Decision Trees and Ensemble Methods

Experts vs Group of Laymen

Decision Trees

The 20 Questions Approach

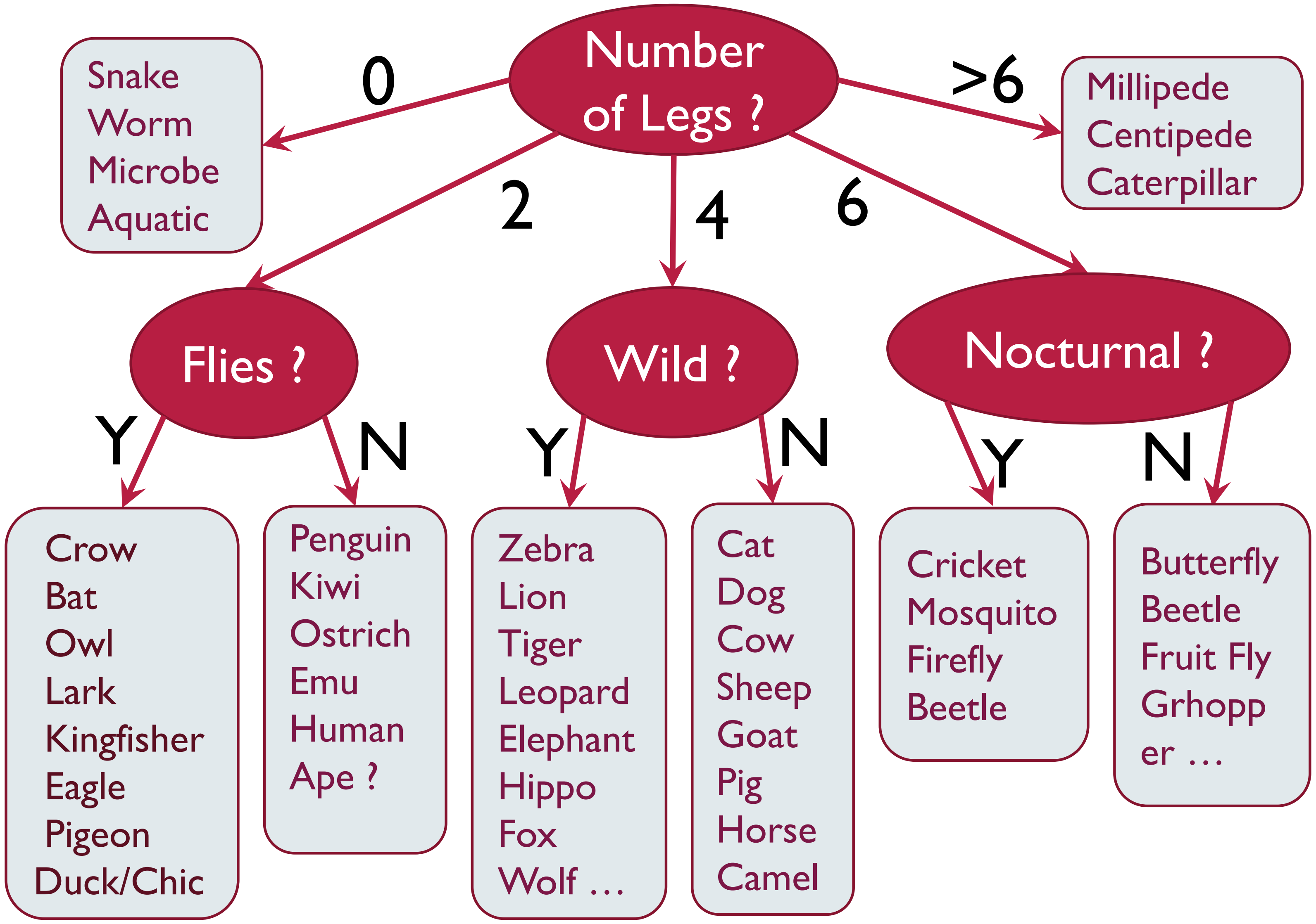
Lets Play a Game: Guess the Animal

- I am thinking of an Animal
- You can ask a set of questions (on features of the animal)
- Can you guess the animal based on my answers?
- Conditions:
  - Only Yes/No questions or questions on a single attribute
  - No questions based on the animal name itself
- Let us play

Guess the Animal

Questions

- How many legs?
- Does it fly?
- Is it a wild animal?
- Is it nocturnal?
- Fur/Feather?
- Farm Animal? …



What are we doing? (Larger Picture)

- We have possible animals
- Each has a set of attributes
- Look at 1 attribute at a time
- Narrow down the class label
- Goal:
  - Arrive at a single class label
- Can we learn the tree?
  - Which question to ask at any point?

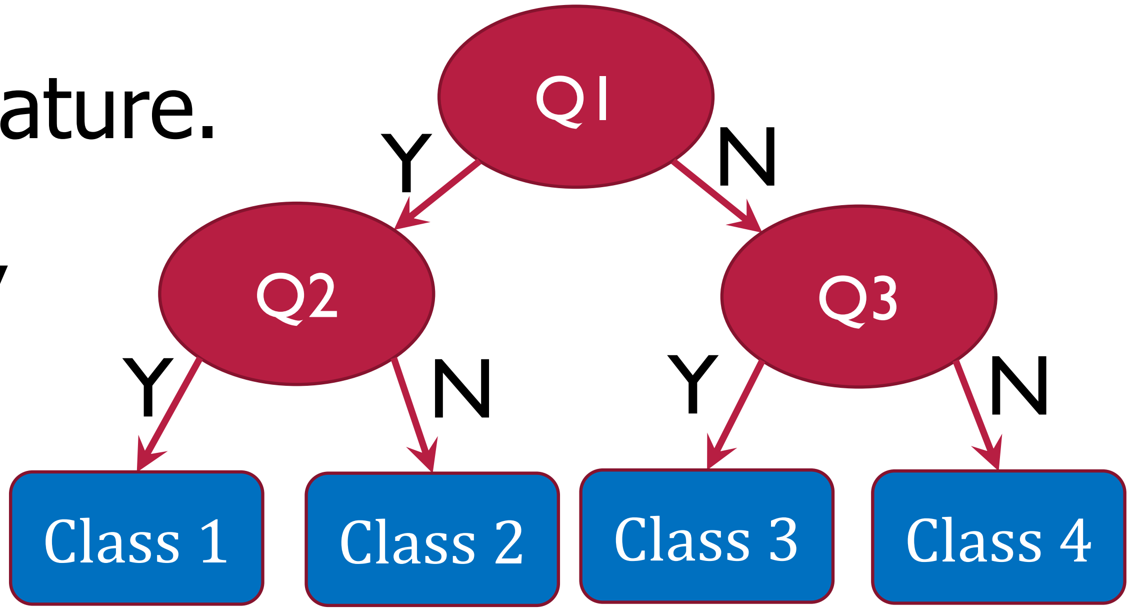
Animal	Legs	Wild	Flies	Noct	Fur/Feather	Farm
Zebra	4	Y	N	N	N	N
Horse	4	Y/N	N	N	N	Y
Cow	4	N	N	N	N	Y
Cat	4	Y/N	N	Y/N	Y	N
Penguin	2	Y	N	N	N	N
Owl	2	Y	Y	Y	Y	N
Fish	0	Y	N	Y/N	N	N
Snake	0	Y	N	Y/N	N	N
Millipede	1000	Y	N	Y	N	N
Firefly	6	Y	Y	Y	N	N
Butterfly	6	Y	Y	N	N	N

What is a Good Question?

- Which question if answered will reduce the possible number of animals the most?
- More precisely, try to reduce our Uncertainty the most
- Mathematically, reduce our Entropy the most
  - Will look at this in more detail soon.

