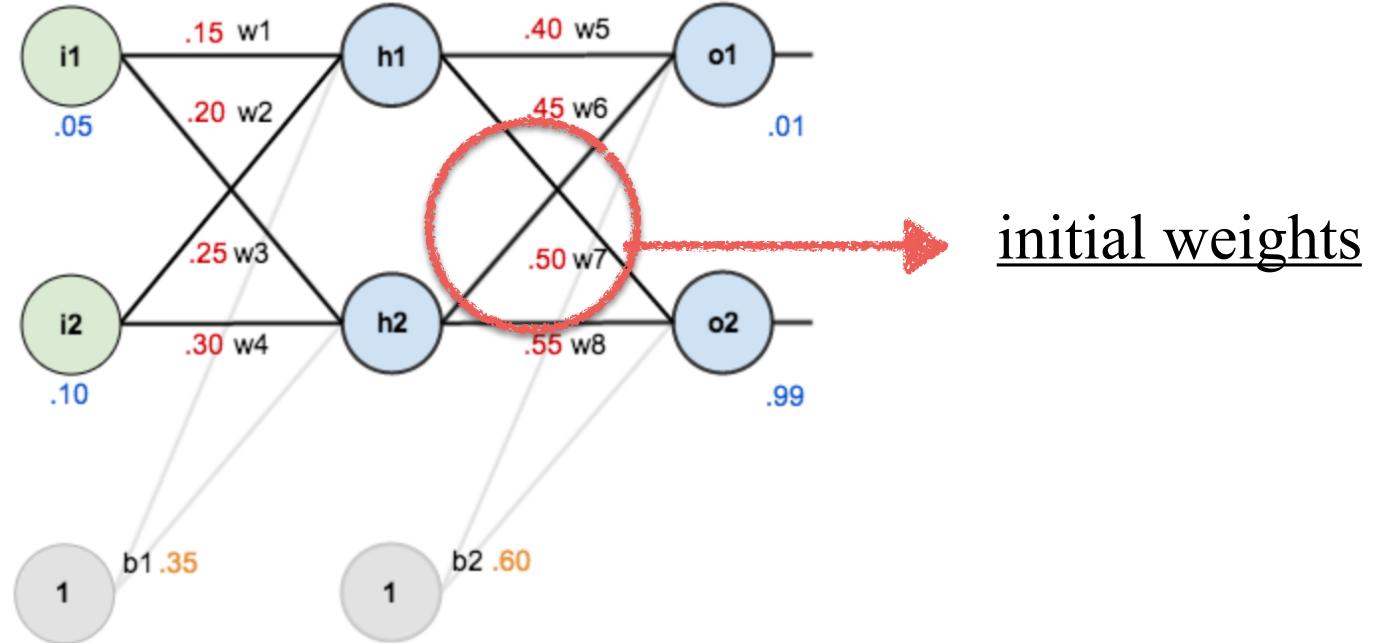


EXAMPLE TAKEN FROM HERE



LET'S ASSUME A VERY SIMPLE TRAINING SET:
 $X=(0.05, 0.10) \rightarrow Y=(0.01, 0.99)$

EXAMPLE TAKEN FROM HERE

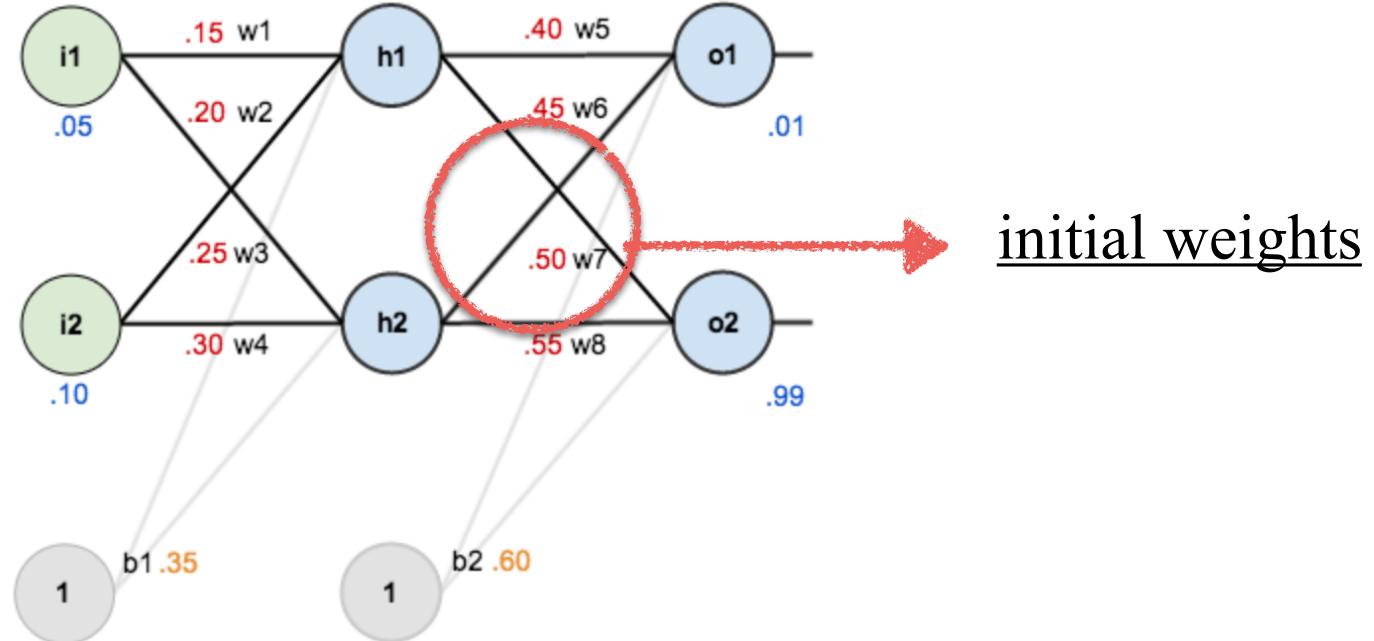


1. THE FORWARD PASS

$$in_{h1} = w_1 i_1 + w_2 i_2 + b_1$$

$$in_{h1} = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 = 0.3775$$

[with initial weights]



1. THE FORWARD PASS

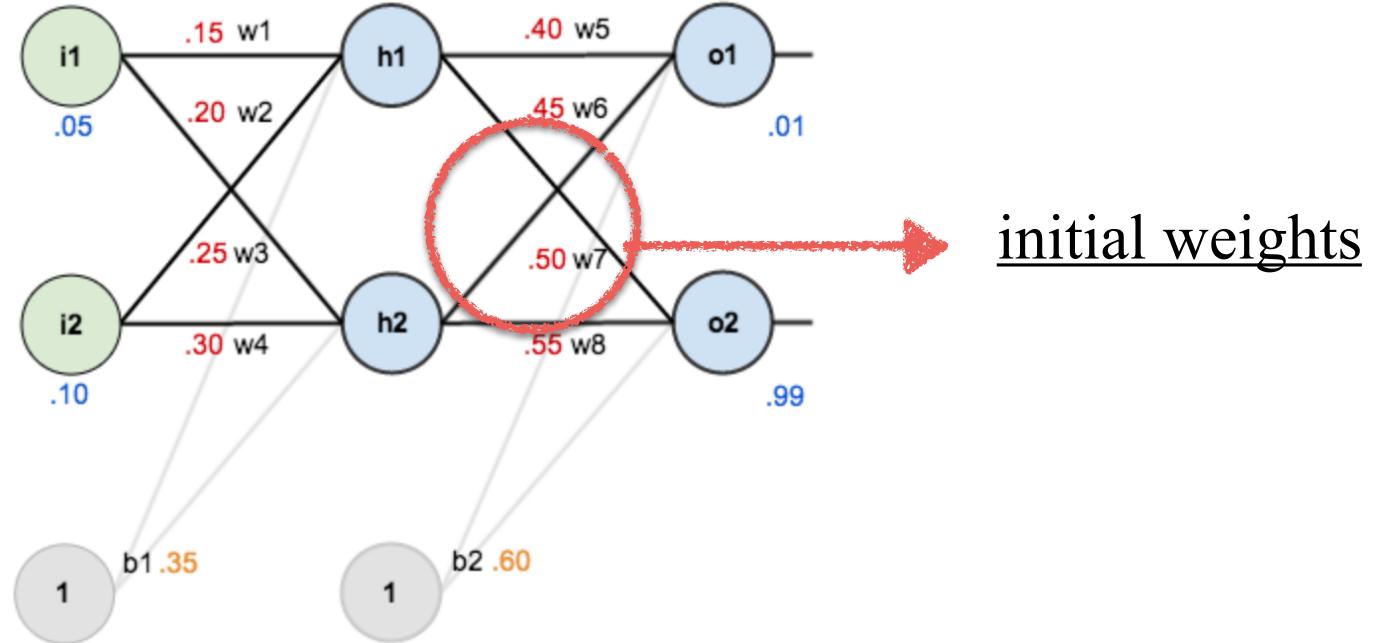
$$in_{h1} = w_1 i_1 + w_2 i_2 + b_1$$

$$in_{h1} = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 = 0.3775$$

[with initial weights]

$$out_{h1} = \frac{1}{1 + e^{-in_{h1}}} = 0.5932$$

[after the activation function]



