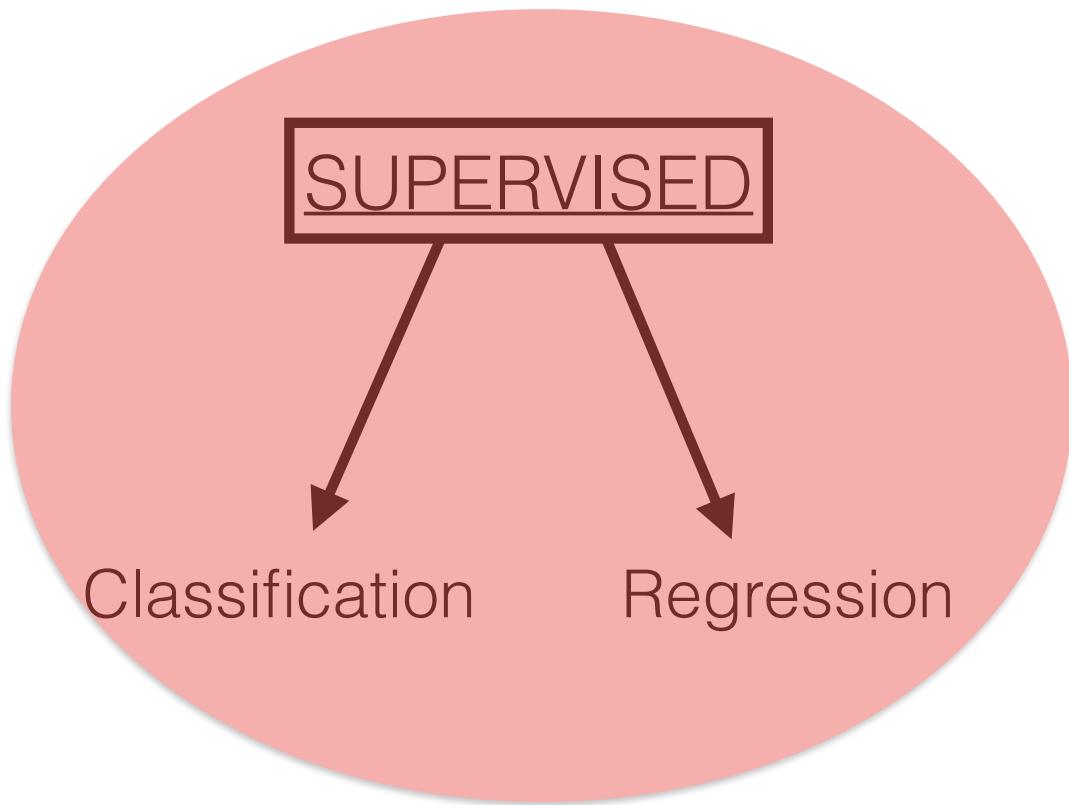


AN INTRODUCTION TO “CLASSICAL” MACHINE LEARNING

WHAT DOES MACHINE LEARNING DO?



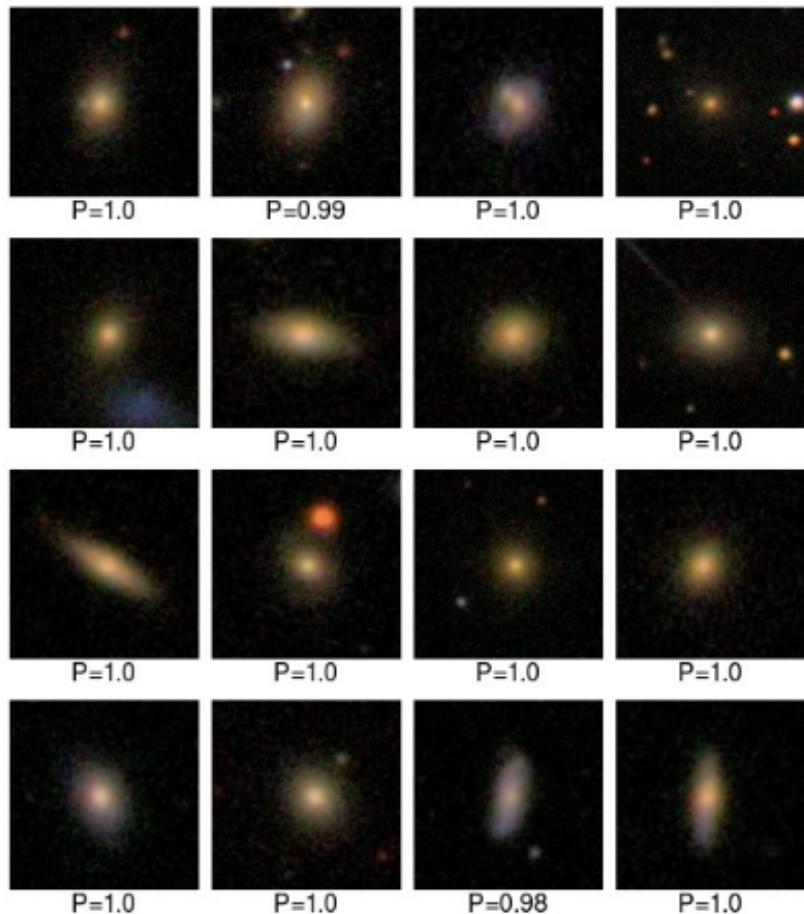
WHAT DOES MACHINE LEARNING DO?



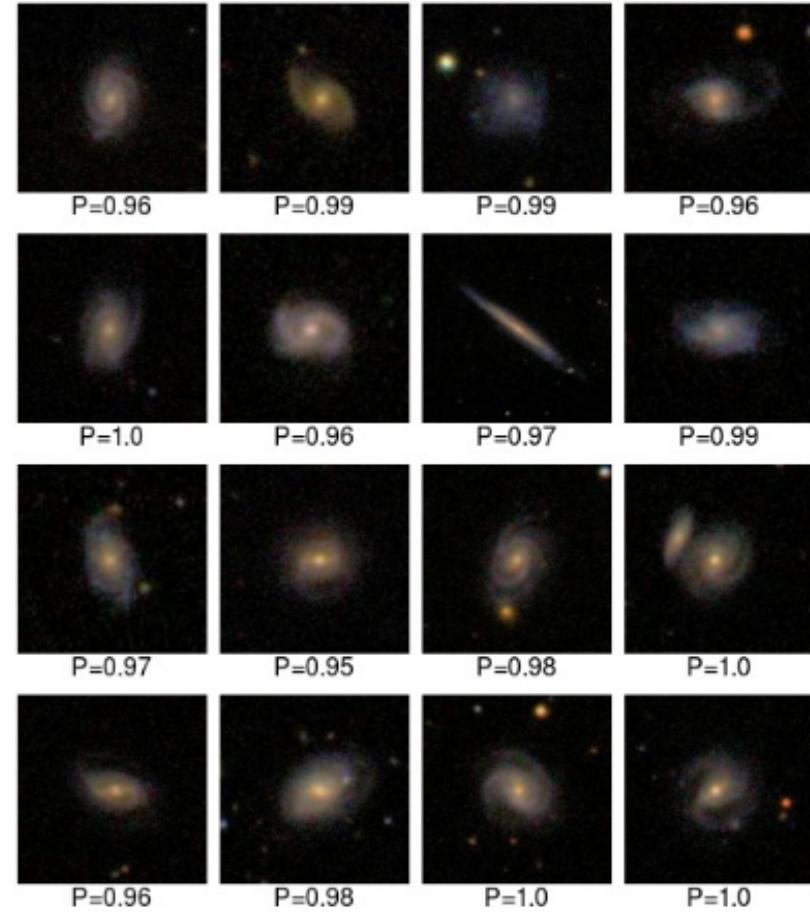
UN-SUPERVISED

Clustering Generative

Classification

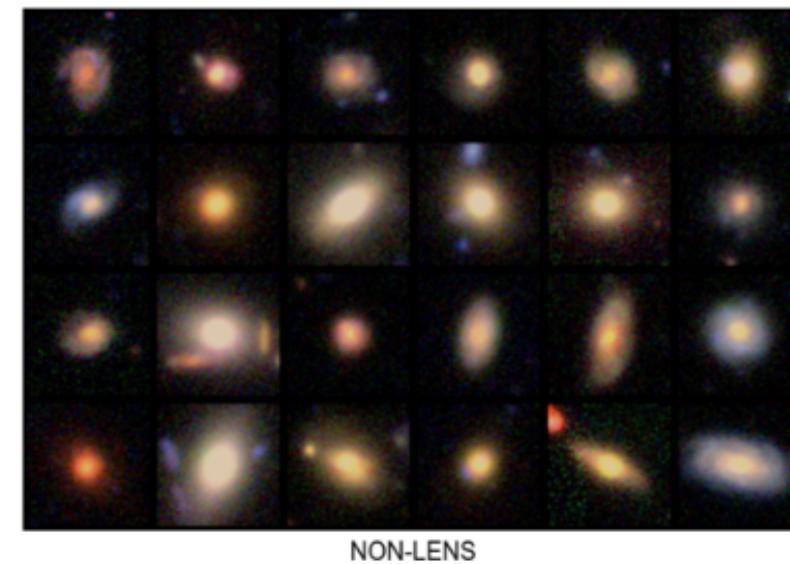
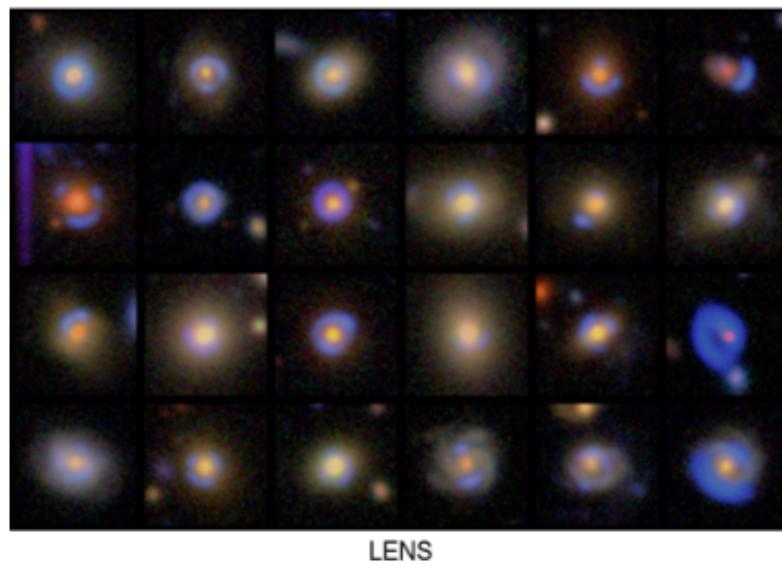


EARLY-TYPE



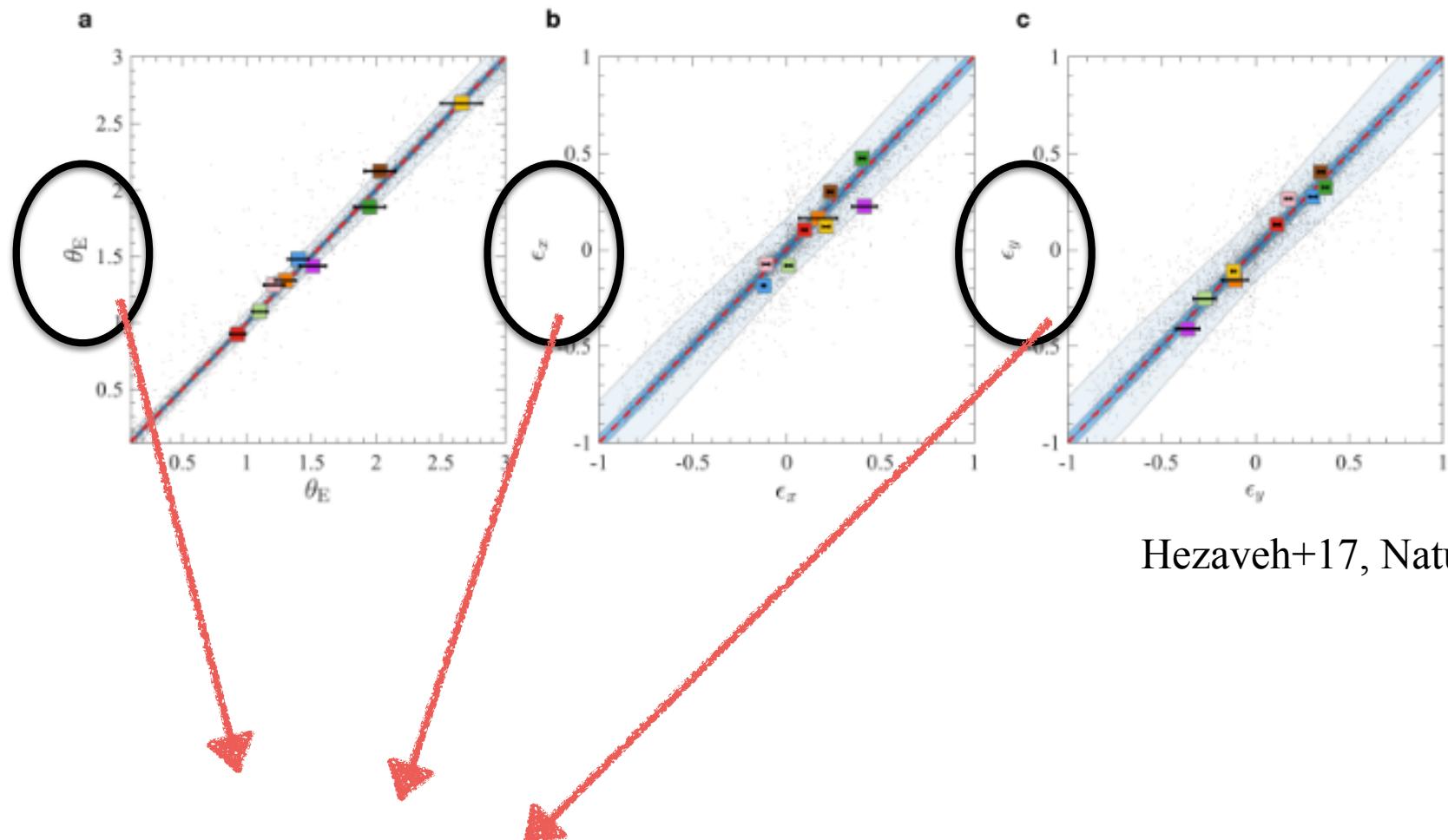
LATE-TYPE

Classification



Jacobs+17

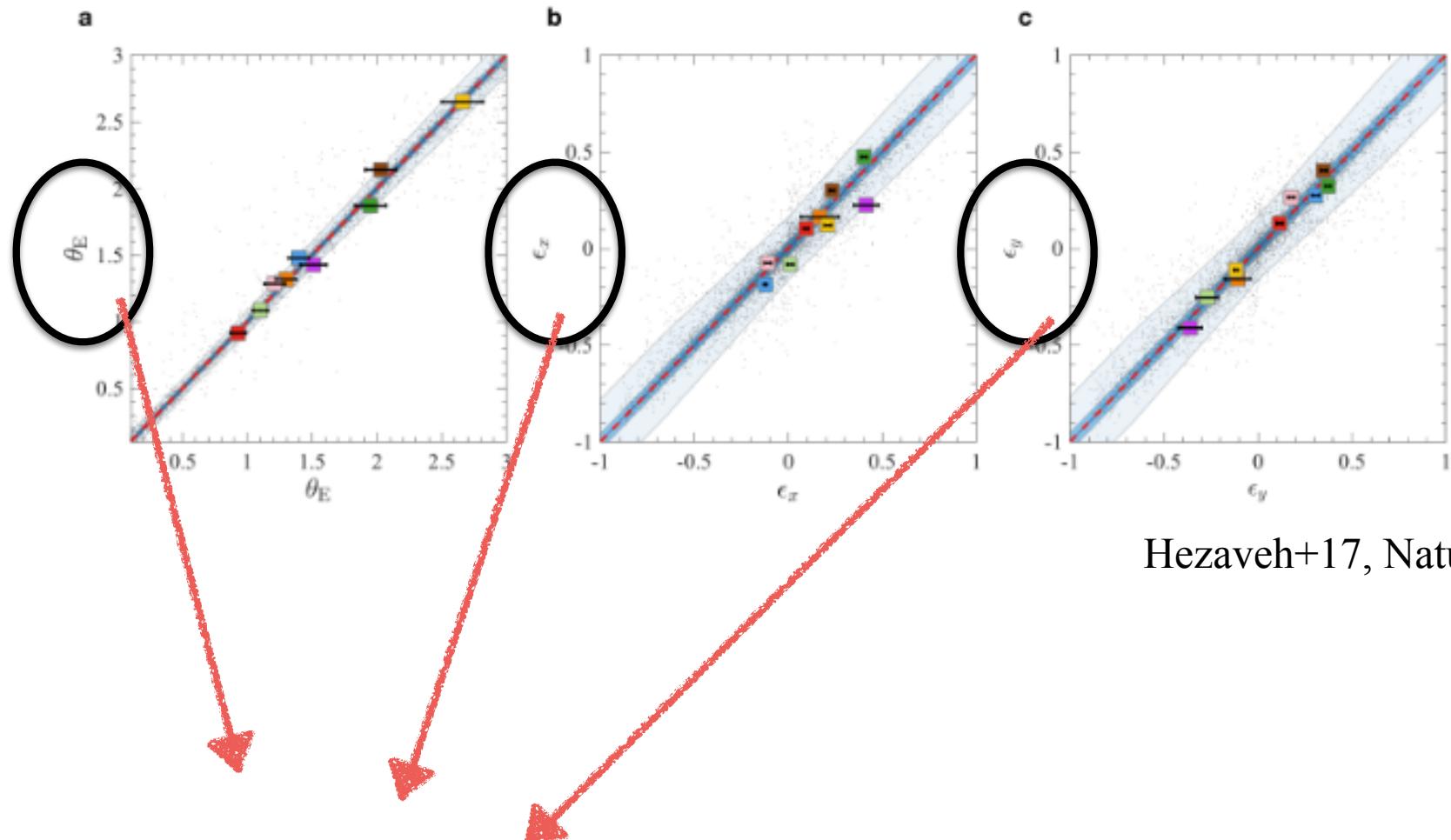
Regression



Hezaveh+17, Nature

REGRESSION ON
STRONG LENSES PARAMETERS

Regression



Hezaveh+17, Nature

REGRESSION ON
STRONG LENSES PARAMETERS

Supervised

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

Training set

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

Measurements
(colors, fluxes, spectra indices...)

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

Label
(morphology, object type, transit ...)

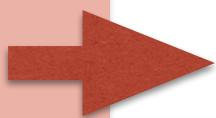
Supervised

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

Training set

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

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

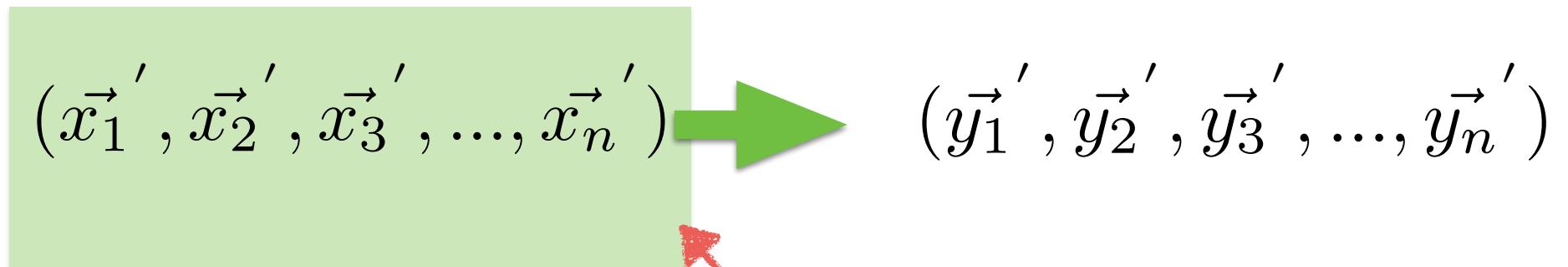


$$f_W(\vec{x}) = \vec{y}$$

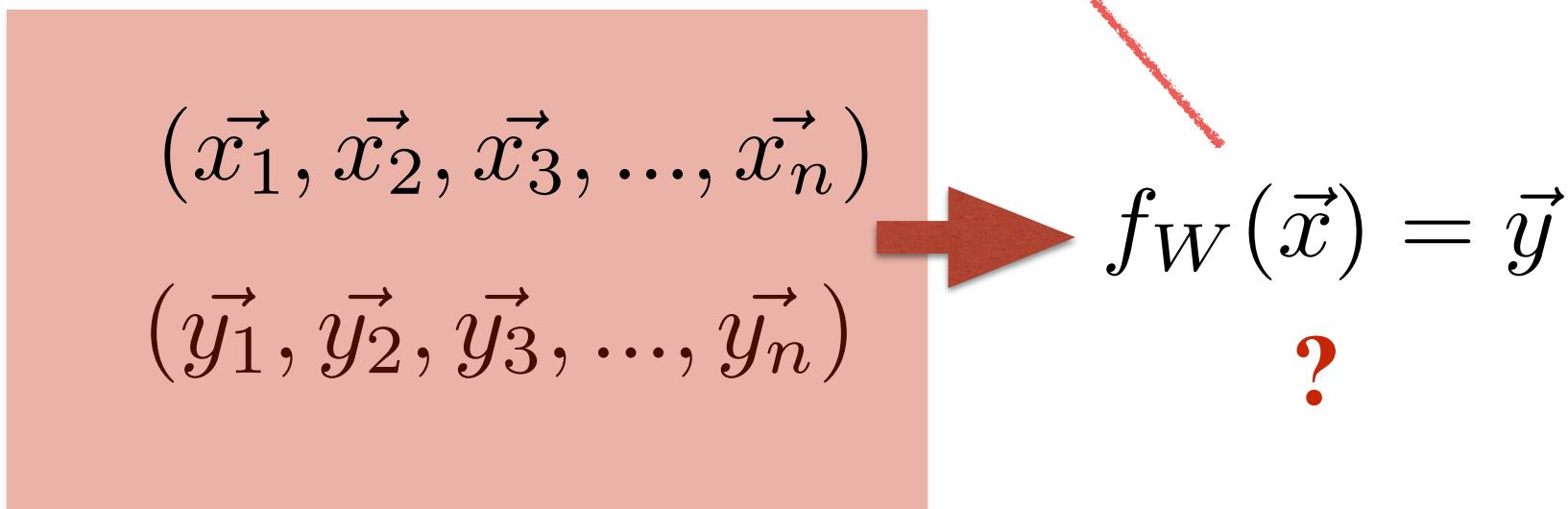
?

Supervised

Unlabeled set



Training set



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

$$\vec{x} \in \mathbb{R}^d$$

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

$$\vec{y} \in \mathbb{R} \quad \vec{y} \in \mathbb{N}$$

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 f such as the prediction error is minimal over all unseen vectors

We need two key elements

A LOSS FUNCTION

**A MINIMIZATION OR OPTIMIZATION
ALGORITHM**

We need two key elements

A LOSS FUNCTION

**A MINIMIZATION OR OPTIMIZATION
ALGORITHM**

**THIS IS COMMON TO ALL MACHINE LEARNING
ALGORITHMS**

1. DEFINE A LOSS FUNCTION

$$loss(F_W(.), \vec{x}_i, \vec{y}_i)$$

Typically: $(F_W(\vec{x}_i) - \vec{y}_i)^2$ Quadratic loss function

2. MINIMIZE THE EMPIRICAL RISK

$$\mathfrak{R}_{empirical}(W) = \frac{1}{N} \sum_i^N [loss(W, \vec{x}, \vec{y})]$$



MINIMIZE THE RISK

EMPIRICAL RISK?

$$\mathfrak{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_i^N [\text{loss}(W, \vec{x}, \vec{y})]$$

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

EMPIRICAL RISK?

$$\mathfrak{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_i^N [\text{loss}(W, \vec{x}, \vec{y})]$$

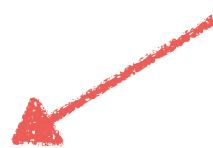
WE ARE MINIMIZING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

OBSERVED DATASET



EMPIRICAL RISK?

$$\mathfrak{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_i^N [\text{loss}(W, \vec{x}, \vec{y})]$$



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

ALL “GALAXIES IN THE UNIVERSE”

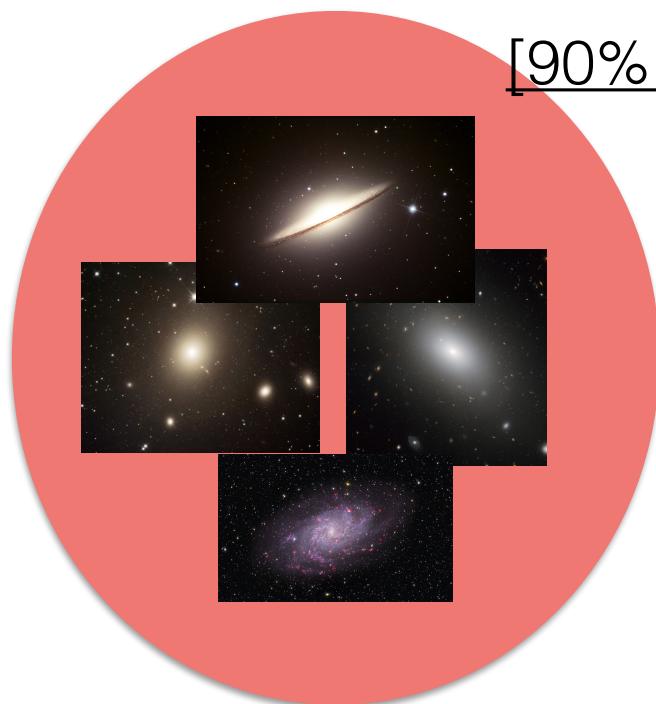
OBSERVED DATASET



THIS IS A VERY IMPORTANT ASSUMPTION. EVERY MACHINE
WILL BE OPTIMIZED TO MINIMIZE THE RISK IN THE
OBSERVED DATASET

THIS IS A VERY IMPORTANT ASSUMPTION. EVERY MACHINE
WILL BE OPTIMIZED TO MINIMIZE THE RISK IN THE
OBSERVED DATASET

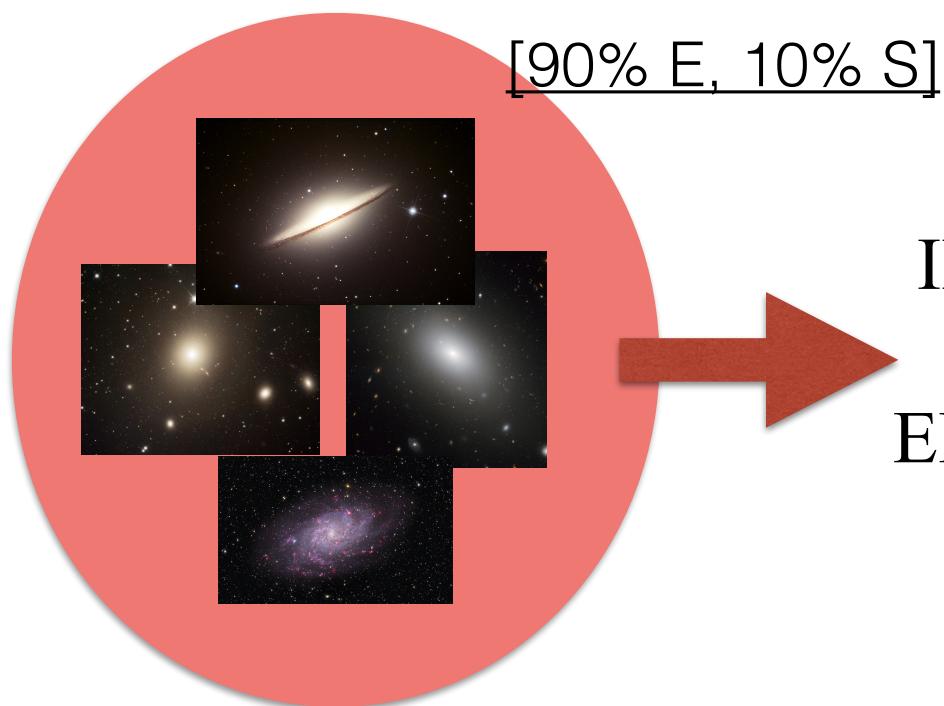
IMAGINE AN EXTREME CASE WITH VERY UNBALANCED
DATA...