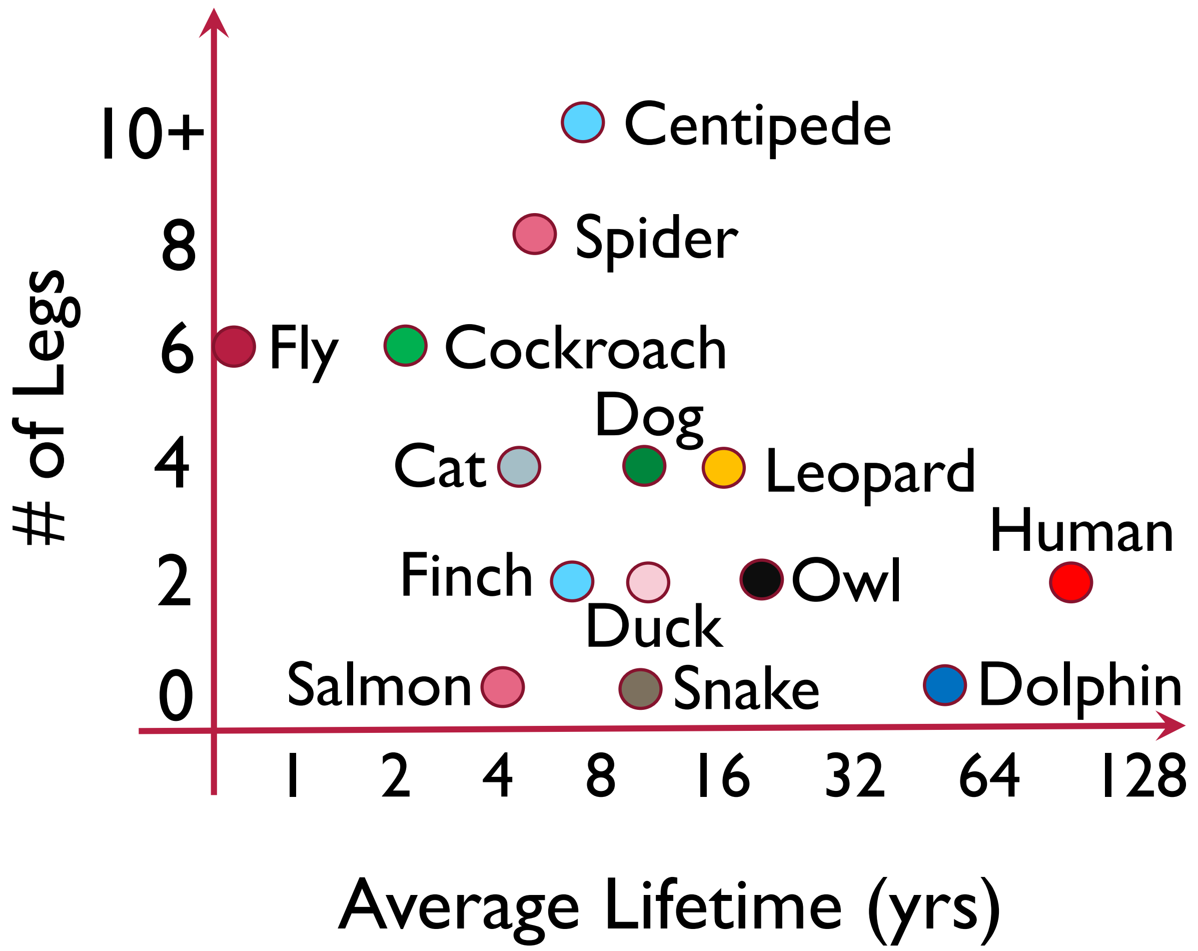
Decision Trees: Summary. Questions?

- At each step, ask the question that minimizes uncertainty
- Once the set contains only a single class, label it.
- The sequence of questions/decisions can be represented as a tree: The Decision Tree.
- Each question is on a single feature.
- Tree Terms: Node, Edge, Root, Leaf, Depth, Height, Path, Parent, Child.

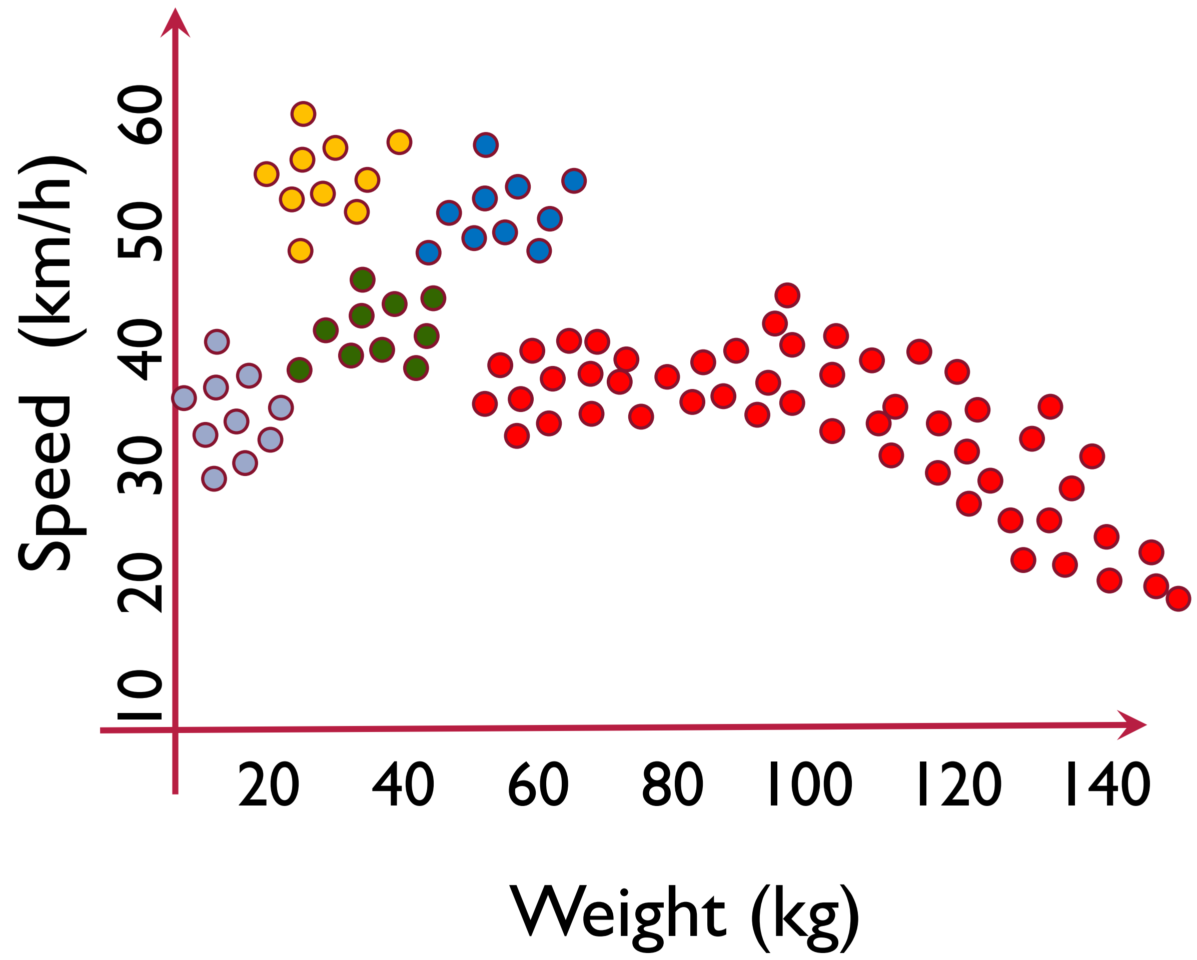


Learning from Examples

Per-Class Features



Per-Sample Features

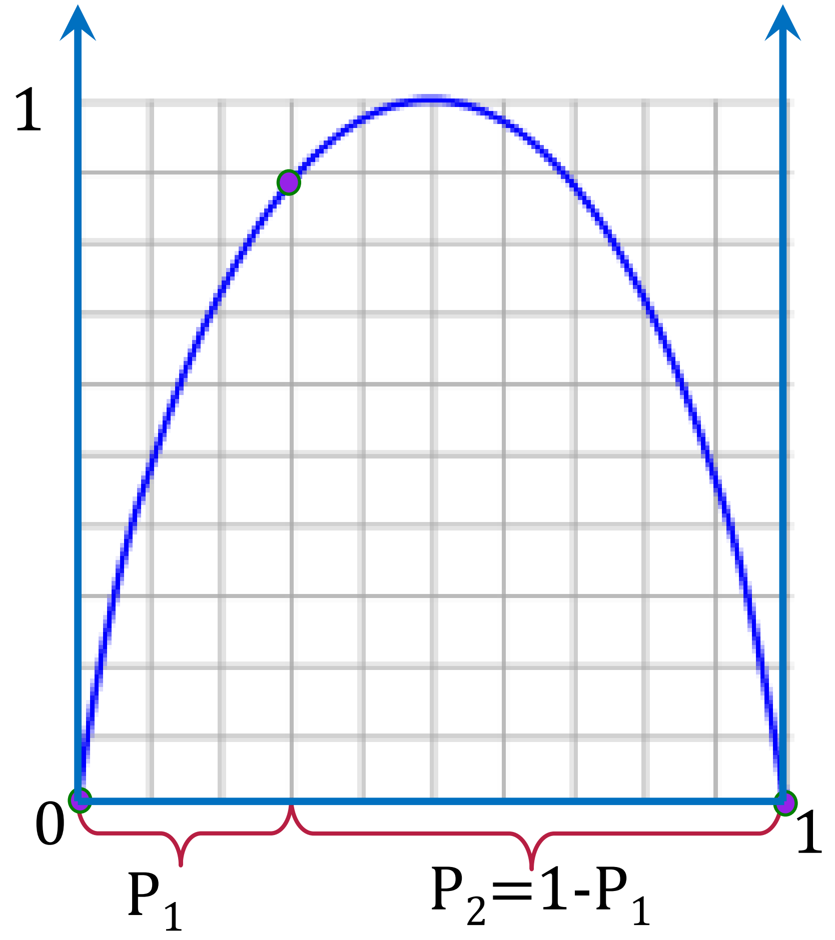


What is Entropy?

