TO SUMMARIZE: we need two key elements

1. A LOSS FUNCTION

2. A MINIMIZATION OR OPTIMIZATION ALGORITHM

TO SUMMARIZE: we need two key elements

1. A LOSS FUNCTION

2. A MINIMIZATION OR OPTIMIZATION ALGORITHM

THIS IS COMMON TO ALL MACHINE LEARNING ALGORITHMS

1. DEFINE A LOSS FUNCTION

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

For example: $(F_W(\vec{x_i}) - \vec{y_i})^2$ (MSE LOSS FUNCTION)

2. MINIMIZE THE EMPIRICAL RISK WITH OPTIMIZATION

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

MINIMIZE THE RISK

1. DEFINE A LOSS FUNCTION

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

For example: $(F_W(\vec{x_i}) - \vec{y_i})^2$ (MSE LOSS FUNCTION)

MORE DETAILS ABOUT LOSS FUNCTIONS IN THE ANN SECTION

2. MINIMIZE THE EMPIRICAL RISK WITH OPTIMIZATION

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

MINIMIZE THE RISK

EMPIRICAL RISK?

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

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

EMPIRICAL RISK?

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

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

OBSERVED DATASET

EMPIRICAL RISK?

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

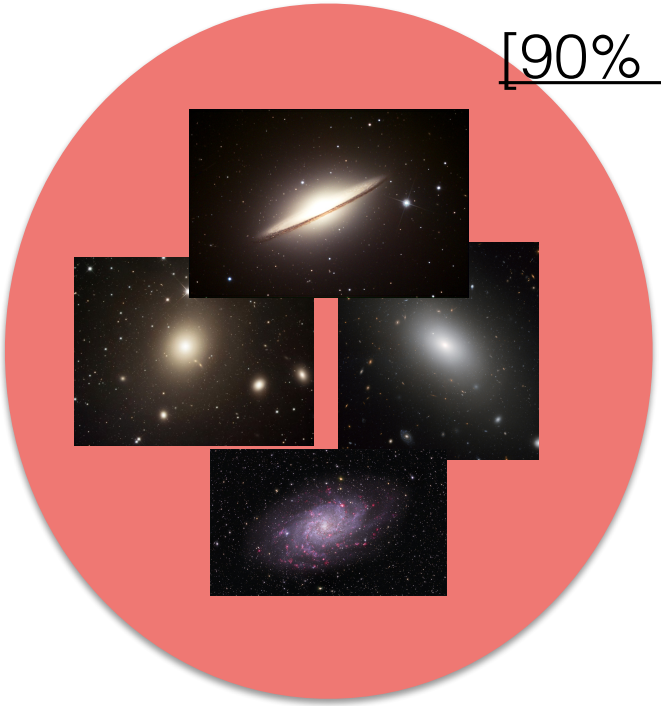
WE ARE MINIMIZING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

ALL “GALAXIES IN THE UNIVERSE”

OBSERVED DATASET

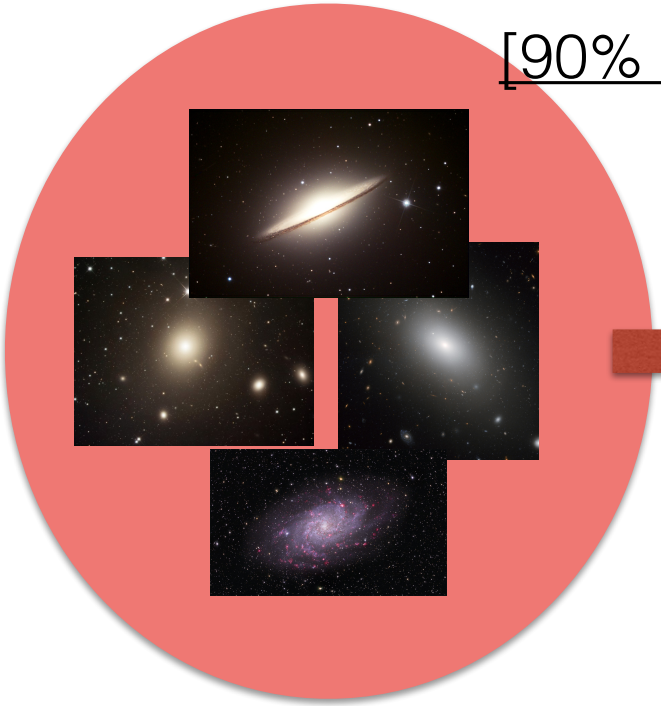
IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

[90% E, 10% S]



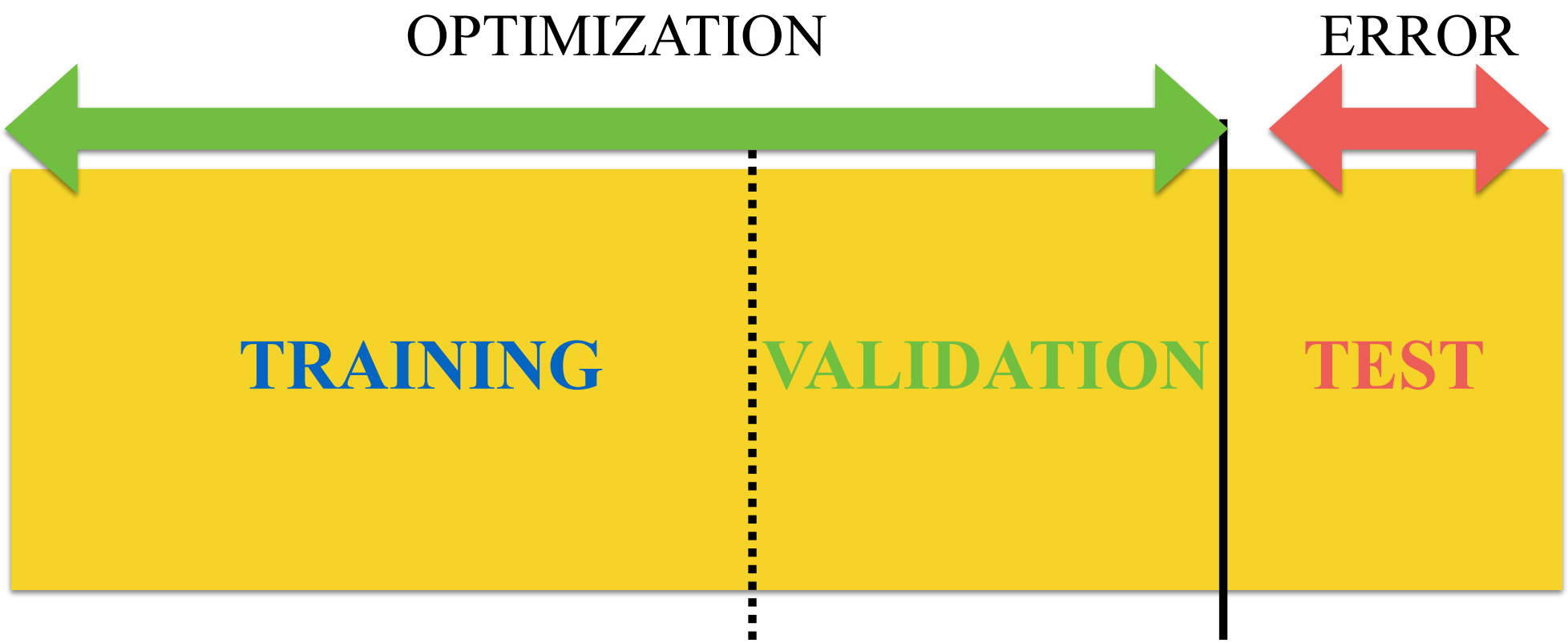
IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

[90% E, 10% S]



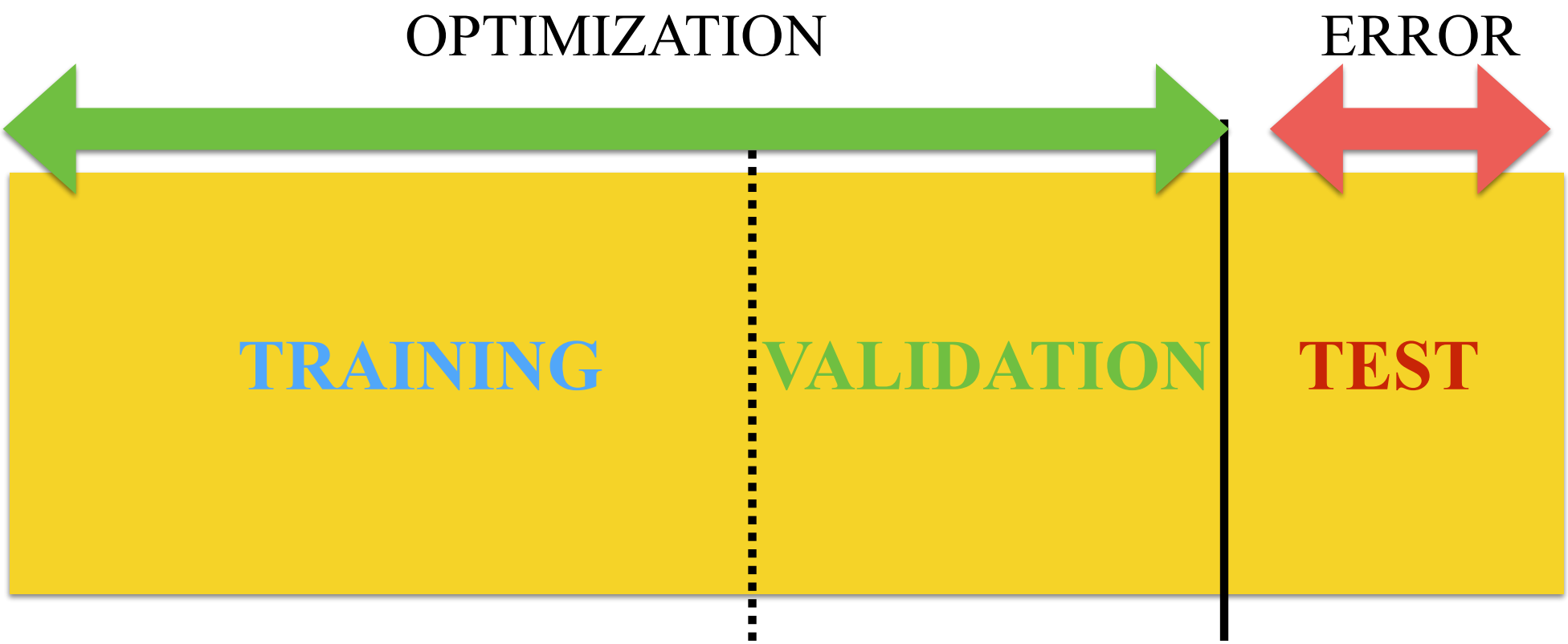
IF I SAY THAT ALL GALAXIES IN THE UNIVERSE ARE ELLIPTICALS I WILL BE RIGHT 90% OF THE TIMES