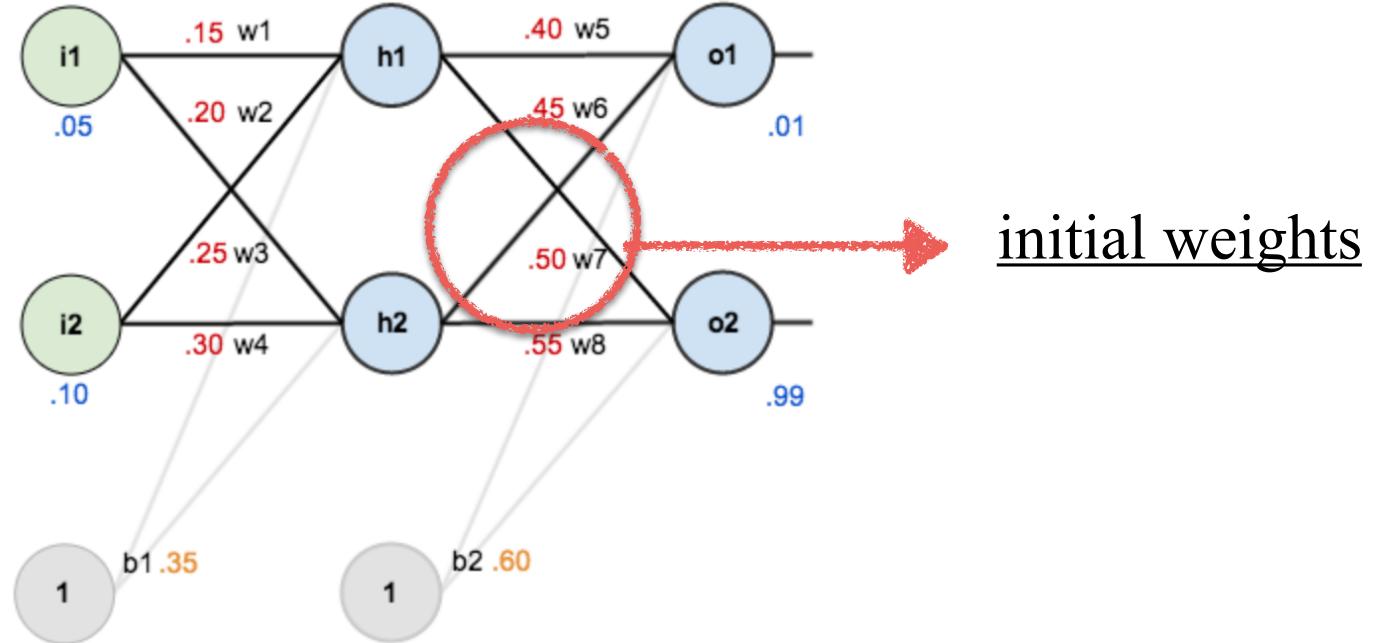
1. THE FORWARD PASS

WE CONTINUE TO o_1

$$in_{o1} = w_5 out_{h1} + w_6 out_{h2} + b_2$$

$$in_{o1} = 0.4 \times 0.593 + 0.45 \times 0.596 + 0.6 = 1.105$$

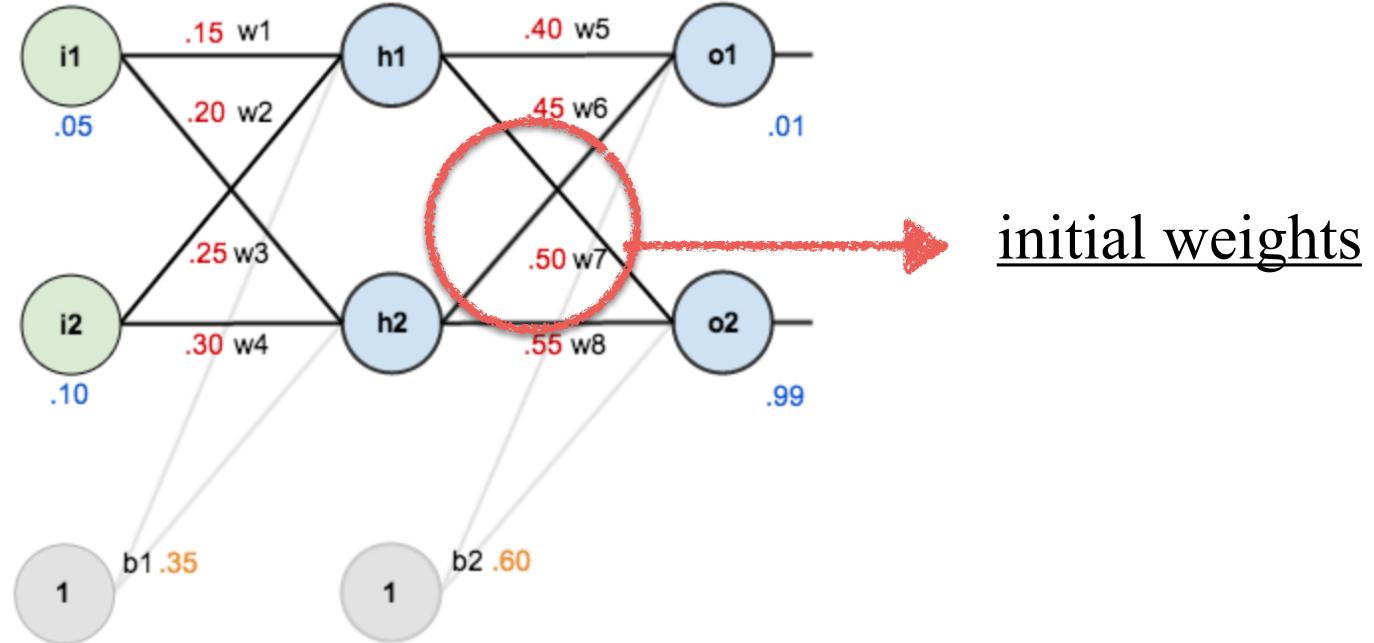
$$out_{o1} = \frac{1}{1 + e^{-1.105}} = 0.751$$



1. THE FORWARD PASS

AND THE SAME FOR o_2

$$out_{o2} = 0.7729$$

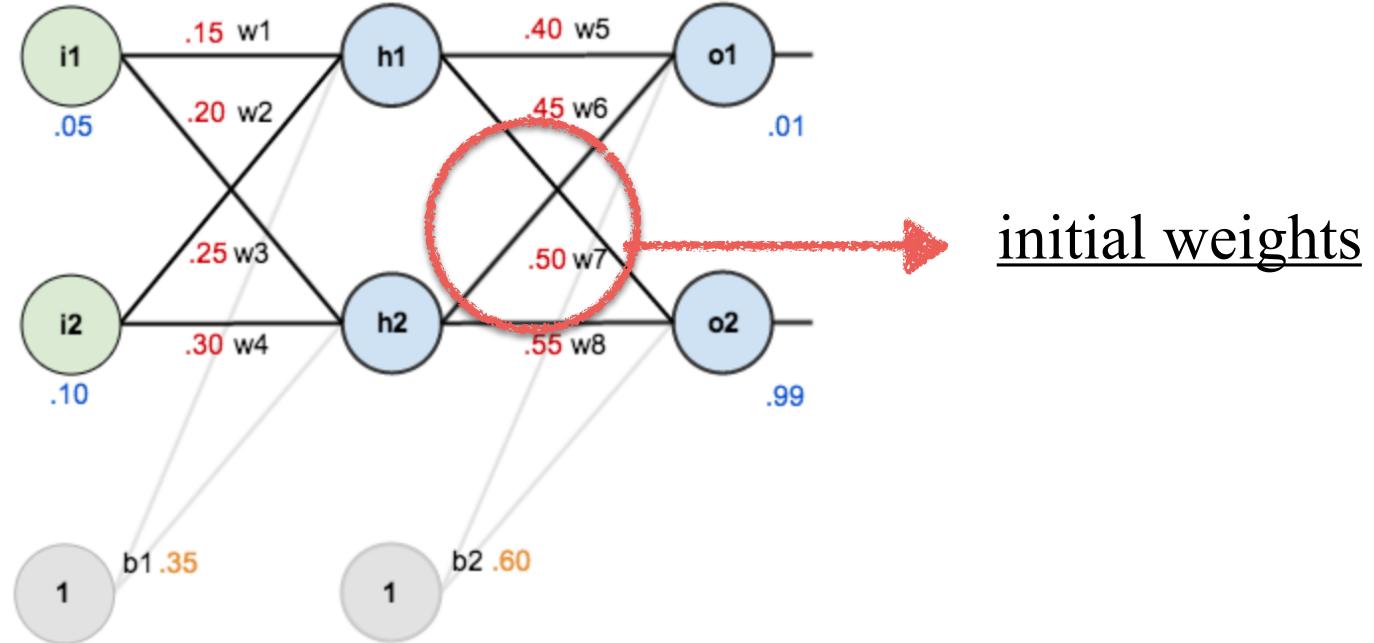


2. THE LOSS FUNCTION

$$L_{total} = \sum 0.5(target - output)^2$$

$$L_{o1} = 0.5(target_{o1} - output_{o1})^2 = 0.5 \times (0.01 - 0.751)^2 = 0.274$$