- Is a measure of Uncertainty.
- Mathematically: $\mathrm{H(X)} = -\sum_i \mathrm{P(i)}\, log_2\, \mathrm{P(i)}$.
- Assume a set contains two classes:

$$\mathrm{H} = -P_1\, log_2\, P_1 - P_2\, log_2\, P_2$$

- Is a measure of impurity
  - Not the only one



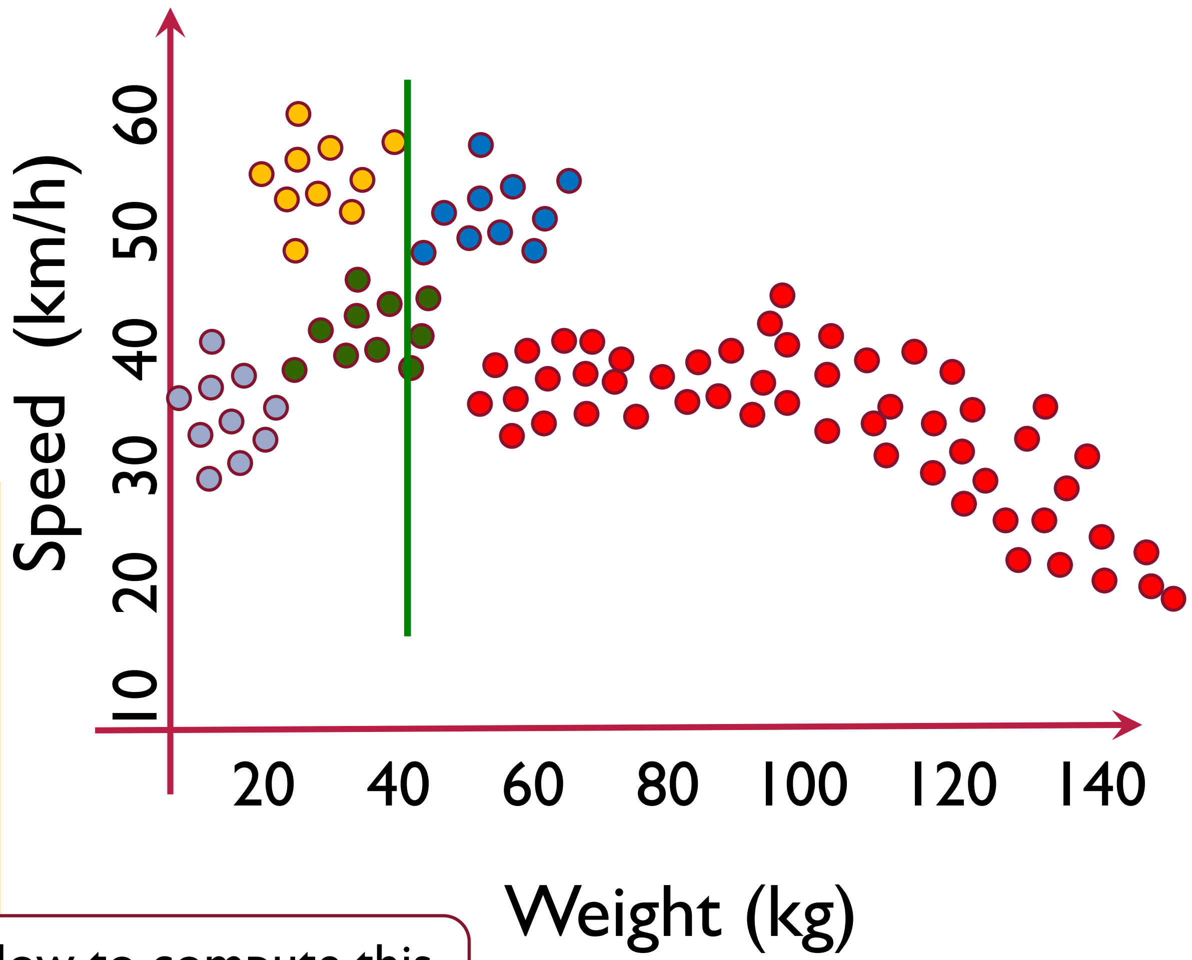
How to Compute Entropy

- Initial Entropy, H:

  $$5 \times (-0.2\, log_2\, 0.2) = 2.32$$

- Q1: Weight < 40?
- Total Entropy of Children:

$$H_1 = \frac{1}{2}\begin{pmatrix}(-0.4\, log_2 0.4) + (-0.4\, log_2 0.4) \\ + (-0.2\, log_2 0.2)\end{pmatrix} + \frac{1}{2}\begin{pmatrix}(-0.4\, log_2 0.4) + (-0.4\, log_2 0.4) \\ + (-0.2\, log_2 0.2)\end{pmatrix} = 1.52$$



How to compute this value on a calculator?

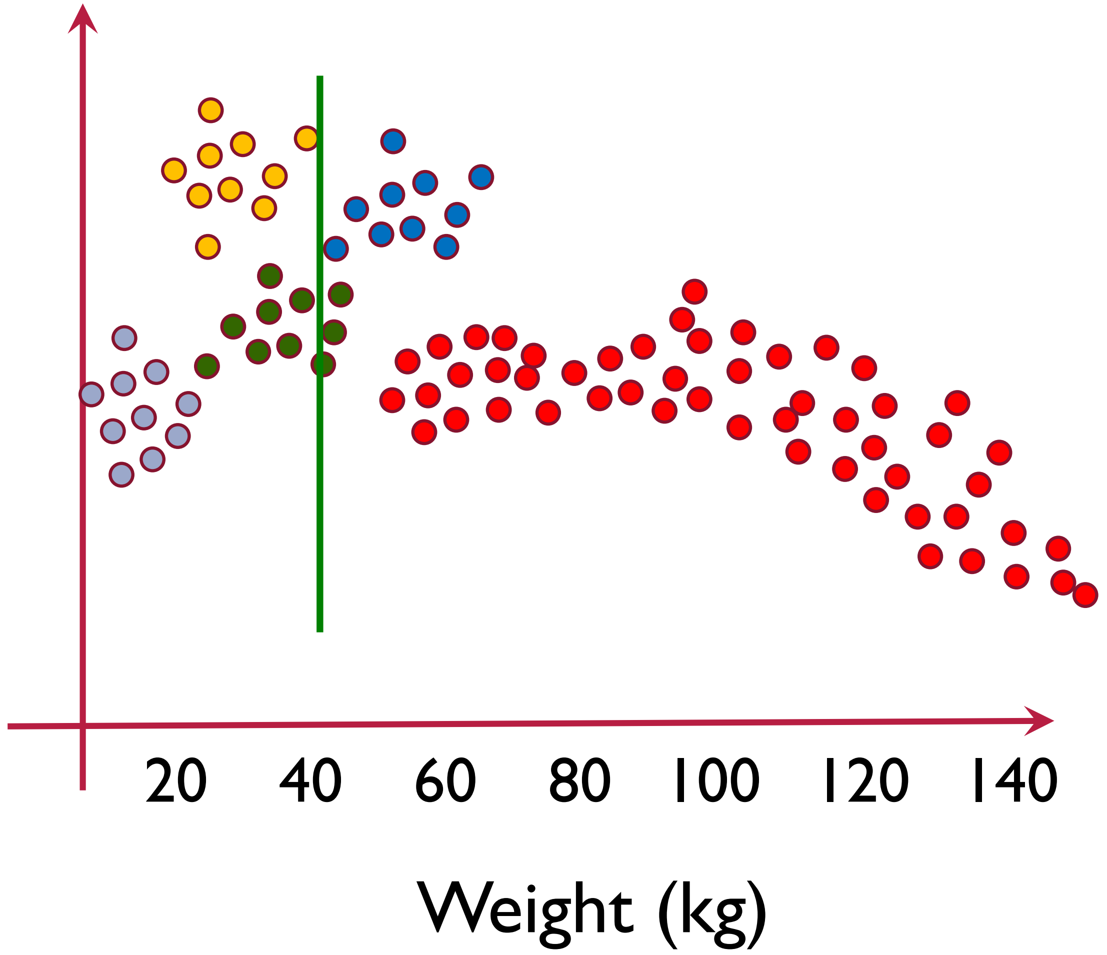
Information Gain

$$\text{Gain}(S, S_v) = \text{Entropy}(S) - \sum_v \frac{|S_v|}{|S|} Entropy(S_v) = H - H_1$$

$$H = 5 \times (-0.2 \log_2 0.2) = 2.32$$

$$H_1 = \frac{1}{2}\left((-0.4 \log_2 0.4) + (-0.4 \log_2 0.4) + (-0.2 \log_2 0.2)\right) + \frac{1}{2}\left((-0.4 \log_2 0.4) + (-0.4 \log_2 0.4) + (-0.2 \log_2 0.2)\right) = 1.52$$