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

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

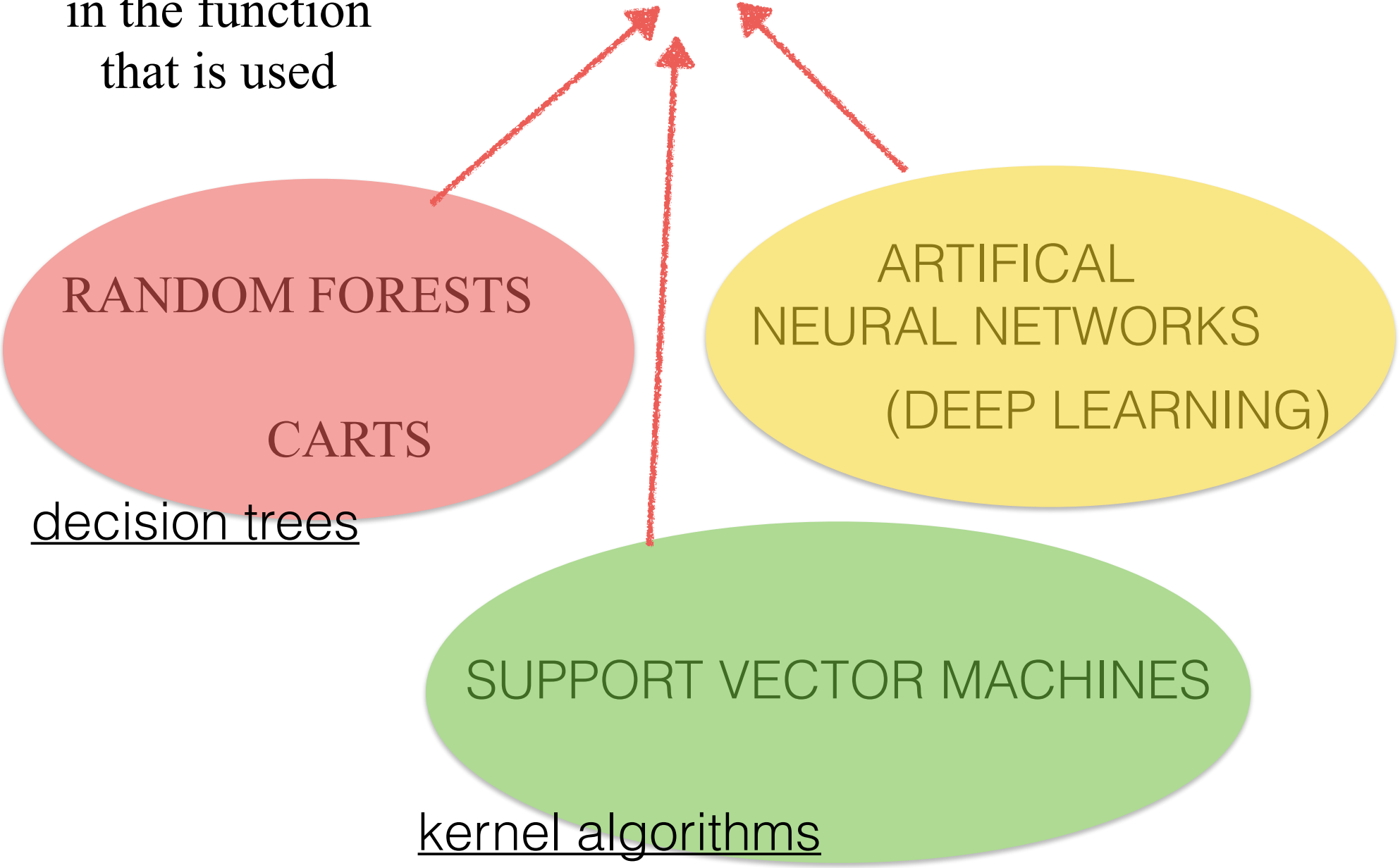
weights to be learned

epoch

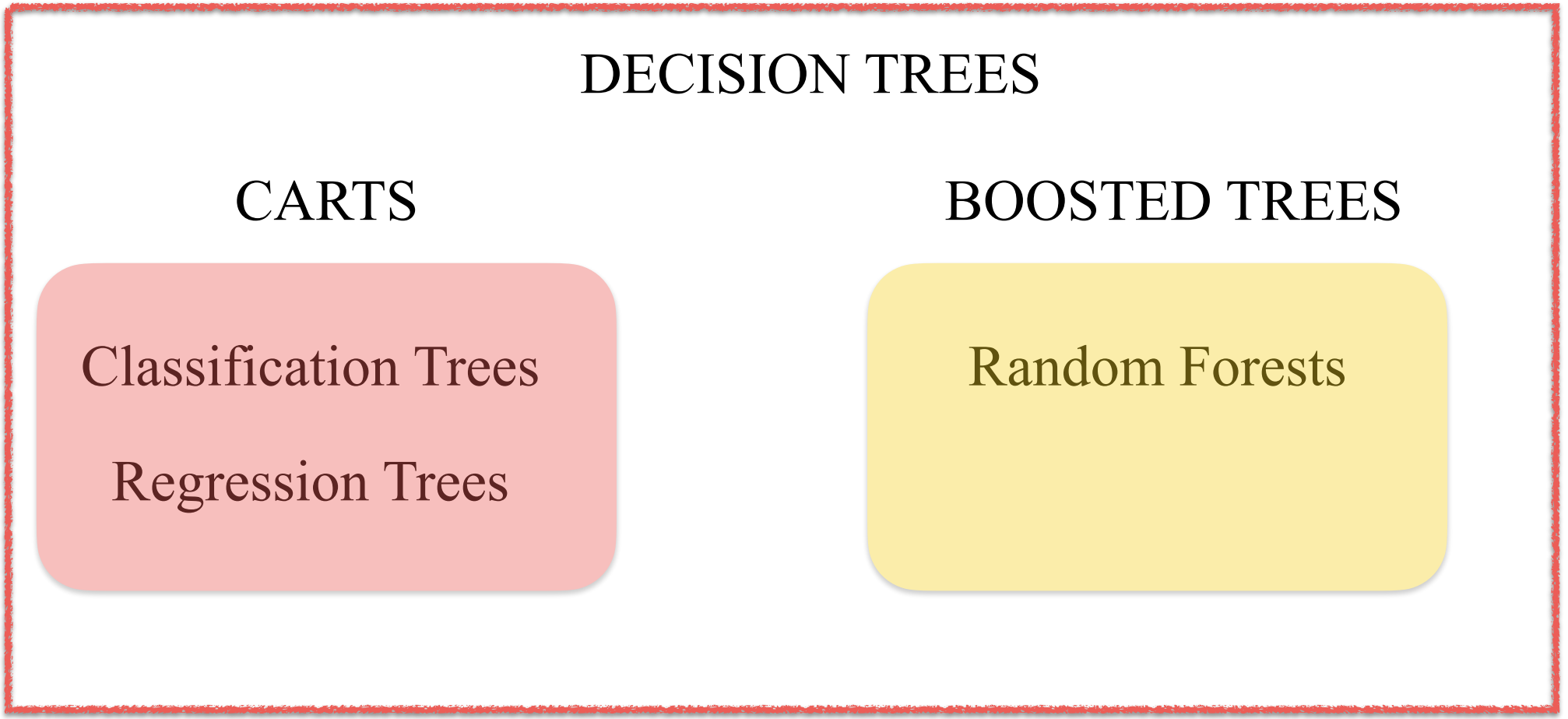
learning rate
(hyper parameter)