$$L_{o2} = 0.023$$

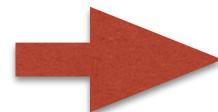


2. THE LOSS FUNCTION

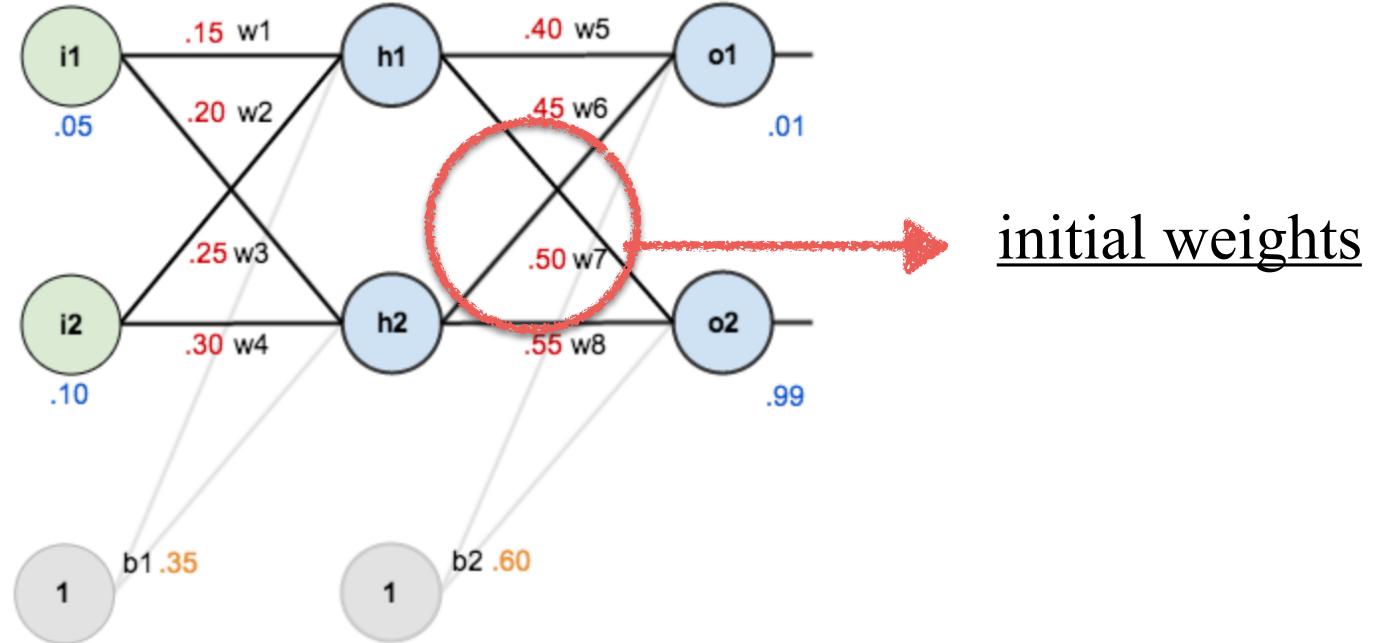
$$L_{total} = \sum 0.5(target - output)^2$$

$$L_{o1} = 0.5(target_{o1} - output_{o1})^2 = 0.5 \times (0.01 - 0.751)^2 = 0.274$$

$$L_{o2} = 0.023$$



$$L_{total} = L_{o1} + L_{o2} = 0.298$$



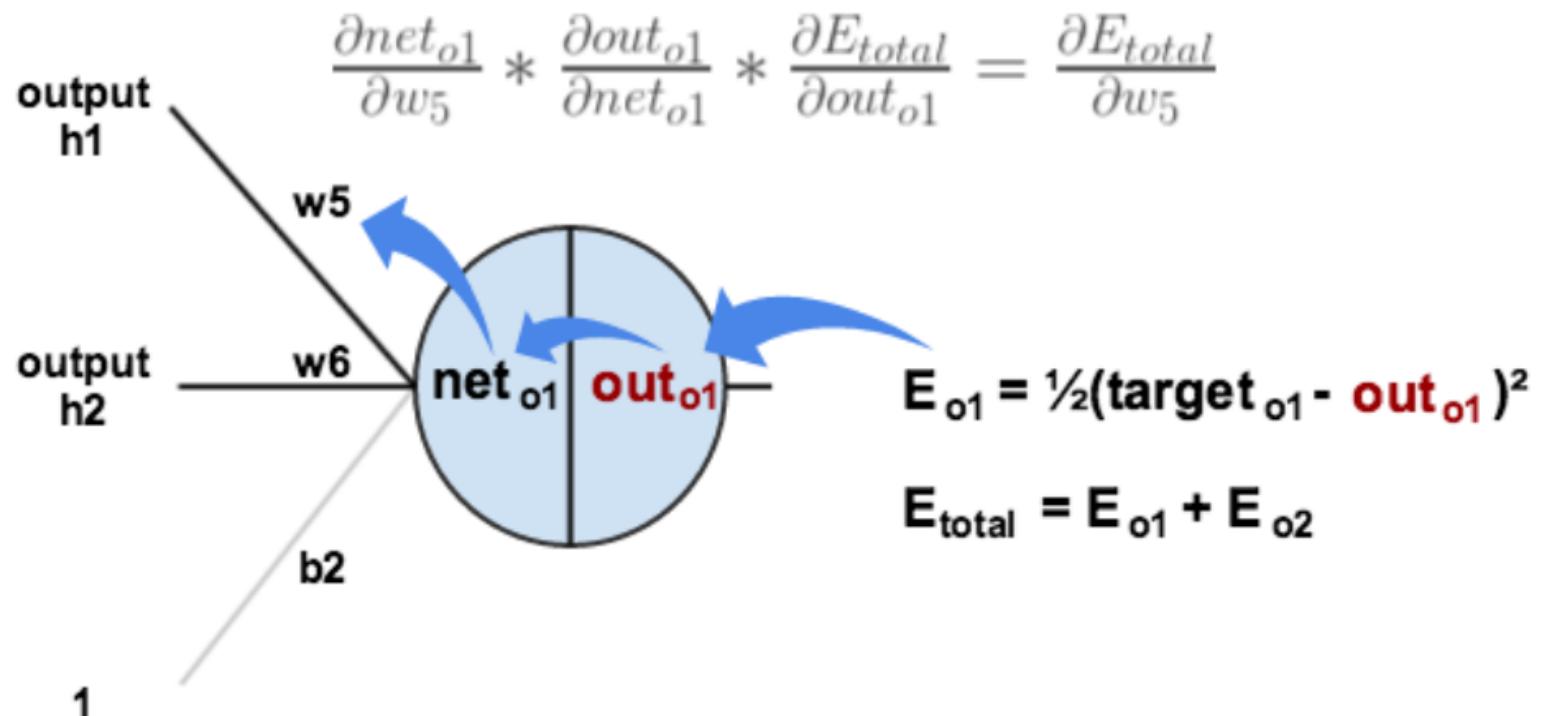
3. THE BACKWARD PASS

FOR *W*₅

WE WANT:

$$\frac{\partial L_{total}}{\partial w_5}$$

[gradient of loss function]



3. THE BACKWARD PASS

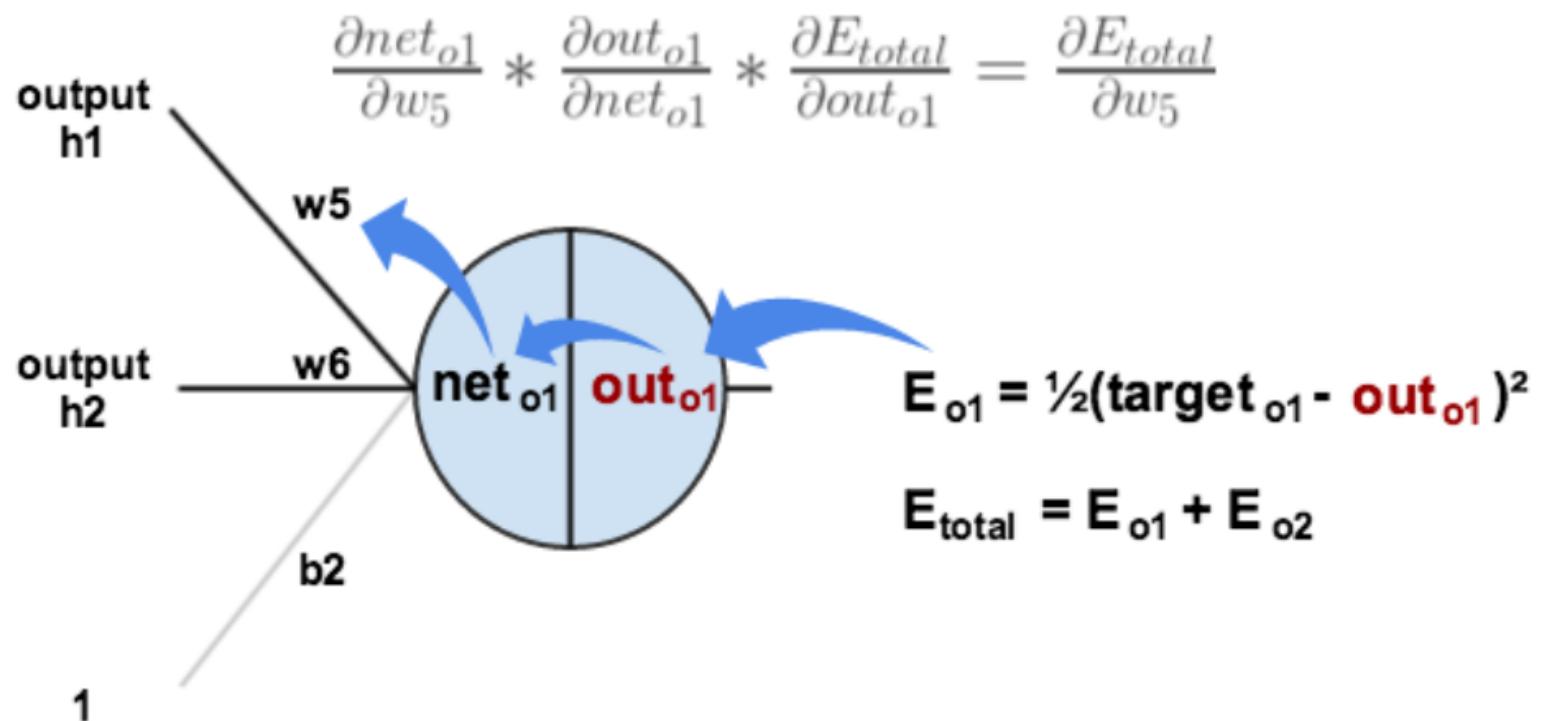
FOR w_5

WE WANT:

$$\frac{\partial L_{\text{total}}}{\partial w_5} \quad [\text{gradient of loss function}]$$