$$\text{Gain}(S, S_v) = H - H_1 = 2.32 - 1.52 = 0.8$$



Best? Maximize Information Gain

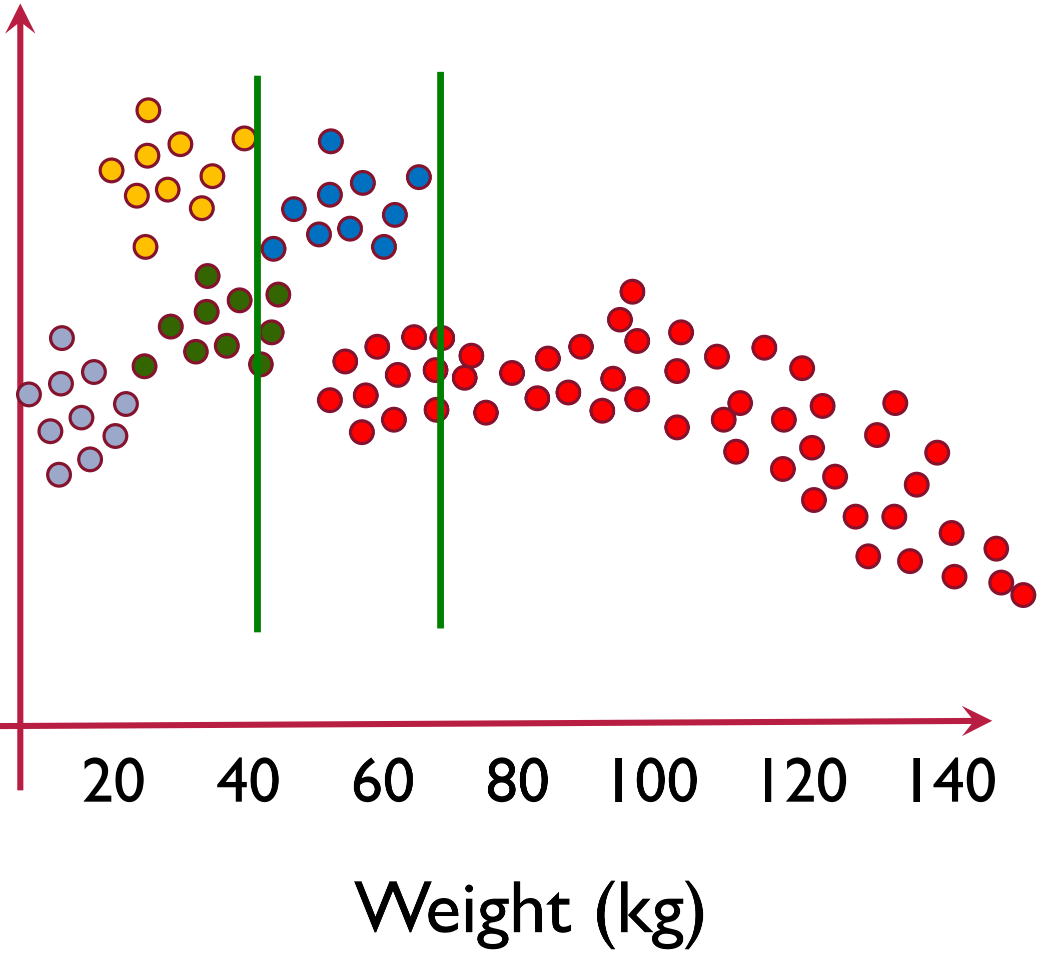
Initial Entropy: $5 \times (-0.2\, log_2\, 0.2) = 2.32$

$$H_1 = \frac{1}{2}\begin{pmatrix}(-0.4\, log_2 0.4) + (-0.4\, log_2 0.4) \\ + (-0.2\, log_2 0.2)\end{pmatrix} + \frac{1}{2}\begin{pmatrix}(-0.4\, log_2 0.4) + (-0.4\, log_2 0.4) \\ + (-0.2\, log_2 0.2)\end{pmatrix} = 1.52$$

Information Gain = 0.8

$$H_2 = \frac{1}{6}(-1.0\;\; log_2 1.0) + \frac{5}{6}\begin{pmatrix}(-0.22\, log_2 0.22) + (-0.22\, log_2 0.22) + \\ (-0.22\, log_2 0.22) + (-0.22\, log_2 0.22) + \\ (-0.12\, log_2 0.12)\end{pmatrix} = 1.91$$

Information Gain = 0.41



Decision Tree Training (ID3 Algorithm)

- Consider the training data and compute the impurity
- At start, all samples are at the root node
- At each step:
  1. Inspect all possible features
  2. Compute the information gain for each
  3. Select the feature that maximizes information gain
  4. Distribute data into child nodes.
  5. Do 1-4 recursively for each child node until pure leaf nodes

Properties of DTs. Qn?

AIML

Advantages

- Fast, Compact and Effective
- Handles categorical variables
  - Ordinal, Nominal
- Interpretable as a set of rules
  - Disjunctive Normal Form
- Can indicate the most useful features

Applications:

- Medical diagnosis
- Credit risk analysis

Disadvantages

- Not suitable for prediction of continuous attribute
- Do not handle non-rectangular regions well
- Computationally expensive to train
  - Sort at each node for each candidate splitting field
  - Some algorithms search for optimal combining weights for features
- Tends to Overfit
  - Perform poorly with many class and small data
  - Solutions even more computationally expensive

Decision Trees

A Look into the Details

Splitting Data at a Node

- Random Split
- N-way split on value
  - Works on categorical features
- Binary split on threshold
  - Works on continuous/ordinal features

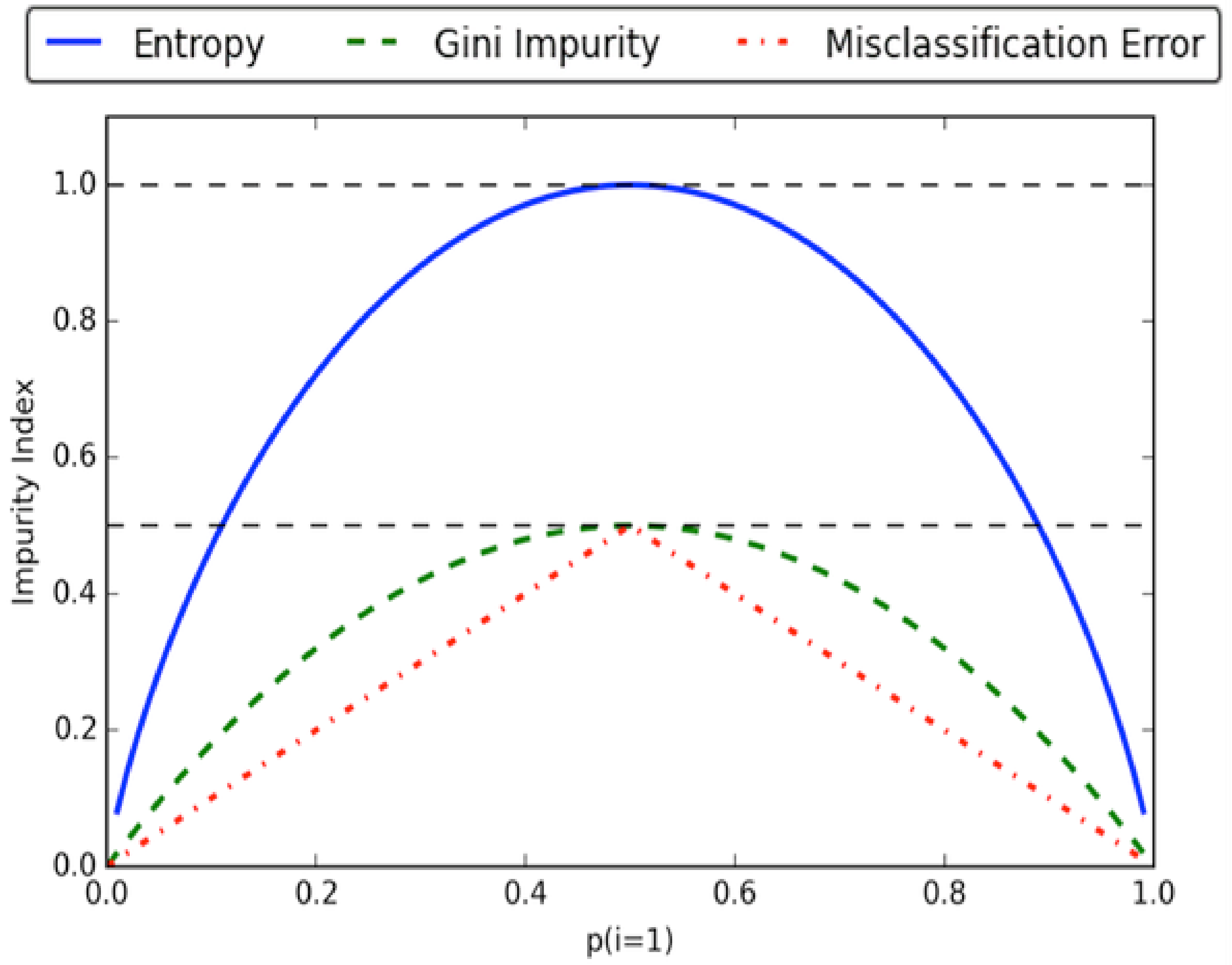
Impurity Metrics

- Entropy: $\mathrm{H(X)} = -\sum_i \mathrm{P(i)}\, log_2\, \mathrm{P(i)}$.
- GINI: $I_G(\mathrm{p}) = \sum_i P_i(1 - P_i)$

$$= 1 - \sum_i {P_i}^2$$

- Misclassification Error:

$$1 - \max_i P_i$$



DT Algorithms: CART

- Classification and Regression Trees (Leo Breiman)
- Recursive Binary Splitting: Greedy Algorithm
  - All values of an attribute are sorted and all split points are tested
  - Test all such attributes and select the split with lowest cost
- Cost Functions:
  - Regression: MSE
  - Classification: Gini

Early Stopping and Pruning

- Decision trees are notorious for overfitting. Solutions:
- Early Stopping
  - Do not split a not beyond a point (of number of items or purity)
- Pruning
  - Once the tree is formed, remove weakest branches.
  - Use validation set to decide when to stop
- Both approaches also reduces the depth and improves classification speed

How to handle missing values?