The differences are in the function that is used

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

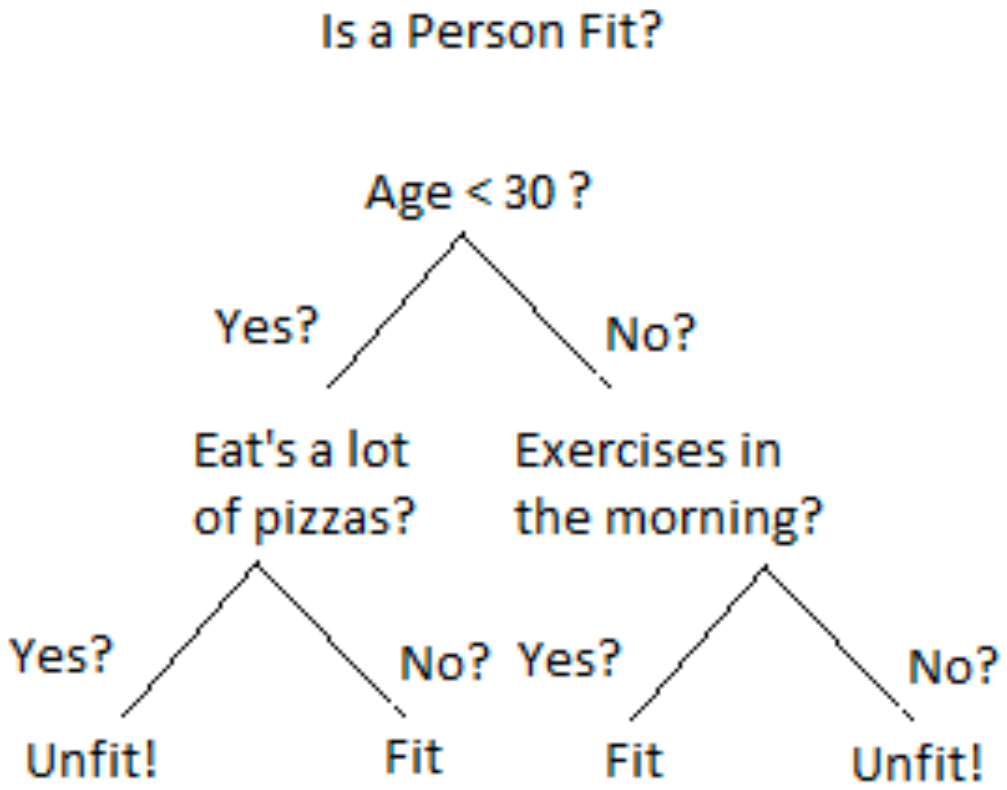


DECISION TREES ALGORITHMS

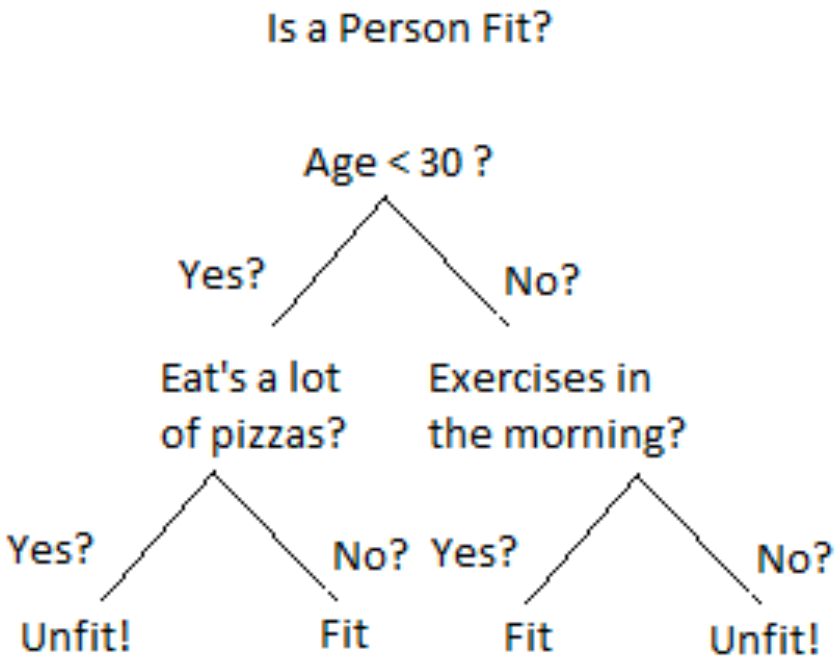


CLASSIFICATION AND REGRESSION TREES (CARTS)

THIS IS THE SIMPLEST AND MORE INTUITIVE MACHINE LEARNING ALGORITHM



DECISION TREES (CARTS)



IT IS BUILT IN AN ITERATIVE WAY

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

2. REPEAT STEP 1 WITH ANOTHER PARAMETER

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

DECISION TREES (CARTS)

TYPICAL METRICS USED:

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

Decision trees (CARTS)

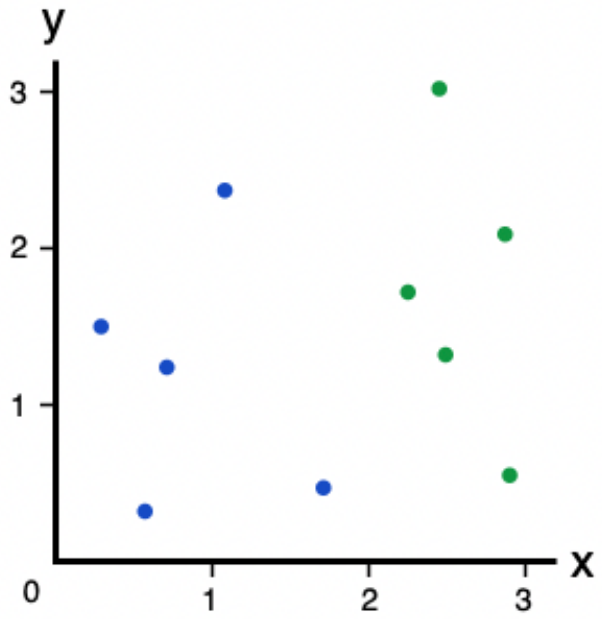
TYPICAL LOSS FUNCTION USED:

GINI IMPURITY

$$\sum_{j=1}^{c} p_j \times (1 - p_j)$$

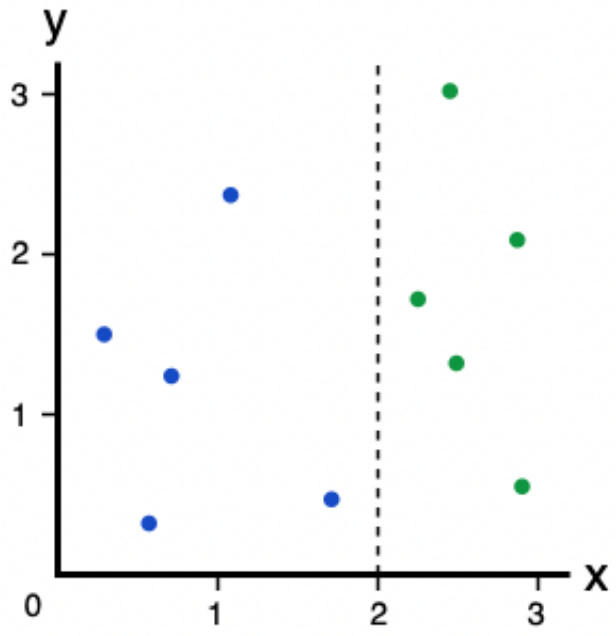
fraction of
objects in each class given
a split value

A PERFECT SPLIT GIVES G=0,
SO WE WANT TO MINIMIZE G



$$G = 0.5 \times (1 - 0.5) + 0.5 \times (1 - 0.5) = 0.5$$

(random selection)



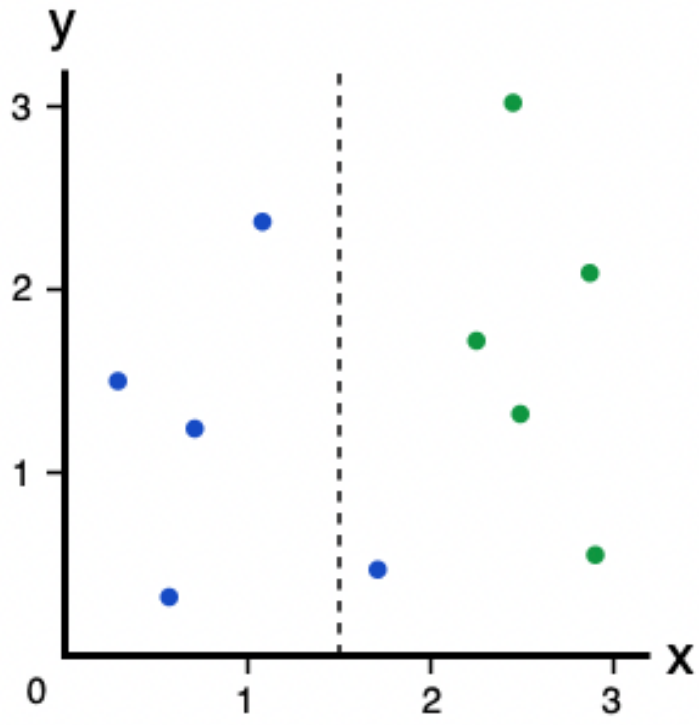
$$G_{left} = 1 \times (1 - 1) + 1 \times (1 - 1) = 0$$

$$G_{right} = 1 \times (1 - 1) + 1 \times (1 - 1) = 0$$

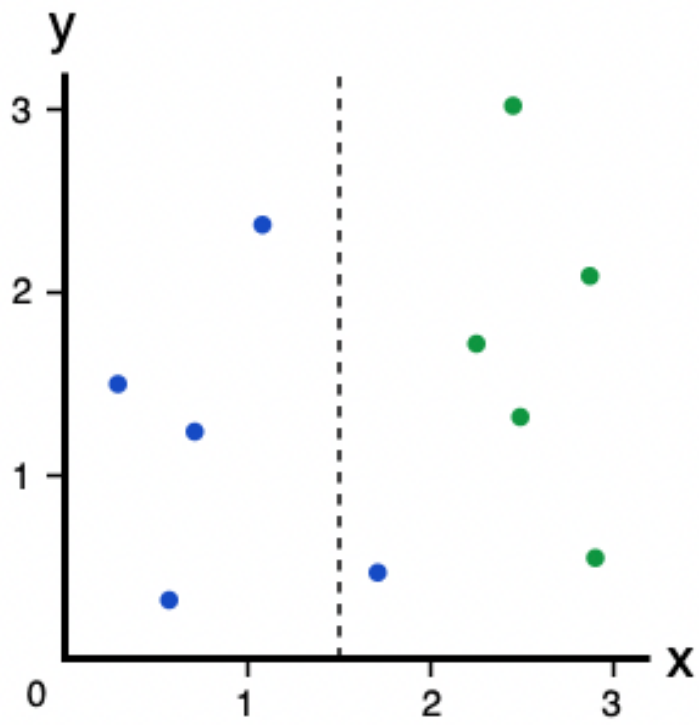
(perfect split)

Everything in between is better than random, worse than perfect

Gini?