WE APPLY THE CHAIN RULE:

$$\frac{\partial L_{\text{total}}}{\partial w_5} = \frac{\partial L_{\text{total}}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

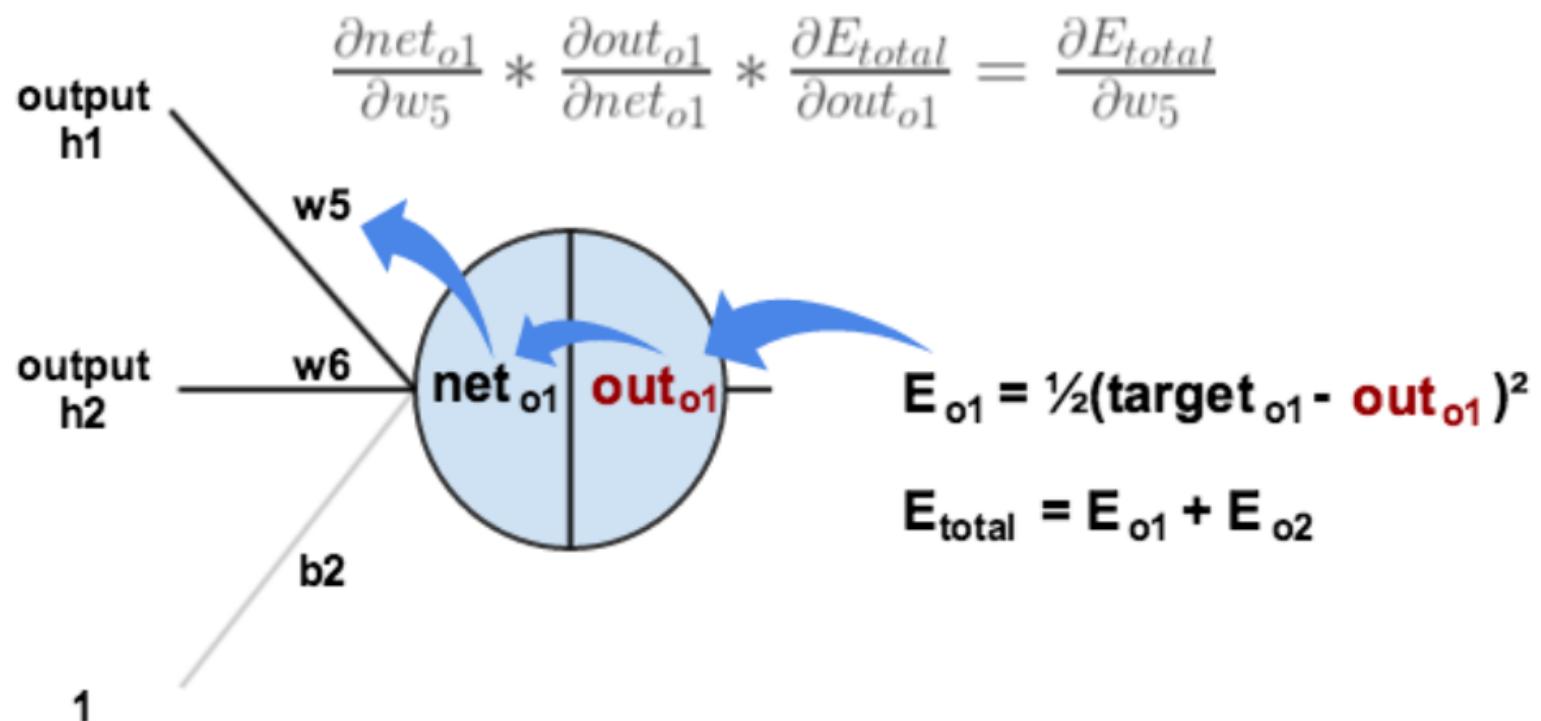


3. THE BACKWARD PASS

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$L_{total} = 0.5(\text{target}_{o1} - \text{out}_{o1})^2 + 0.5(\text{target}_{o2} - \text{out}_{o2})^2$$

$$\frac{\partial L_{total}}{\partial out_{o1}} = 2 \times 0.5(\text{target}_{o1} - \text{out}_{o1}) \times (-1) = 0.741$$

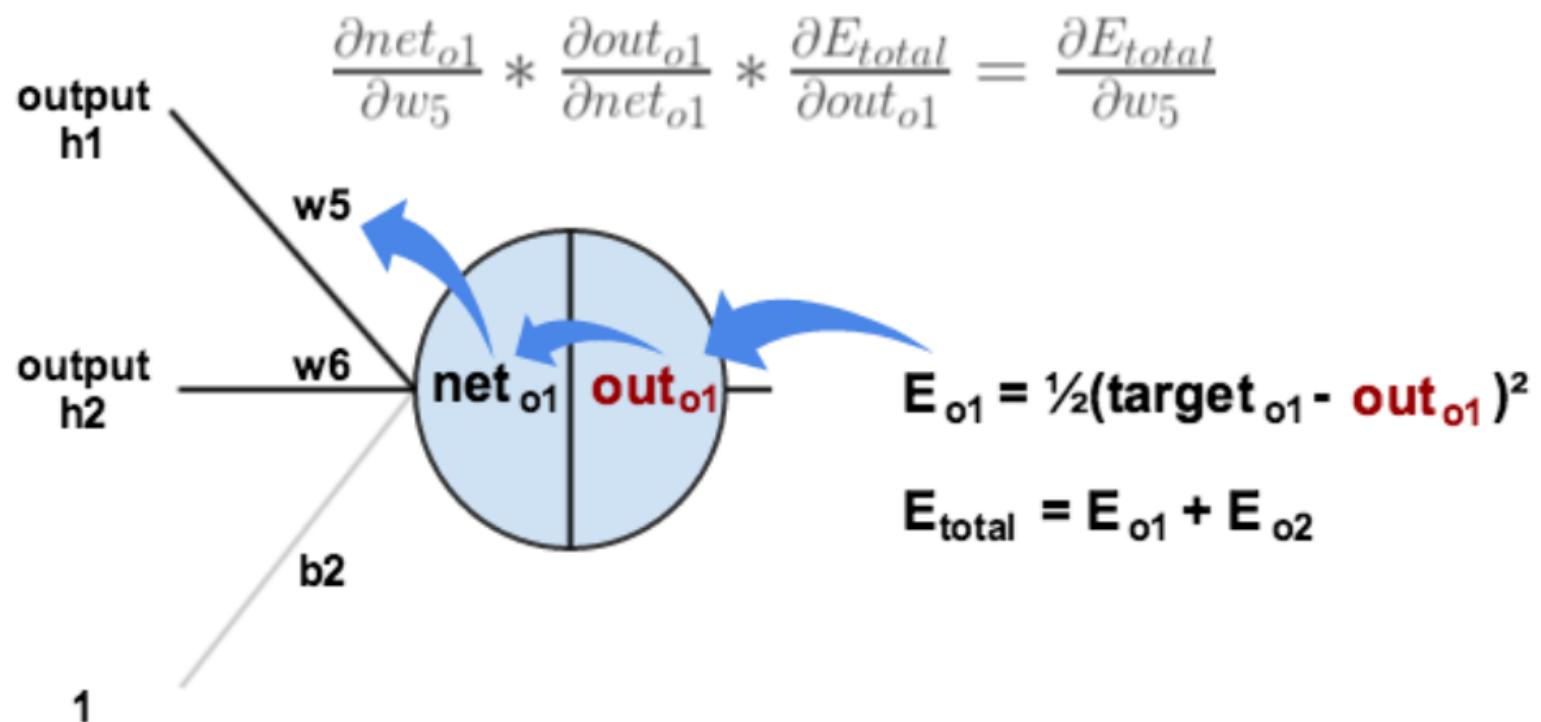


3. THE BACKWARD PASS

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$out_{o1} = \frac{1}{1 + e^{-in_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial in_{o1}} = out_{o1} \times (1 - out_{o1}) = 0.186$$