[90% E, 10% S]

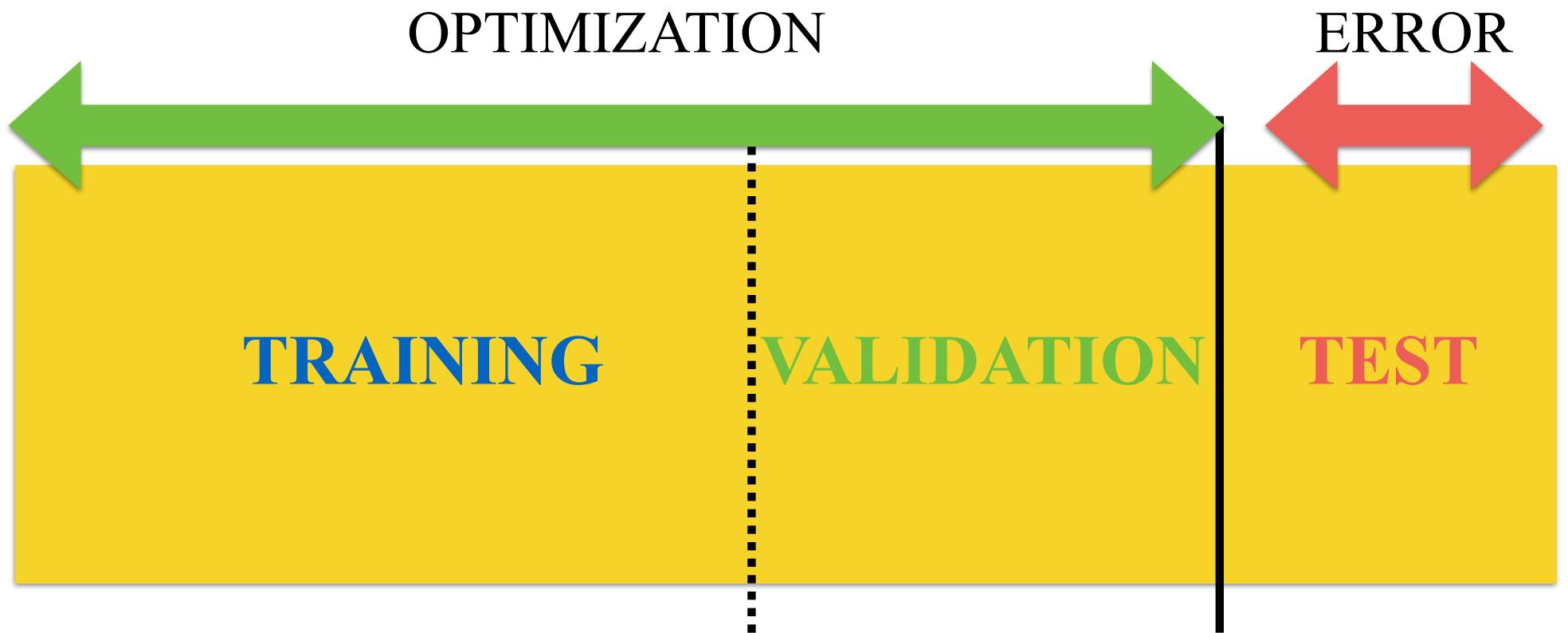
THIS IS A VERY IMPORTANT ASSUMPTION. EVERY MACHINE
WILL BE OPTIMIZED TO MINIMIZE THE RISK IN THE
OBSERVED DATASET

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

In practice

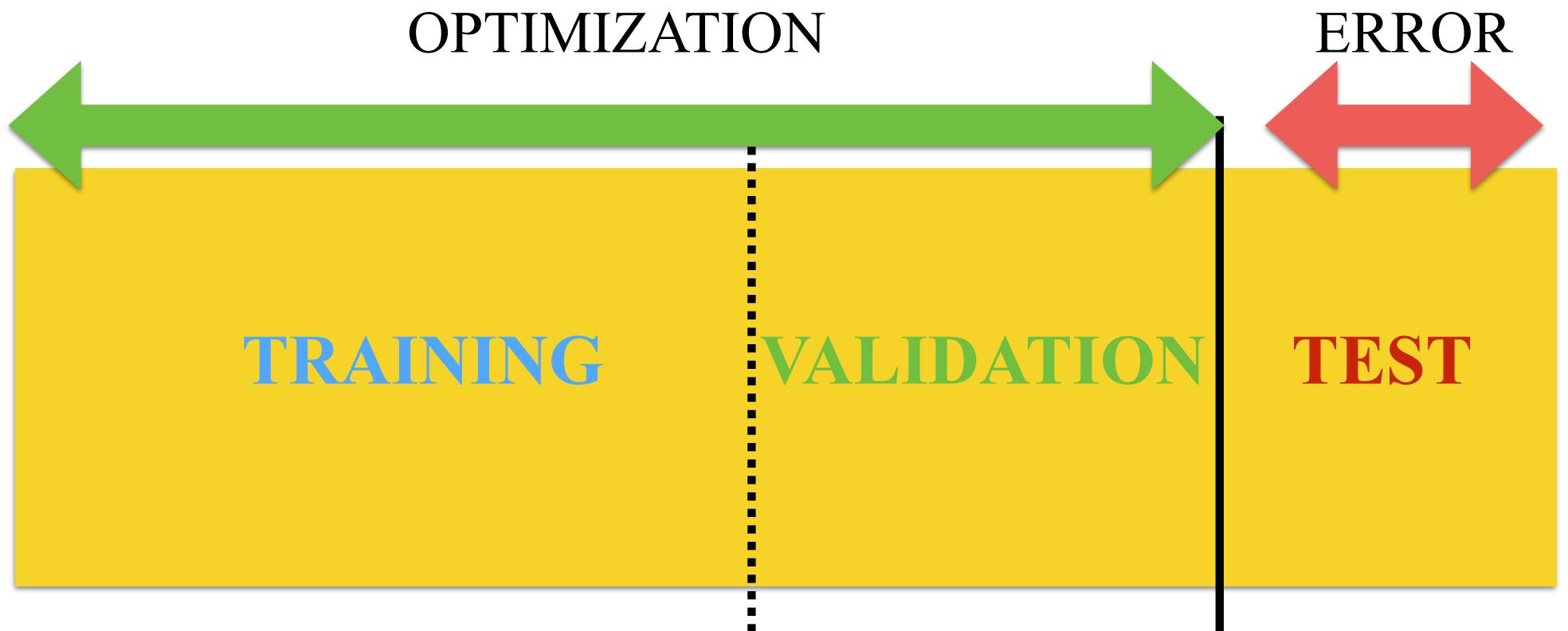


training set: use to train the classifier

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

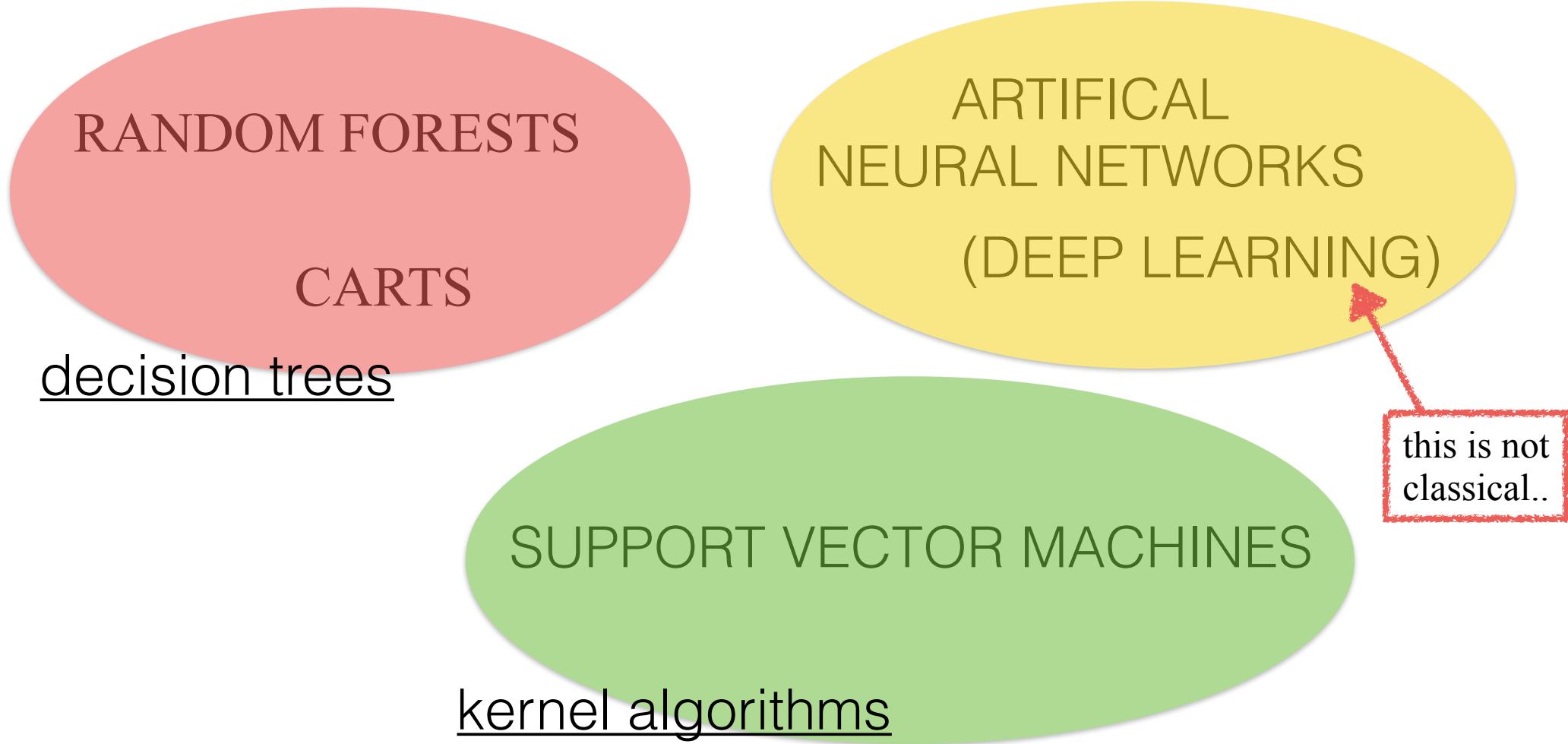
$$W_{t+1} = W_t - \lambda_h \nabla f(W_t)$$

weights to be learned

epoch

learning rate

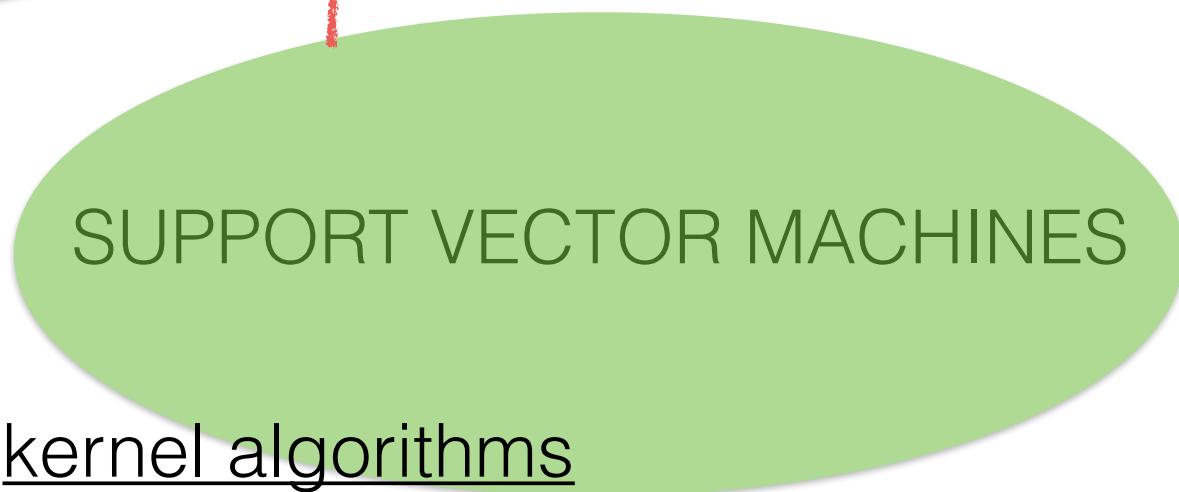
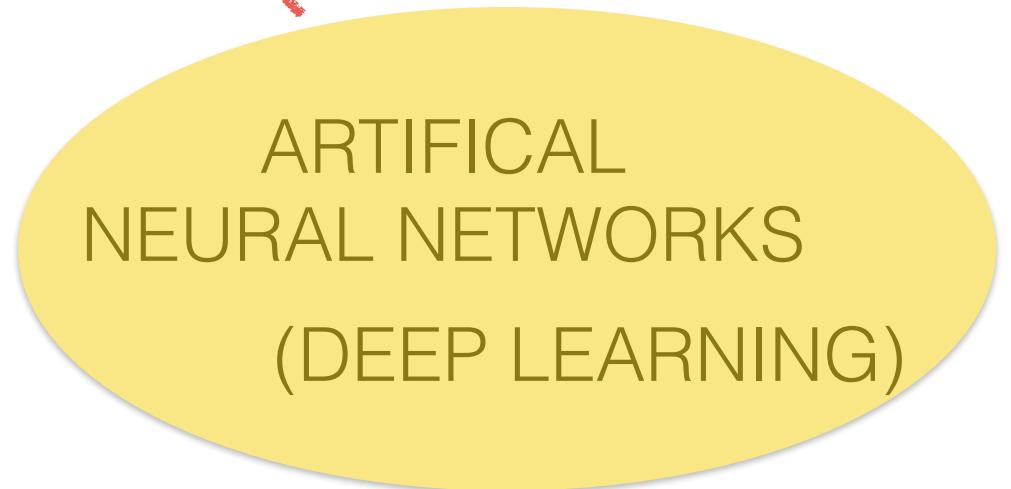
Different “classical” supervised machine learning methods



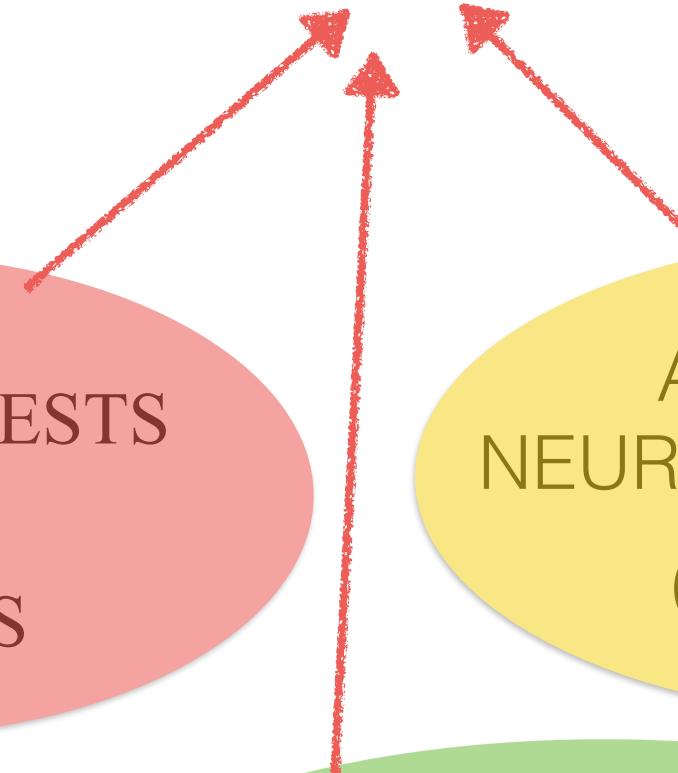
Differences are
in the function
that is used

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

All algorithms
minimize
a loss function



kernel algorithms



Differences are
in the function
that is used

Standard Chi2 model
fitting is a type
of machine learning ...
with a priori known
parameters

All algorithms
minimize
a loss function

RANDOM FORESTS

CARTS

decision trees

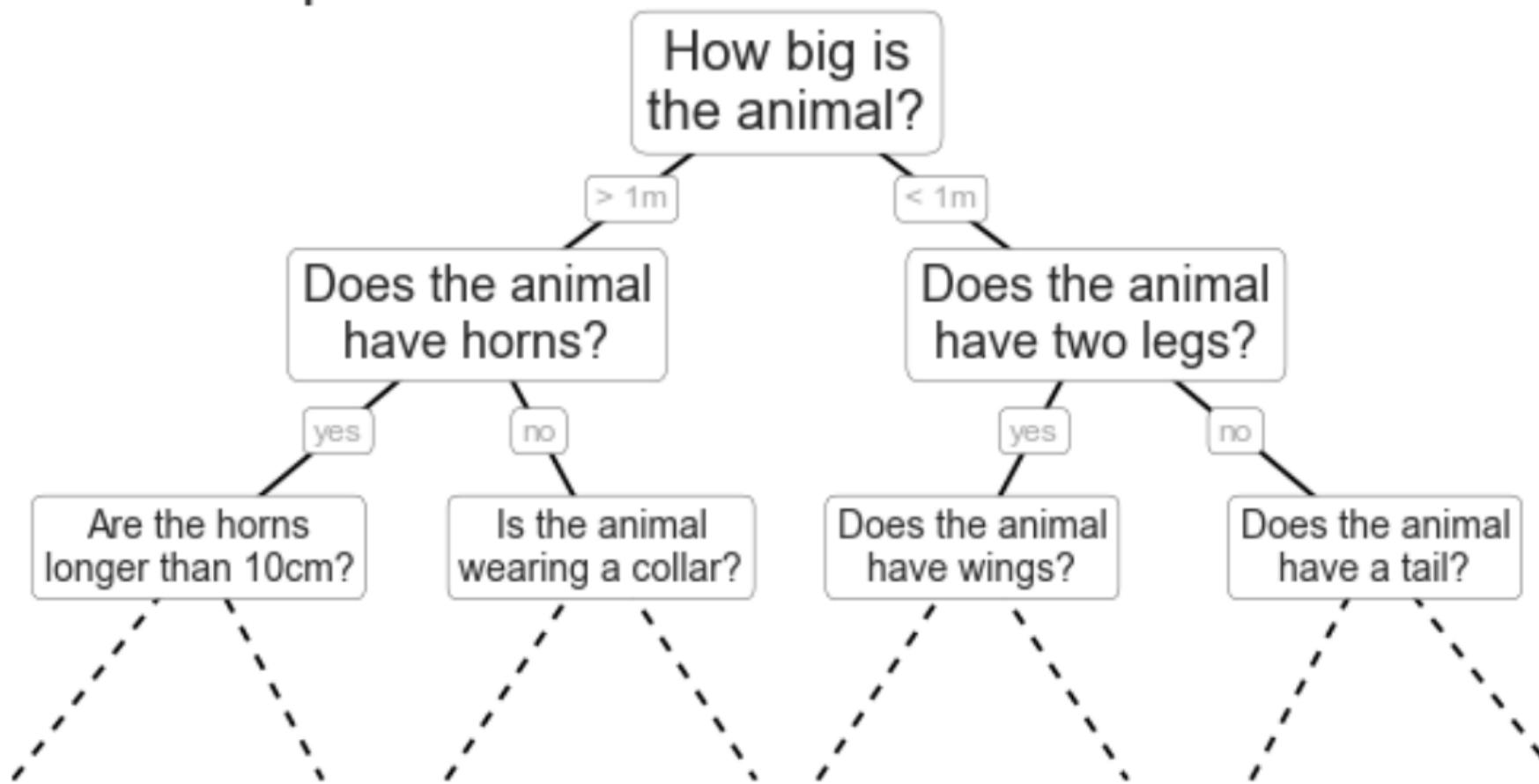
ARTIFICAL
NEURAL NETWORKS
(DEEP LEARNING)

SUPPORT VECTOR MACHINES

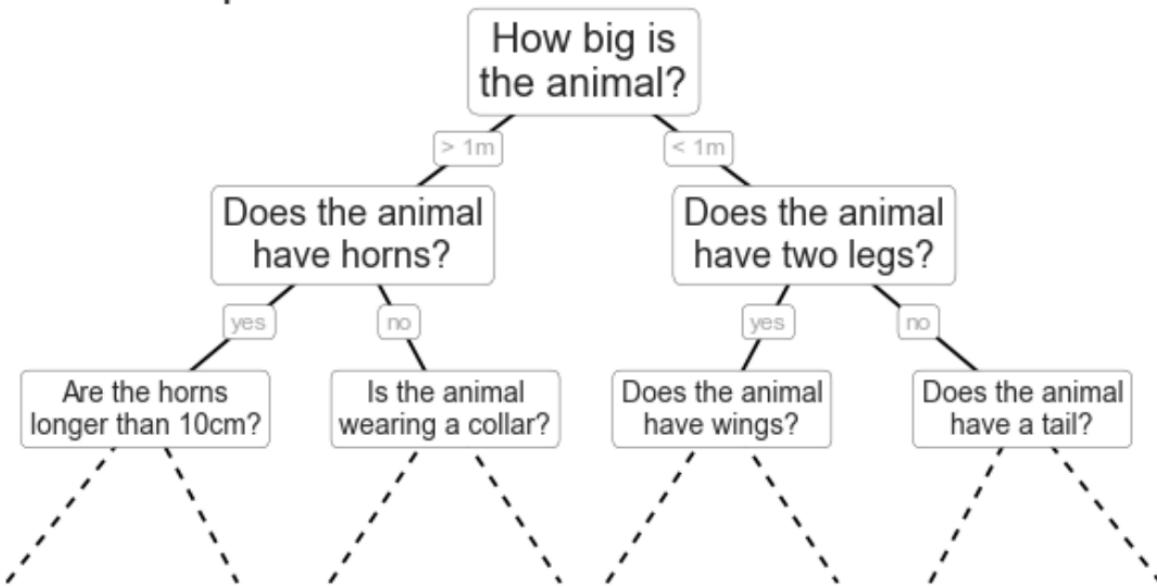
kernel algorithms

Classification/ Regression (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 SEPARATE INTO 2 GROUPS **[I.E. MINIMIZES SOME LOSS USING A METRIC]**
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 IN ONE CLASS]

Decision trees (CARTS)

TYPICAL METRICS USED:

GINI IMPURITY

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

Decision trees (CARTS)

TYPICAL METRICS 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 METRICS 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)

TYPICAL METRICS USED:

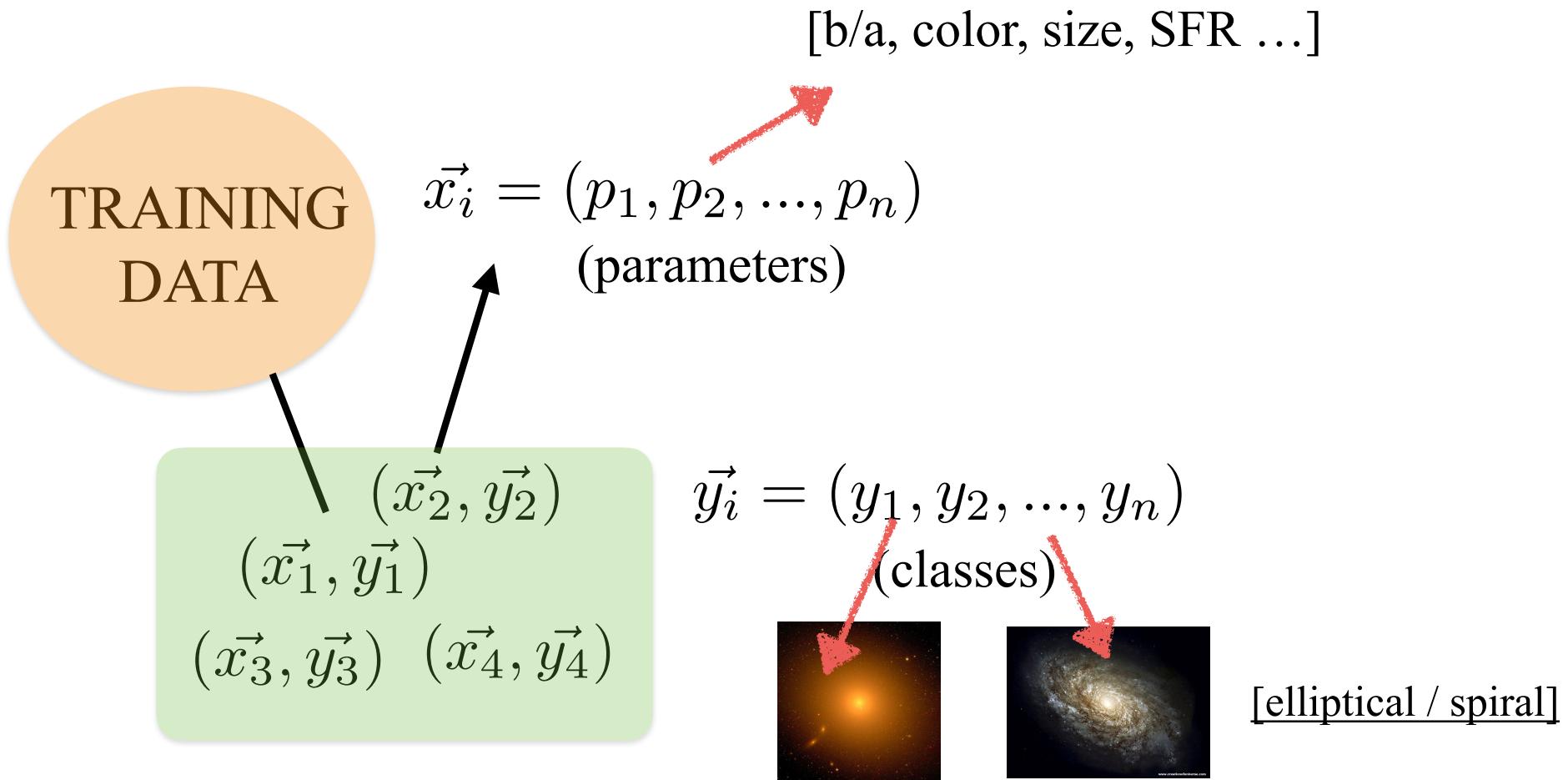
CROSS ENTROPY $E = - \sum_{j=1}^c p_j \log(p_j)$



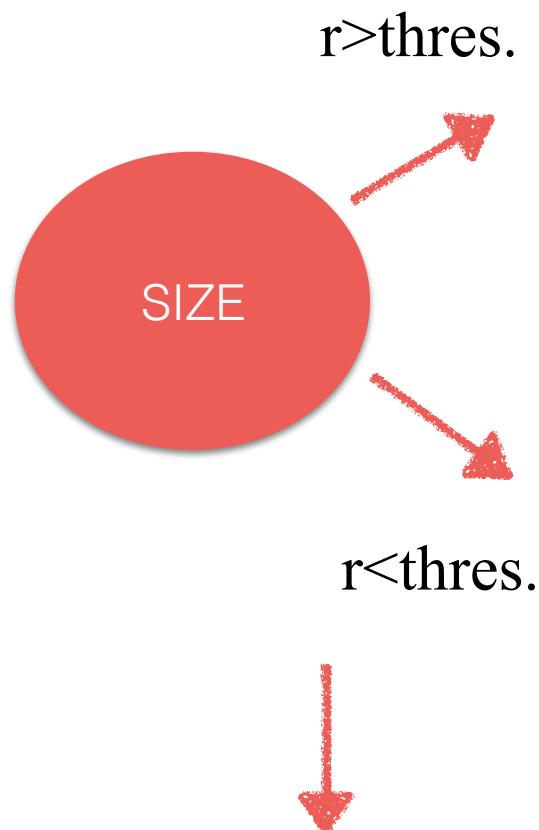
IF THERE IS ONLY ONE CLASS
THE CROSS ENTROPY GOES TO 0

fraction of
objects in each class given
a split value

Decision trees (CARTS)



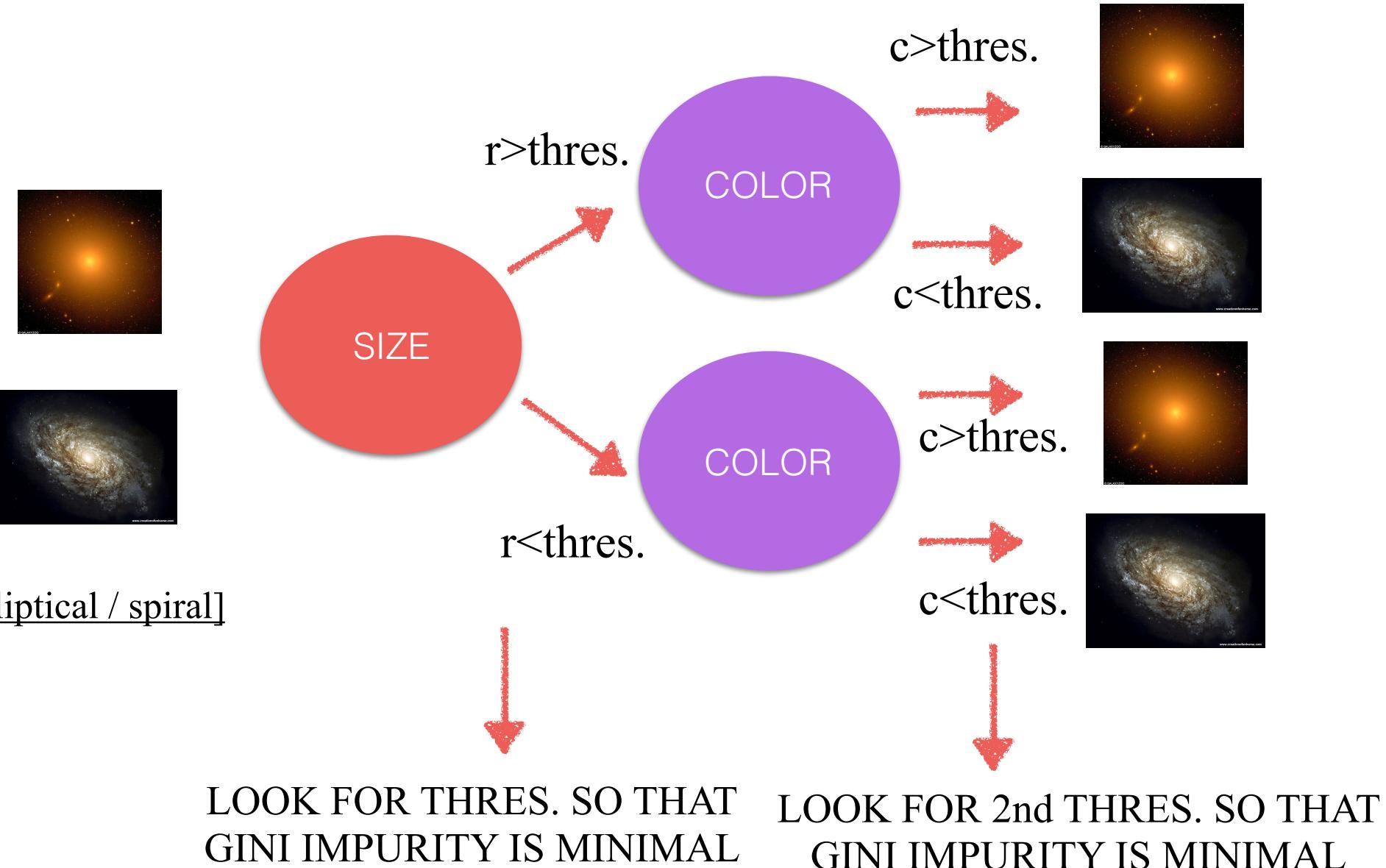
Decision trees (CARTS)