Gini?



$$G_{right} = \frac{1}{6} \times (1 - \frac{1}{6}) + \frac{5}{6} \times (1 - \frac{5}{6}) = 0.278$$

$$G_{left} = 0$$

$$G_{avg} = (0.4 \times 0) + (0.6 \times 0.278) = 0.167$$

DECISION TREES (CARTS)

TRAINING DATA

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

(parameters)

[b/a, color, size, SFR …]

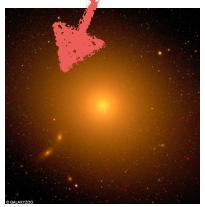
$(\vec{x_2}, \vec{y_2})$

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

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

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

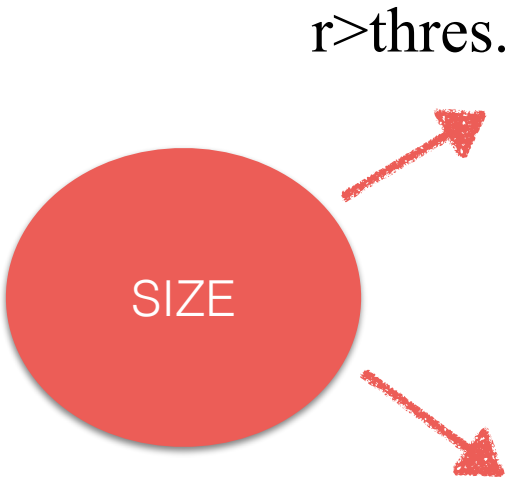
(classes)



[elliptical / spiral]

DECISION TREES (CARTS)

[elliptical / spiral]



LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

DECISION TREES (CARTS)

[elliptical / spiral]

SIZE

r>thres.

COLOR

c>thres.

c<thres.

r<thres.

COLOR

c>thres.

c<thres.

LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

Let's assume the following sample of galaxies:

Label	Size
	3 kpc
	10 kpc
	4 kpc
	8 kpc
	4 kpc
	9 kpc

When poll is active, respond at **pollev.com/marchuertasc257**

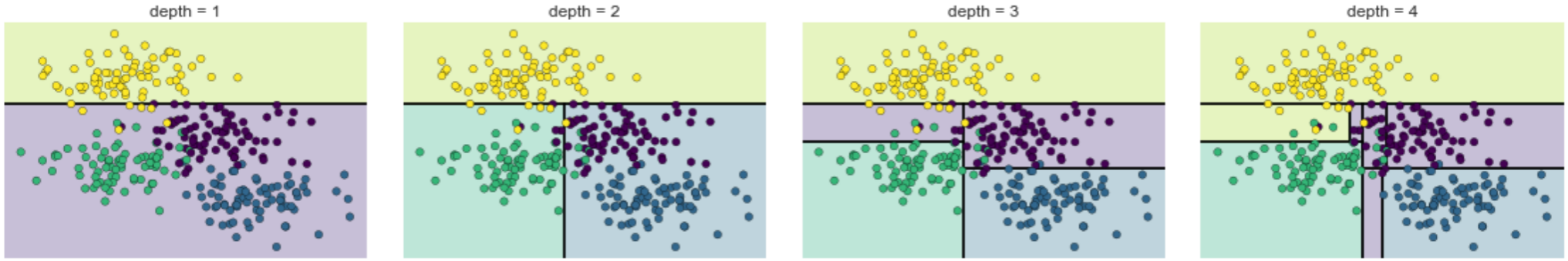
What would be the Gini Impurity with a split value of 6 kpc

- 5/9
- 0
- 1
- 3/8
- None of the above

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

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

[b/a, color, size, SFR …]

TRAINING DATA

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

(parameters)

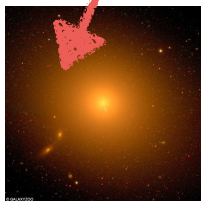
$(\vec{x_2}, \vec{y_2})$

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

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

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

(classes)



[elliptical / spiral]

RANDOM FORESTS

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

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

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

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

RANDOM FORESTS

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

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

[elliptical / spiral]

SIZE

r>thres.

COLOR

c>thres.

c<thres.

r<thres.

COLOR

c>thres.

c<thres.

LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

RANDOM FORESTS

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

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

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

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

[elliptical / spiral]

SIZE

r>thres.

Sersic

s>thres.

s<thres.

r<thres.

Sersic

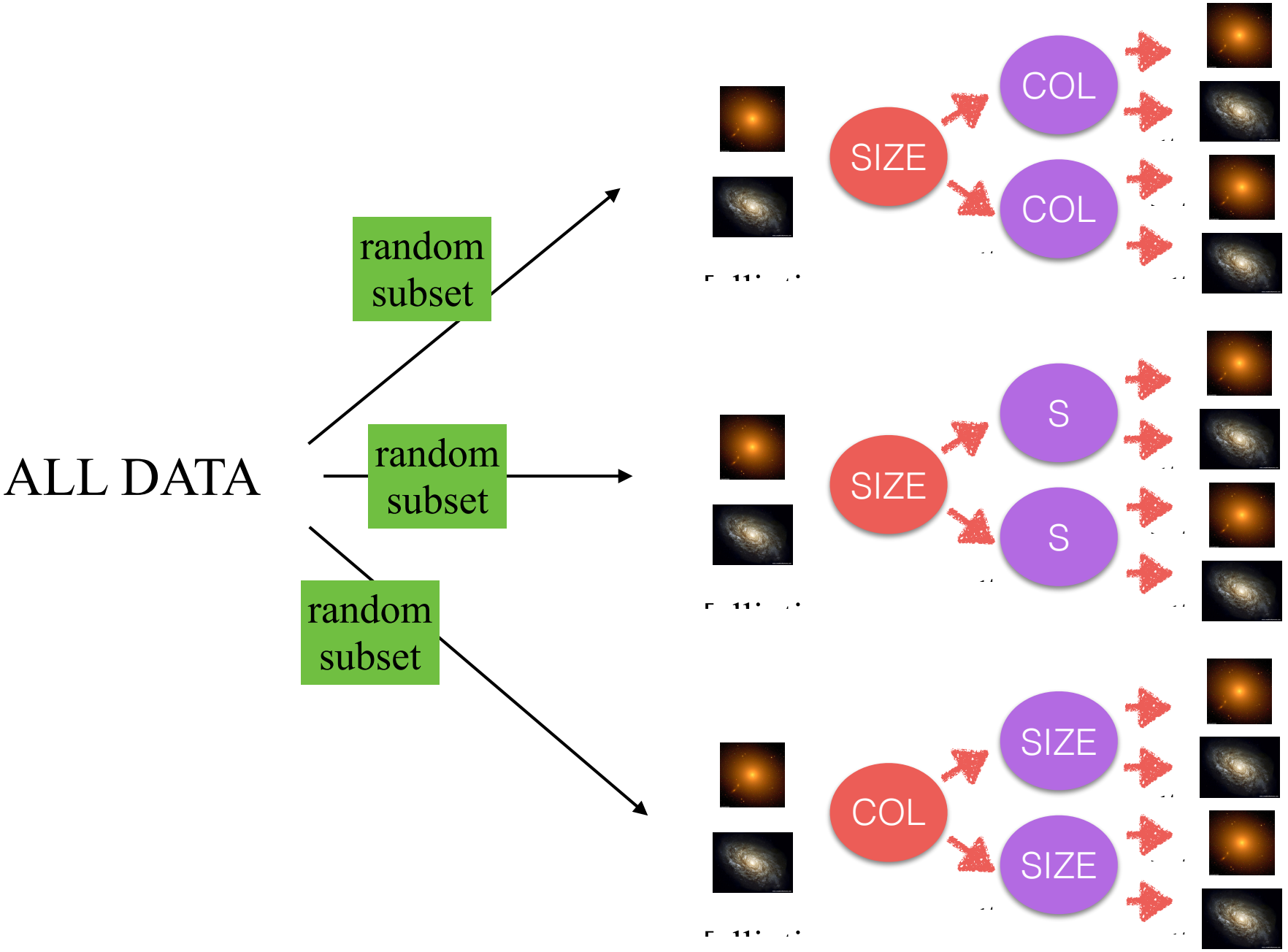
s>thres.

s<thres.