- Why did the data go missing?
  - Radom, but dependent on observed variables
  - Radom, but dependent on unobserved variables
  - Dependent on the value itself!!
- Throw out data with missing values
  - What are the implications?
- Impute values
  - Impute 0
  - Mean/median imputation
  - Impute from observed values: build predictor
  - Impute from last observation

C4.5: An extension of ID3

- Split data into training-validation and build DT on training data
- Uses Gain Ratio
    - If we have an attribute D that has a distinct value for each record, then Info(D,T) is 0, thus Gain(D,T) is maximal.
    - To compensate for this use the following ratio instead of Gain
        - GainRatio(D,T) = Gain(D,T) / SplitInfo(D,T)
        - SplitInfo(D,T) is the information due to the split of T on the basis of value of categorical attribute D.
- Move misclassifications in validation to training

DT in Scikit Learn

Training

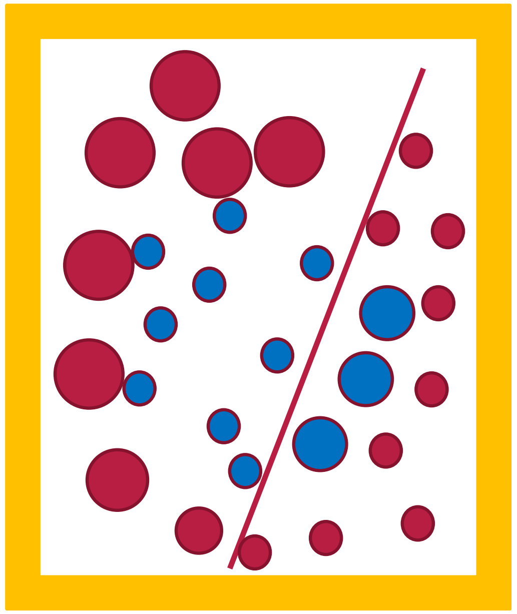
```
from sklearn.tree import DecisionTreeClassifier

clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100,
    max_depth=5, min_samples_leaf=1)

 clf_gini.fit(X_train, y_train)
```

Testing

```
 Z = clf_gini.predict(x,y)
```



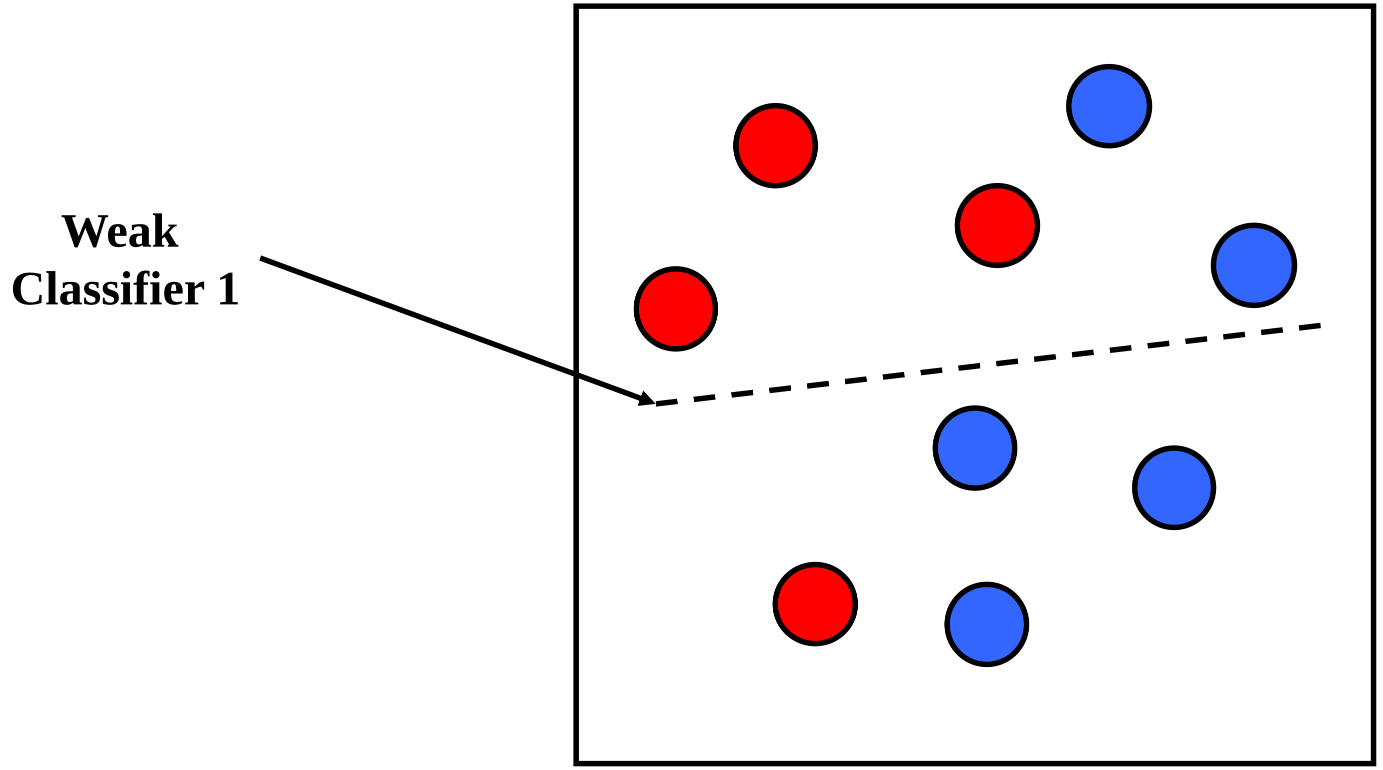
Boosting

Combining Weak Learners

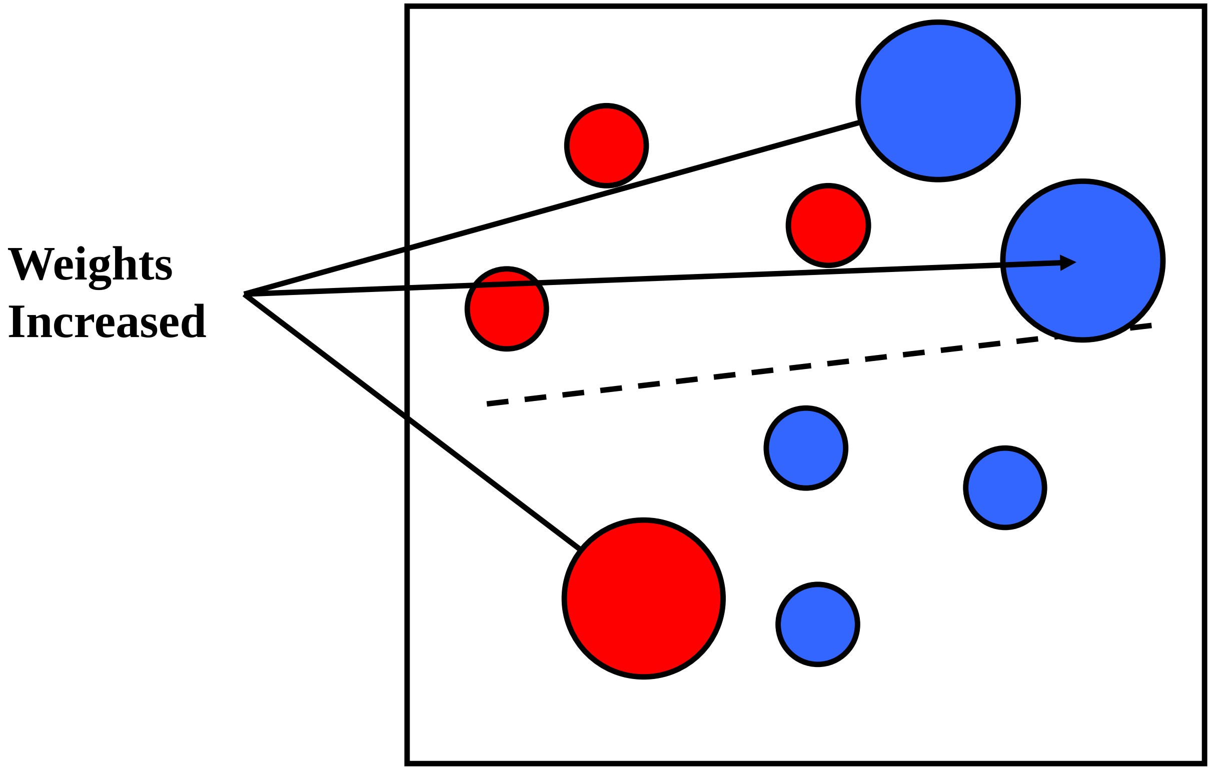
Boosting and Adaboost

- Generate a set of weak classifiers
- Combine them using a weighted combination (probabilities)
- Weights proportional to their performance on validation set
- AdaBoost:
  - Popular variant of boosting
  - Generate classifiers by weighted sampling