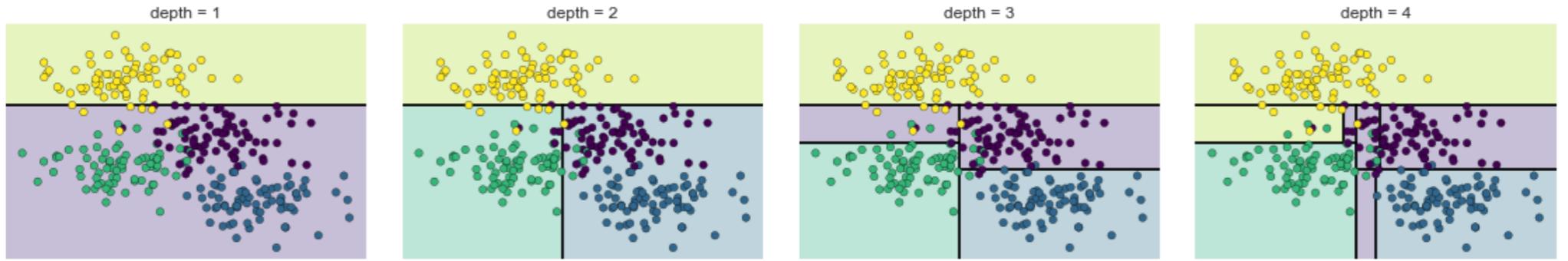
[elliptical / spiral]

LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

Decision trees (CARTS)



Decision trees (CARTS)



AT THE IT IS SIMPLY A PARTITION OF THE PARAMETER SPACE
WITH CONSTANT BOUNDARIES

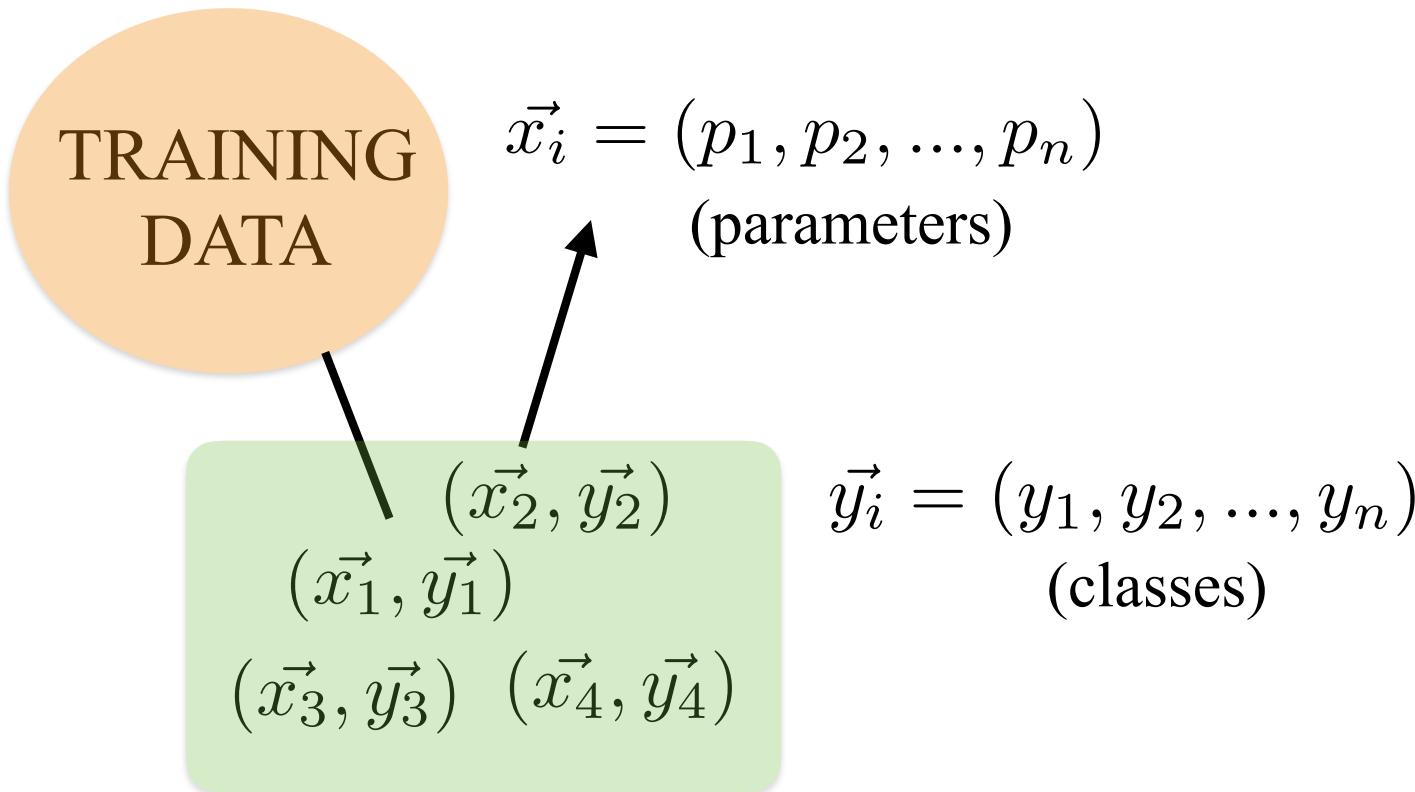
Random Forests

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

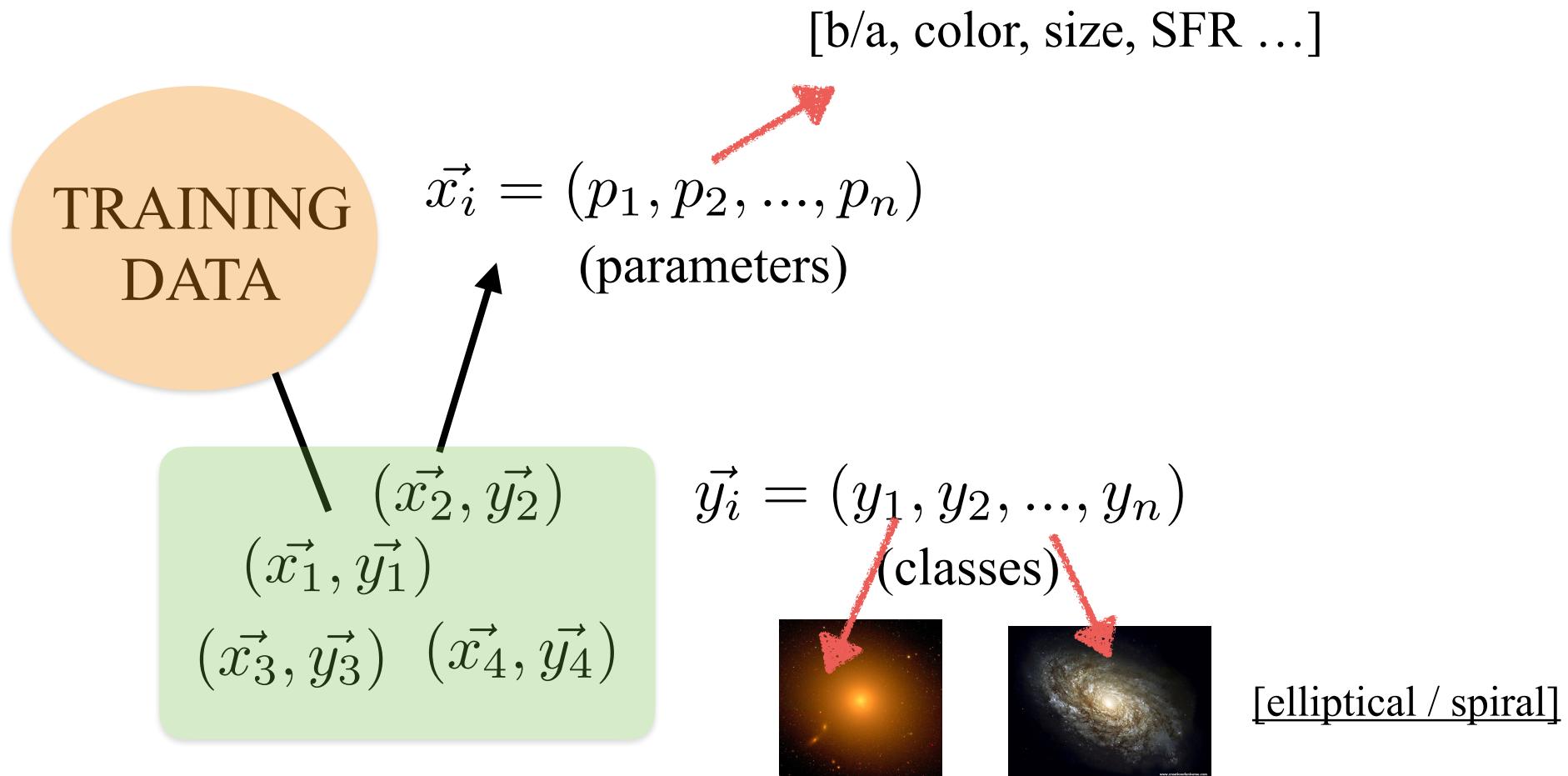
Random Forests

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



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

(\vec{x}_2, \vec{y}_2)
 (\vec{x}_1, \vec{y}_1)
 (\vec{x}_3, \vec{y}_3) (\vec{x}_4, \vec{y}_4)

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



[elliptical / spiral]

SIZE

$r > \text{thres.}$

$r < \text{thres.}$



COLOR

$c > \text{thres.}$



$c < \text{thres.}$



$c > \text{thres.}$



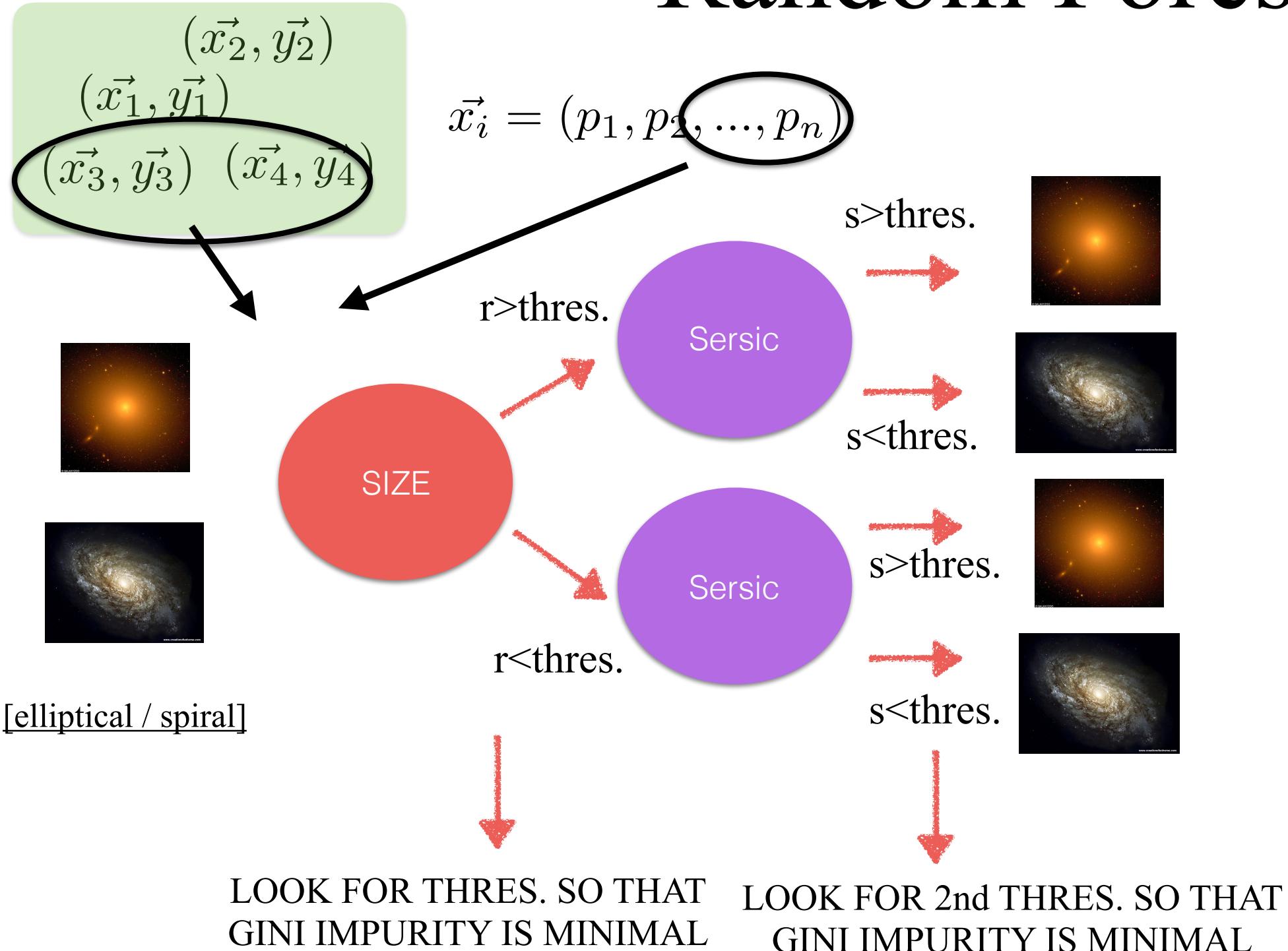
$c < \text{thres.}$



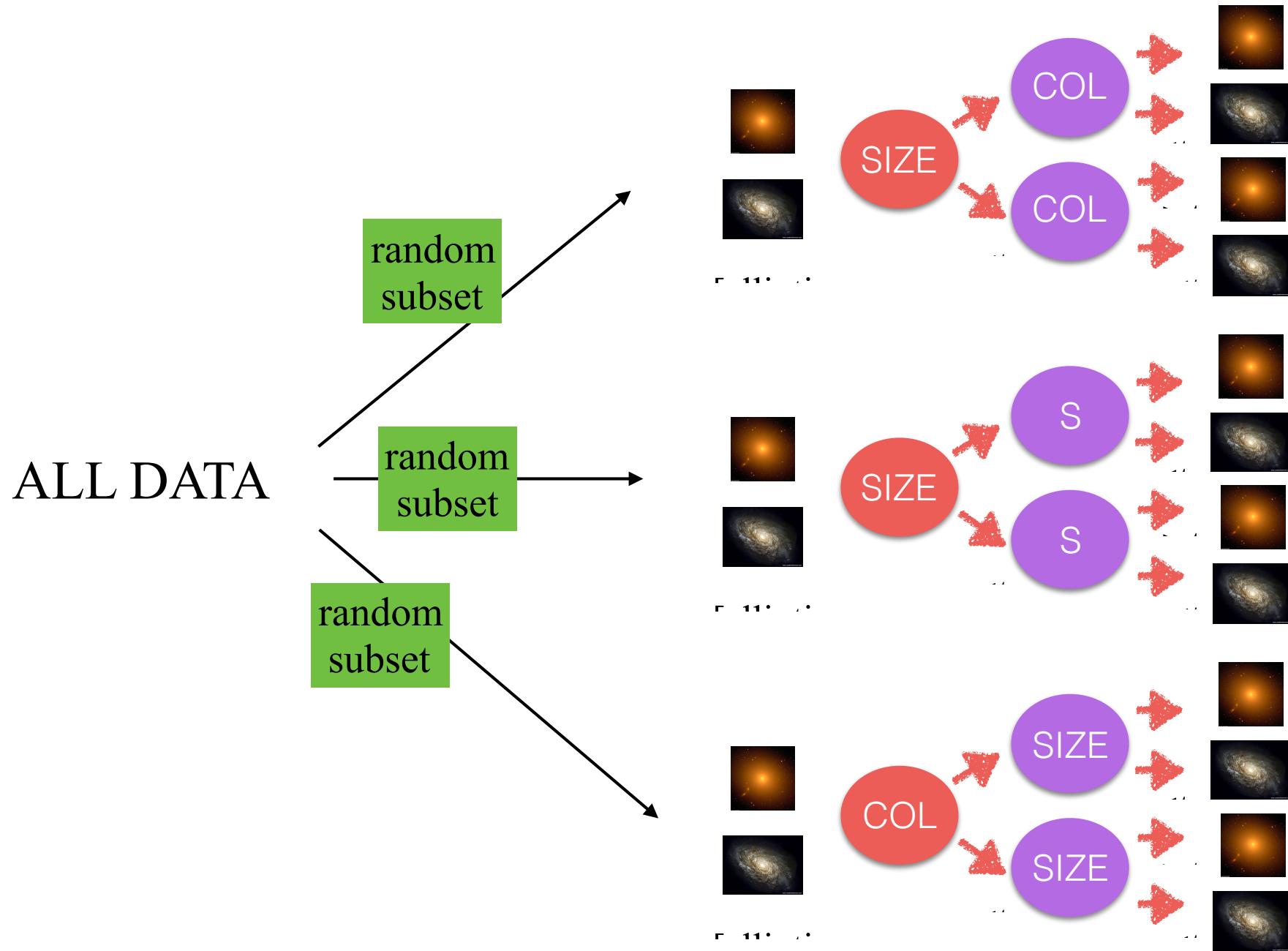
LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

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

Random Forests

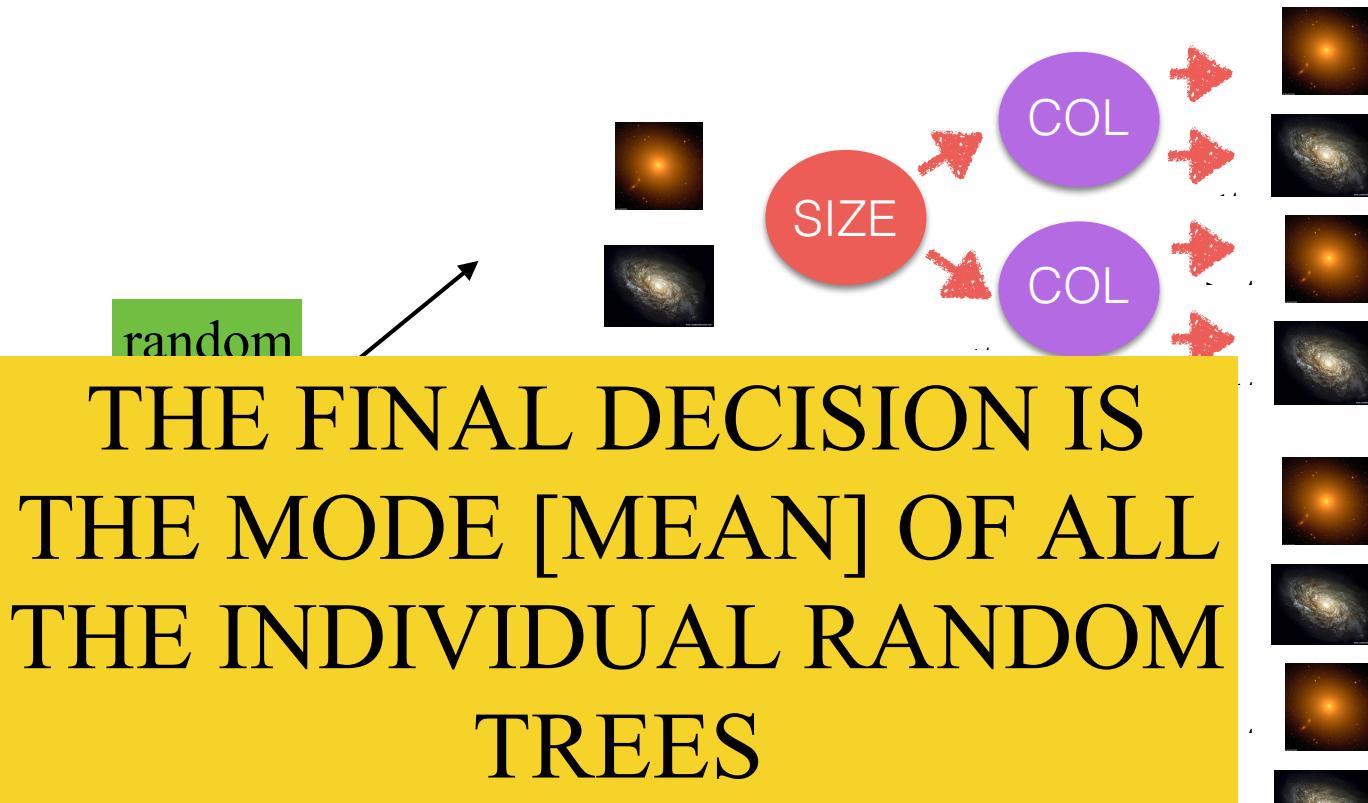


Random Forests



Random Forests

ALL DATA



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

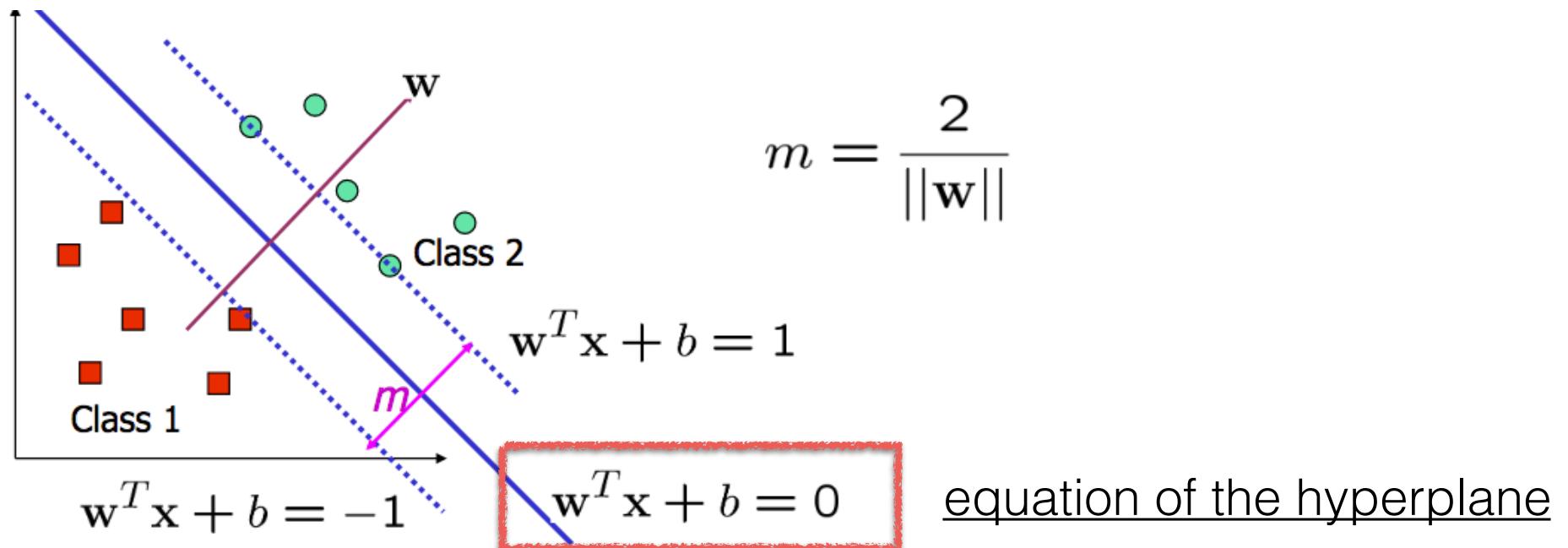
ONE KEY ADVANTAGE OF DECISION TREE ALGORITHMS IS THAT THEY ARE VERY EASY TO INTERPRET

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

| Rank | Property | AUC | Success label ^a |
|------|--------------------|-----------------|----------------------------|
| 1 | ALL | 0.9074 ± 0.0106 | Outstanding |
| 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 |

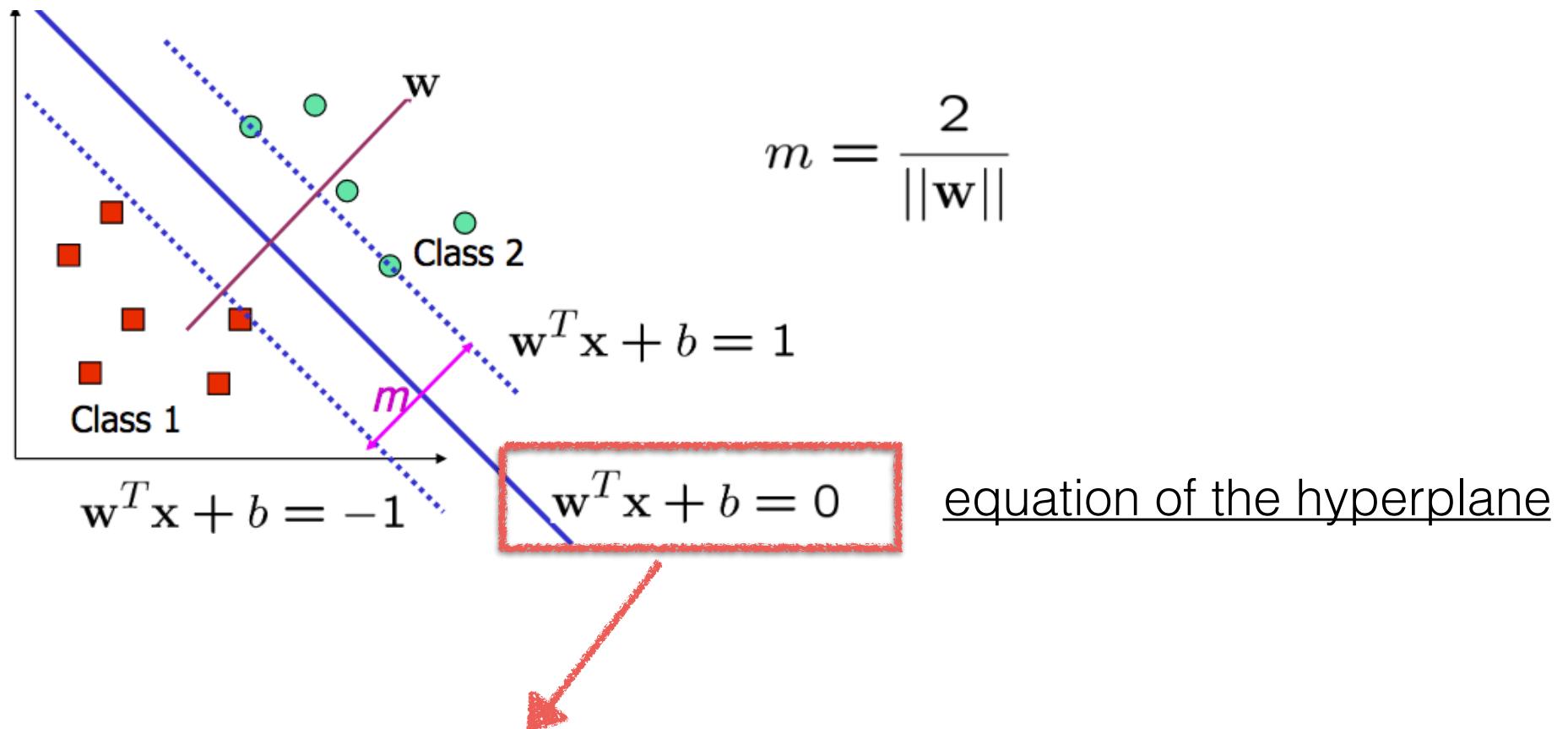
IMPORTANCE OF
PARAMETERS TO PREDICT
IF A GALAXY IS QUENCHED

Support Vector Machines



[Credit: M. Law]

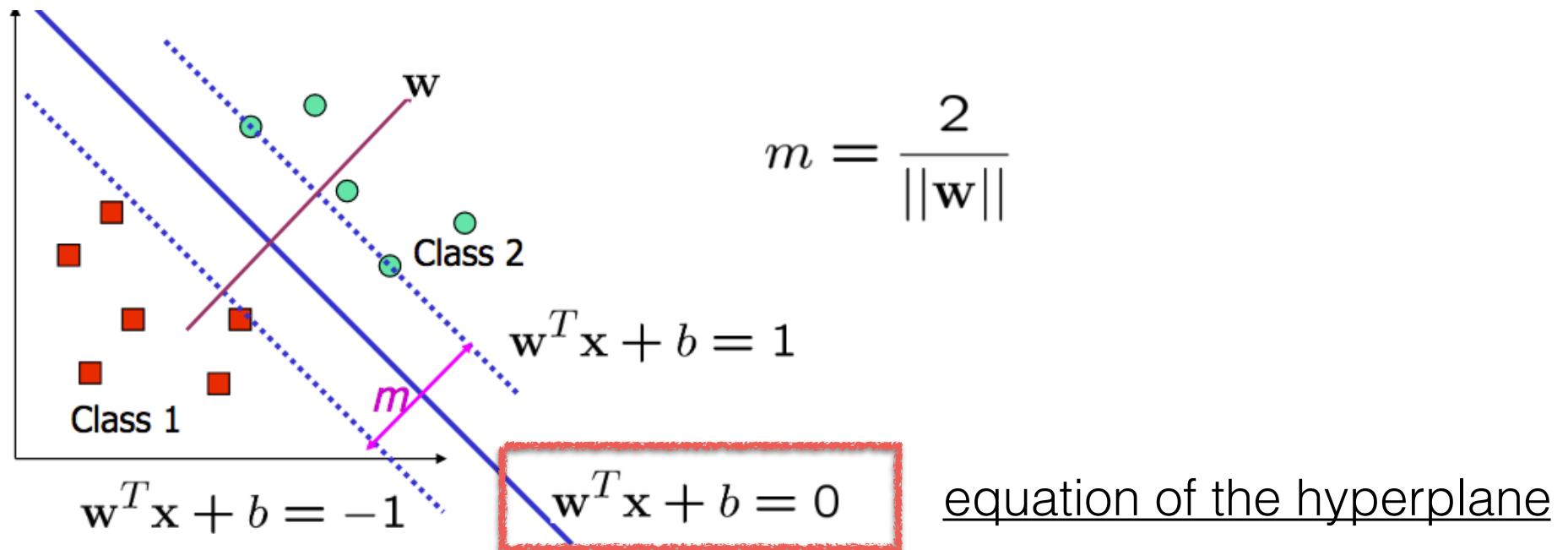
Support Vector Machines



THE GAME HERE IS TO FIND THE W THAT
MAXIMIZES THE MARGIN [MINIMIZE $\|w\|/2$]