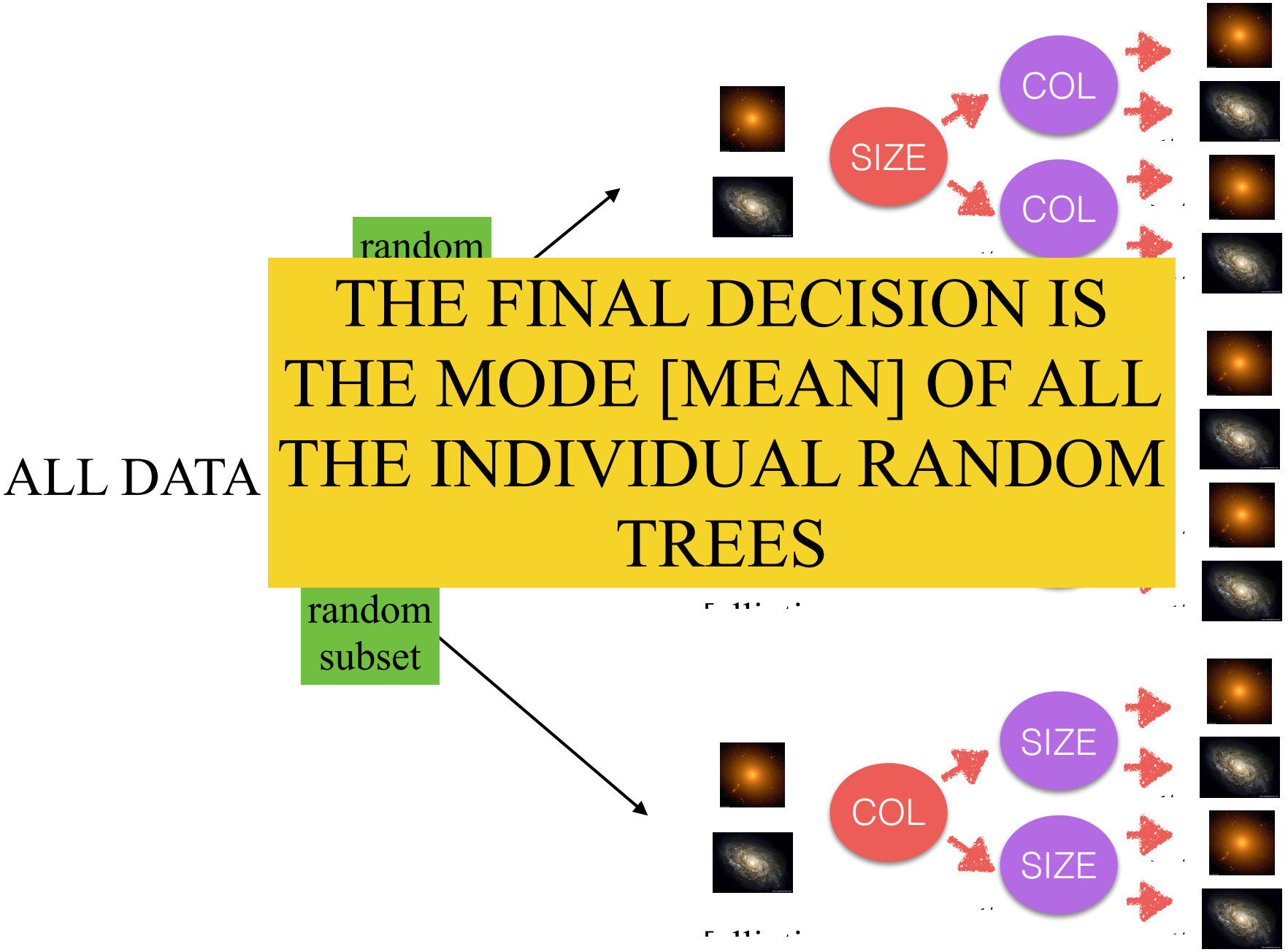
LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

ANDOM FORESTS



RANDOM FORESTS



Random Forests

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

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

RANDOM FORESTS

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

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

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

IMPORTANCE OF PARAMETERS TO PREDICT IF A GALAXY IS QUENCHED

Teimiroonia+16

PRACTICAL NOTE: *Python scikit learn*

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

sklearn.ensemble.RandomForestClassifier

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

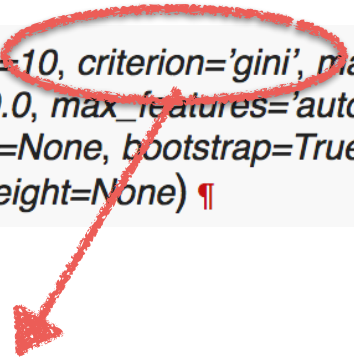
NUMBER OF RANDOM TREES

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]



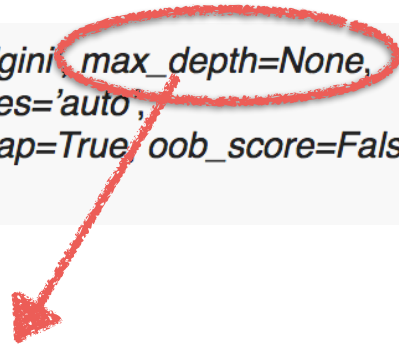
METRIC TO MINIMIZE

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

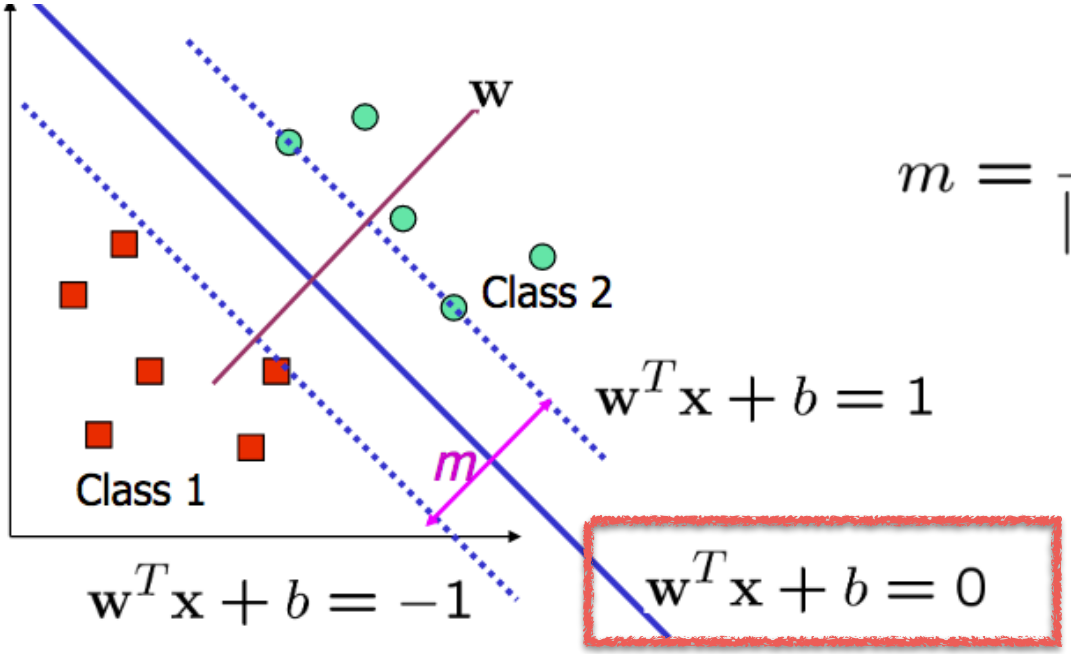
class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, 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,
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

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

Support Vector Machines

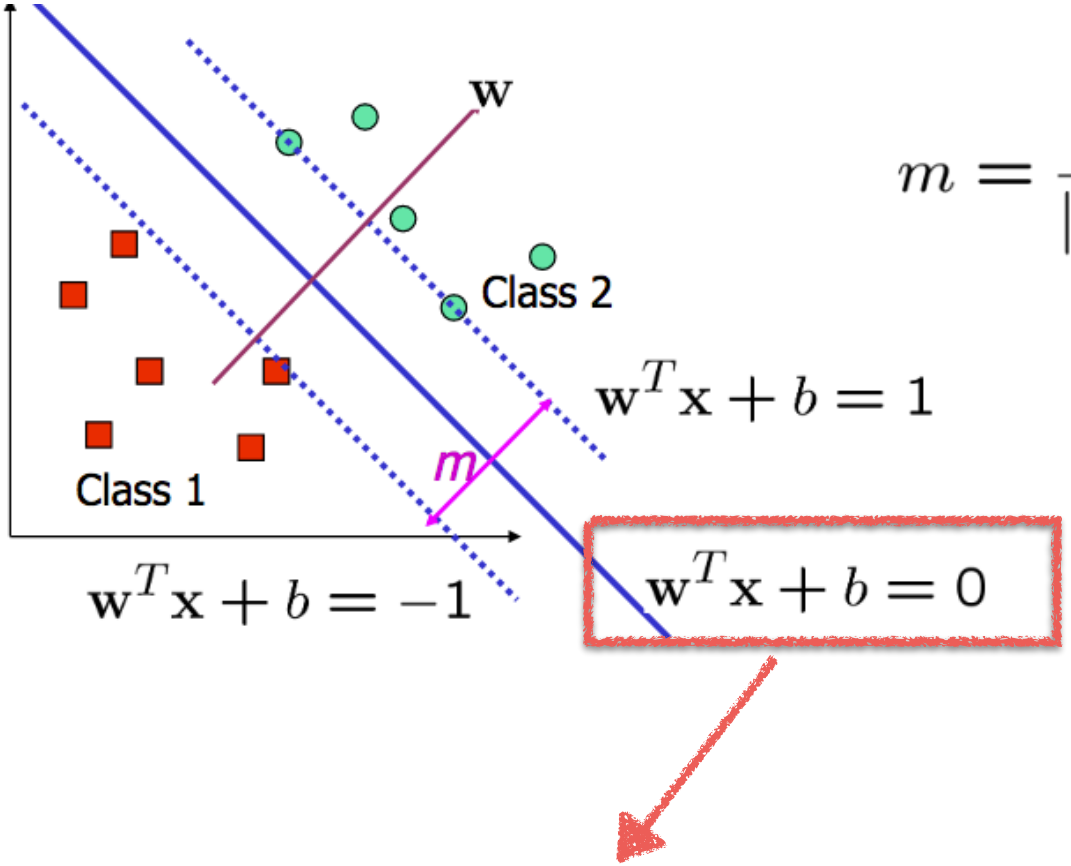


$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

[Credit: M. Law]

Support Vector Machines



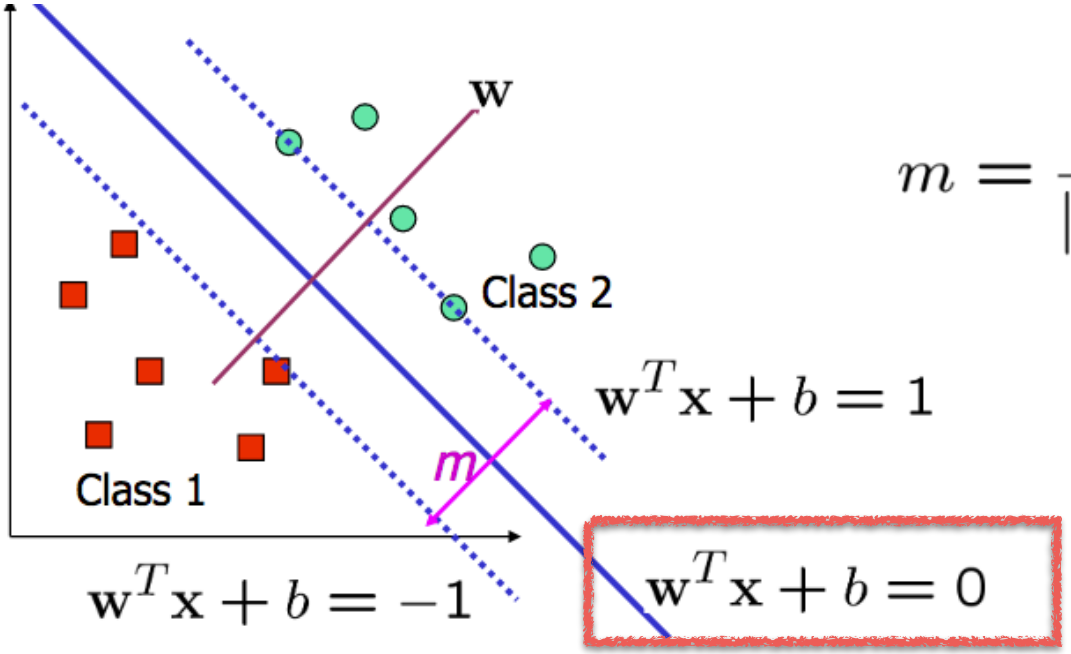
$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