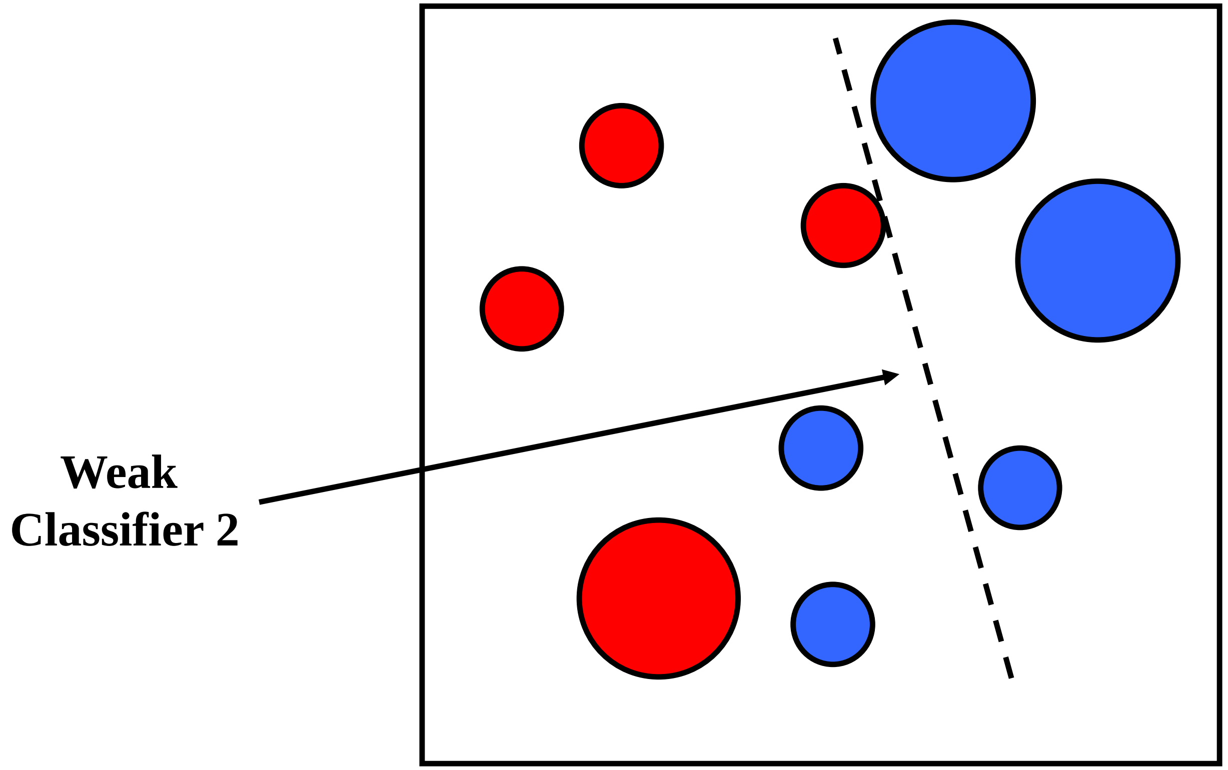
Boosting Illustration



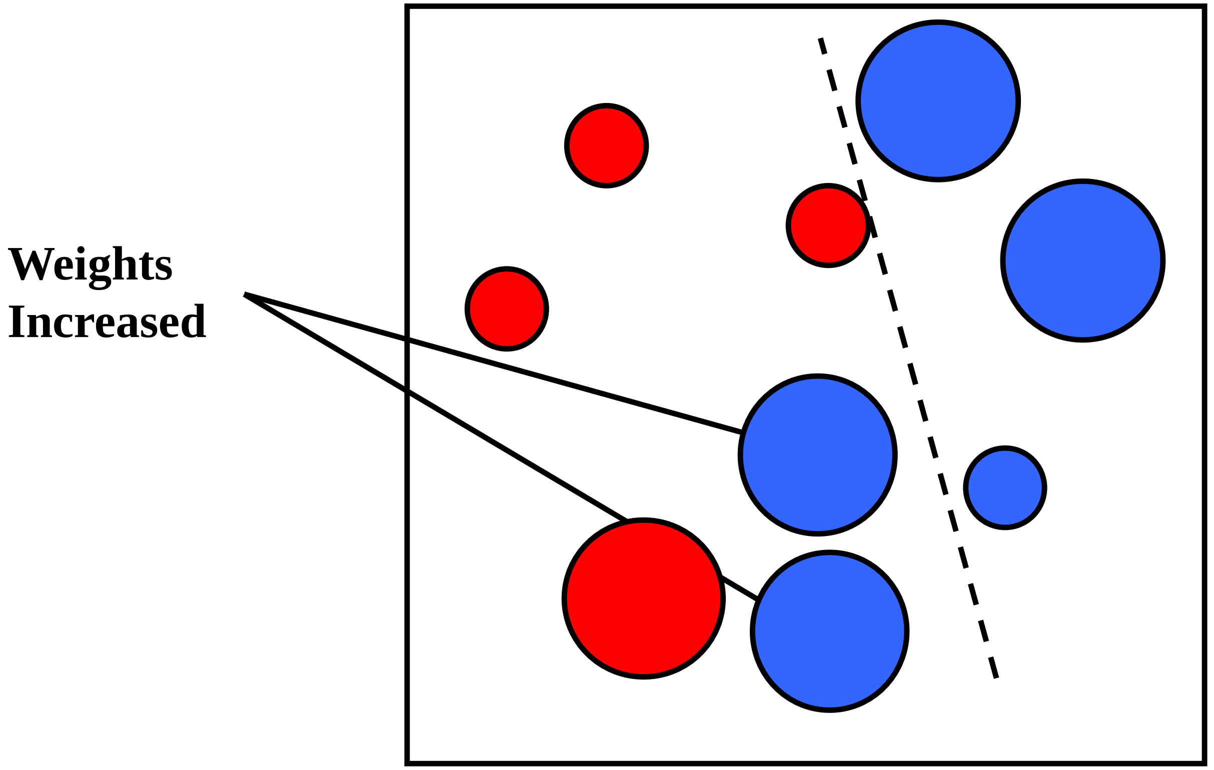
Boosting Illustration



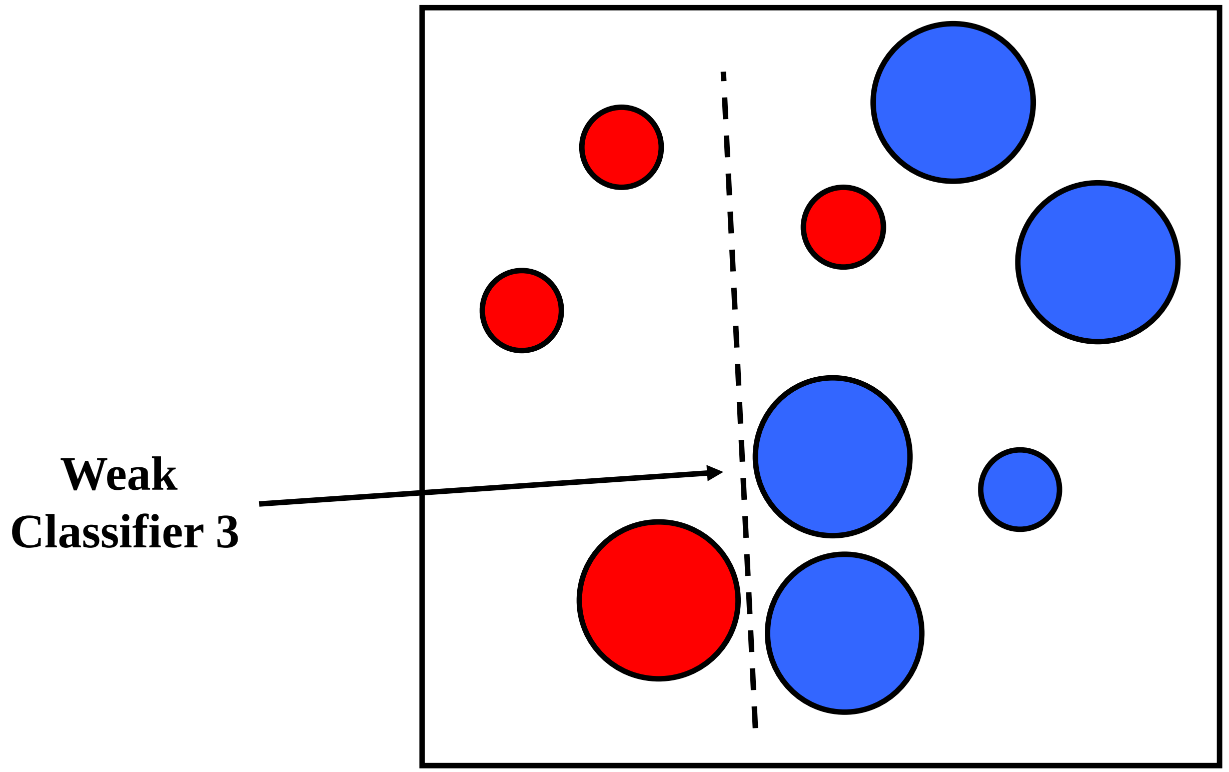
Boosting Illustration



Boosting Illustration

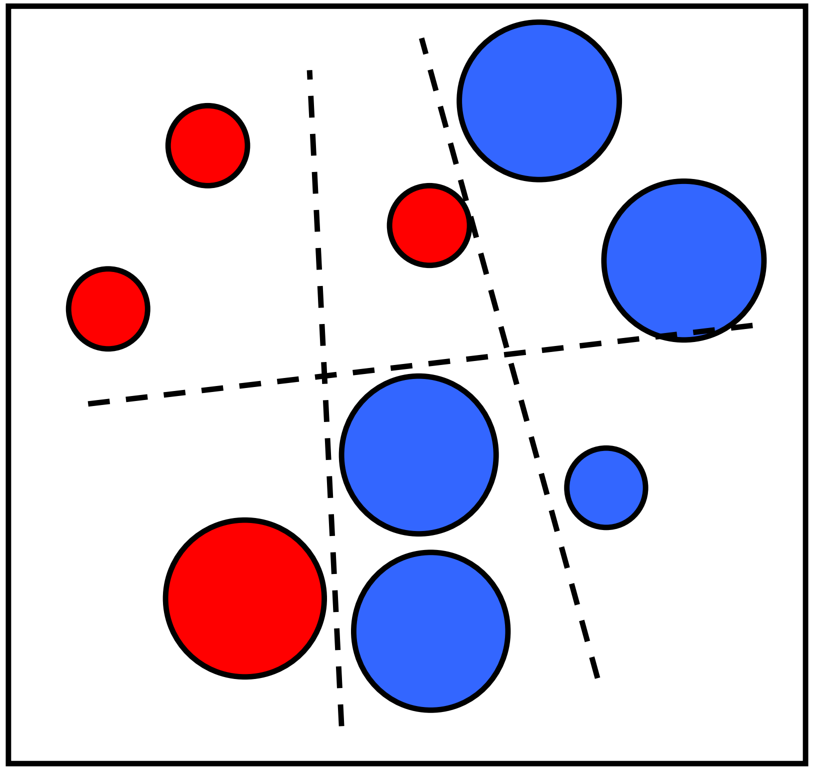


Boosting Illustration



Boosting Illustration

Final classifier is a combination of weak classifiers



Boosting: Summary. Questions?