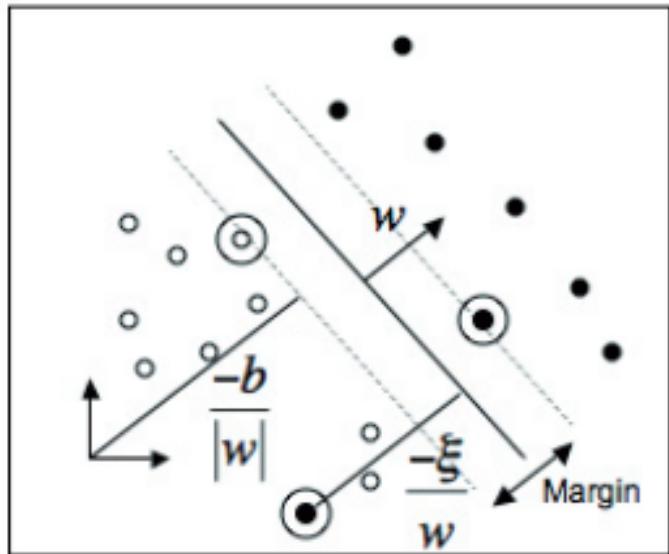
[Credit: M. Law]

Support Vector Machines



IF THE PROBLEM IS LINEARLY SEPARABLE, THERE IS
AN IDEAL SOLUTION [see figure]

Support Vector Machines



IN MOST CASES THE DATA
ARE NON LINEARLY
SEPARABLE

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \epsilon_i$$

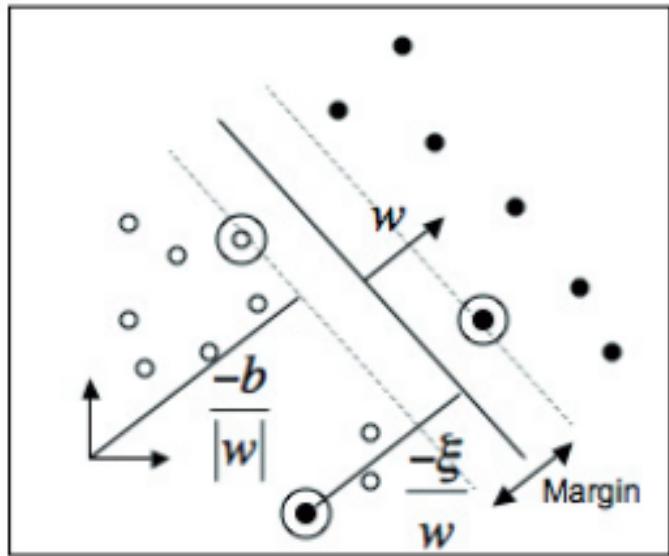
$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 + \epsilon_i$$

Huertas-Company+08

THE CONDITION IS RELAXED BY ADDING A POSITIVE
STACK VARIABLE

Huertas-Company+08

Support Vector Machines



IN MOST CASES THE DATA
ARE NON LINEARLY
SEPARABLE

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \epsilon_i$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 + \epsilon_i$$

Huertas-Company+08

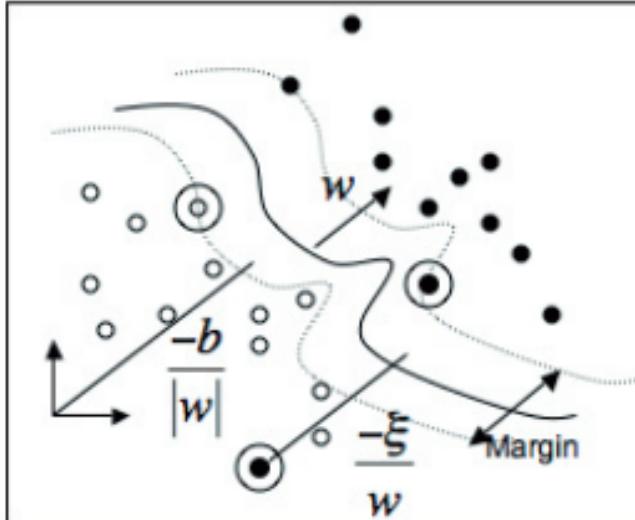
THE CONDITION IS RELAXED BY ADDING A POSITIVE
STACK VARIABLE

THE FUNCTION TO MINIMIZE:

$$\frac{\|\mathbf{w}\|^2}{2} + T[\sum \epsilon_i]$$

THE LARGER T, THE
MORE ERRORS
ARE PENALIZED

Support Vector Machines



THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

WE MAP THE PLANE INTO
A NEW EUCLIDIAN SPACE
WHERE THE POINTS ARE
SEPARABLE

$$\Phi : \mathbb{R} \rightarrow H$$

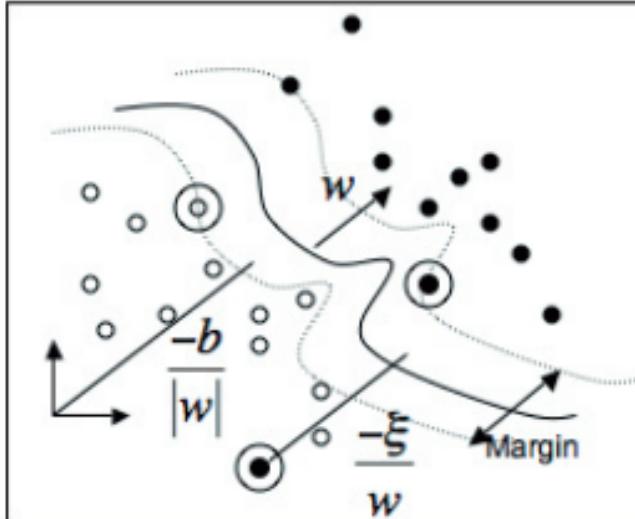
SINCE WE ONLY NEED
TO COMPUTE INNER
PRODUCTS IN THIS
SPACE

$$x_i \cdot x_j$$

THE TRAINING
ALGORITHM WOULD
ONLY DEPEND ON THE
DATA THROUGH THE
PRODUCTS IN H

$$\Phi(x_i) \cdot \Phi(x_j)$$

Support Vector Machines



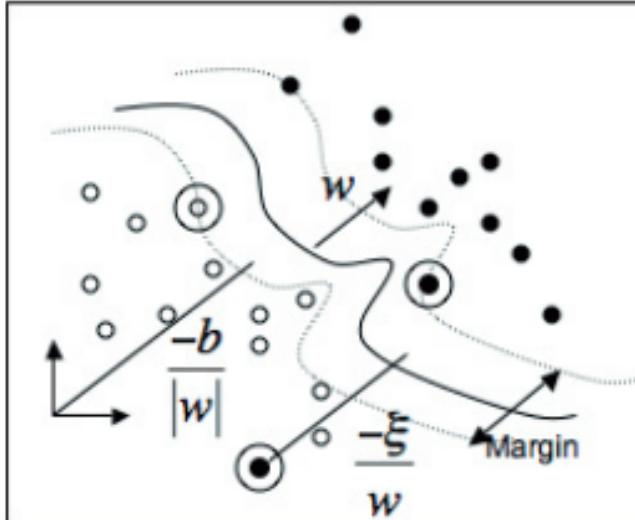
THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

IF THERE IS A **KERNEL FUNCTION** SO THAT:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

THEN THERE IS NO NEED TO EXPLICITLY
KNOW Φ

Support Vector Machines



THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

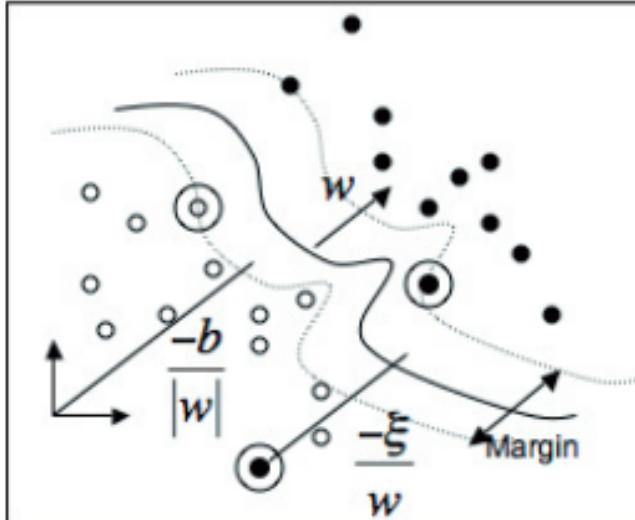
IF THERE IS A **KERNEL FUNCTION** SO THAT:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

THEN THERE IS NO NEED TO EXPLICITLY
KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

Support Vector Machines



THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

IF THERE IS A **KERNEL FUNCTION** SO THAT:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

THEN THERE IS NO NEED TO EXPLICITLY
KNOW Φ

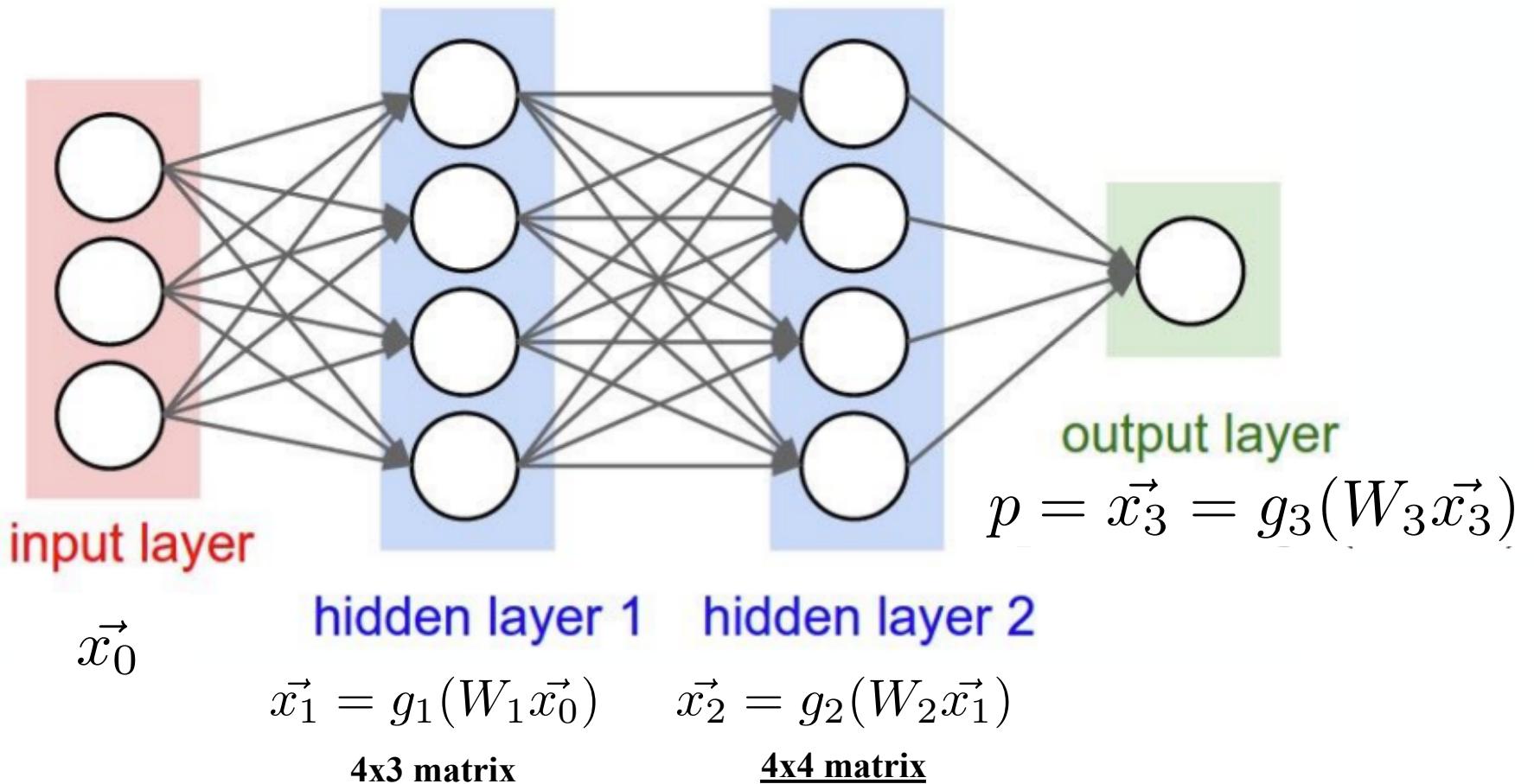
SVM IS CALLED **A KERNEL METHOD**

EXAMPLE OF KERNEL:

$$K(x, y) = e^{-g||x-y||^2}$$

[GAUSSIAN RBF]

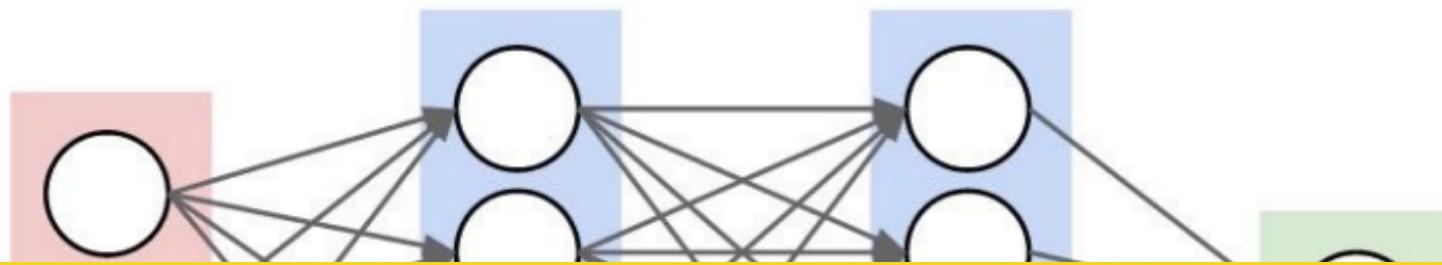
Neural Networks



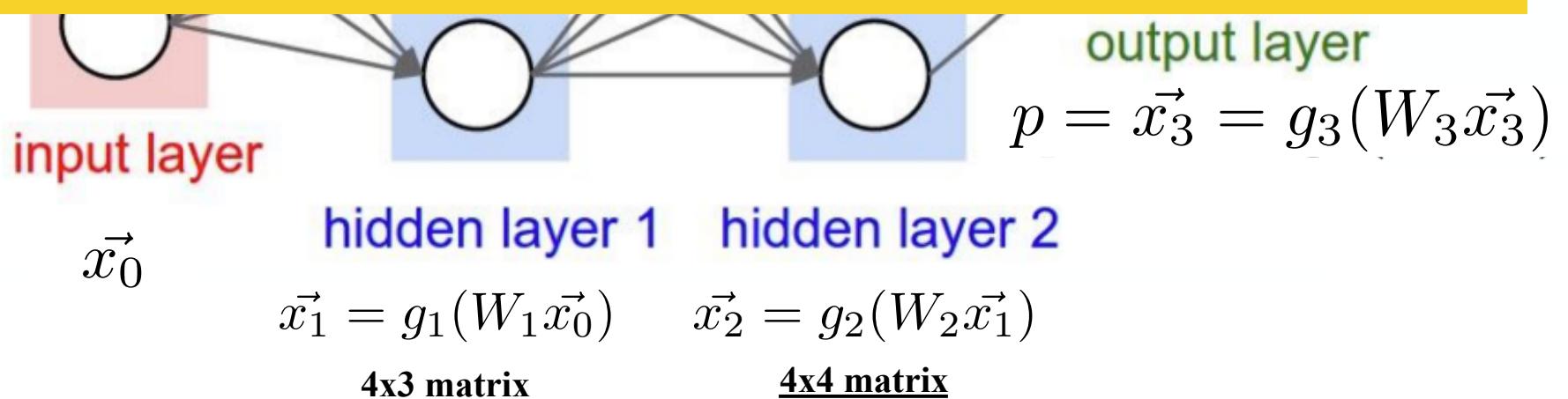
W matrices are called “weights”

The functions g are called “activations functions”

Neural Networks



THESE ARE THE SEED OF DEEP LEARNING.
WE WILL EXPLAIN THEM IN DETAIL LATER ON.



W matrices are called “weights”
The functions g are called “activations functions”

In practice: *Python scikit learn*

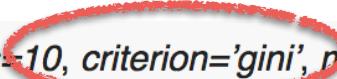
- All these different methods are standard and available in Python
- Very easy to use
- All info here

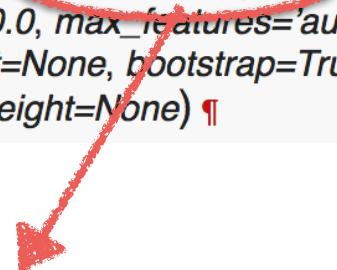
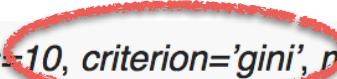
sklearn.ensemble.RandomForestClassifier

```
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

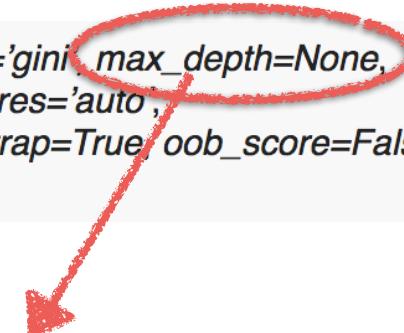
sklearn.ensemble.RandomForestClassifier

```
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

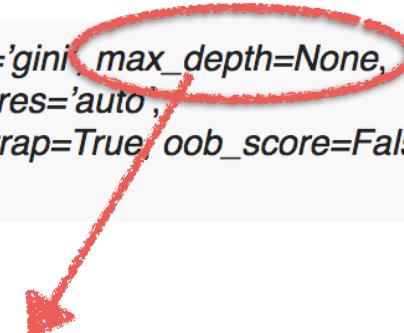
```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
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,
```

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

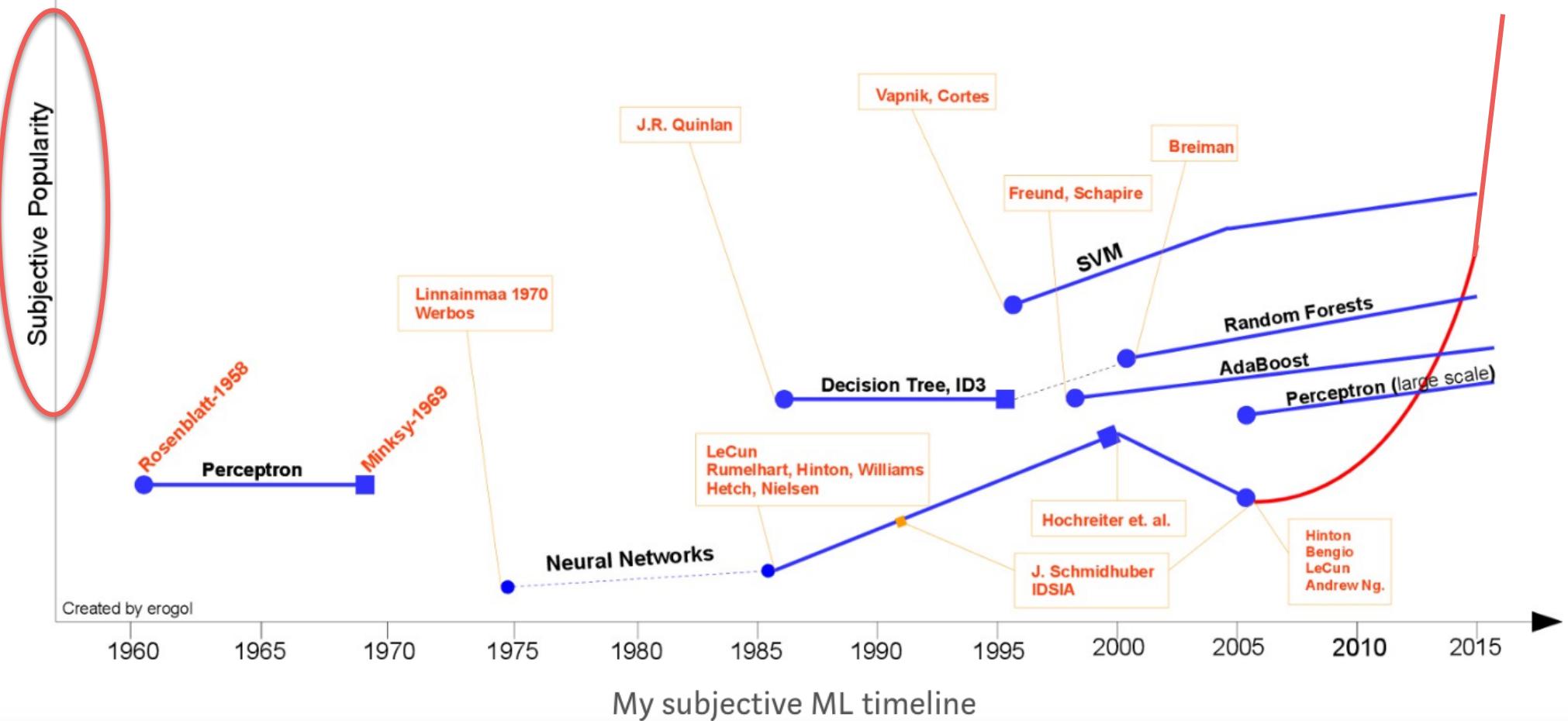
```
>>> print(clf.predict([[0, 0, 0, 0]]))  
[1]
```

How to choose your classical classifier?

NO RULE OF THUMB - REALLY DEPENDS ON APPLICATION

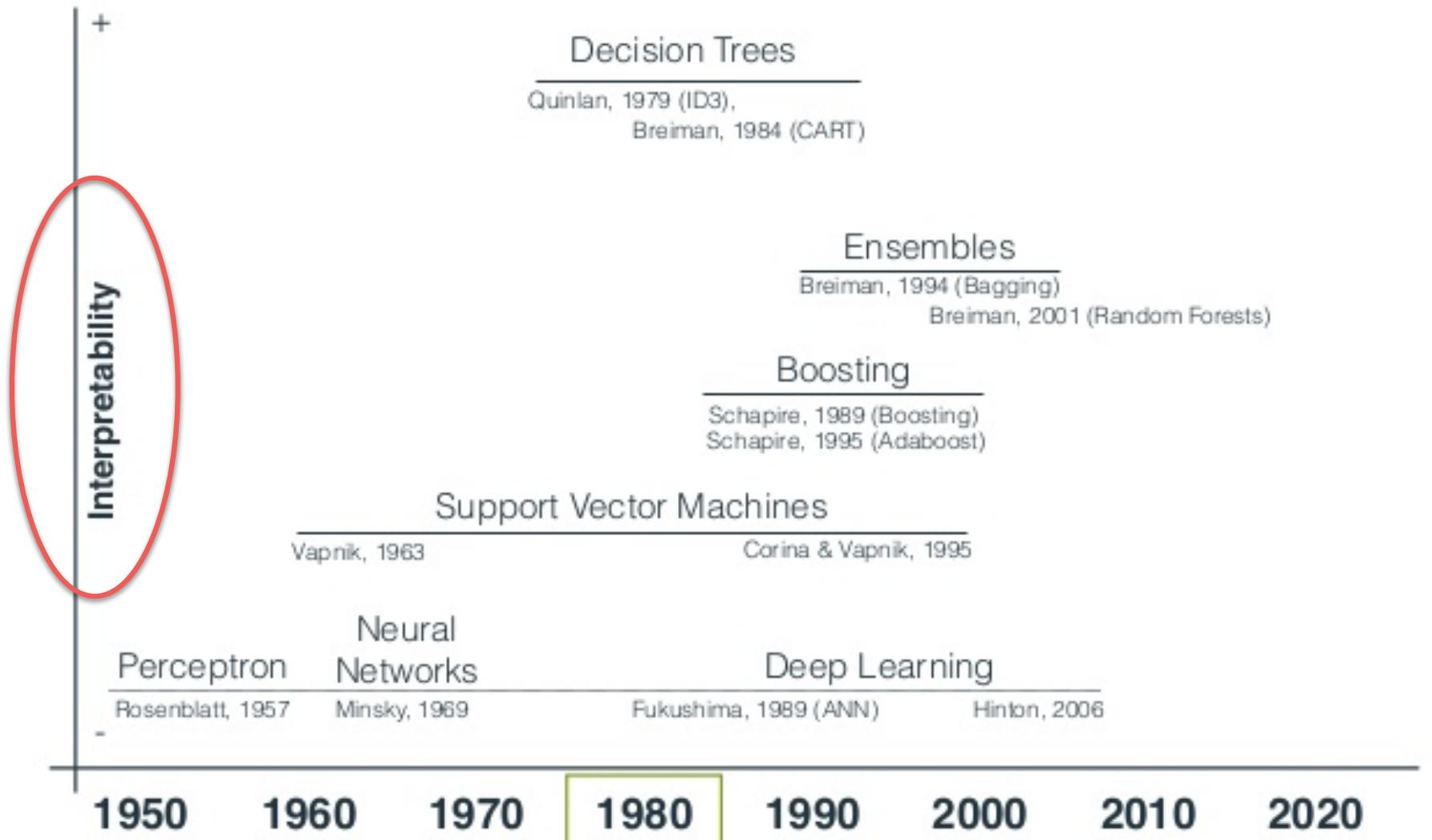
| ML METHOD | ++ | - | Python |
|------------------------------|---|---|---|
| CARTS / RANDOM FOREST | Easy to interpret (“White box”) Litte data preparation Both numerical + categorical | Over-complex trees Unstable Biased tress if some classes dominate | sklearn.ensemble.RandomForestClassifier sklearn.ensemble.RandomForestRegressor |
| 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.MLPClassifier sklearn.neural_network.MLPRegressor |

ALSO INFLUENCED BY “MAINSTREAM” TRENDS



Source

CAN DEPEND ON YOUR MAIN INTEREST



credit

Evaluation of results

- 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..

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE
TOTAL ACCURACY

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

true positives true negatives



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

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

MEASURES HOW MANY OBJECTS ARE CORRECTLY
CLASSIFIED

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.