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

[Credit: M. Law]

Support Vector Machines



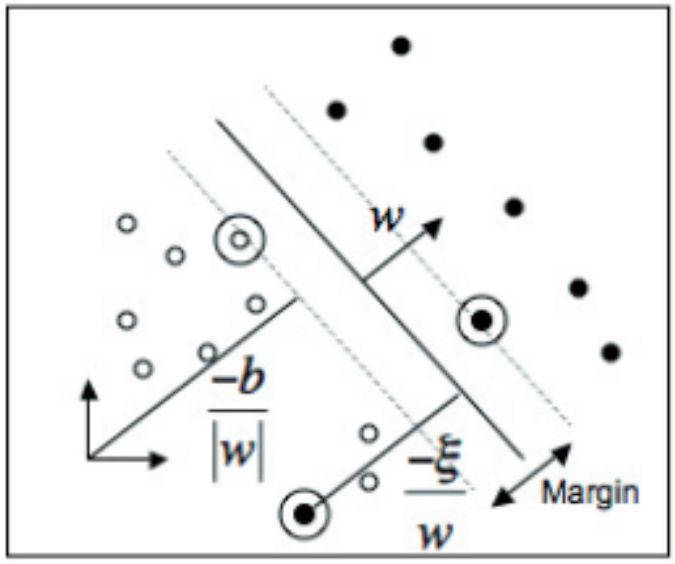
$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

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

[Credit: M. Law]

Support Vector Machines



Huertas-Company+08

IN MOST CASES THE DATA ARE NON LINEARLY SEPARABLE

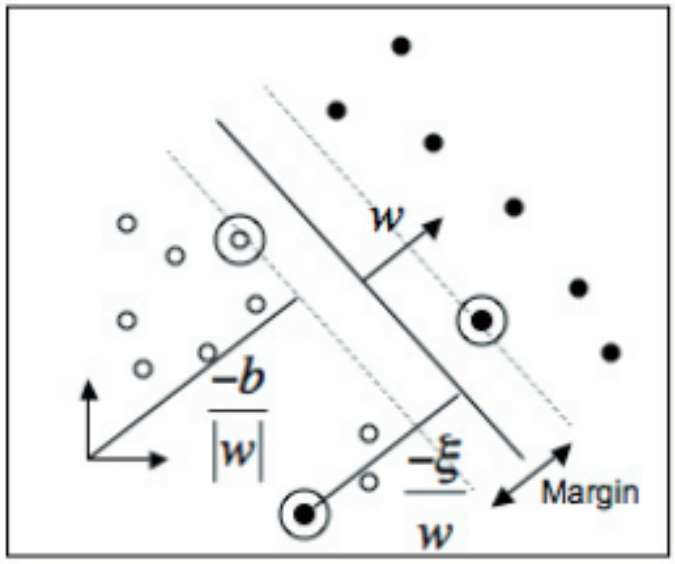
$$\mathbf{x_i}.\mathbf{w} + b \geq +1 - \epsilon_i$$

$$\mathbf{x_i}.\mathbf{w} + b \geq +1 + \epsilon_i$$

THE CONDITION IS RELAXED BY ADDING A POSITIVE STACK VARIABLE

Huertas-Company+08

Support Vector Machines



Huertas-Company+08

IN MOST CASES THE DATA ARE NON LINEARLY SEPARABLE

$$\mathbf{x_i}.\mathbf{w} + b \geq +1 - \epsilon_i$$

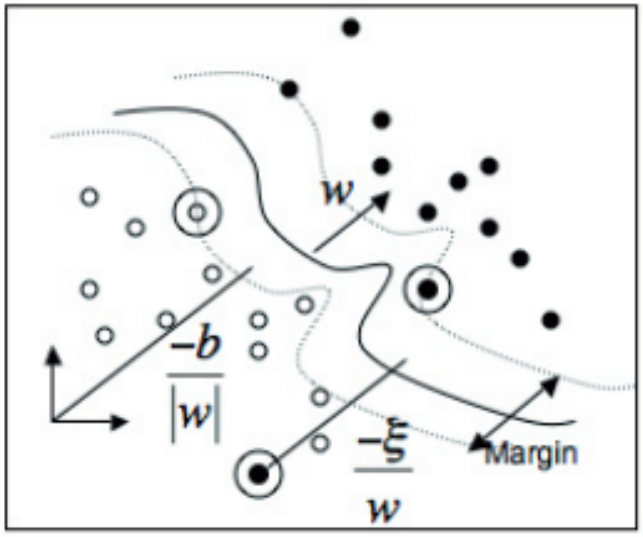
$$\mathbf{x_i}.\mathbf{w} + b \geq +1 + \epsilon_i$$

THE CONDITION IS RELAXED BY ADDING A POSITIVE STACK VARIABLE

THE FUNCTION TO MINIMIZE: $\frac{||w^2||}{2} + T[\sum \epsilon_i]$

THE LARGER T, THE MORE ERRORS ARE PENALIZED

Support Vector Machines



Huertas-Company+08

THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

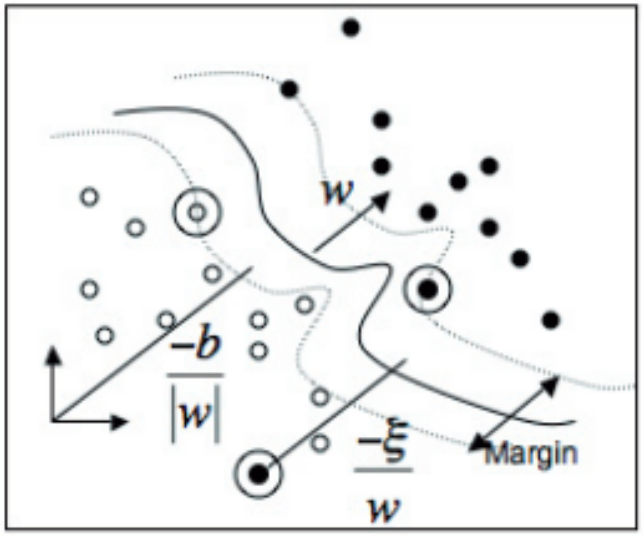
SINCE WE ONLY NEED TO COMPUTE INNER PRODUCTS IN THIS SPACE

$$x_i . x_j$$

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

$$\Phi(x_i).\Phi(x_j)$$

Support Vector Machines



Huertas-Company+08

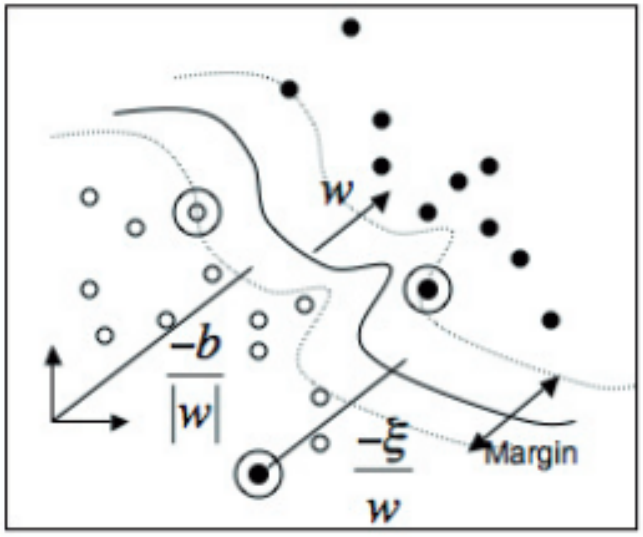
THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

Support Vector Machines



Huertas-Company+08

THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

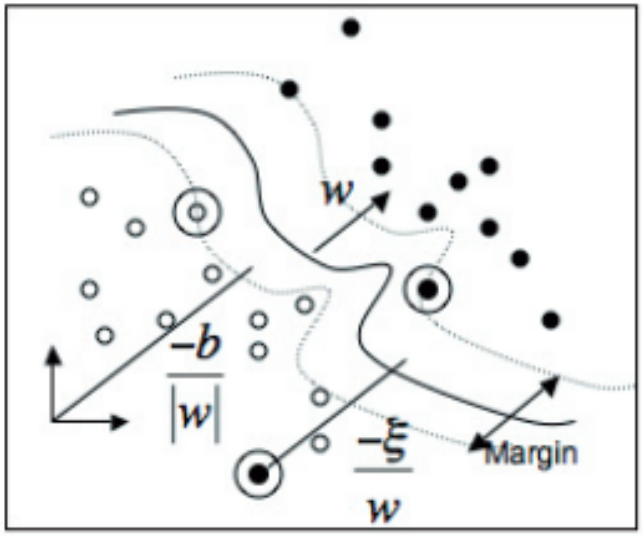
IF THERE IS A **KERNEL FUNCTION** SO THAT:

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

THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

Support Vector Machines



Huertas-Company+08

THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

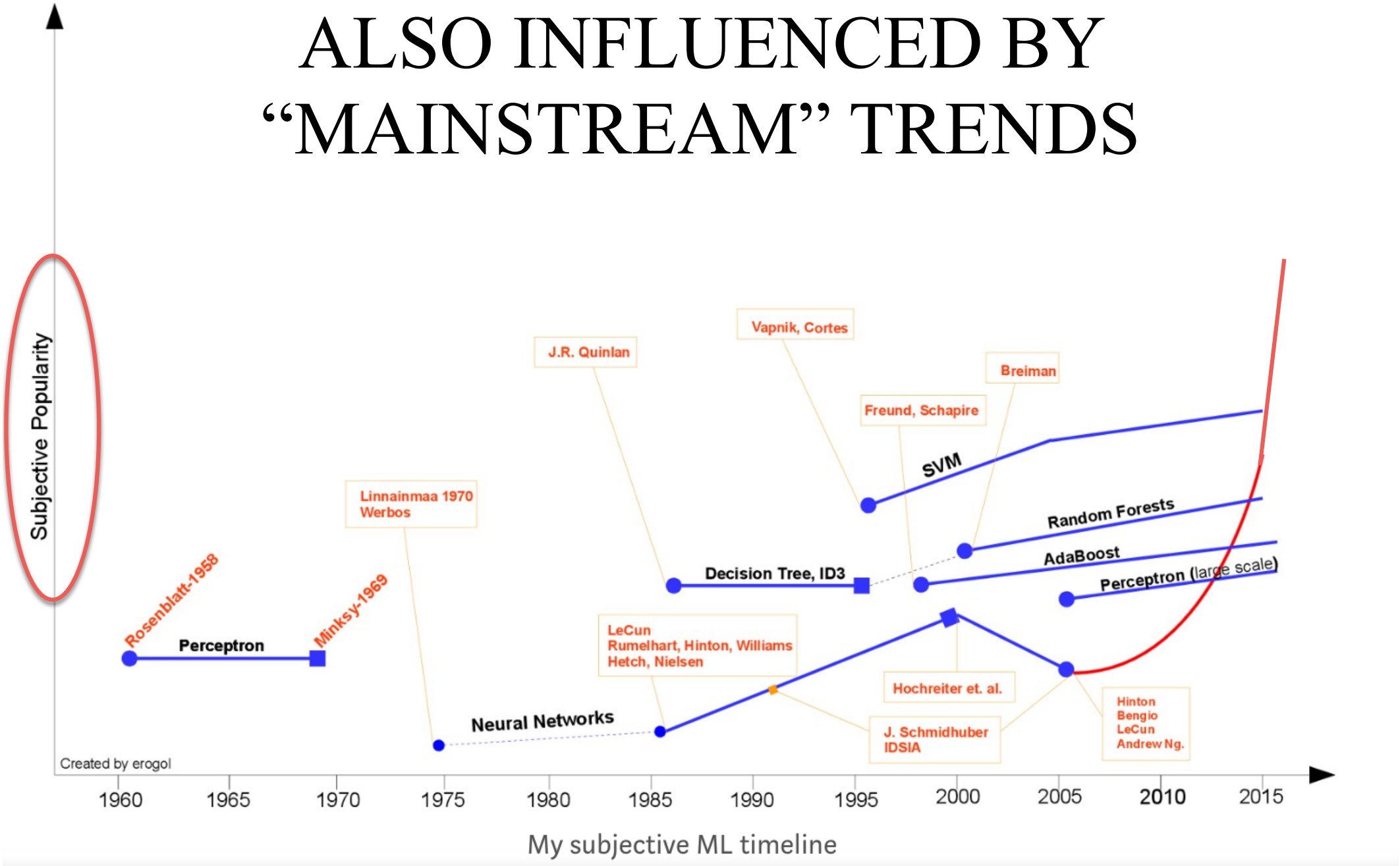
THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

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

[GAUSSIAN RBF]

ALSO INFLUENCED BY "MAINSTREAM" TRENDS

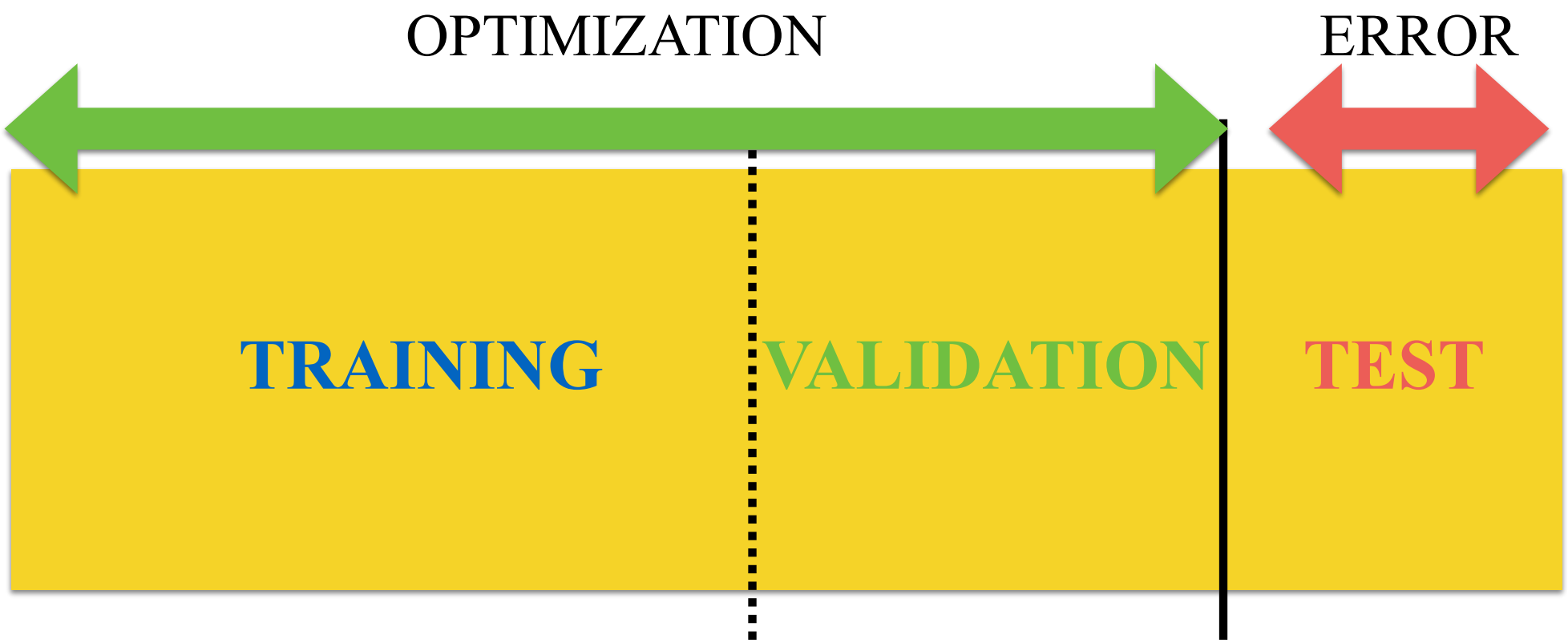


Source

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

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

REMEMBER



training set: use to train the classifier

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

test set: use to train the classifier

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY

true positives

true negatives

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

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY

NOT VERY INFORMATIVE IF UNBALANCED CLASSES

true po ue negatives

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

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

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.]

- ver Operating Characteristic)

YOU WANT THIS TO BE AS BIG AS POSSIBLE

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1. $TPR = \frac{TP}{TP +$ TRUE POSITIVE RATE [Completeness]

YOU WANT THIS TO BE AS SMALL AS POSSIBLE

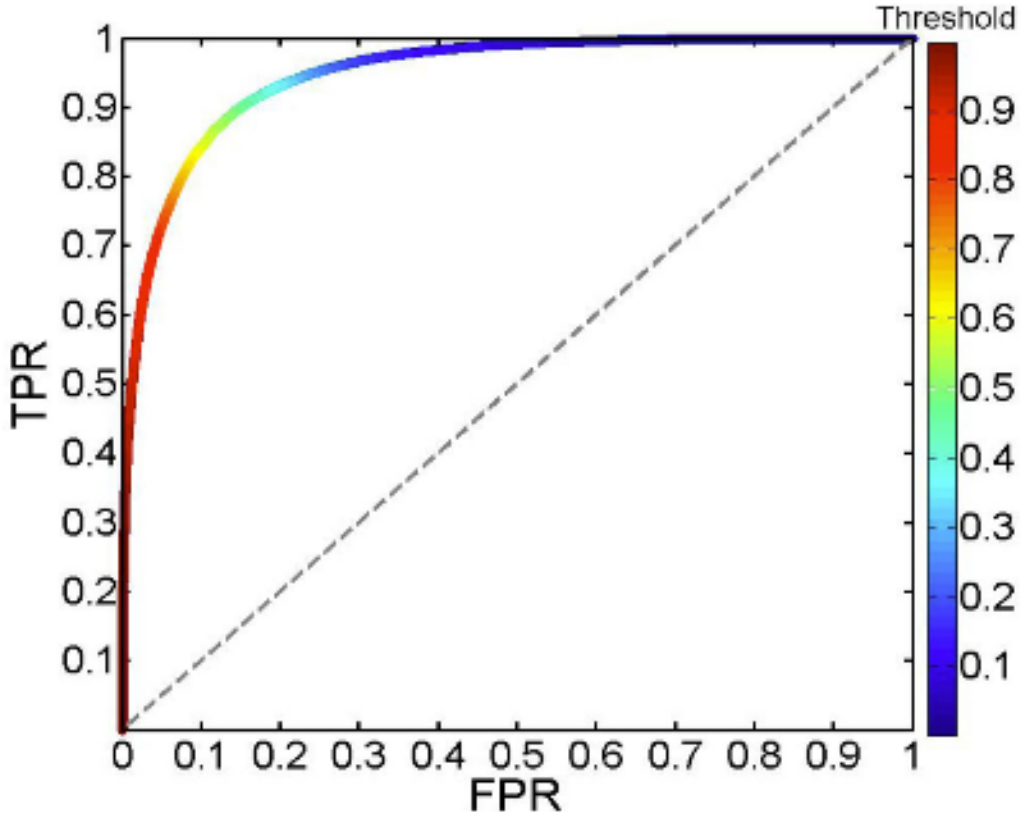
"Fraction of positive examples classified correctly"