- Weak Learners combine to form strong classifiers
- Does not work that well with strong learners
- Boosting combined with trees gives some of the most powerful classifiers
    - Gradient Boosted Trees (GBT, xGBT)

Random Forests

A simple way of combining Decision Trees

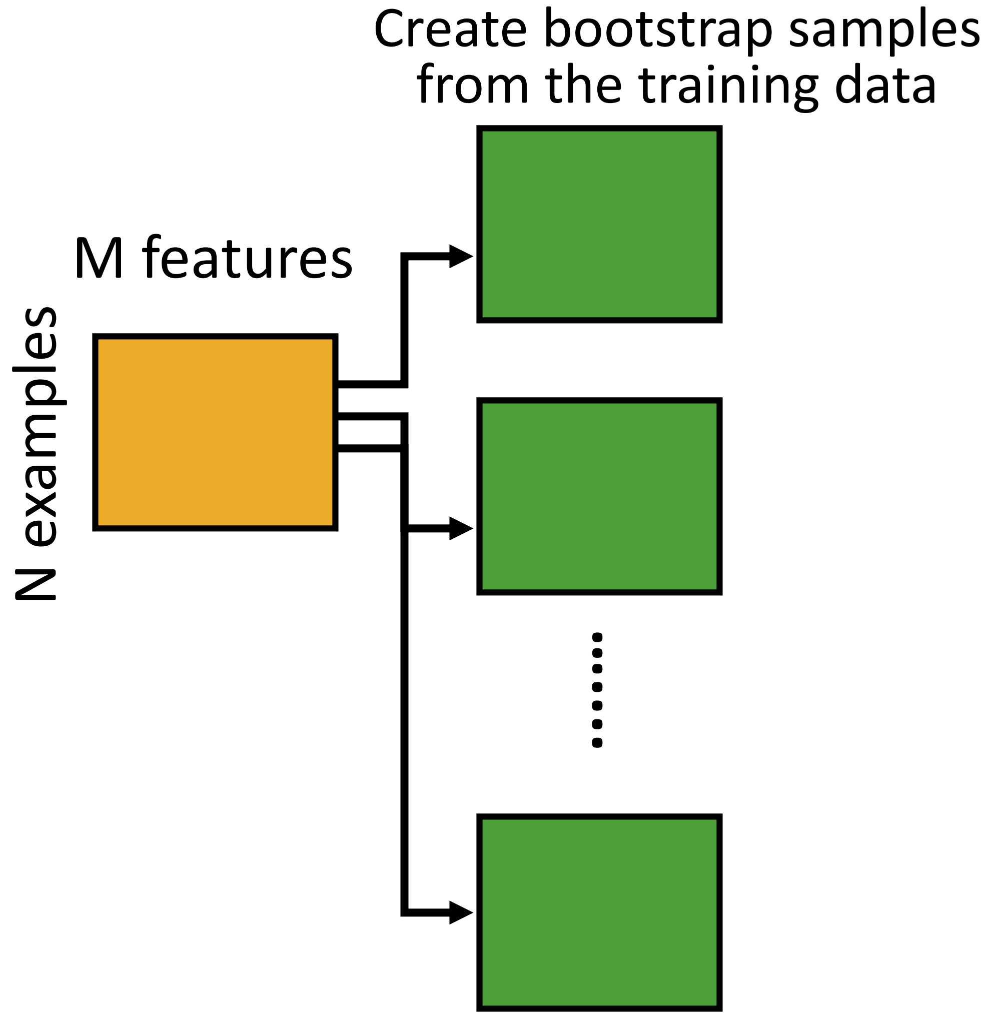
Random Forest

- **Random forest** (or **random forests**) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- The term came from **random decision forests** that was first proposed by Tin Kam Ho of Bell Labs in 1995.
- The method combines Breiman's "bagging" idea and the random selection of features.

Bagging

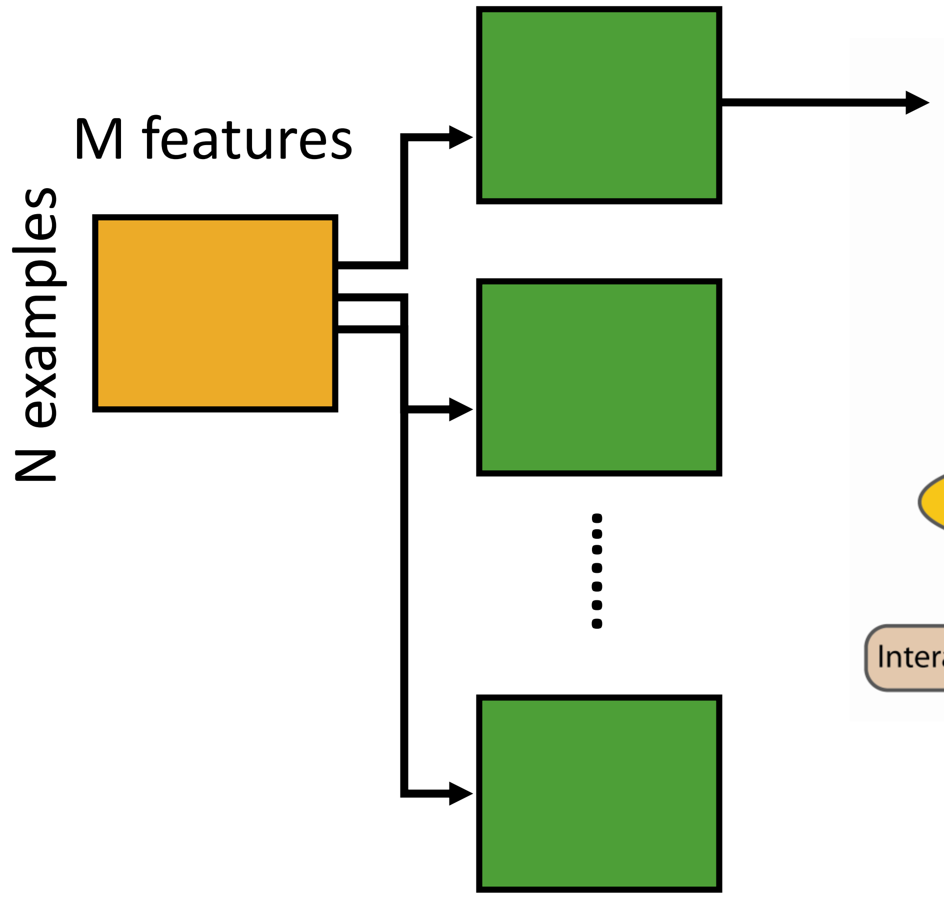
- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.
- Bootstrap: Randomly draw datasets *with replacement* from the training data, each sample *the same size as the original training set*
- For classification, a *committee* of trees each cast a vote for the predicted class.
- A simpler way to generate samples than boosting