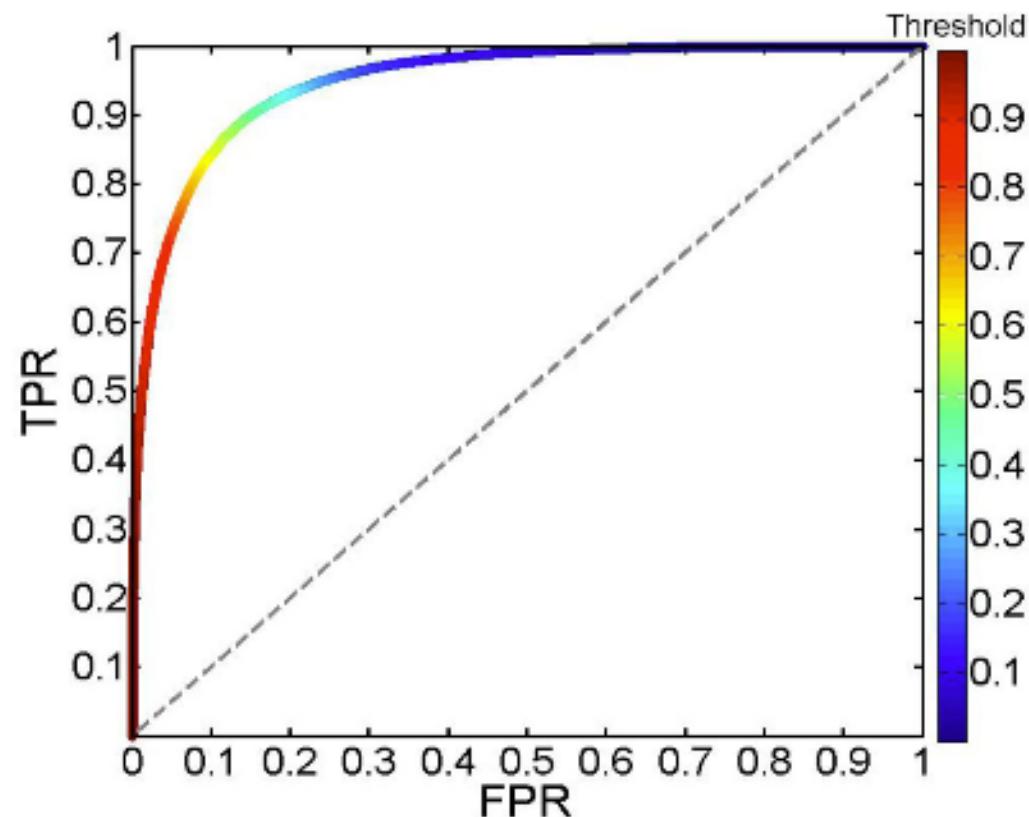
$$FPR = \frac{FP}{FP + TN}$$

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

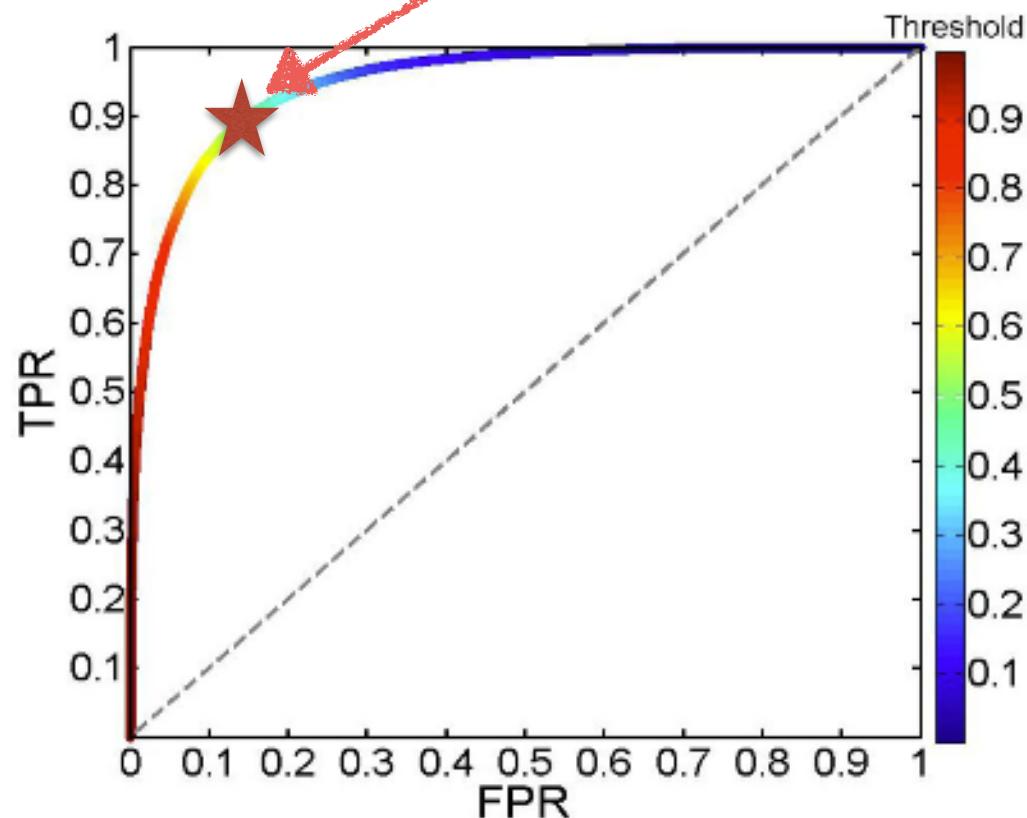
ROC CURVE



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

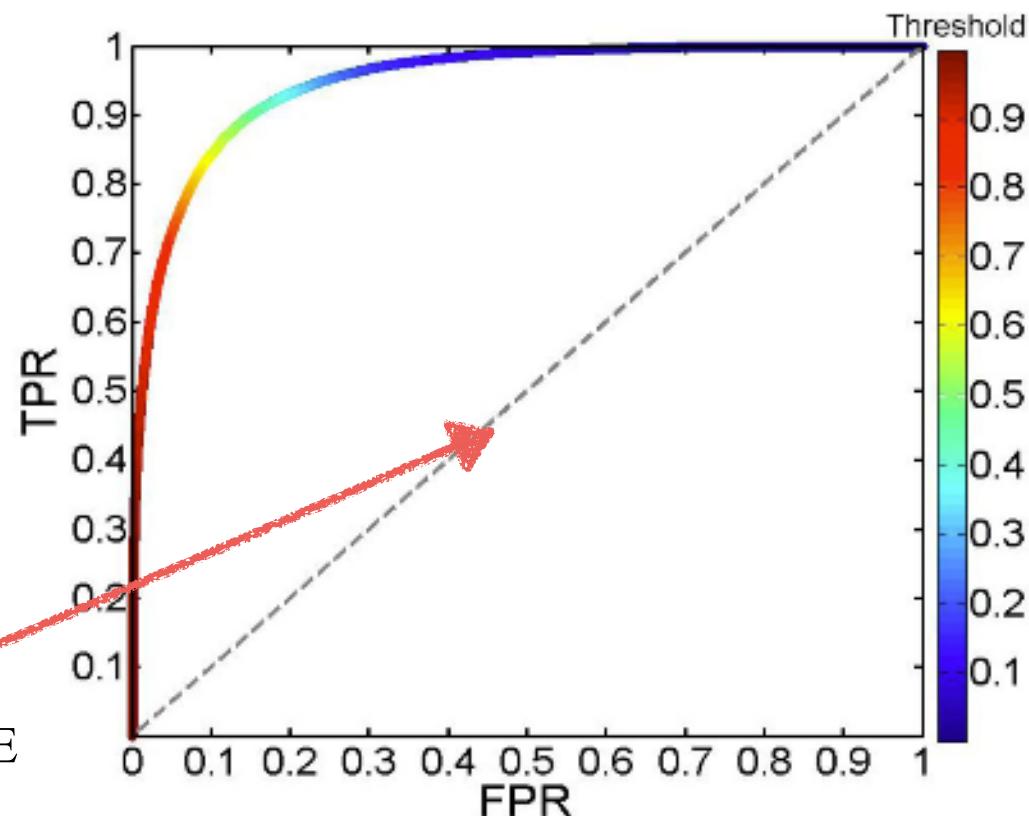
EACH POINT HERE SHOWS THE VALUES OF TPR AND
FPR FOR A GIVEN THRESHOLD

ROC CURVE



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

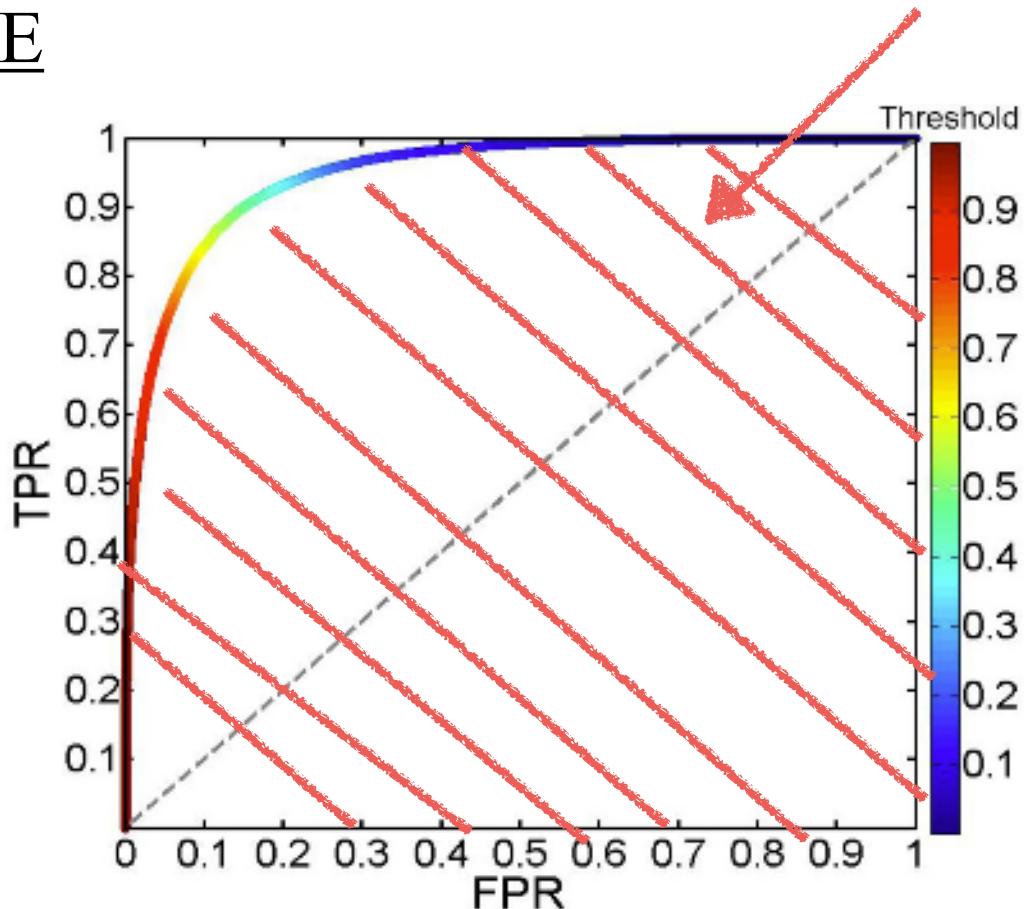
ROC CURVE



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)

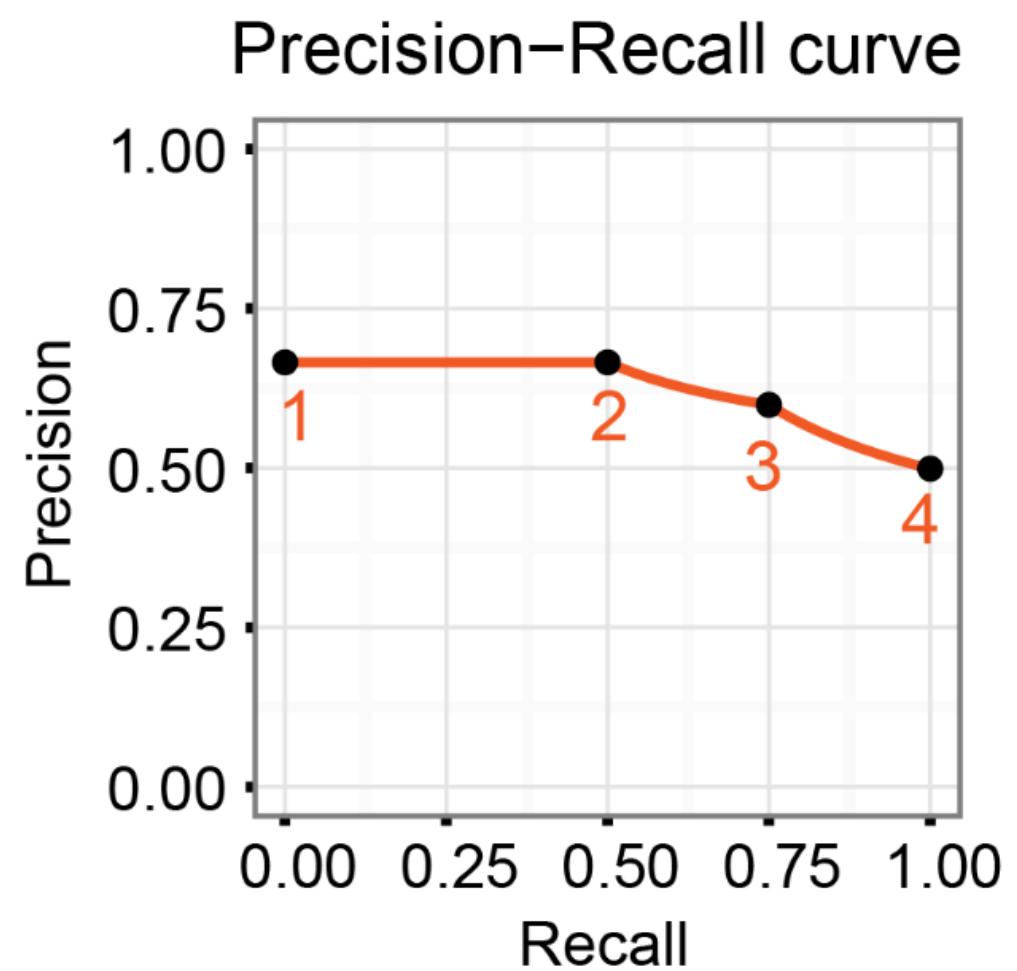
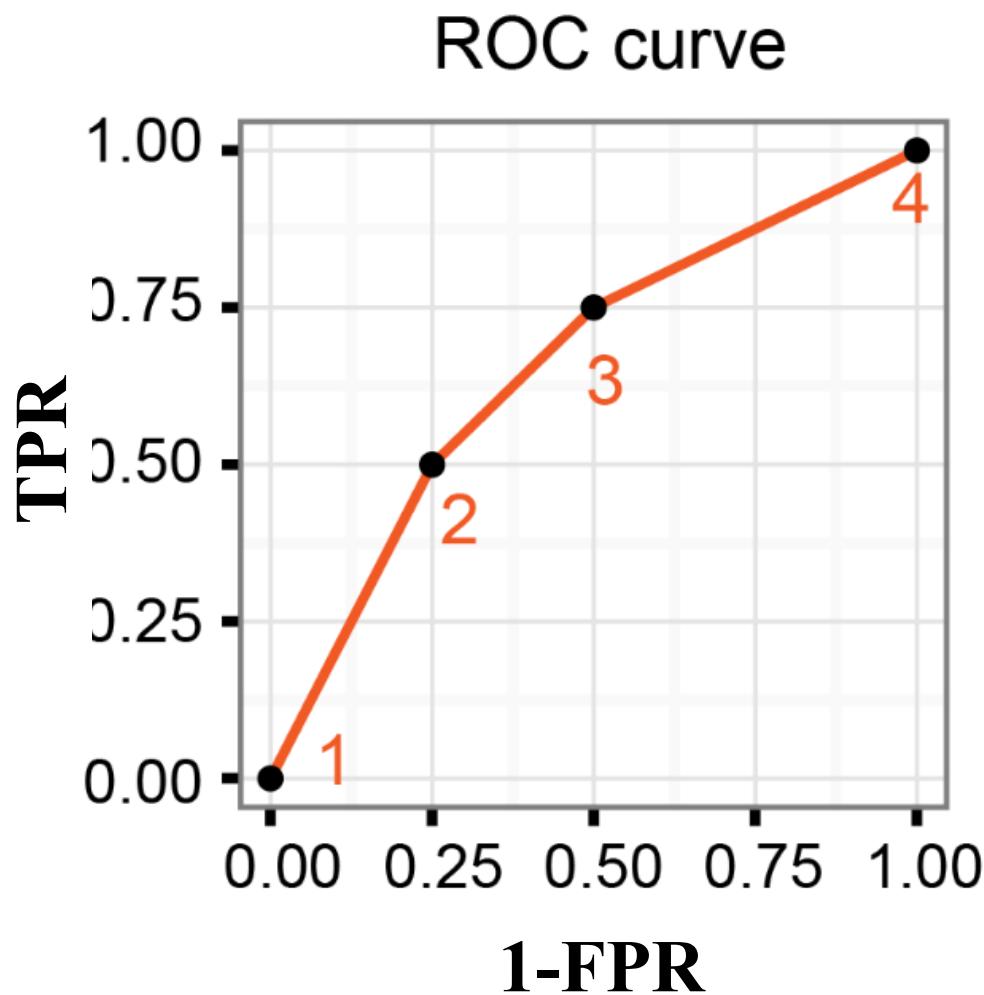
$$Recall = \frac{TP}{TP + FN} = TPR \quad [\text{completeness}]$$

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



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) = $\frac{\text{LR+}}{\text{LR-}}$ | $F_1 \text{ score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$ |
| | 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-) = $\frac{\text{FNR}}{\text{TNR}}$ | | |

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 VETY 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.

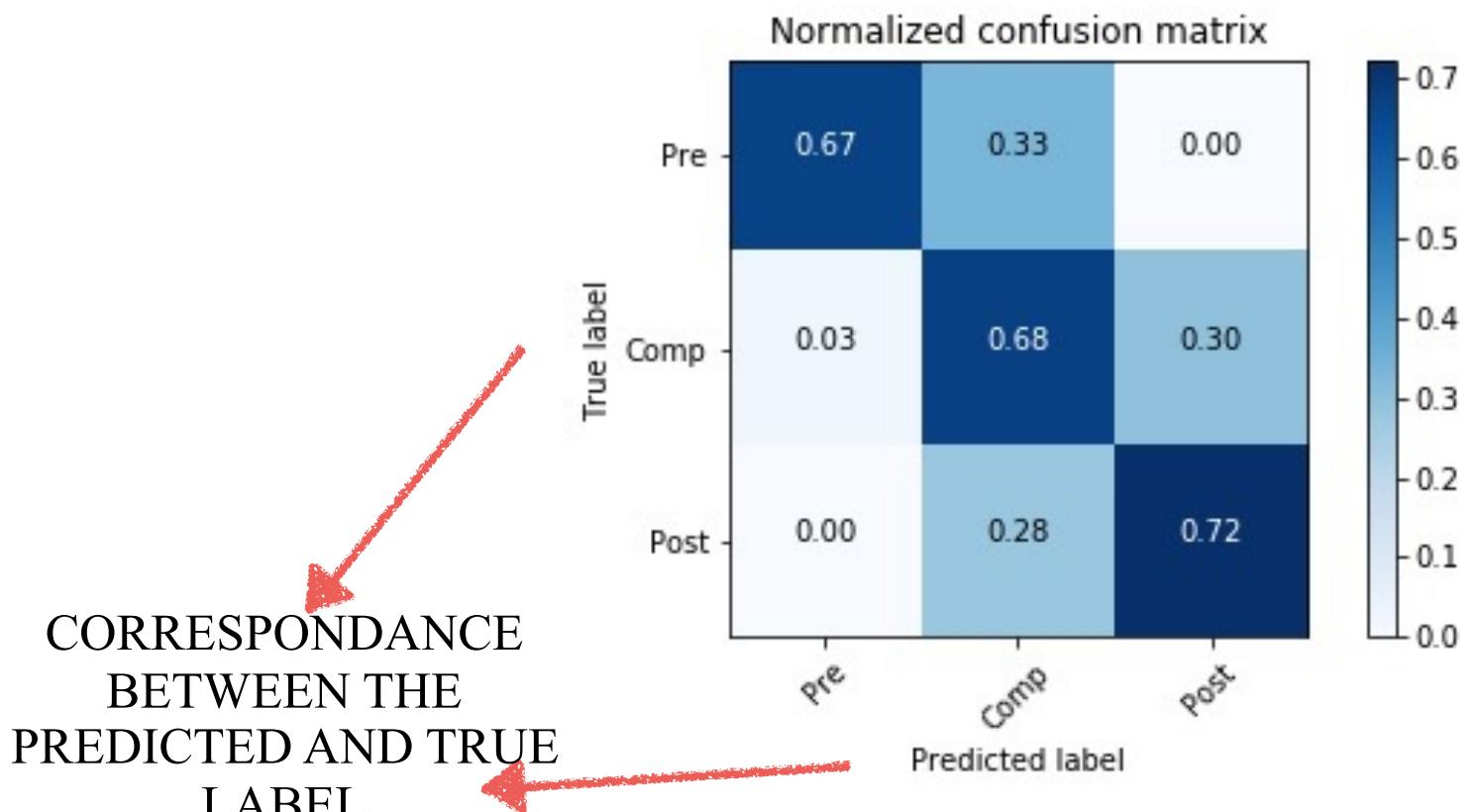
The recall is the ratio $\text{tp} / (\text{tp} + \text{fn})$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

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.

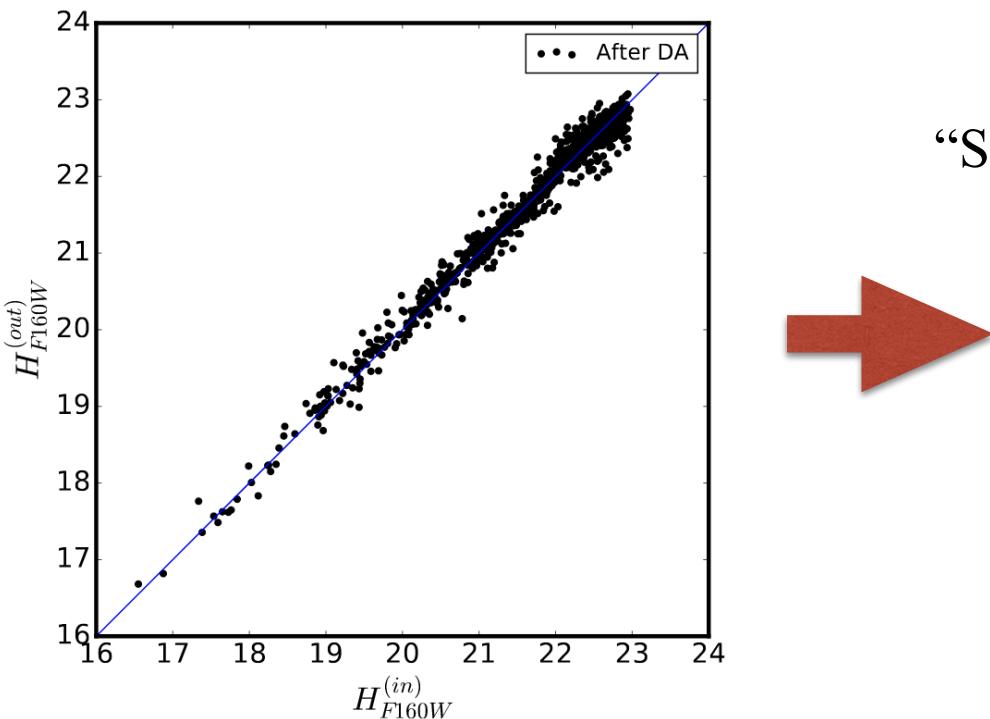
Evaluation of results

[multi-class]

CONFUSION MATRIX



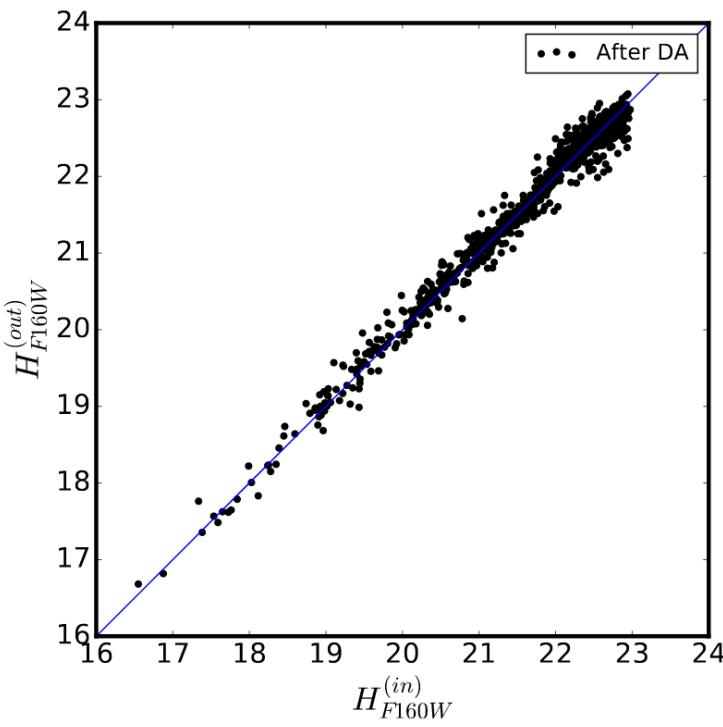
Evaluation of results [regression]



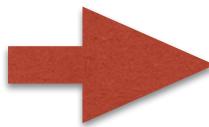
FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS



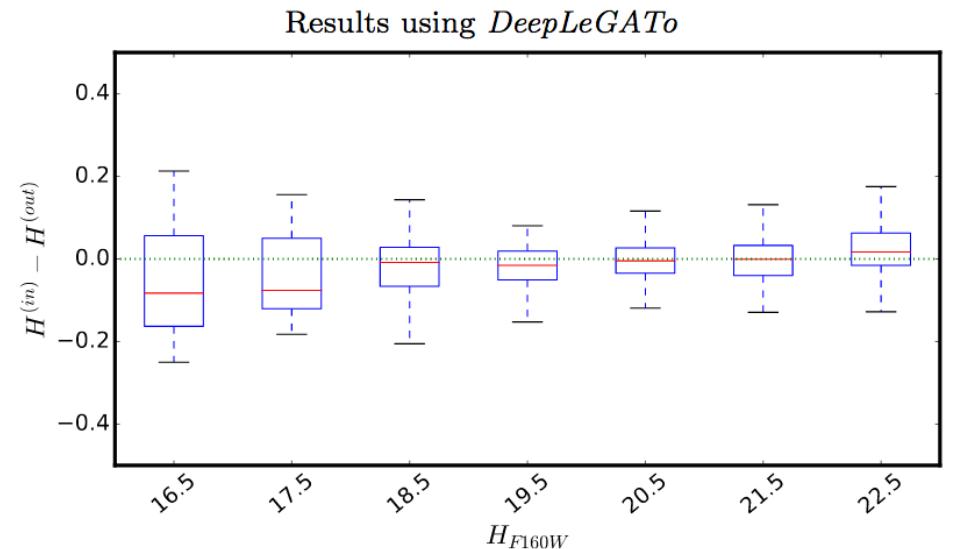
Evaluation of results [regression]



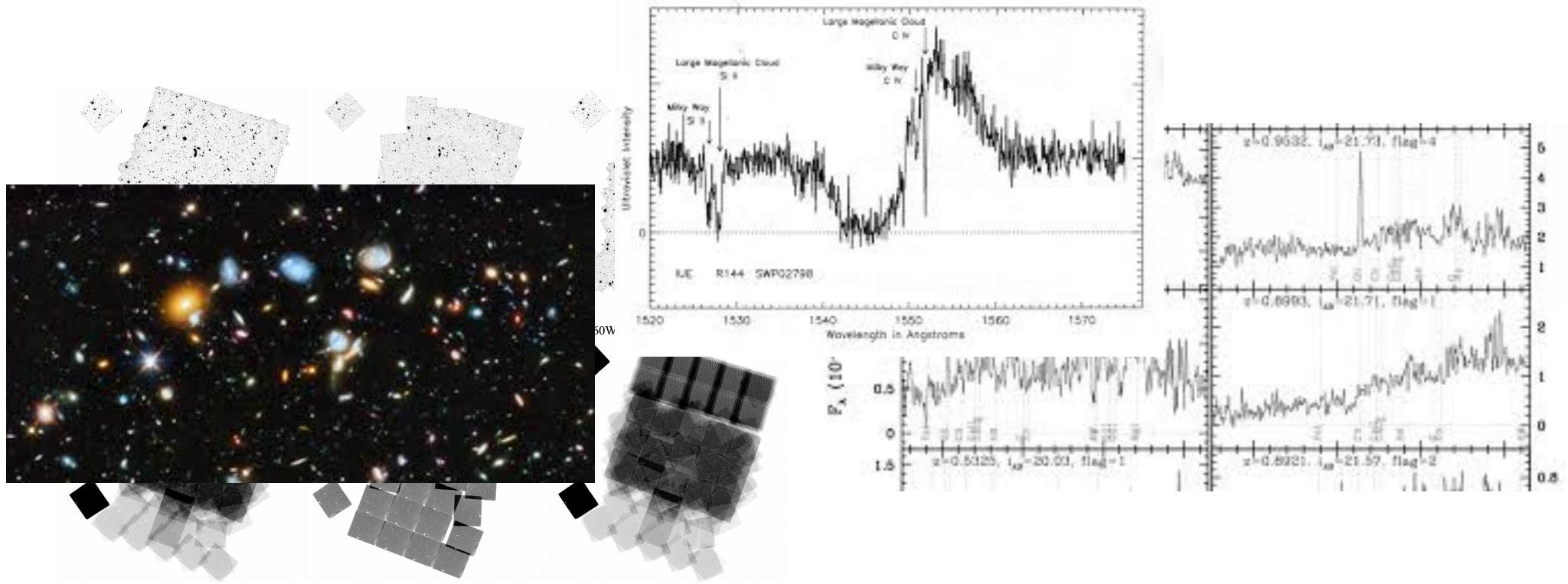
FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS



BIAS, SCATTER ... YOU KNOW!

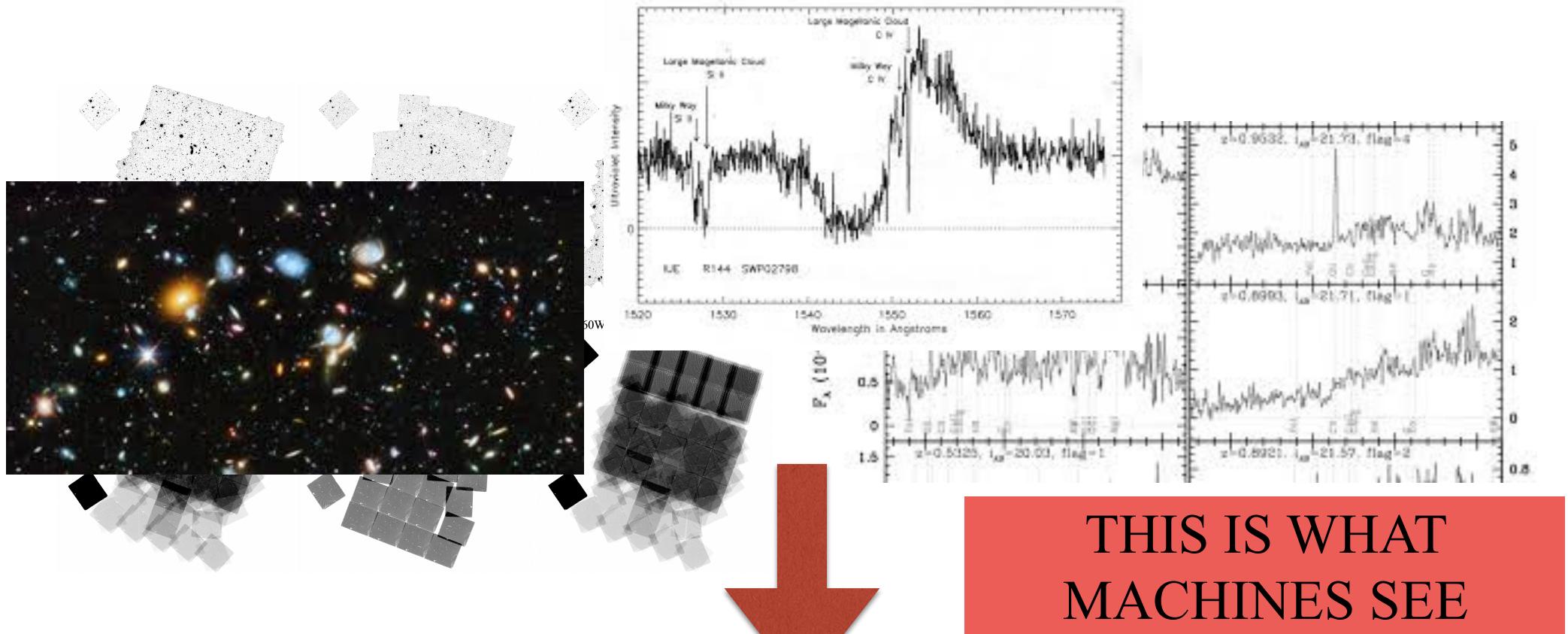


What do we put as input?