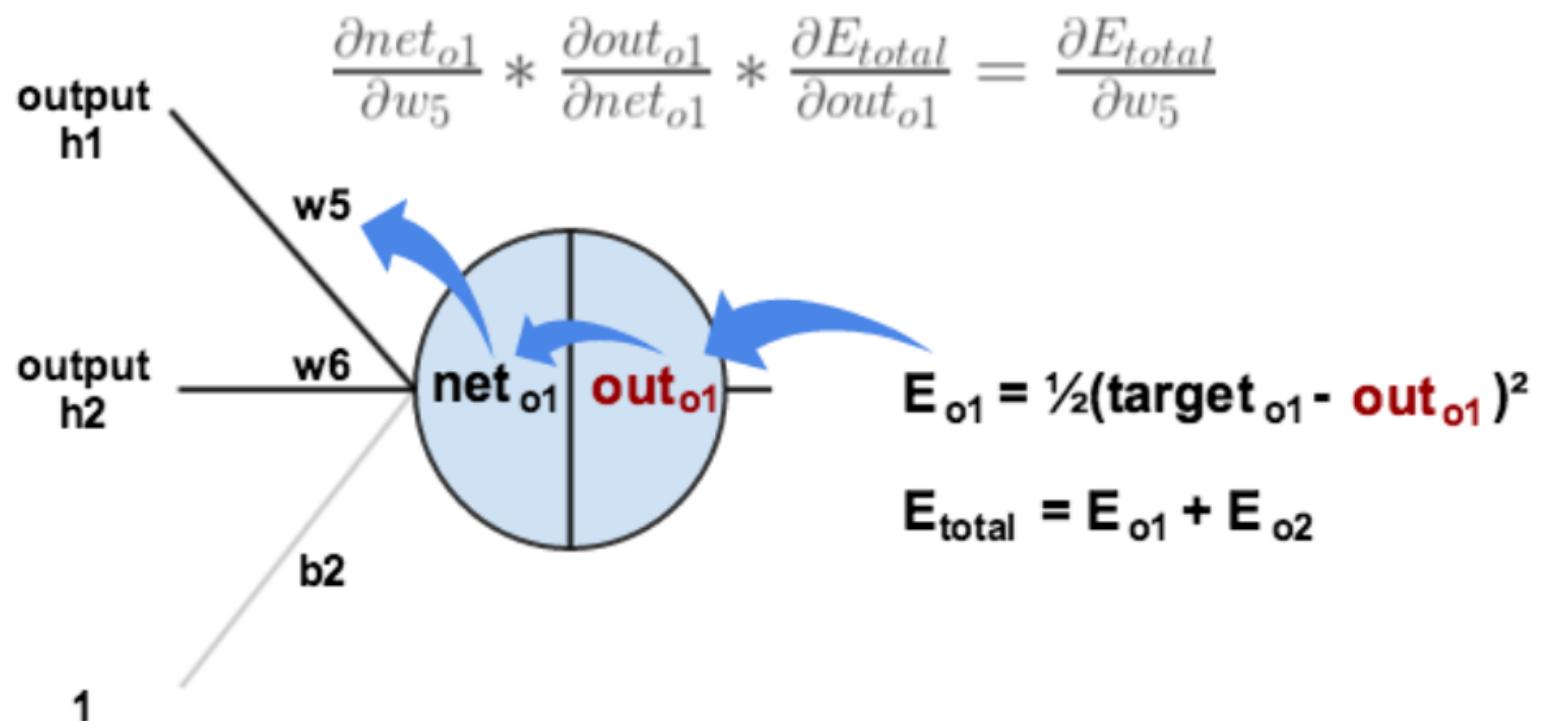


3. THE BACKWARD PASS

$$\frac{\partial L_{\text{total}}}{\partial w_5} = \frac{\partial L_{\text{total}}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$in_{o1} = w_5 \times out_{h1} + w_6 \times out_{h2} + b_2$$

$$\frac{\partial in_{o1}}{\partial w_5} = out_{h1} \times w_5^{1-1} = out_{h1} = 0.593$$

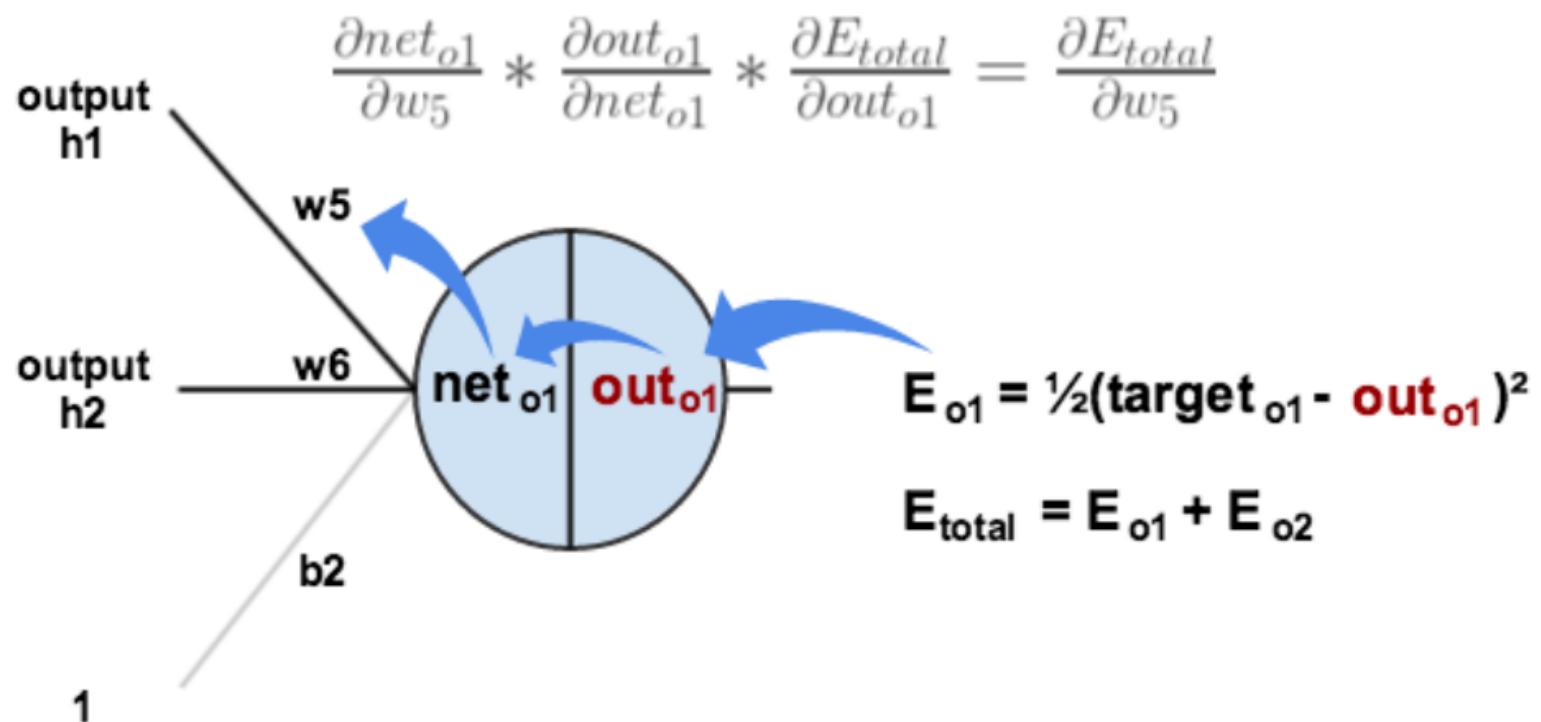


3. THE BACKWARD PASS

ALL TOGETHER:

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

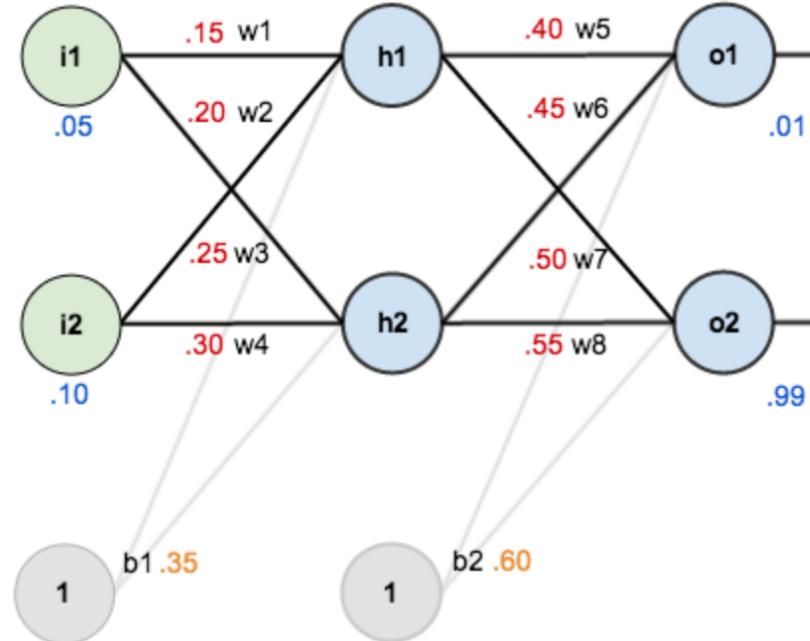
$$\frac{\partial L_{total}}{\partial w_5} = 0.741 \times 0.186 \times 0.593 = 0.082$$