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

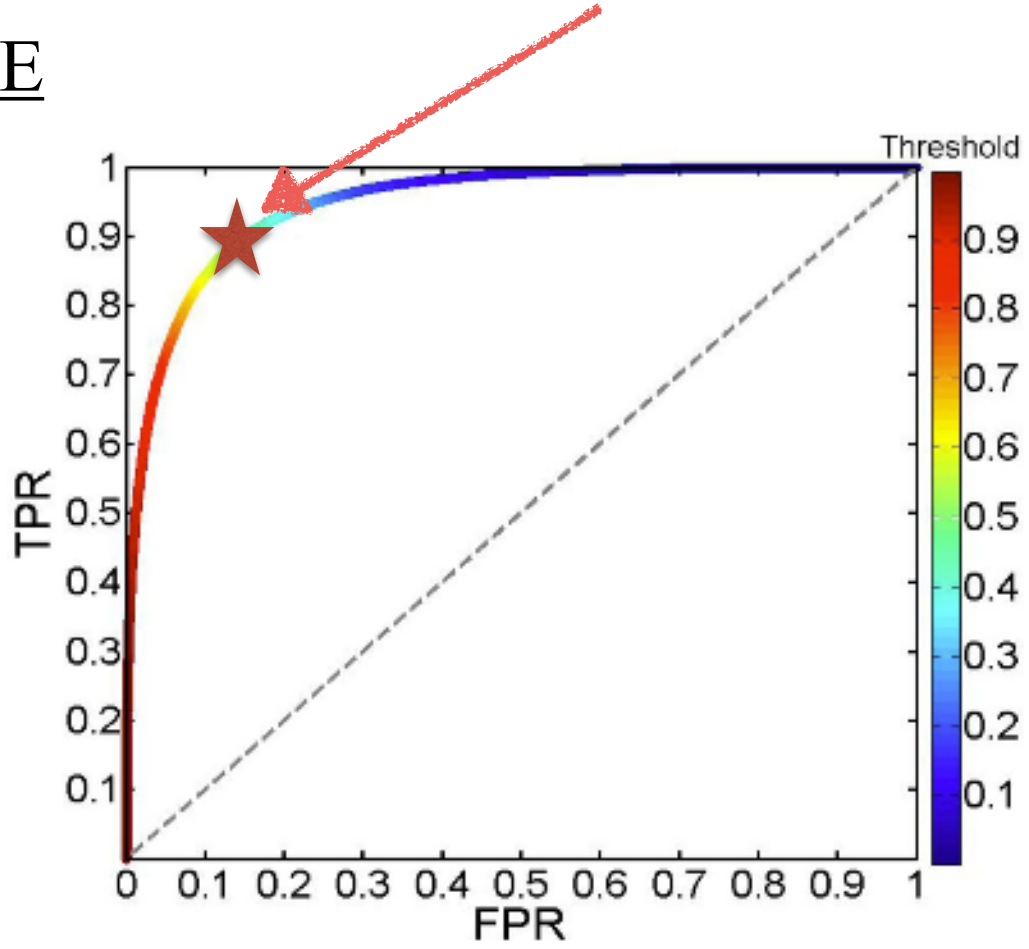


Teimiroonia+16

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 FOR FOR A GIVEN THRESHOLD

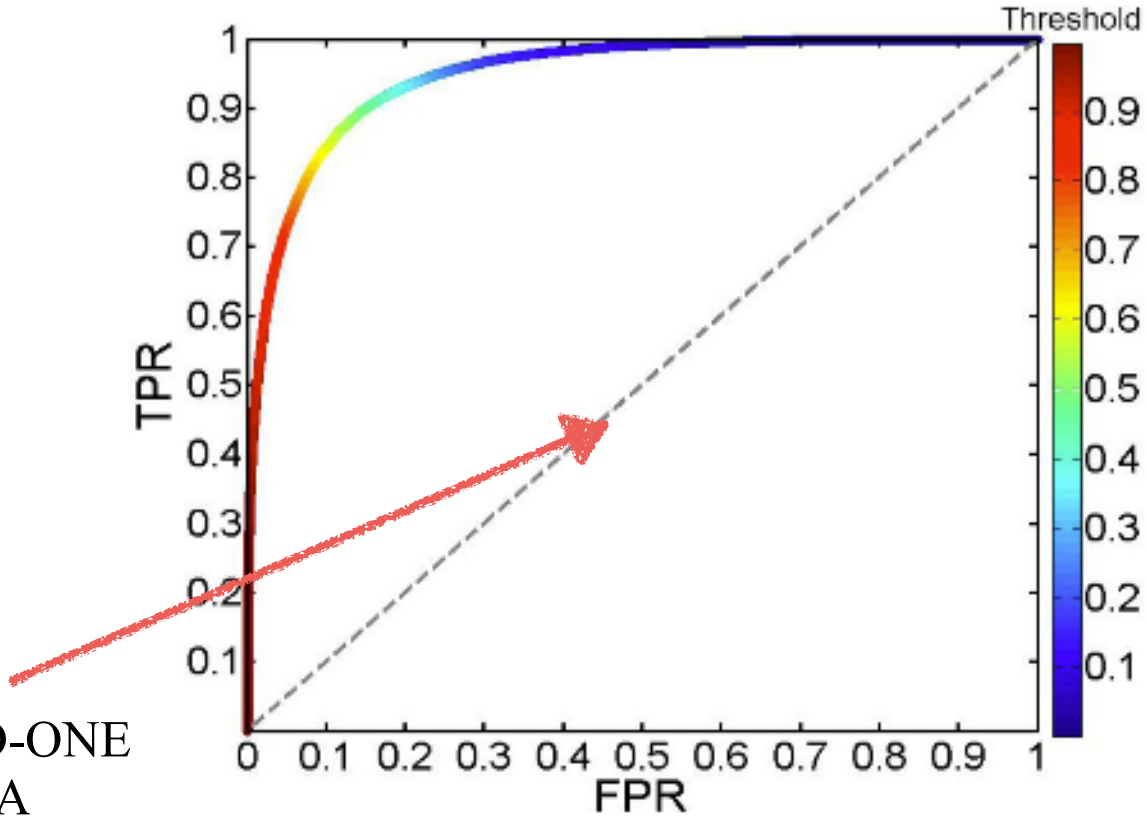
ROC CURVE



Teimiroonia+16

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

ROC CURVE

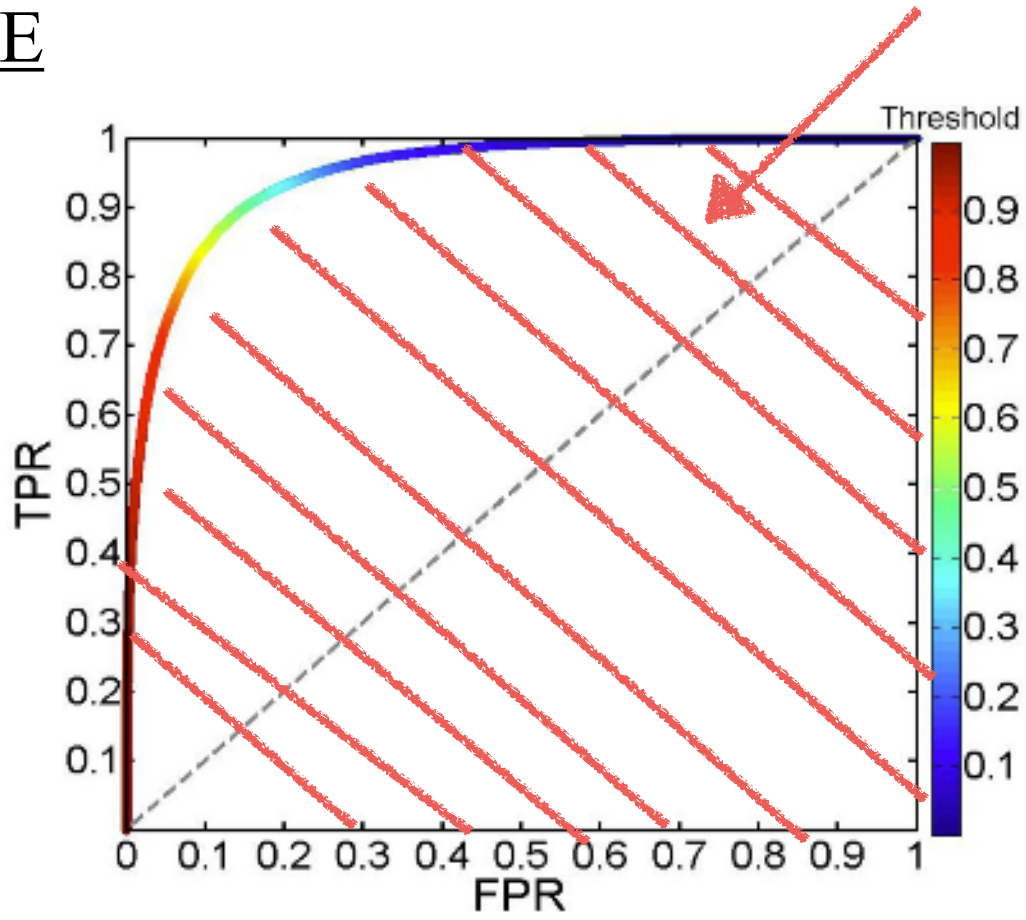


THE ONE-TO-ONE LINE IS A RANDOM CLASSIFICATION

Teimiroonia+16

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

Teimiroonia+16

Evaluation of results

THE P-R CURVE (Precision - Recall)

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

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

Evaluation of results

THE P-R CURVE (Precision - Recall)

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

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

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