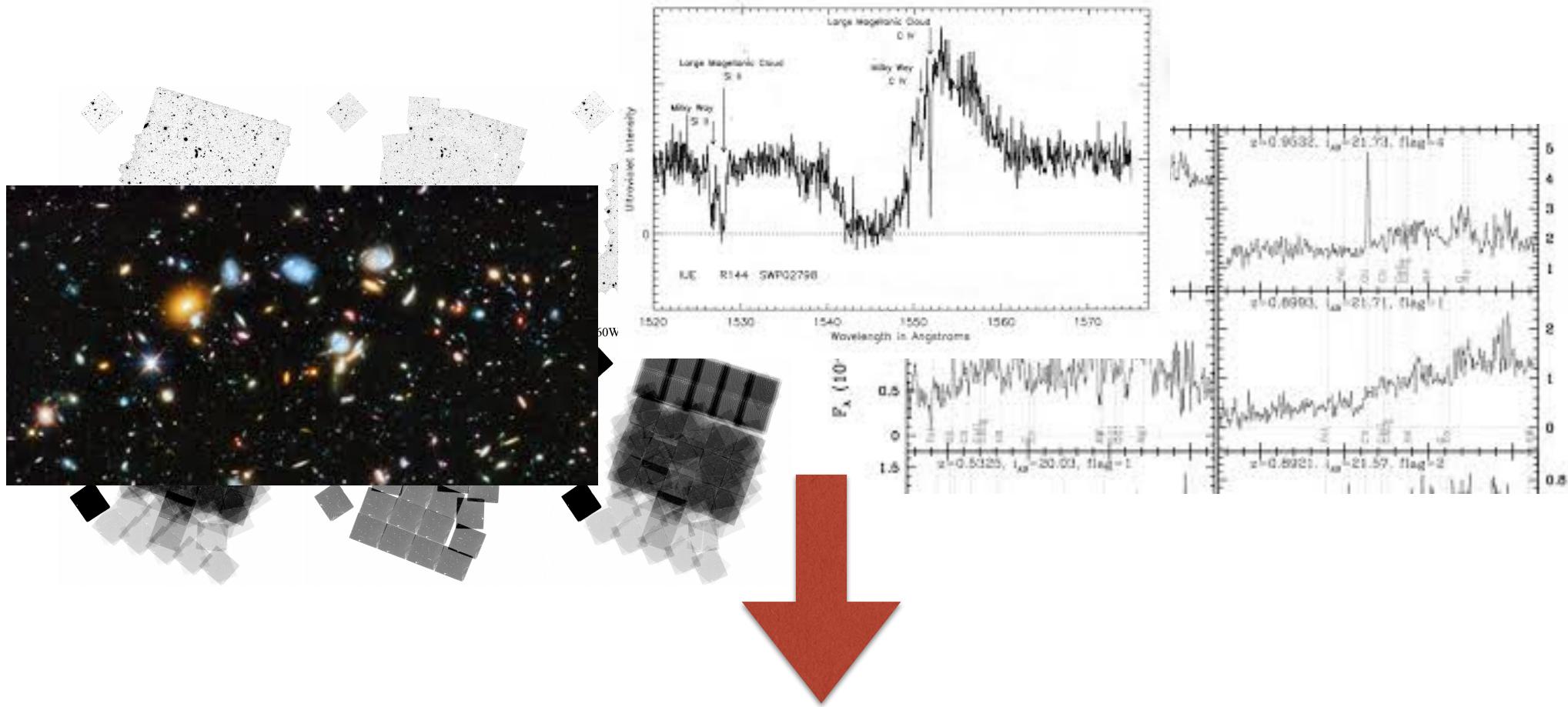
THIS IS WHAT “WE” [HUMANS] SEE

What do we put as input?



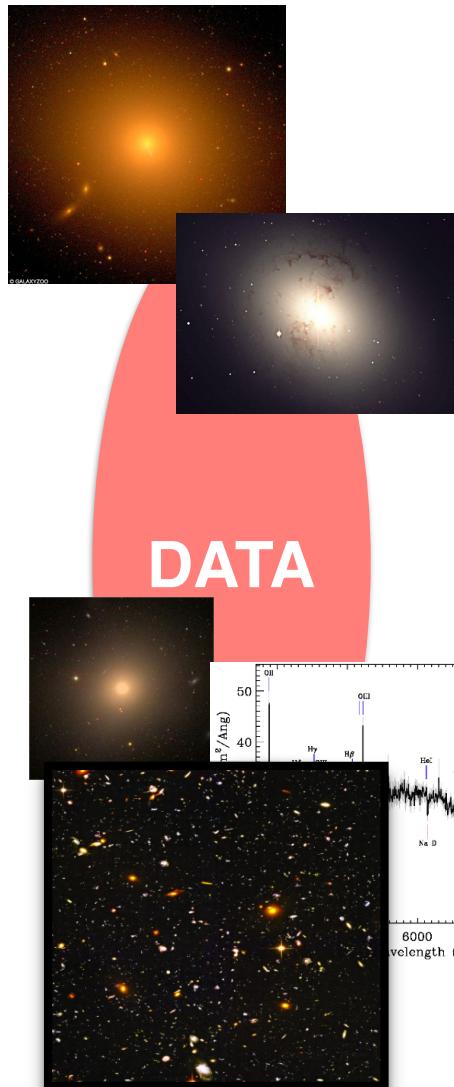
THIS IS WHAT MACHINES SEE

What do we put as input?



PRE-PROCESS DATA TO EXTRACT MEANINGFUL INFORMATION

THIS IS GENERALLY CALLED **FEATURE EXTRACTION**



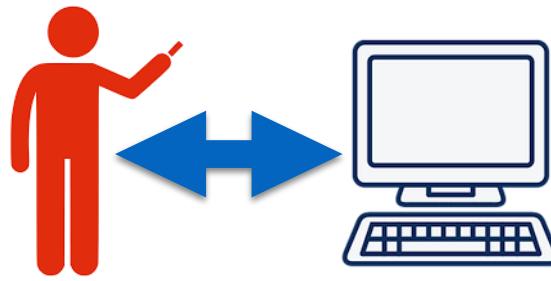
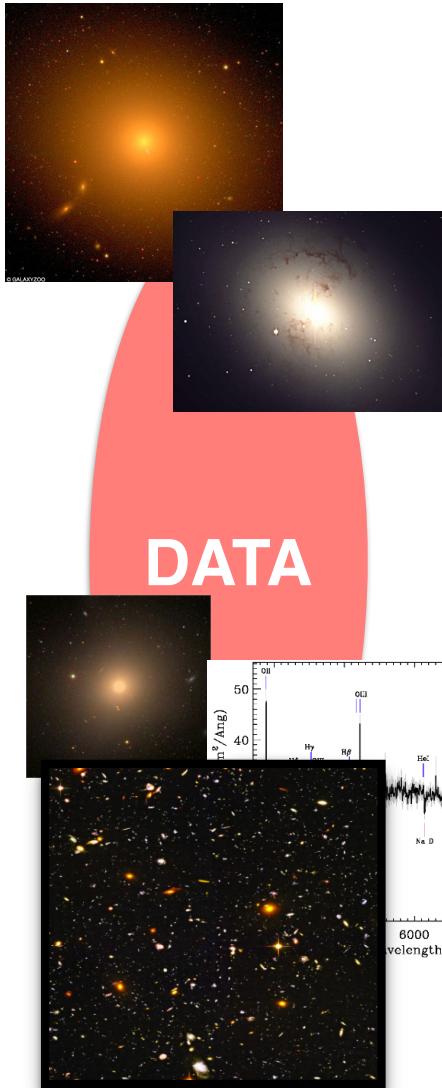
Spiral!

Emission line!

Merger!

Clump!

AGN!



Spiral!

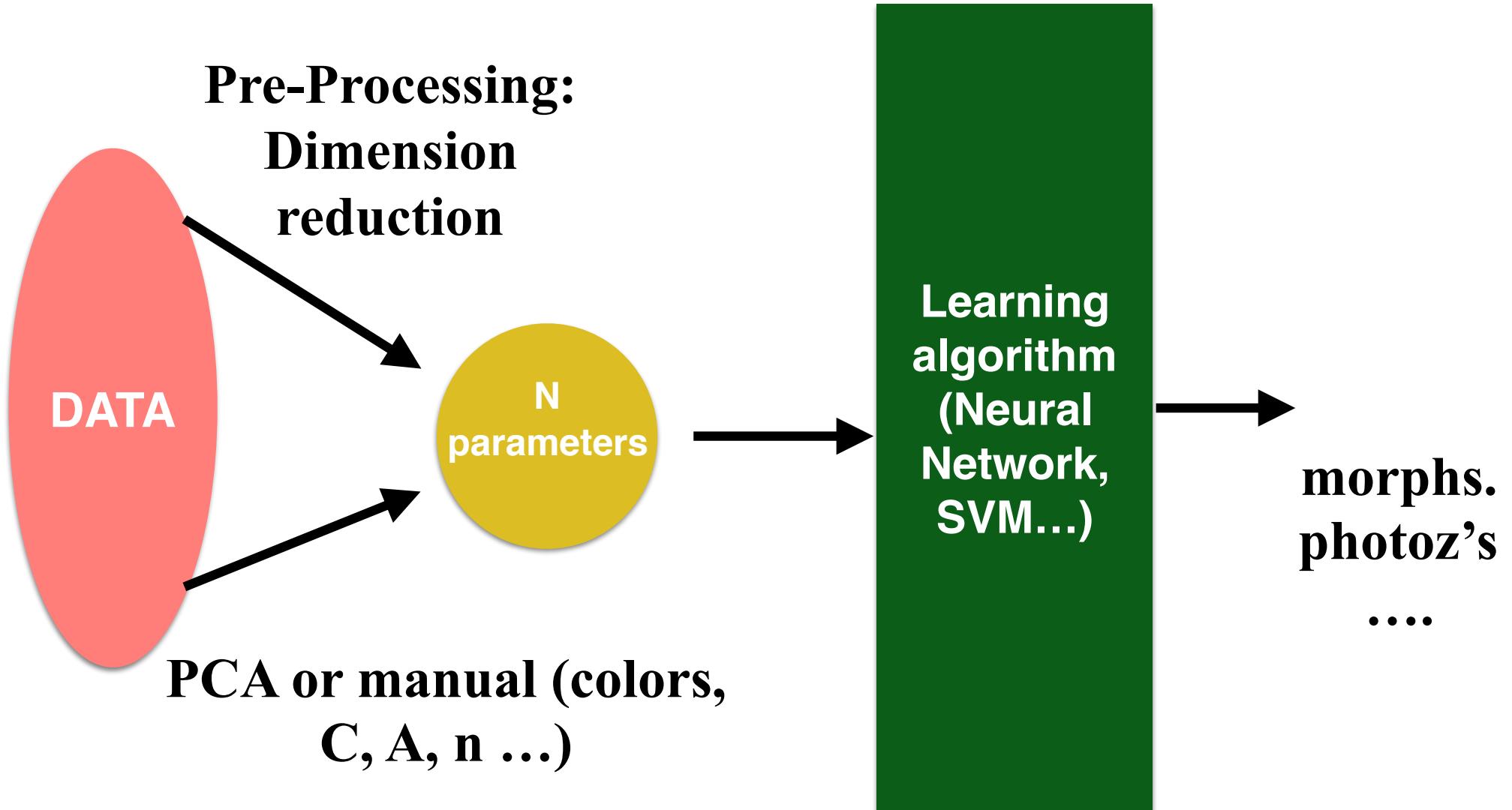
Emission line!

Merger!

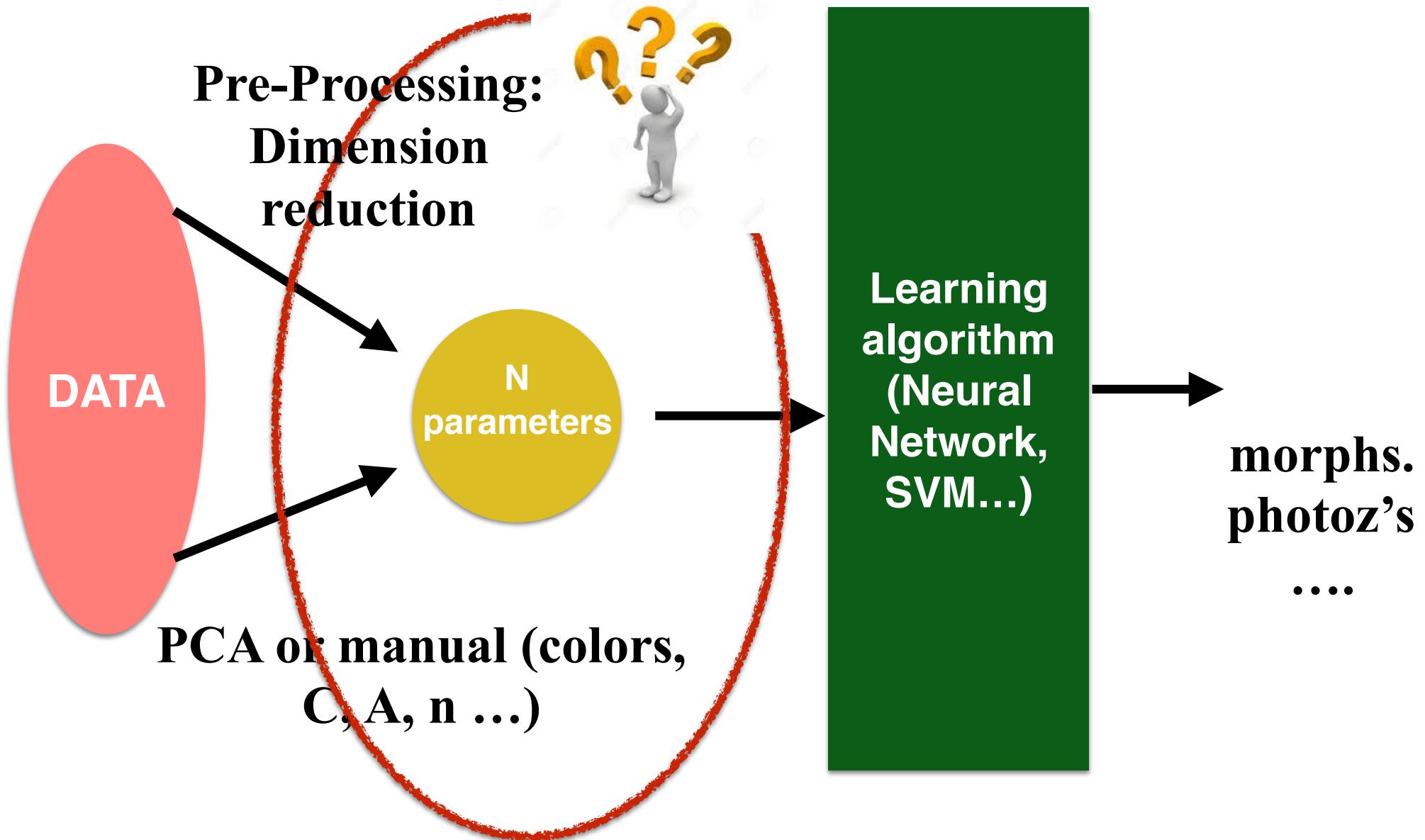
Clump!

AGN!

THE “CLASSICAL” APPROACH

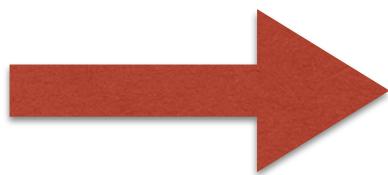


“CLASSICAL” MACHINE LEARNING

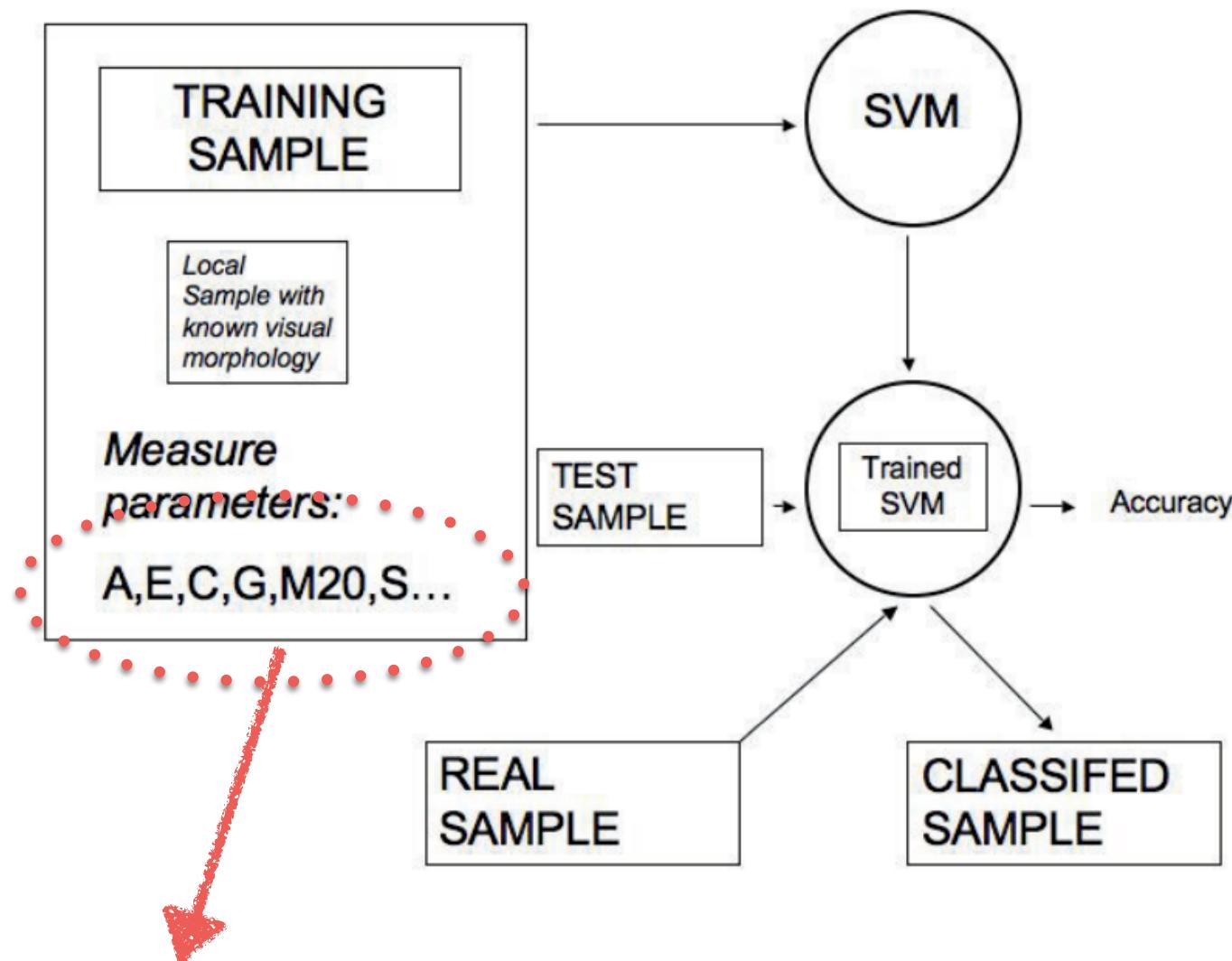


In Astronomy

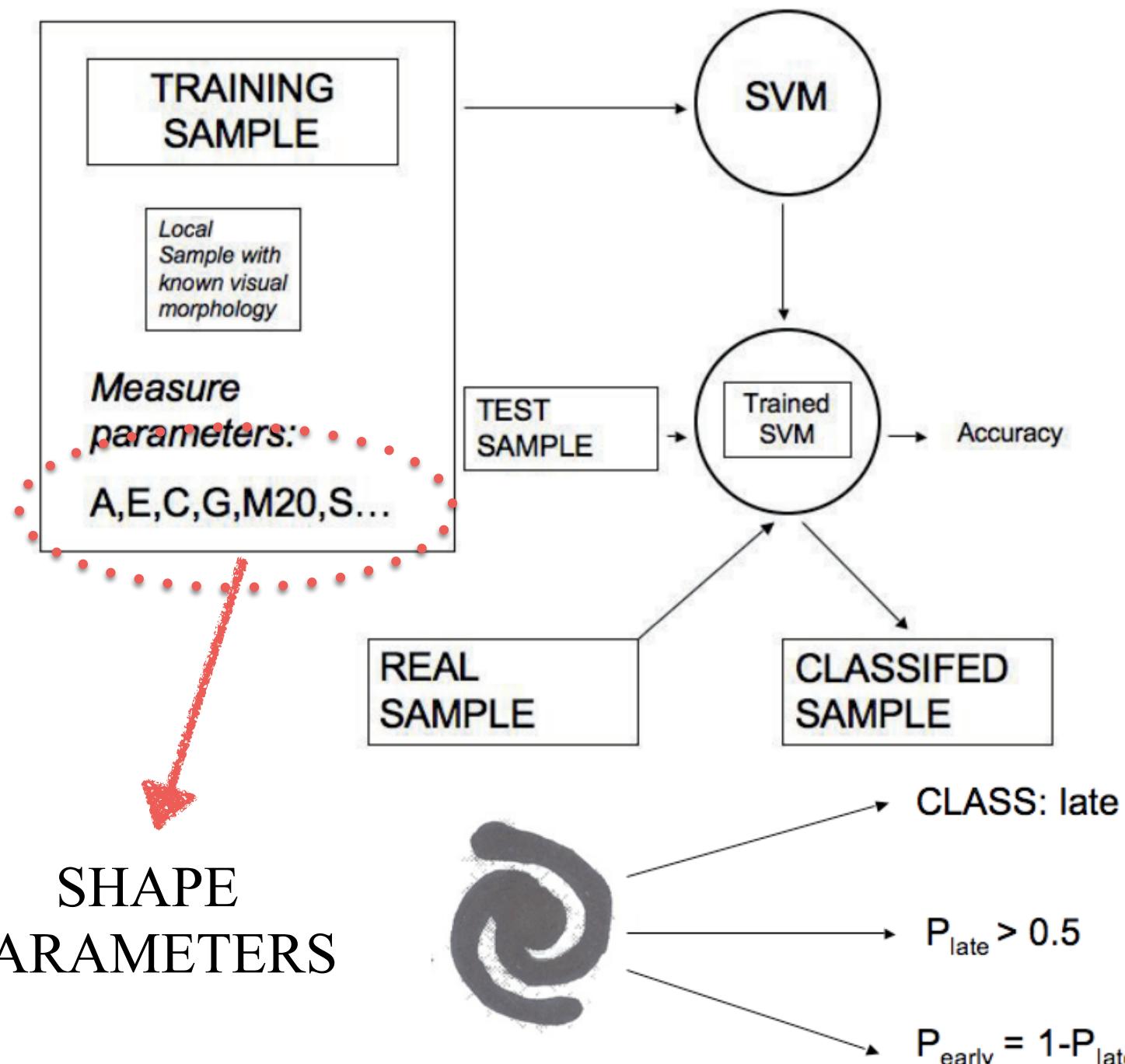
- Colors, Fluxes
- Shape indicators
- Line ratios, spectral features
- Stellar Masses, Velocity Dispersions

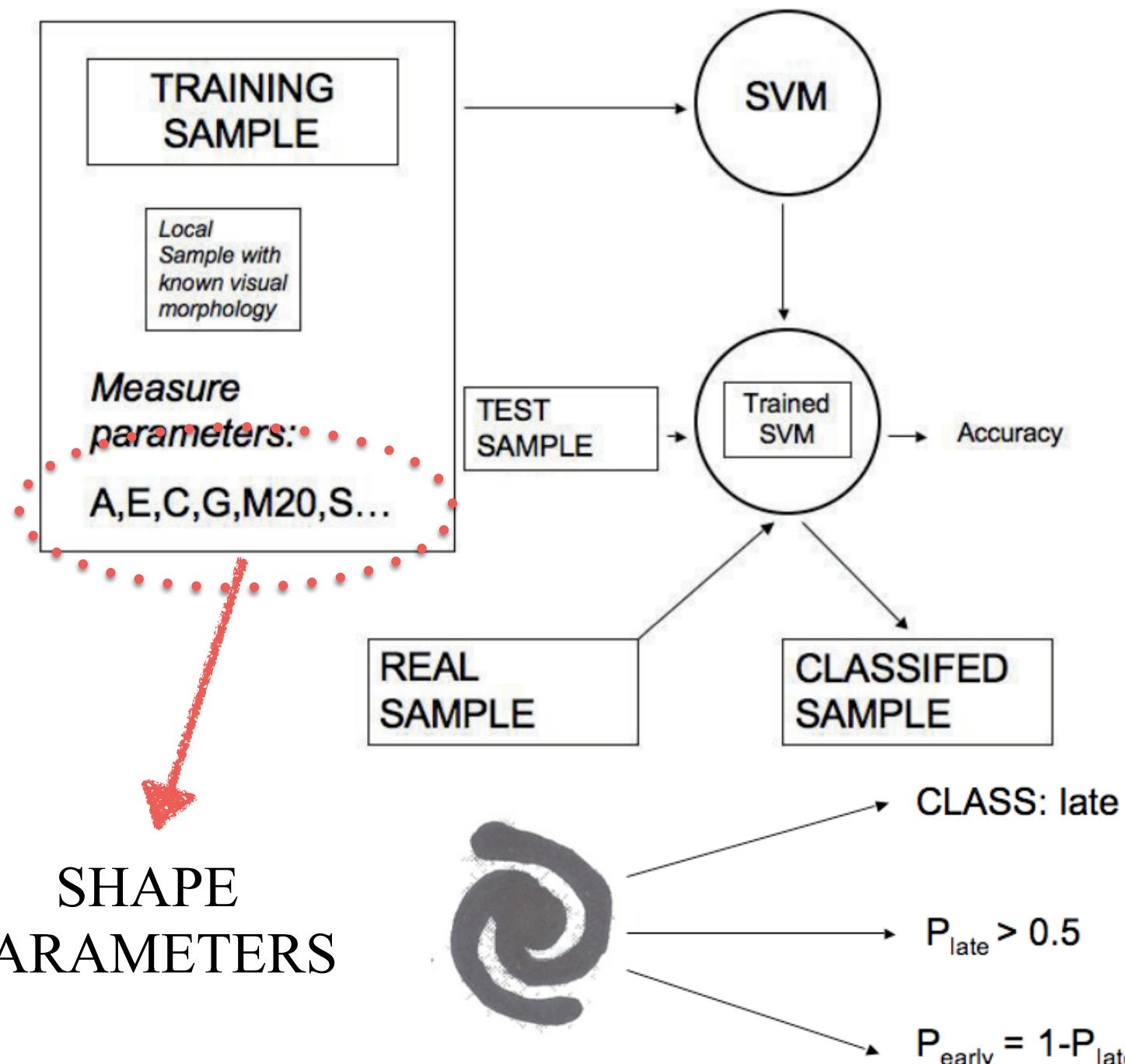


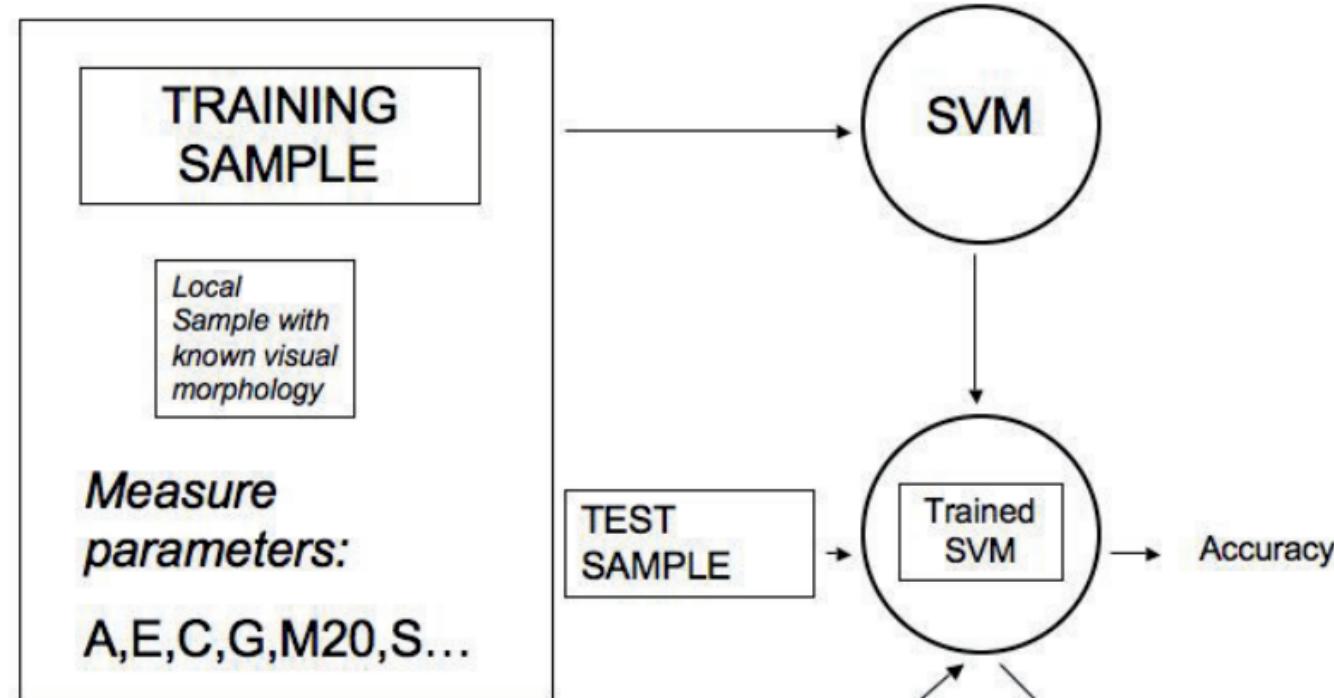
Requires specialized software before feeding the machine learning algorithm



SHAPE
PARAMETERS

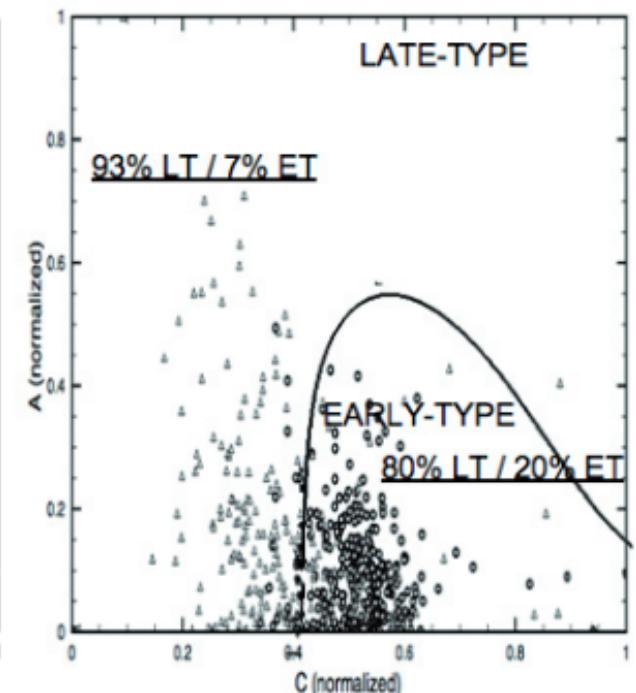
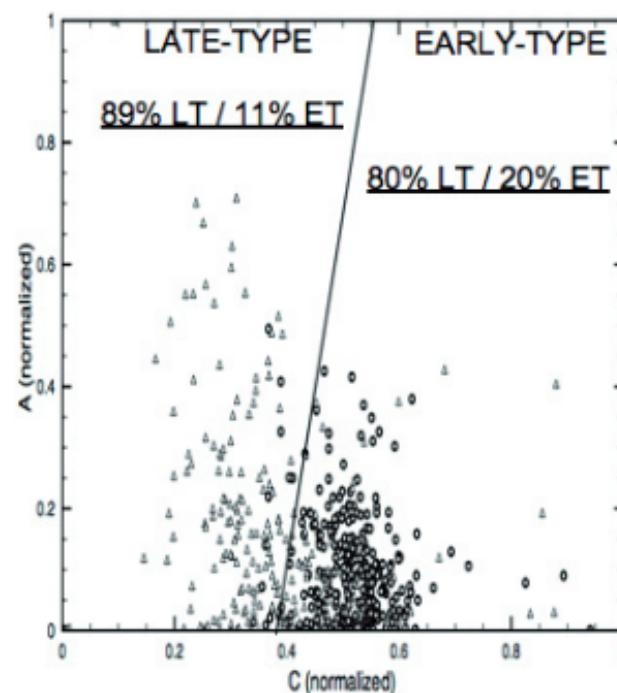




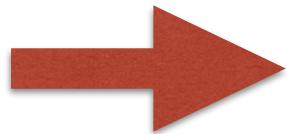


Measure parameters:
A,E,C,G,M20,S...