4. UPDATE WEIGHTS WITH GRADIENT AND LEARNING RATE

$$w_5^{t+1} = w_5 - \lambda \times \frac{\partial L_{total}}{\partial w_5}$$

$$w_5^{t+1} = 0.4 - 0.5 \times 0.082 = 0.358$$



**THIS IS REPEATED FOR THE OTHER WEIGHTS
OF THE OUTPUT LAYER**

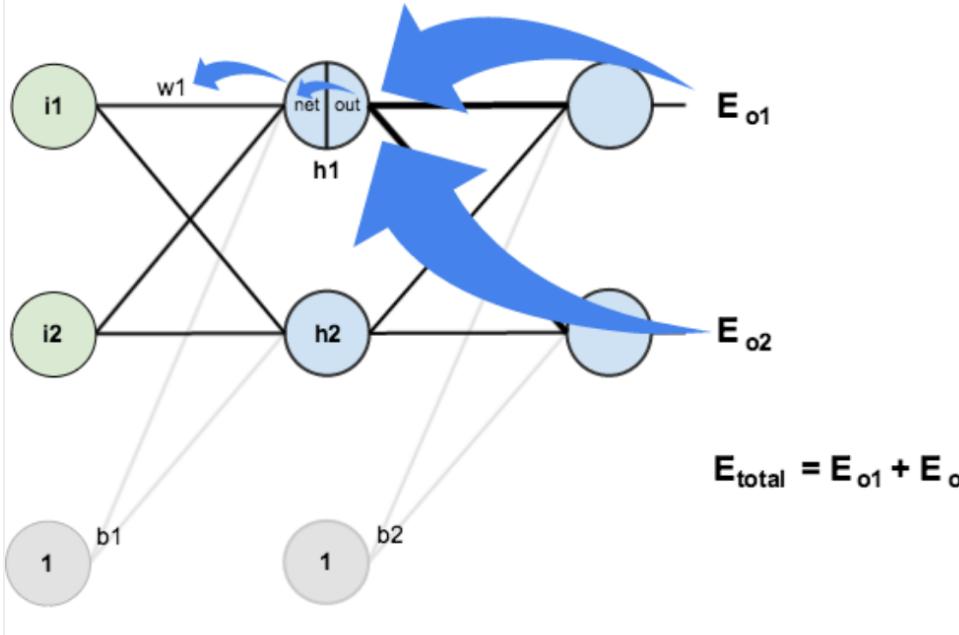
$$w_6^{t+1} = 0.408$$

$$w_7^{t+1} = 0.511$$

$$w_8^{t+1} = 0.561$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

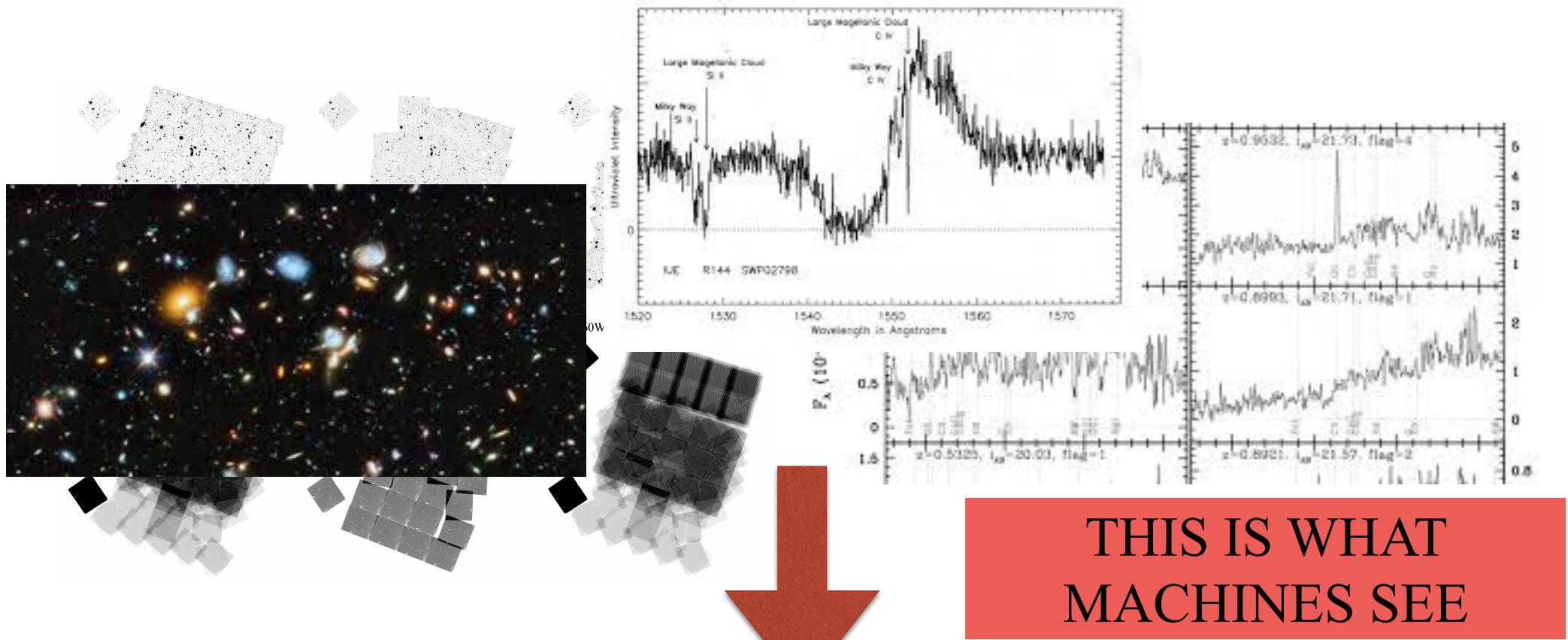


AND BACK-PROPAGATED TO THE HIDDEN LAYERS

CAN WE GO DEEP NOW?

ALMOST THERE...LET'S THINK FOR A
MOMENT ABOUT WHAT WE PUT AS
INPUT...

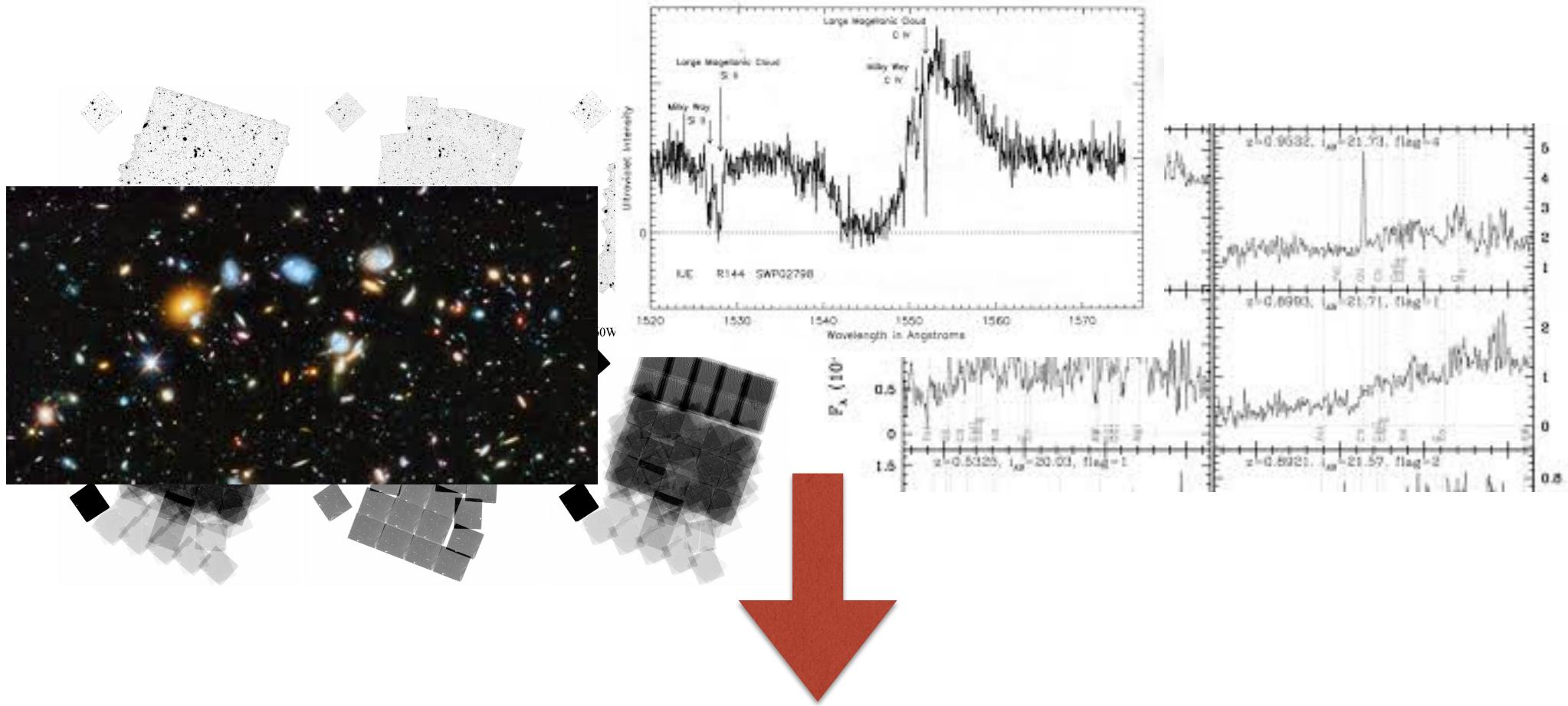
What do we put as input?



THIS IS WHAT MACHINES SEE

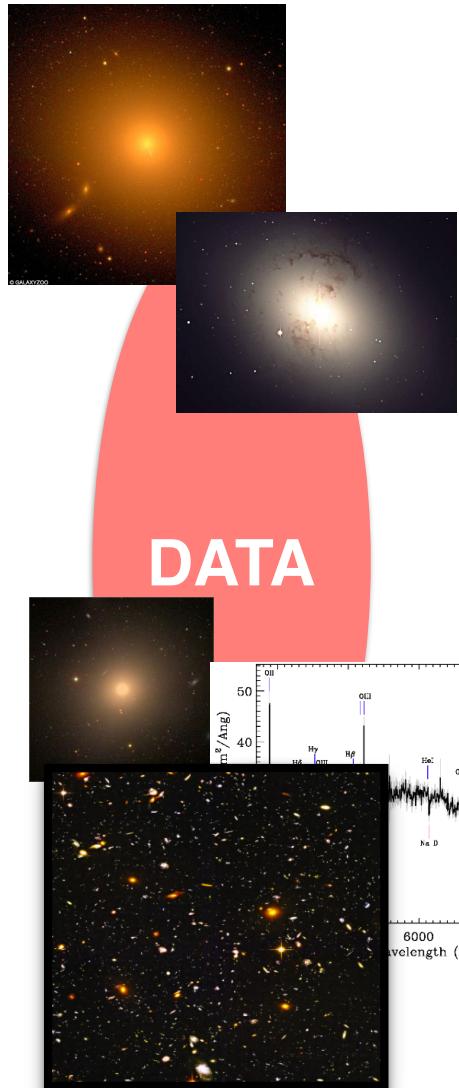
A magnifying glass is positioned over a grid of binary digits (0s and 1s) on a light blue background. The digits are arranged in a repeating pattern. The magnifying glass is focused on a specific group of four binary digits: 1, 1, 0, 1. This group is highlighted with a red arrow pointing to it from the top right.