REAL SAMPL



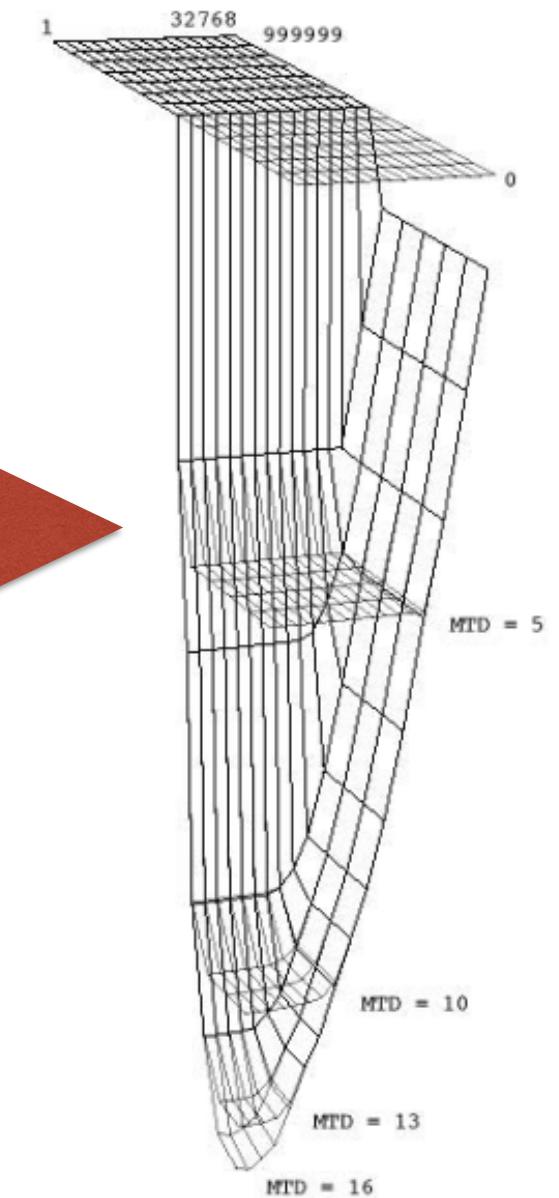
Star-galaxy separation with CARTS



$g-r$

$r-i$

Z



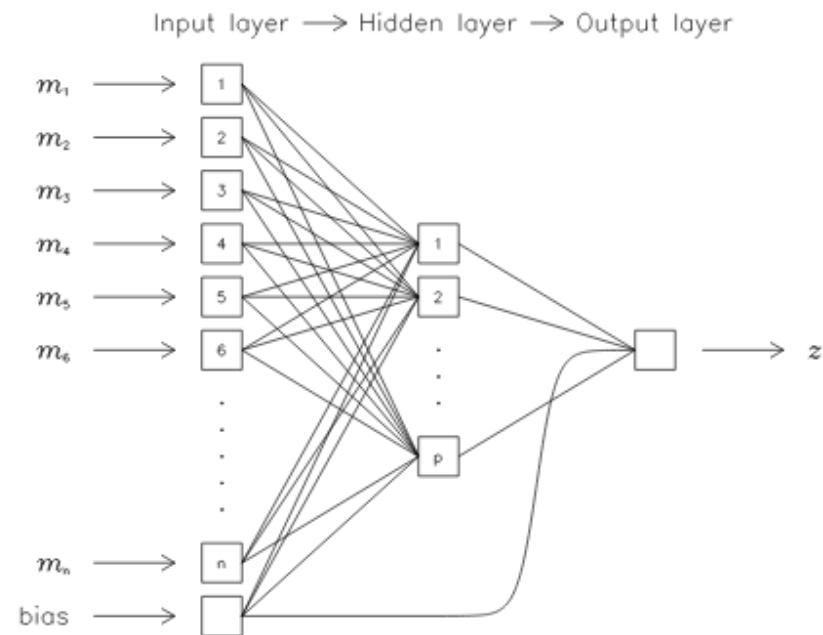
Ball+06

PHOTOMETRIC REDSHIFTS

SDSS

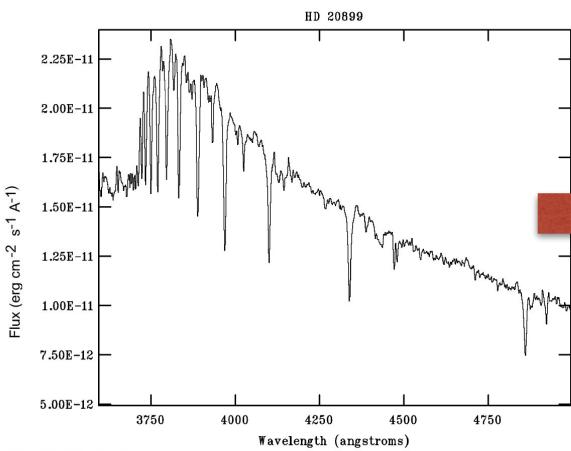


g
r
i
z

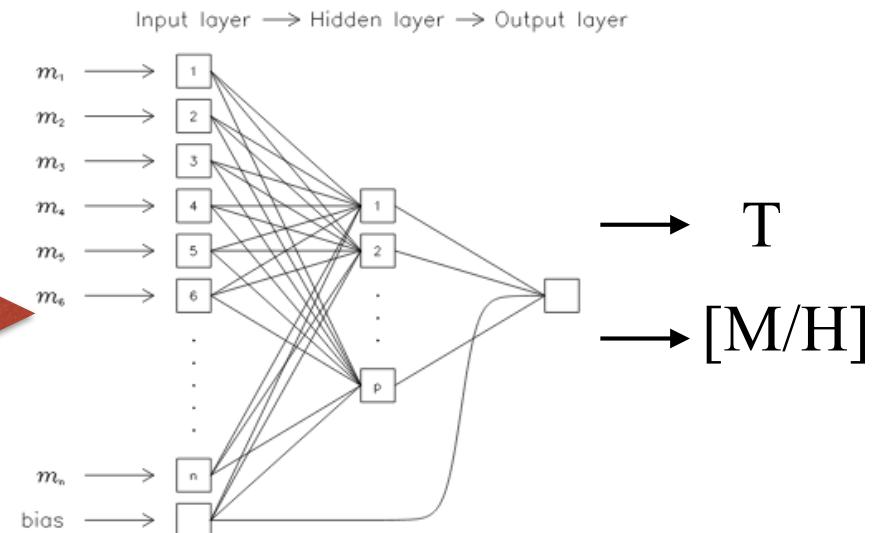


Collister+08

STELLAR PARAMETERS FROM MEDIUM BAND FILTERS



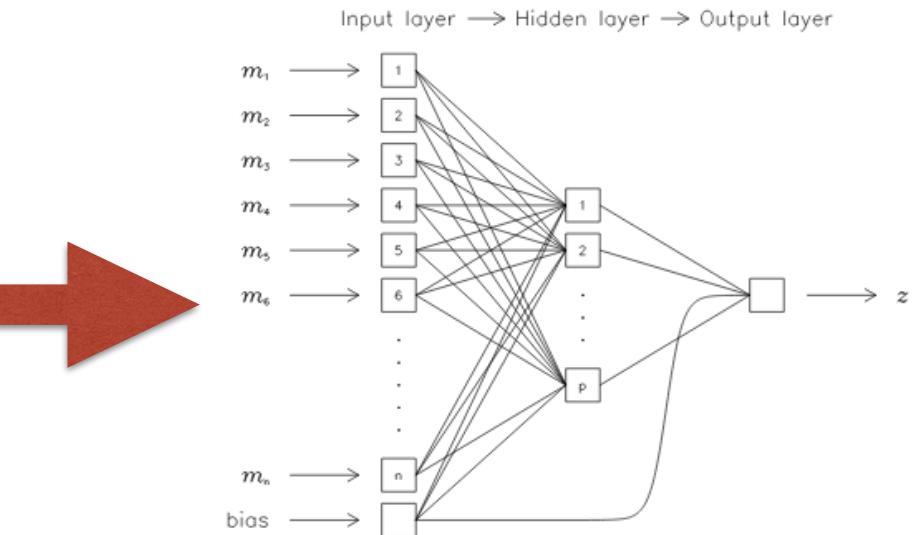
MEDIUM
BAND
FLUXES



Bailer-Jones+00

| No. | Symbol | Description | Scale ^a |
|-----|--------------------|-----------------------------------|--------------------|
| 1 | CVD | Central velocity dispersion | ~1 kpc |
| 2 | M_{bulge} | Bulge stellar mass | 0.5–4 kpc |
| 3 | R_e | Bulge effective radius | 0.5–4 Kpc |
| 4 | B/T | Bulge-to-total stellar mass ratio | 0.5–8 kpc |
| 5 | M_* | Total stellar mass | 2–8 kpc |
| 6 | M_{disc} | Disc stellar mass | 4–10 kpc |
| 7 | M_{halo} | Group halo mass | 0.1–1 Mpc |
| 8 | δ_5 | Local density parameter | 0.5–3 Mpc |

Notes. ^a Approximate 1σ range from centre of galaxy. For photometric quantities half-light radii are used.

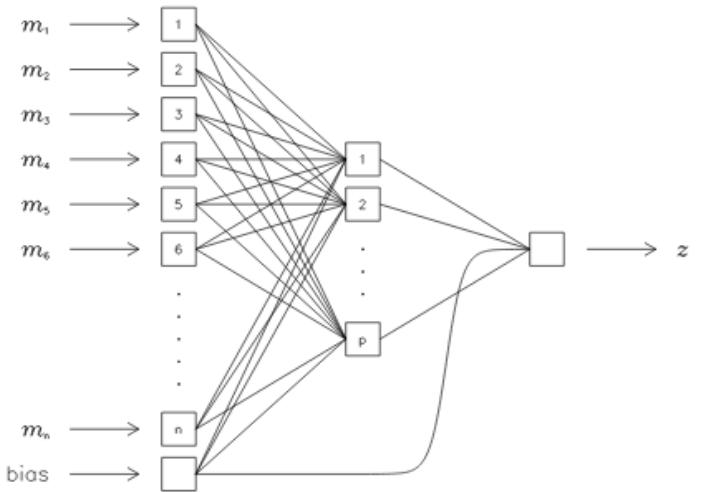


HEAVILY PROCESSED DATA

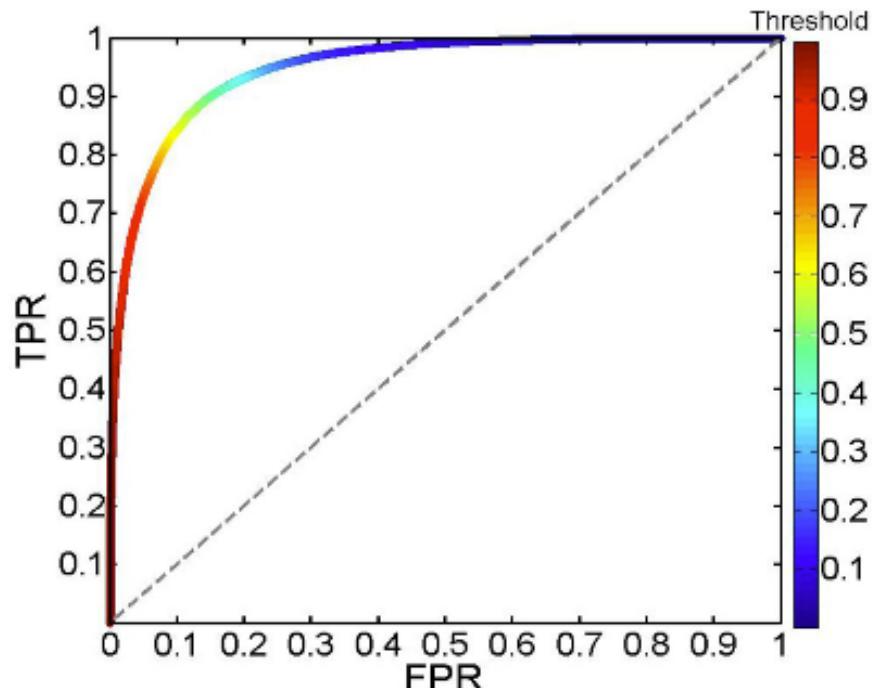
| No. | Symbol | Description | Scale ^a |
|-----|--------------------|-----------------------------------|--------------------|
| 1 | CVD | Central velocity dispersion | ~1 kpc |
| 2 | M_{bulge} | Bulge stellar mass | 0.5–4 kpc |
| 3 | R_e | Bulge effective radius | 0.5–4 Kpc |
| 4 | B/T | Bulge-to-total stellar mass ratio | 0.5–8 kpc |
| 5 | M_* | Total stellar mass | 2–8 kpc |
| 6 | M_{disc} | Disc stellar mass | 4–10 kpc |
| 7 | M_{halo} | Group halo mass | 0.1–1 Mpc |
| 8 | δ_5 | Local density parameter | 0.5–3 Mpc |

Notes. ^a Approximate 1σ range from centre of galaxy. For photometric quantities half-light radii are used.

Input layer → Hidden layer → Output layer



HEAVILY PROCESSED DATA

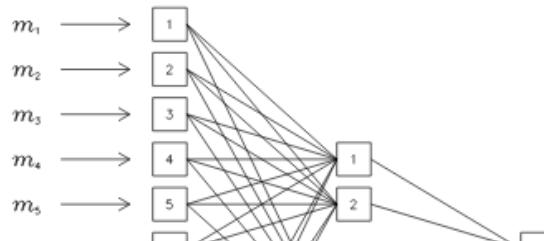


Teimiroonia+16

| No. | Symbol | Description | Scale ^a |
|-----|--------------------|-----------------------------------|--------------------|
| 1 | CVD | Central velocity dispersion | ~1 kpc |
| 2 | M_{bulge} | Bulge stellar mass | 0.5–4 kpc |
| 3 | R_e | Bulge effective radius | 0.5–4 Kpc |
| 4 | B/T | Bulge-to-total stellar mass ratio | 0.5–8 kpc |
| 5 | M_* | Total stellar mass | 2–8 kpc |
| 6 | M_{disc} | Disc stellar mass | 4–10 kpc |
| 7 | M_{halo} | Group halo mass | 0.1–1 Mpc |
| 8 | δ_5 | Local density parameter | 0.5–3 Mpc |

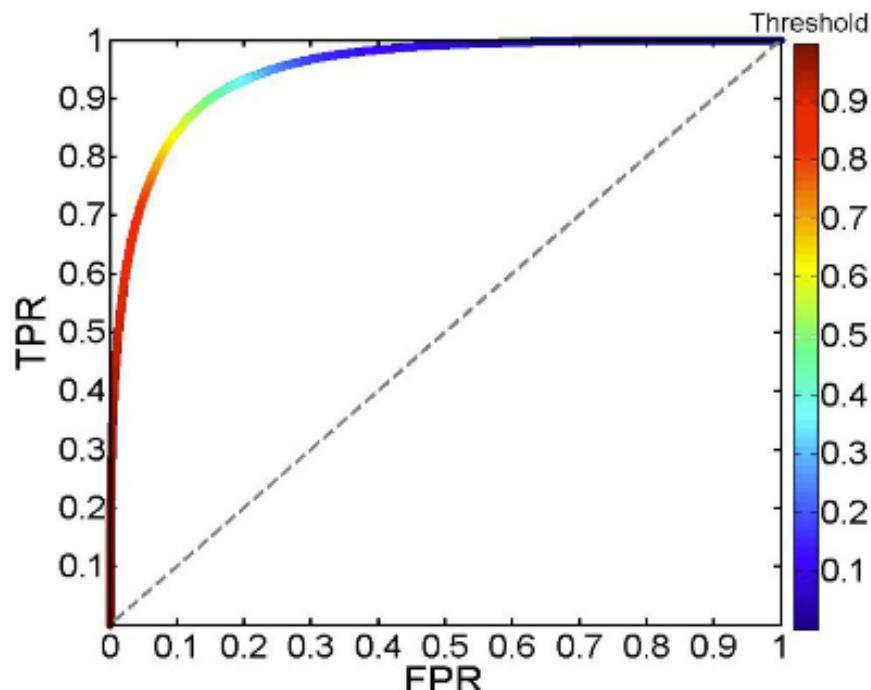
Notes. ^a Approximate 1σ range from centre of galaxy. For photometric quantities half-light radii are used.

Input layer → Hidden layer → Output layer

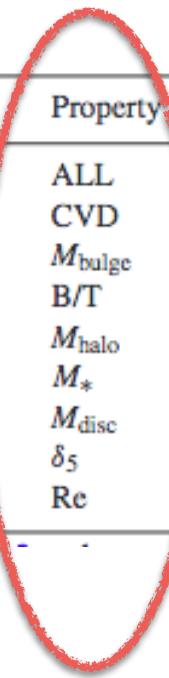


A KEY ADVANTAGE IS THAT
CONTROL IS KEPT
ABOUT THE IMPACT OF THE
INPUT PARAMETERS

HEAVILY PROCESSED DATA



| 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 |
| ... | ... | ... | ... |



Teimiroonia+16

Other general computer vision features [for images!]

- Pixel Concatenation
- Color histograms
- Texture Features
- Histogram of Gradients
- SIFT

FOR MANY YEARS COMPUTER VISION RESEARCHERS HAVE BEEN TRYING TO FIND THE MOST GENERAL FEATURES

Other general computer vision features [for images!]

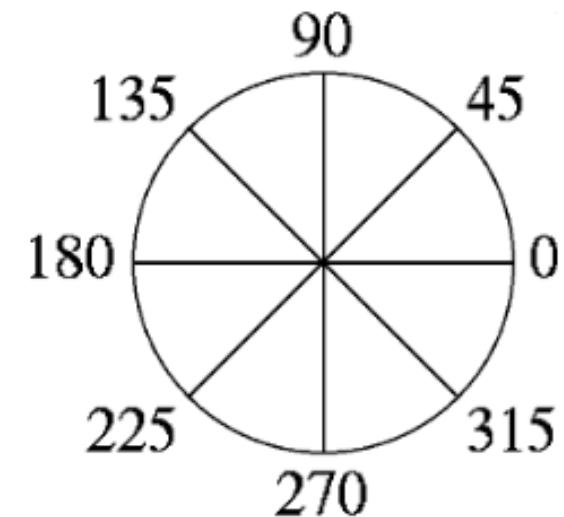
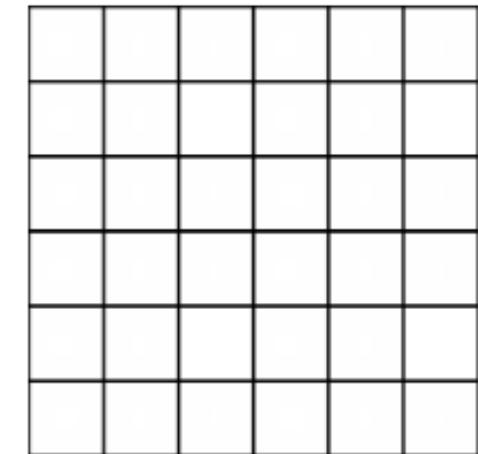
- Pixel Concatenation
- Color histograms
- Texture Features
- Histogram of Gradients
- SIFT

FOR MANY YEARS COMPUTER VISION RESEARCHERS HAVE BEEN TRYING TO FIND THE MOST GENERAL FEATURES

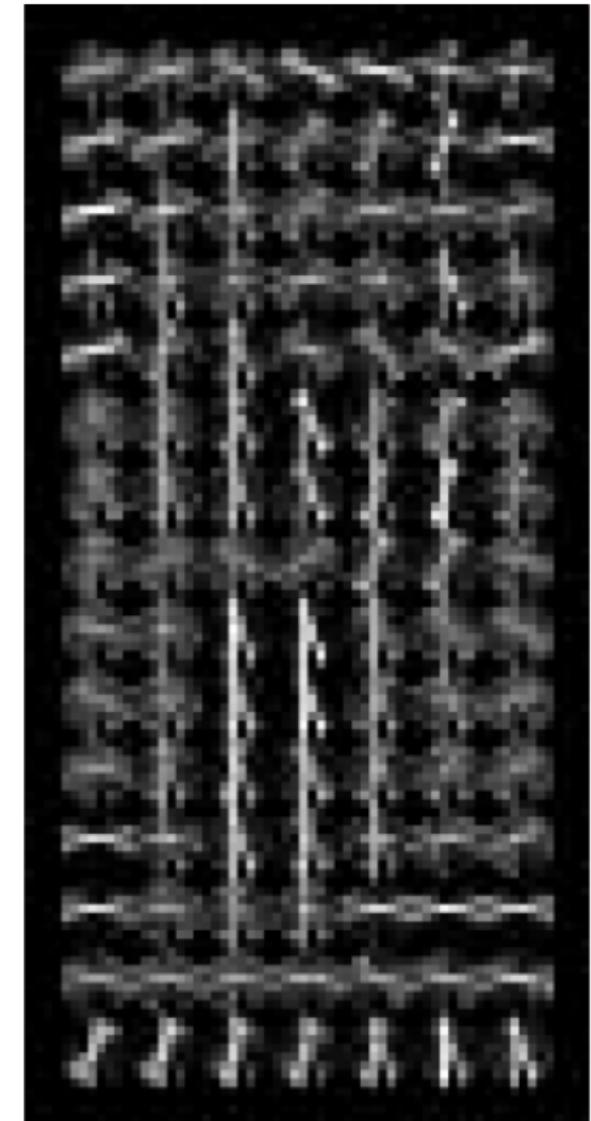
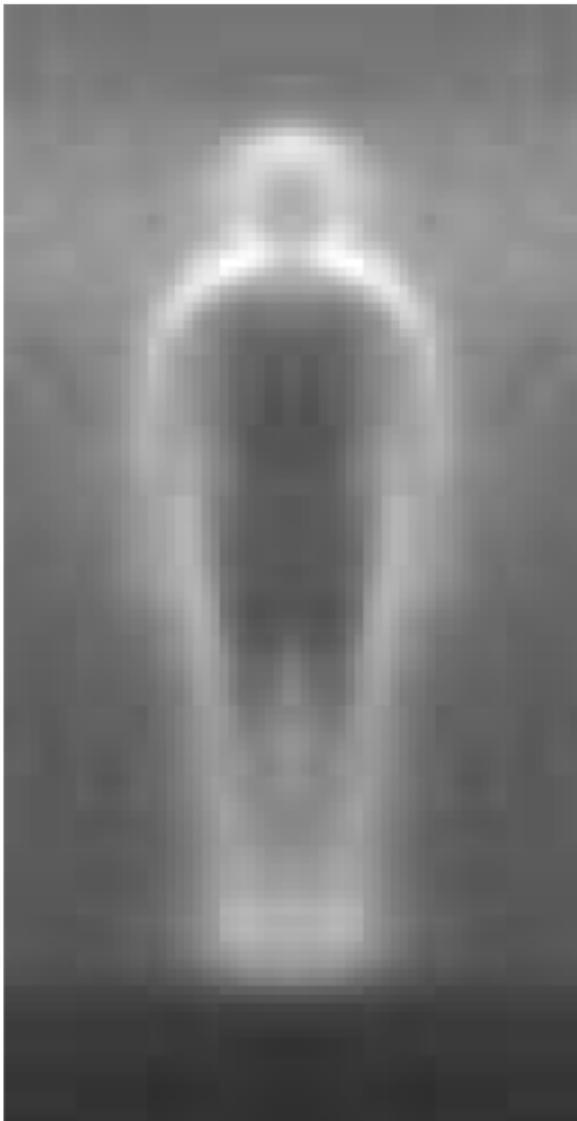
THE BEST CLASSICAL SOLUTION [BEFORE 2012] WHERE BASED ON LOCAL FEATURES

HISTOGRAM OF ORIENTED GRADIENTS (HoG)

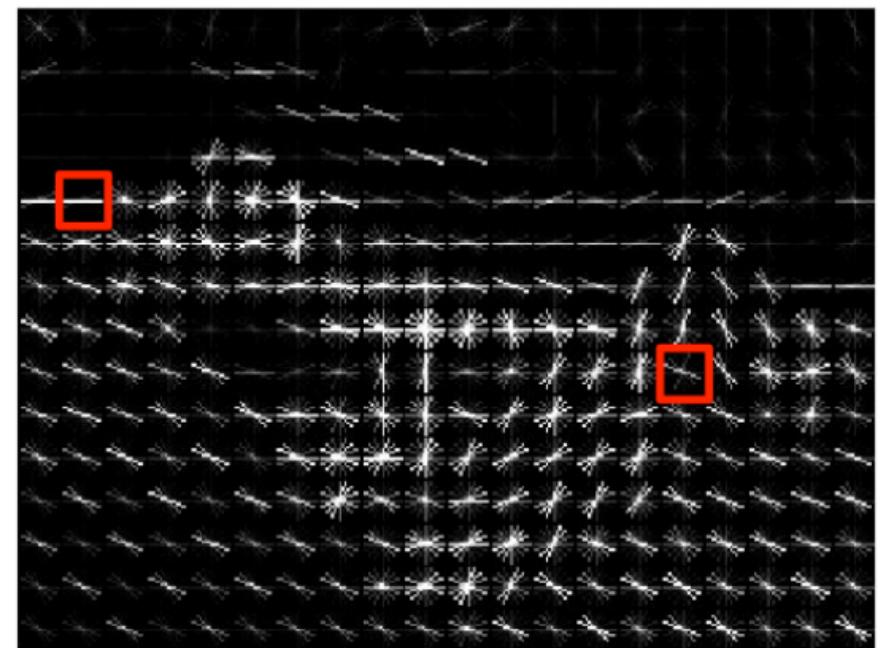
1. DIVIDE IMAGE INTO SMALL SPATIAL REGIONS CALLED CELLS
2. COMPUTE INTENSITY GRADIENTS OVER N DIRECTIONS [TYPICALLY 9 FOR IMAGE]
3. COMPUTE WEIGHTED 1-D HISTOGRAM OF ALL DIRECTIONS. A CELL IS REDUCED TO N NUMBERS



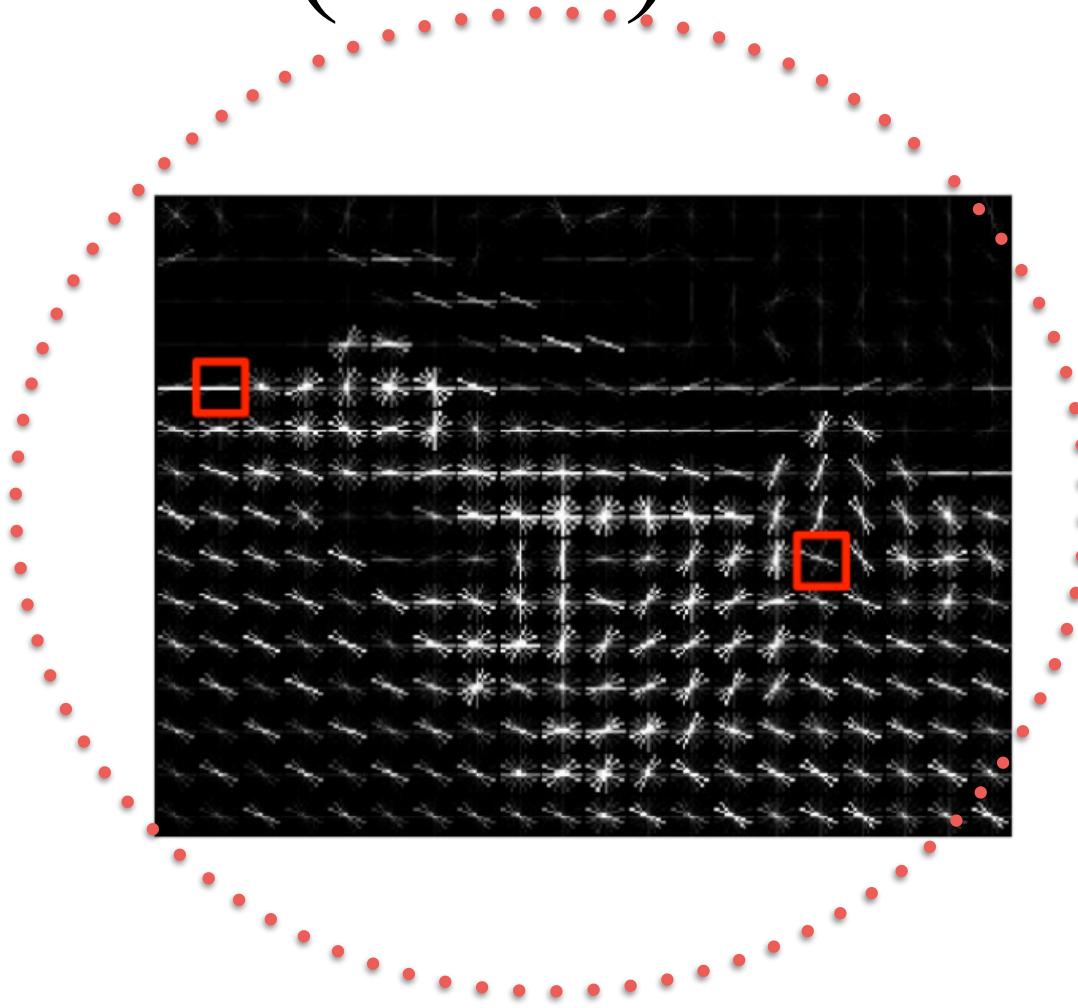
HISTOGRAM OF ORIENTED GRADIENTS (HoG)



HISTOGRAM OF ORIENTED GRADIENTS (HoG)



HISTOGRAM OF ORIENTED GRADIENTS (HoG)



KEEP THIS IMAGE IN MIND FOR LATER...