What do we put as input?



PRE-PROCESS DATA TO EXTRACT MEANINGFUL INFORMATION

THIS IS GENERALLY CALLED **FEATURE EXTRACTION**



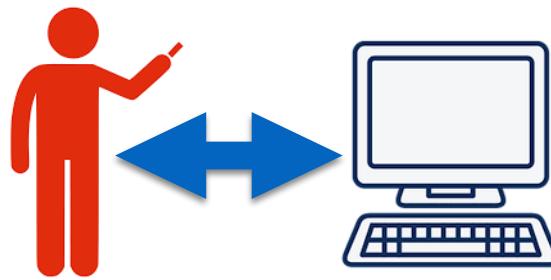
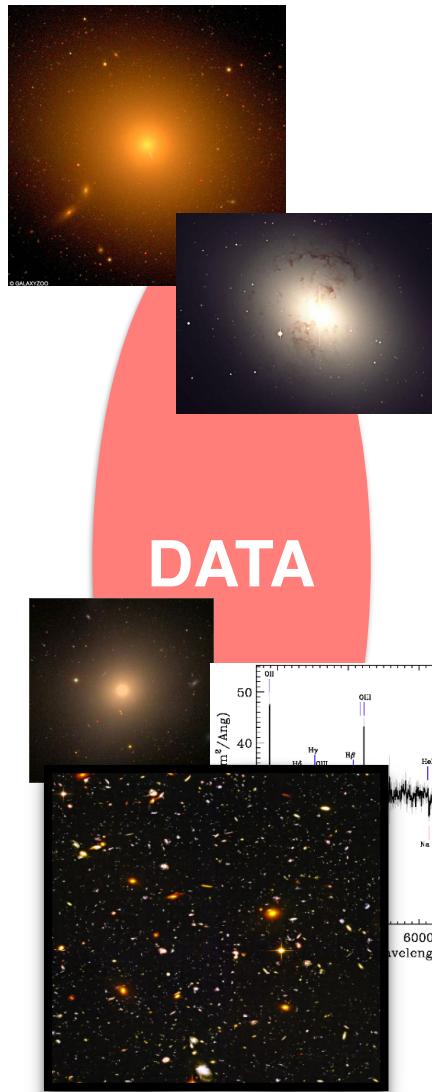
Spiral!

Emission line!

Merger!

Clump!

AGN!



Spiral!

Emission line!

Merger!

Clump!

AGN!

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

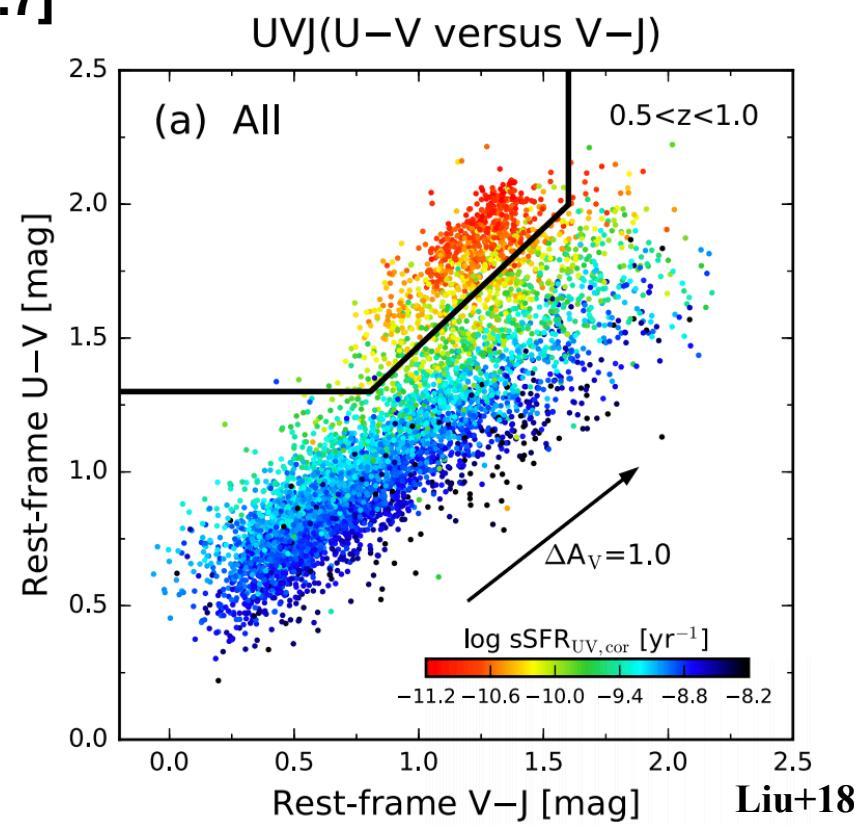
Q(0) , SF(1)