SIFT FEATURES

[Scale Invariant Feature Transform]

1. FIND LOCAL REGIONS OF INTEREST
WITHIN THE IMAGE

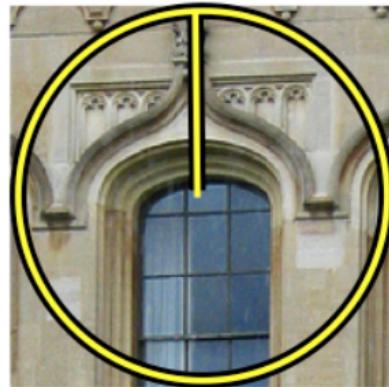
2. REMOVE EFFECTS OF ORIENTATION
AND SCALE

3. COMPUTE HISTOGRAM
OF GRADIENTS IN DEFINED REGIONS

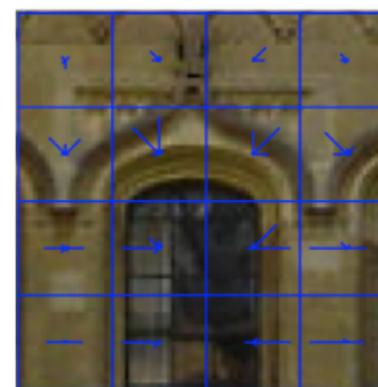
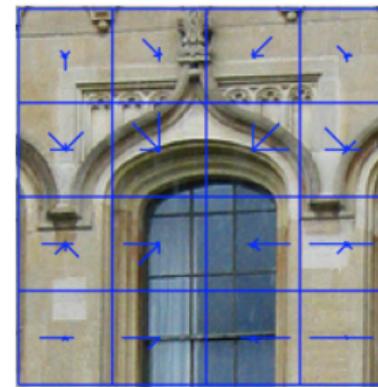
SIFT FEATURES

[Scale Invariant Feature Transform]

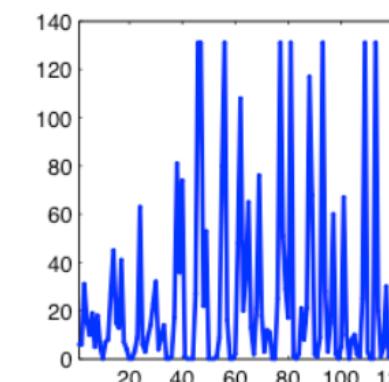
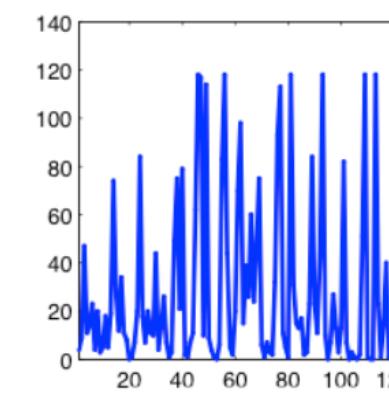
normalised features



spatial histogram of gradients



SIFT descriptor



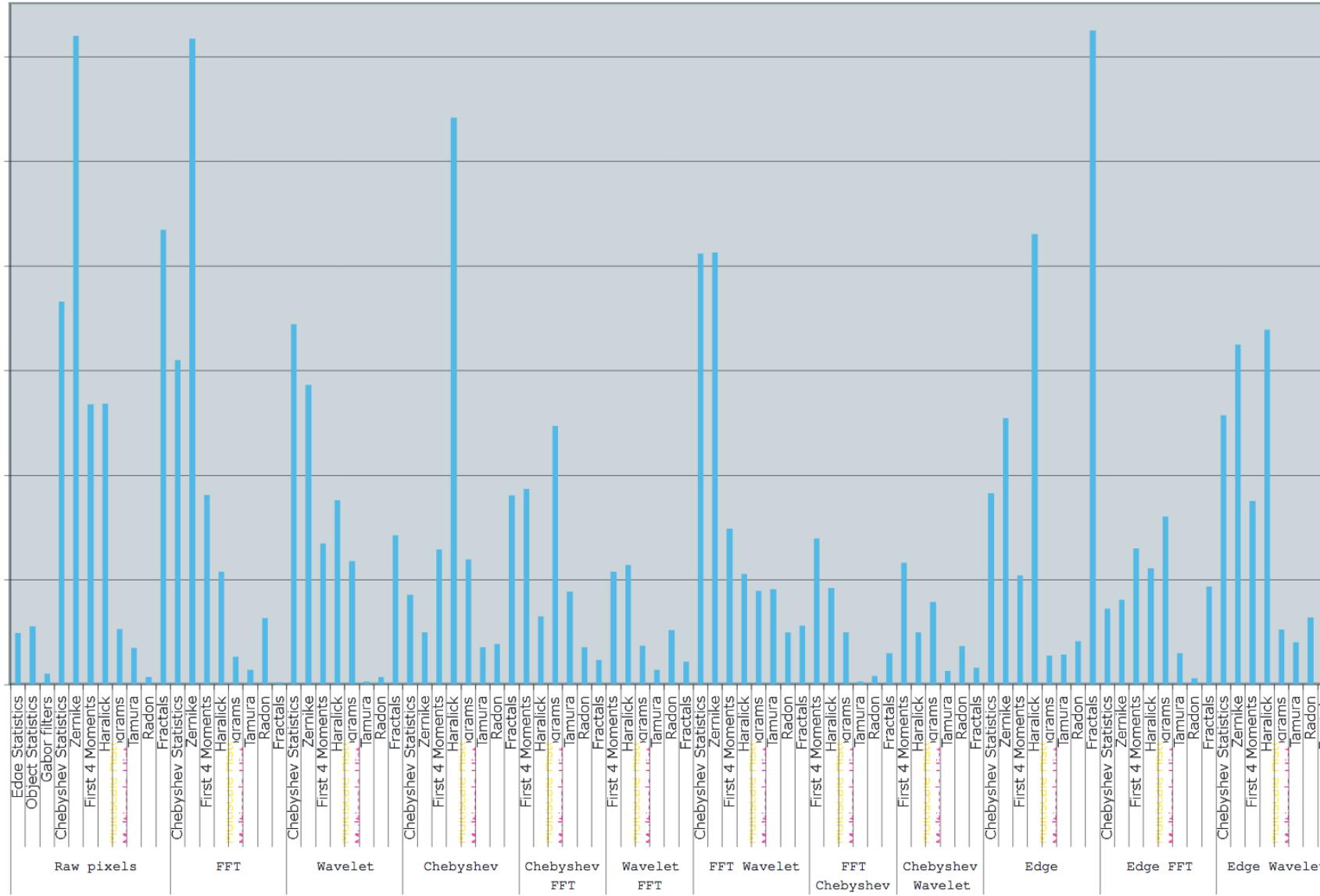
d

d'

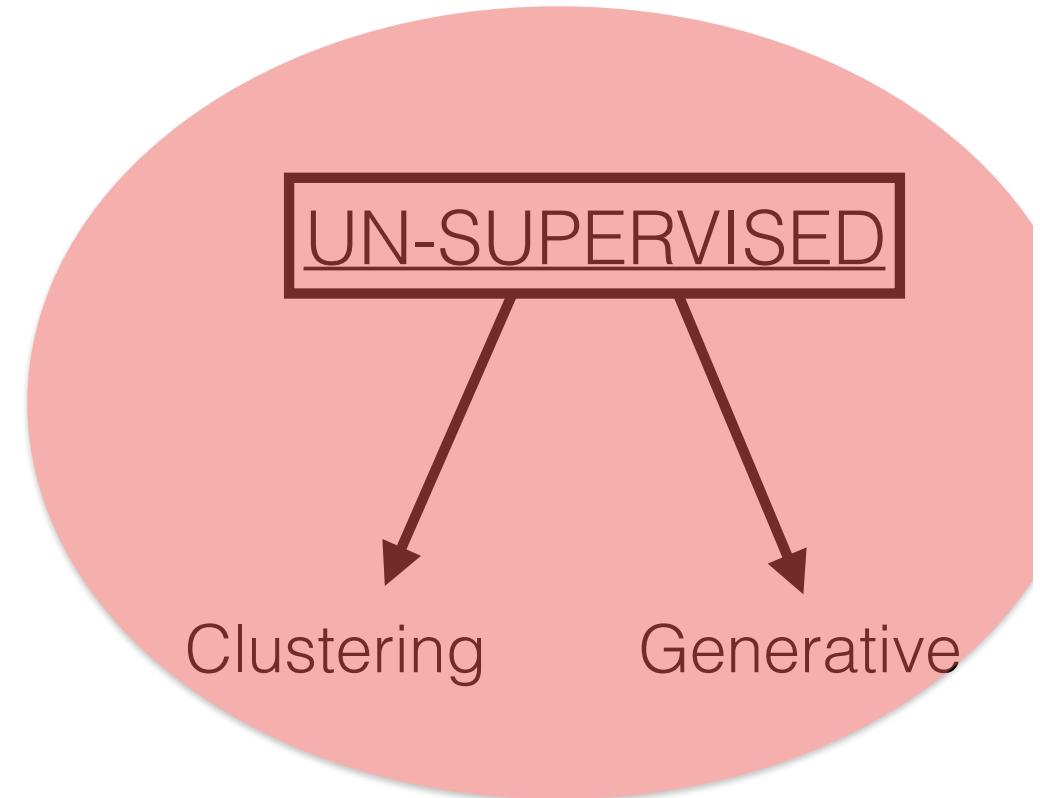
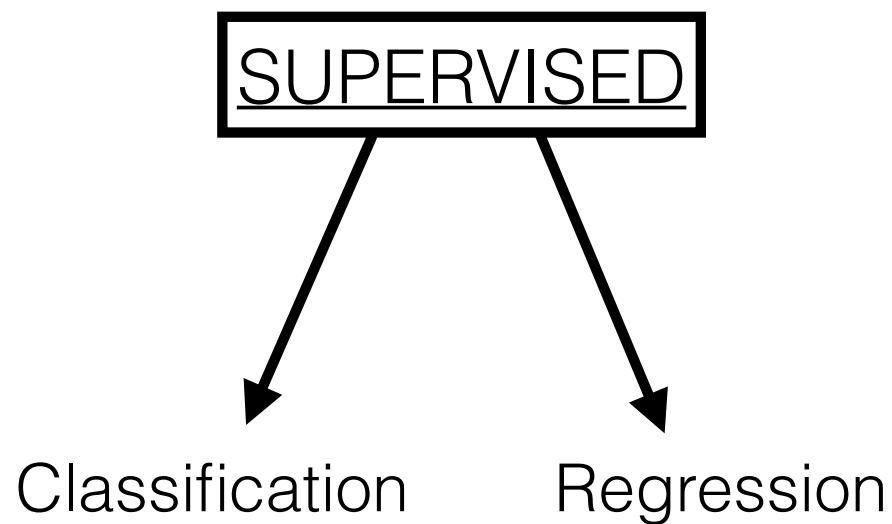
TRADITIONAL FEATURES HAVE GENERALLY IN COMMON...

- 1.ACT LOCALLY
- 2.LOOK FOR DIFFERENT ORIENTATIONS
- 3.TRY TO BE SCALE / TRANSLATION / ROTATION
INVARIANT

A LARGE AMOUT OF FEATURES CAN BE COMBINED

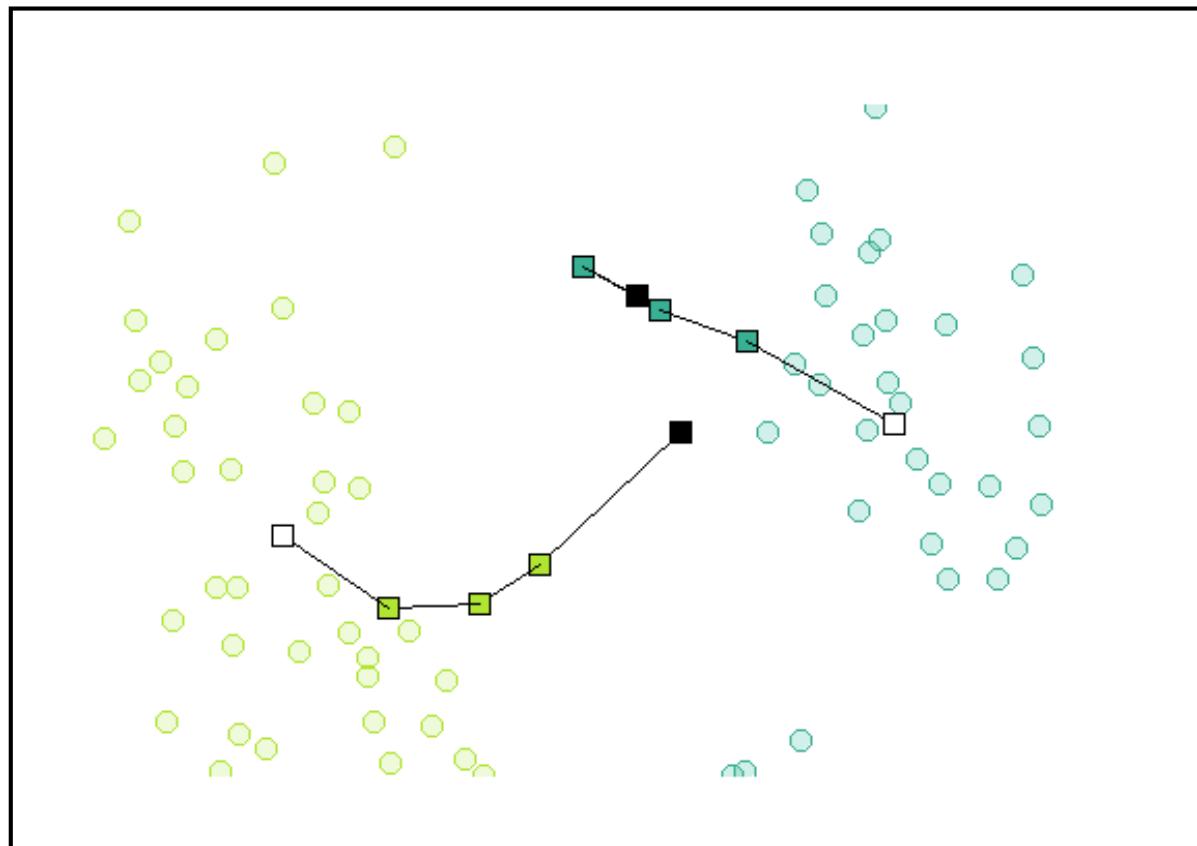


WHAT DOES MACHINE LEARNING DO?



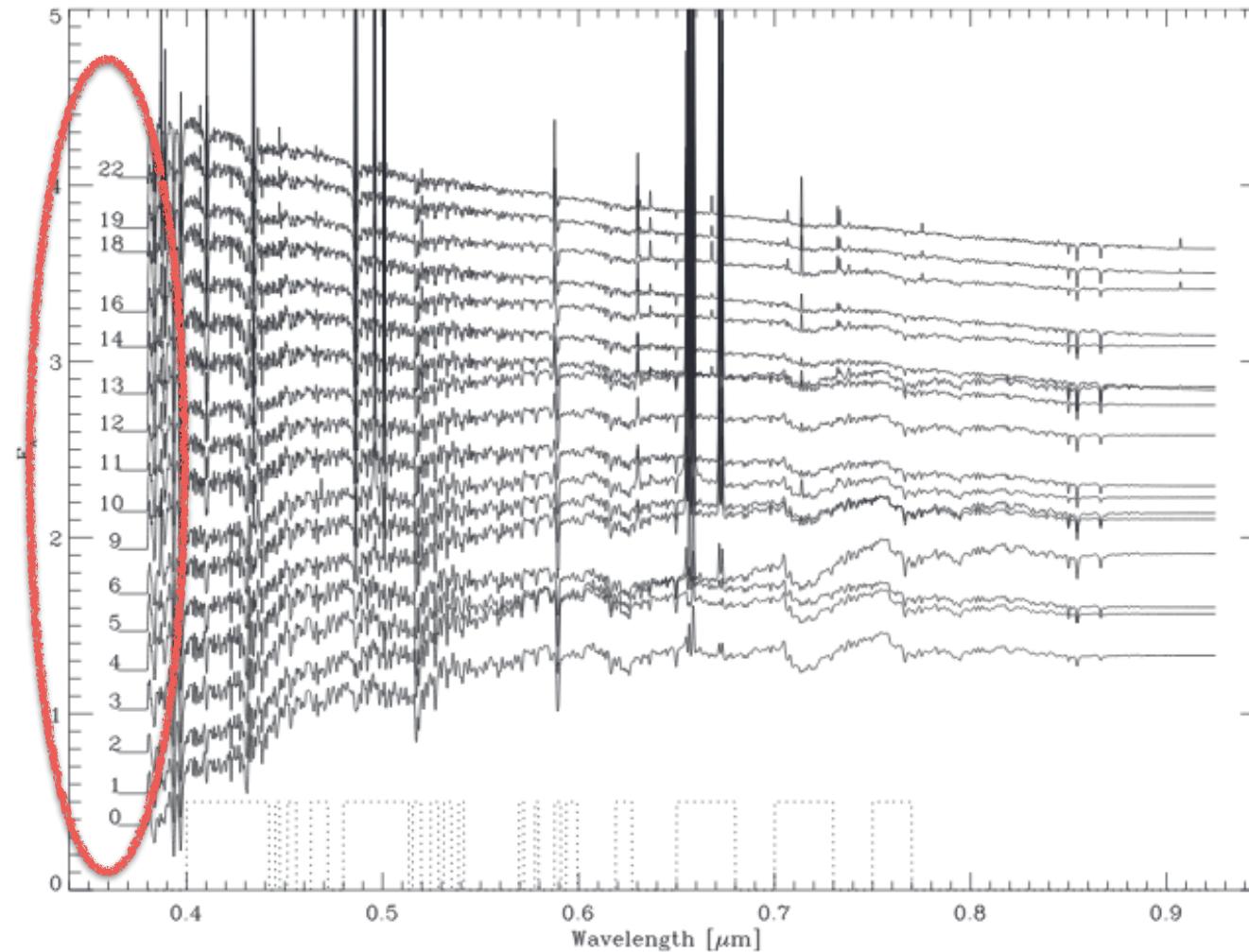
MOST DATA ARE ACTUALLY UNLABELLED!

K-means clustering



K-MEANS APPLIED TO ALL SDSS SPECTRA

NUMBER
OF CLASSES
FOUND



KEY ISSUES WITH K-MEANS CLUSTERING

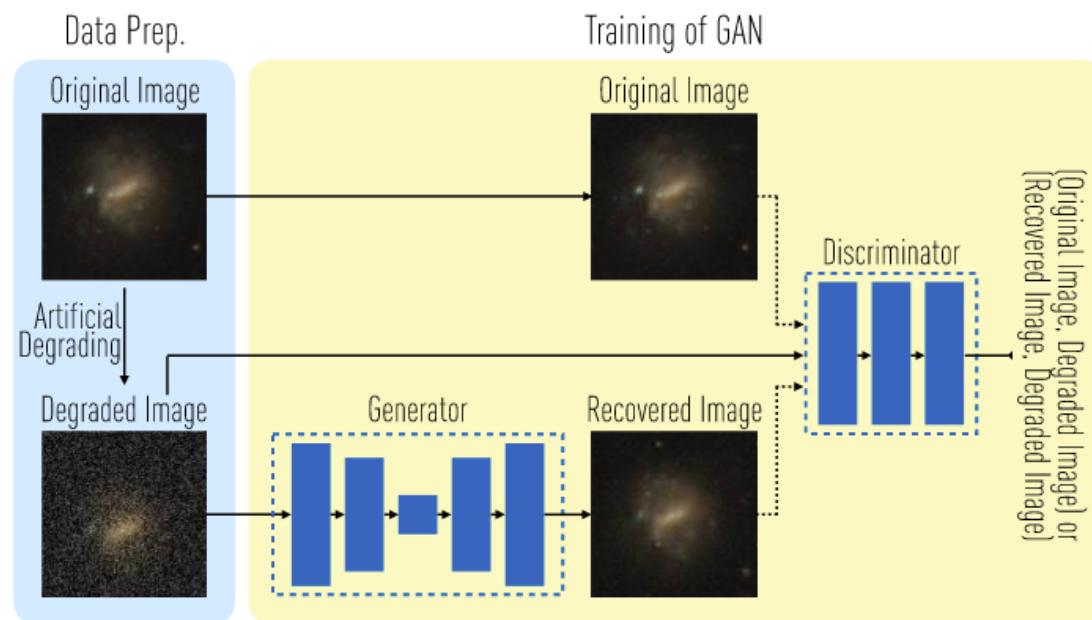
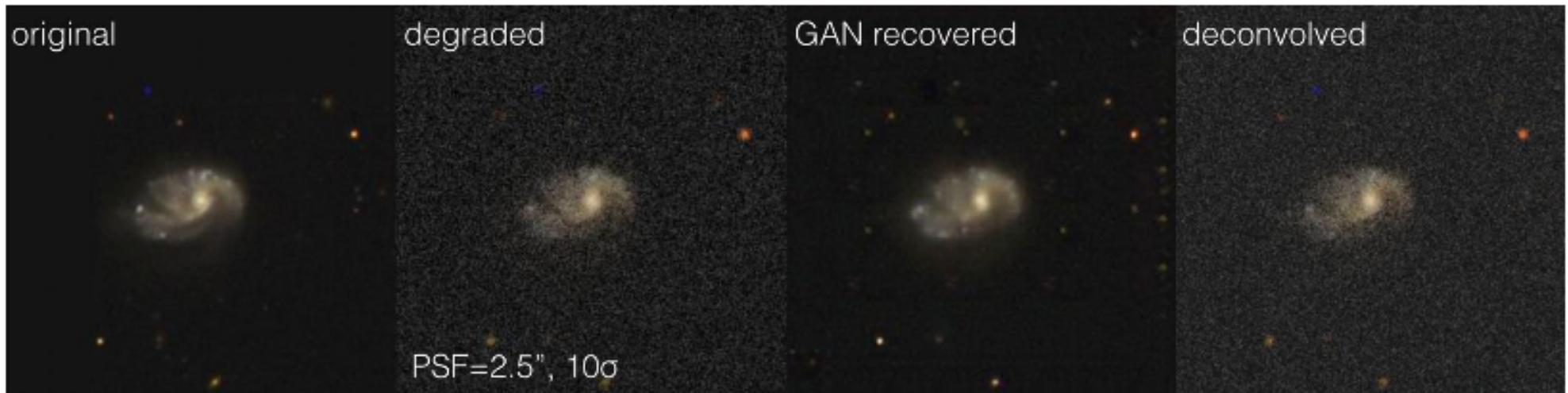
NUMBER OF CLASSES / CENTERS?

ARE THEY SIGNIFICANT?

INITIALIZATION

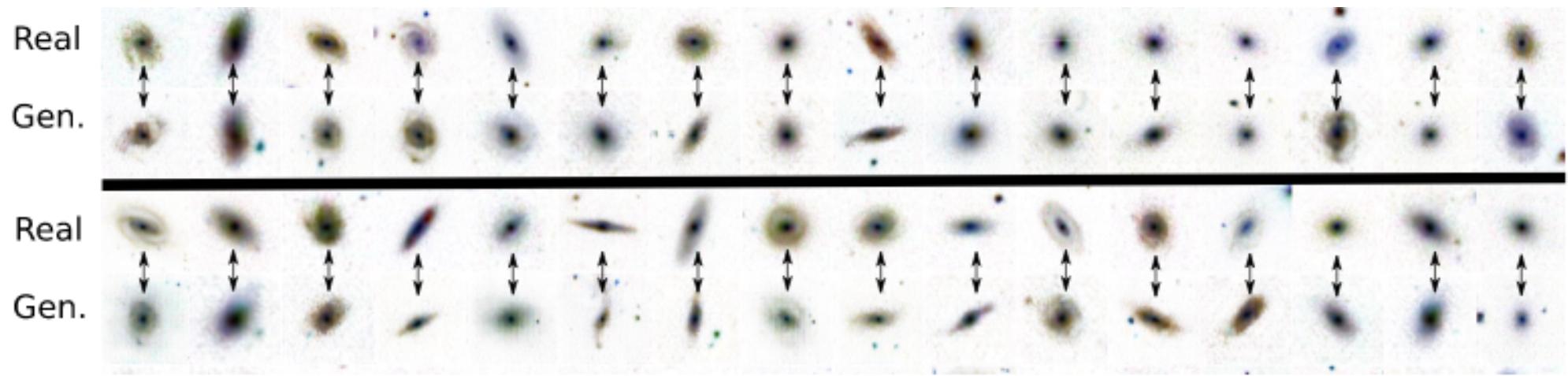
WHERE DO I CHOOSE THE CLUSTER CENTERS?

Generative Models



Schawinsky+17

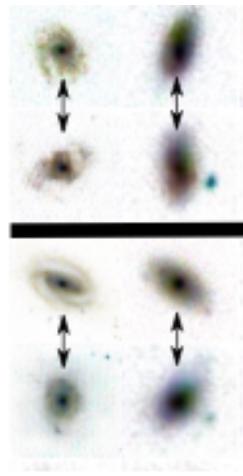
Generative Models



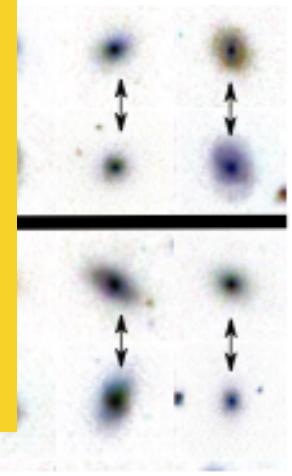
Ravanbakhsh+16

Generative Models

Real
Gen.
Real
Gen.



**A BIT MORE OF THIS ON
FRIDAY!
[TALK BY S. VELASCO-
FORERO]**



Ravanbakhsh+16