NETWORK FUNCTION

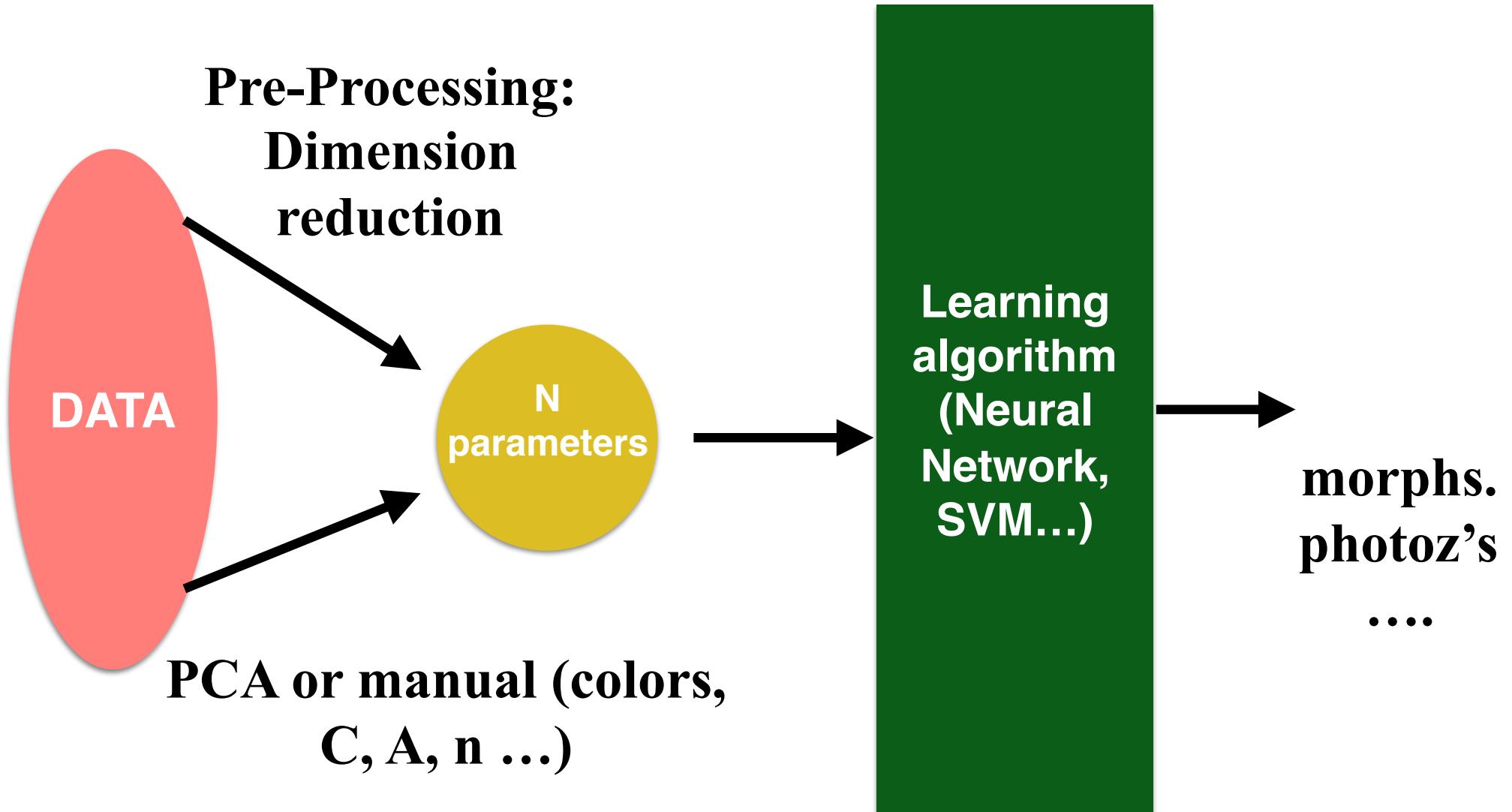
(U-V, V-J) FEATURES

$$\text{sgn}[(u-v)-0.8*(v-j)-0.7]$$

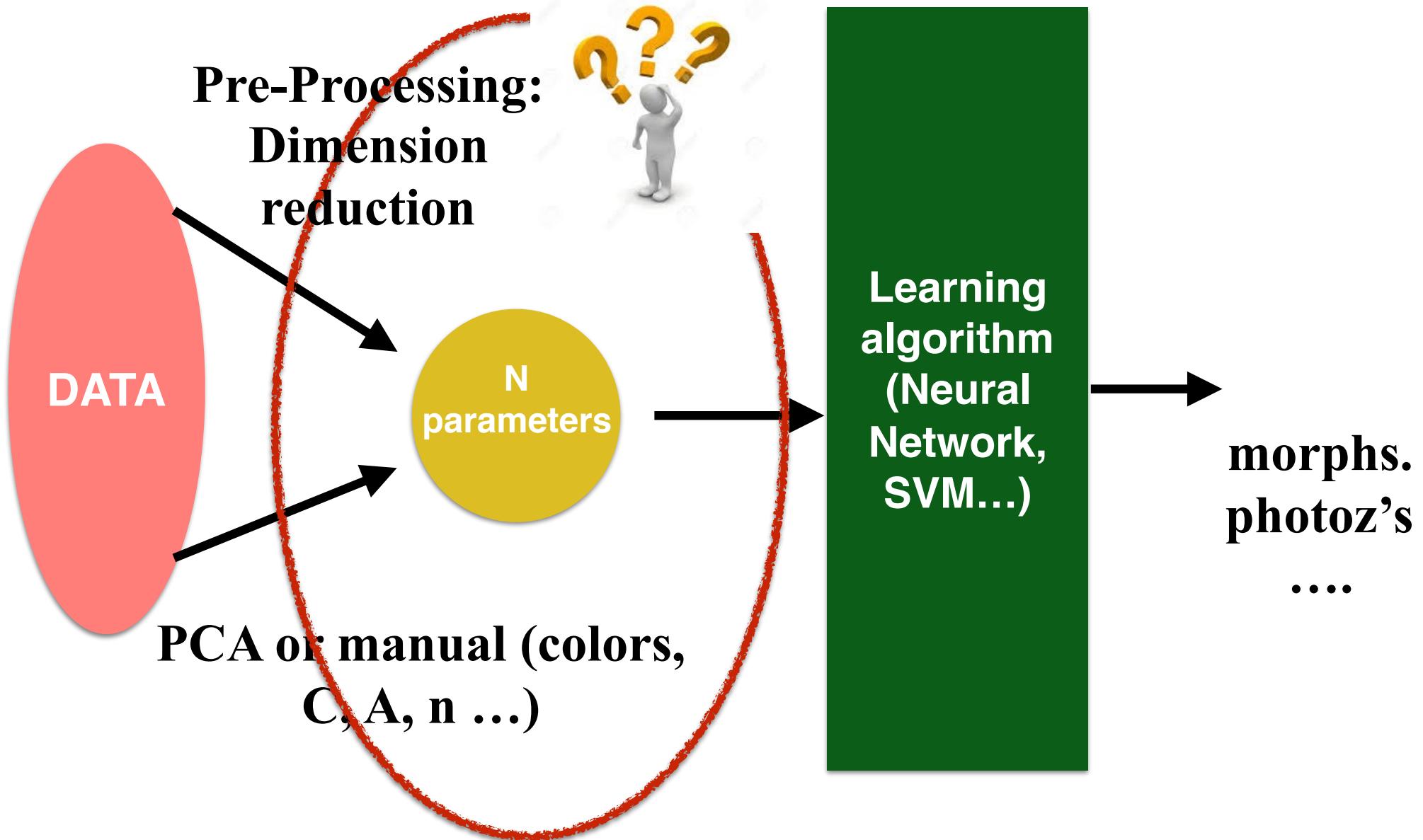
WEIGHTS



THE “CLASSICAL” APPROACH

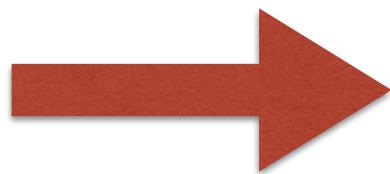


“CLASSICAL” MACHINE LEARNING



In Astronomy

- Colors, Fluxes
- Shape indicators
- Line ratios, spectral features
- Stellar Masses, Velocity Dispersions



Requires specialized software before feeding the machine learning algorithm

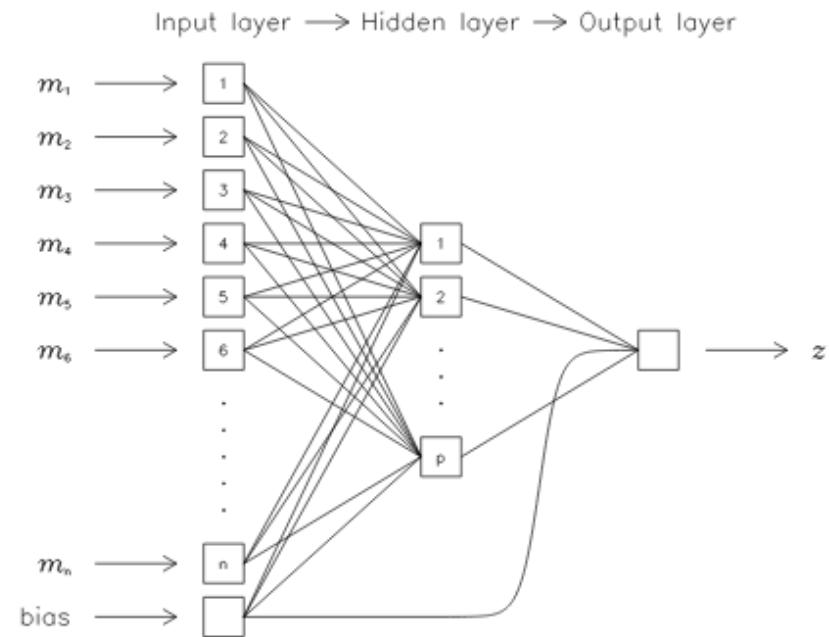
IT IMPLIES A DIMENSIONALITY REDUCTION!

PHOTOMETRIC REDSHIFTS

SDSS



g
r
i
z



Collister+08

**EVERYTHING IS IN THE FEATURES....WHAT IF I
IGNORED SOME IMPORTANT FEATURES?**



**EVERYTHING IS IN THE FEATURES....WHAT IF I
IGNORED SOME IMPORTANT FEATURES?**



WHAT ABOUT USING RAW DATA?

ALL INFORMATION IS IN THE INPUT DATA

WHY REDUCING ?

LET THE NETWORK FIND THE INFO

WHAT ABOUT USING RAW DATA?

ALL INFORMATION IS IN THE INPUT DATA

WHY REDUCING ?

LET THE NETWORK FIND THE INFO

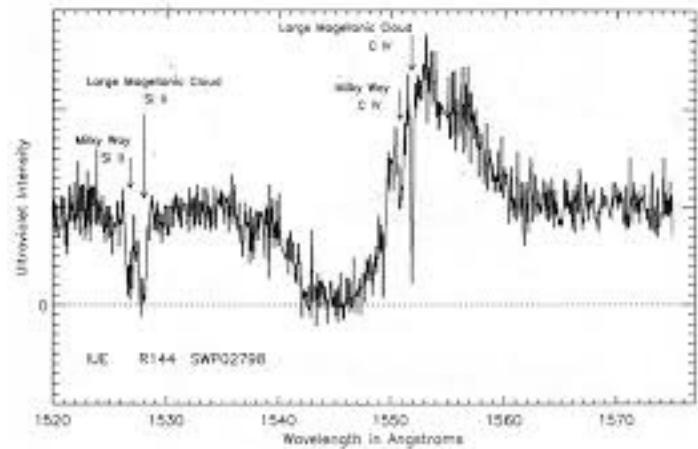
LARGE DIMENSION SIGNALS SUCH AS IMAGES OR SPECTRA WOULD REQUIRE TREMENDOUSLY LARGE MODELS

A 512x512 image as input of a fully connected layer producing output of same size:

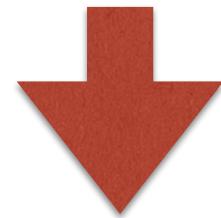
$$(512 \times 512)^2 = 7e10$$

BUT

FEEDING INDIVIDUAL RESOLUTION ELEMENTS IS NOT
VERY EFFICIENT SINCE IT LOOSES ALL INVARIANCE TO
TRANSLATION AND IGNORES CORRELATION IN THE DATA
AT ALL SCALES

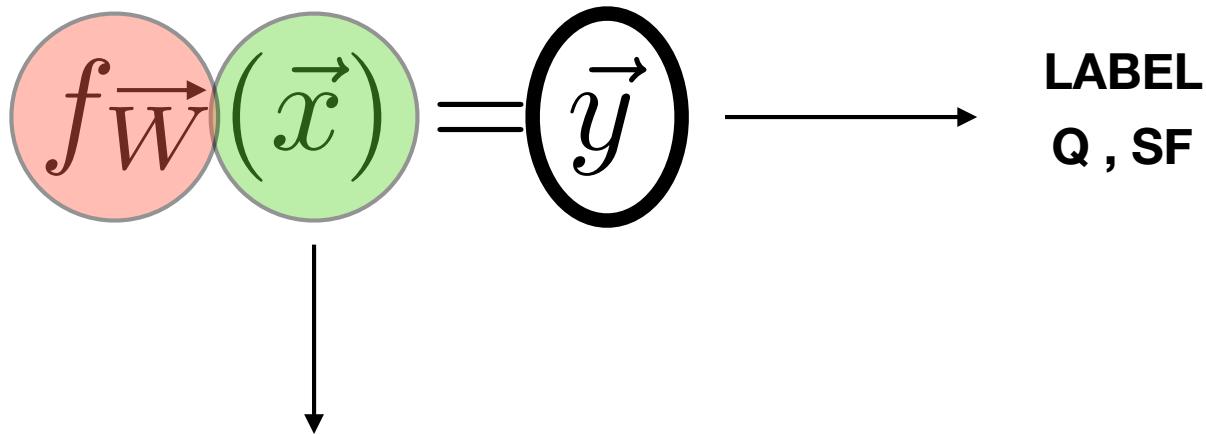


FEEDING INDIVIDUAL RESOLUTION ELEMENTS IS NOT
VERY EFFICIENT SINCE IT LOSES ALL INVARIANCE TO
TRANSLATION



SO?

DEEP LEARNING



LET THE MACHINE FIGURE THIS OUT (“unsupervised feature extraction”)

LET'S GO A STEP FORWARD INTO LOOSING CONTROL....