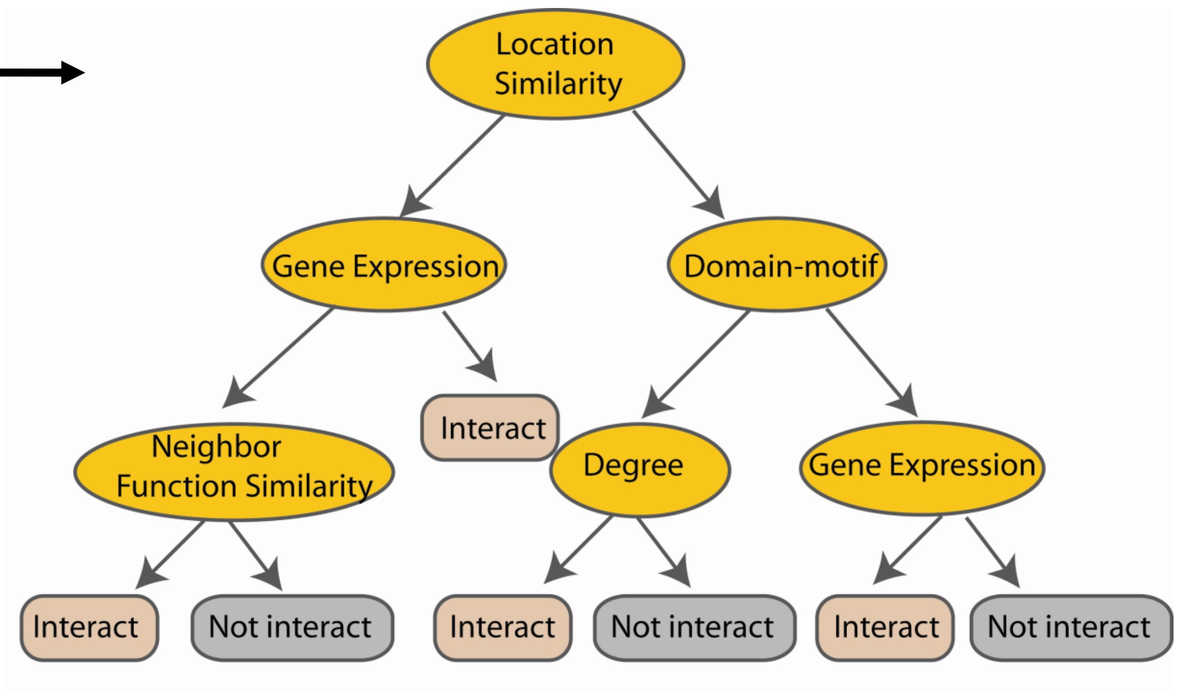
Random Forest



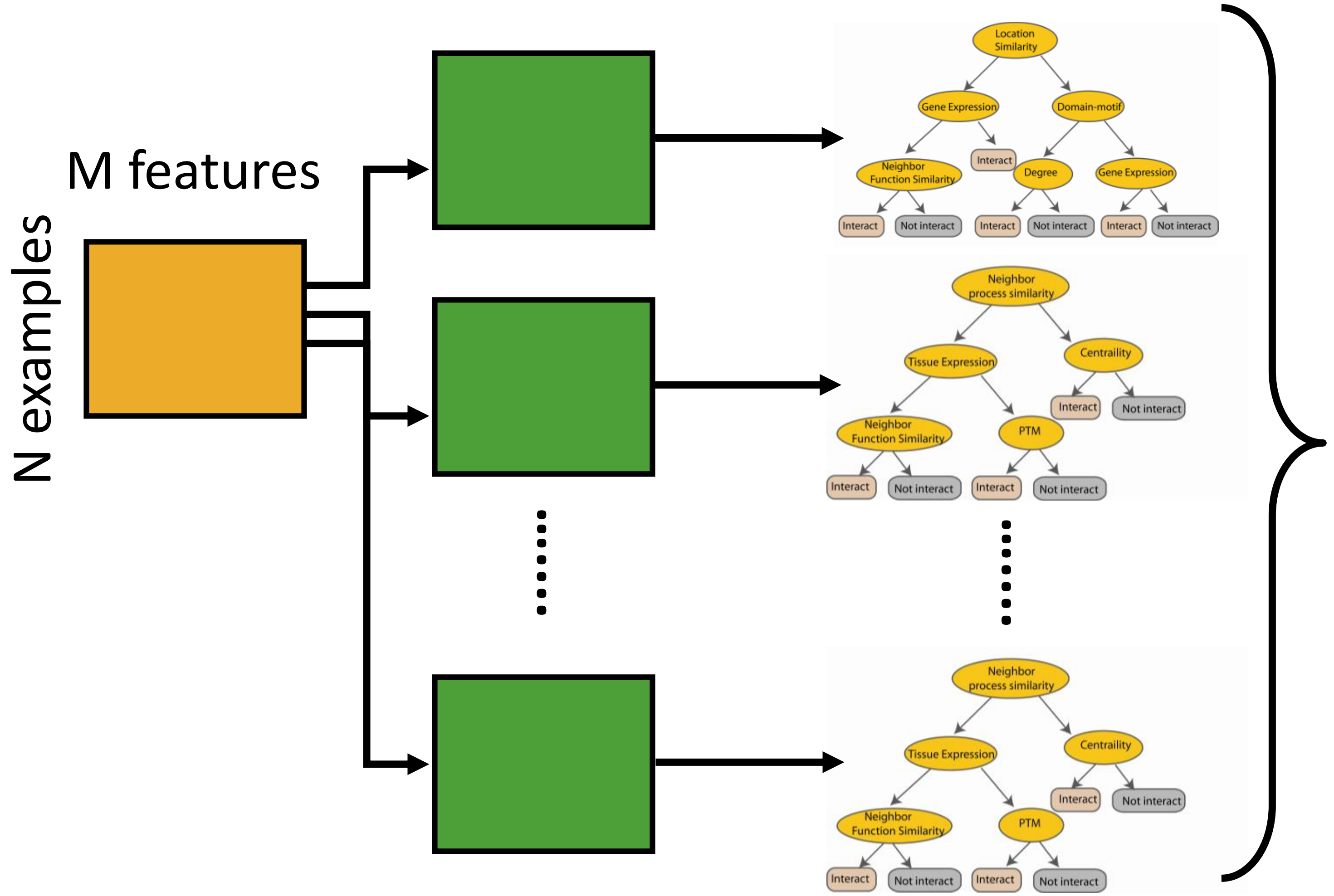
Random Forest

Construct a decision tree



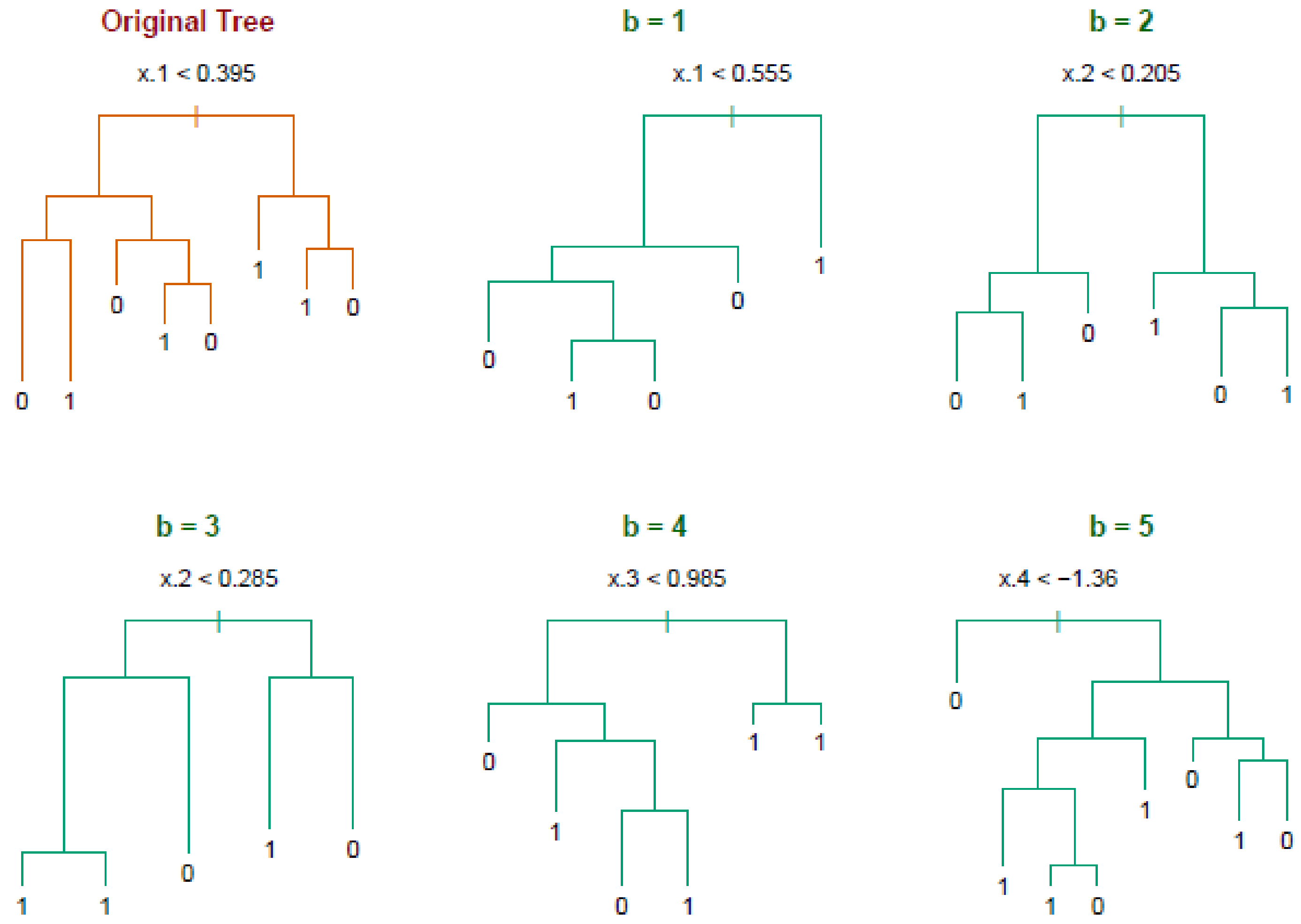


Random Forest



Take the majority vote

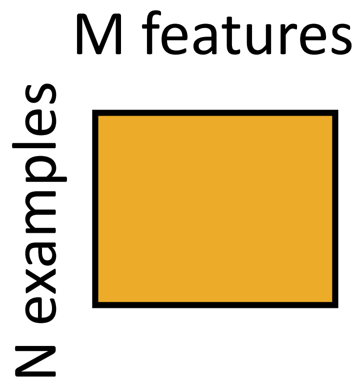
Random Forest



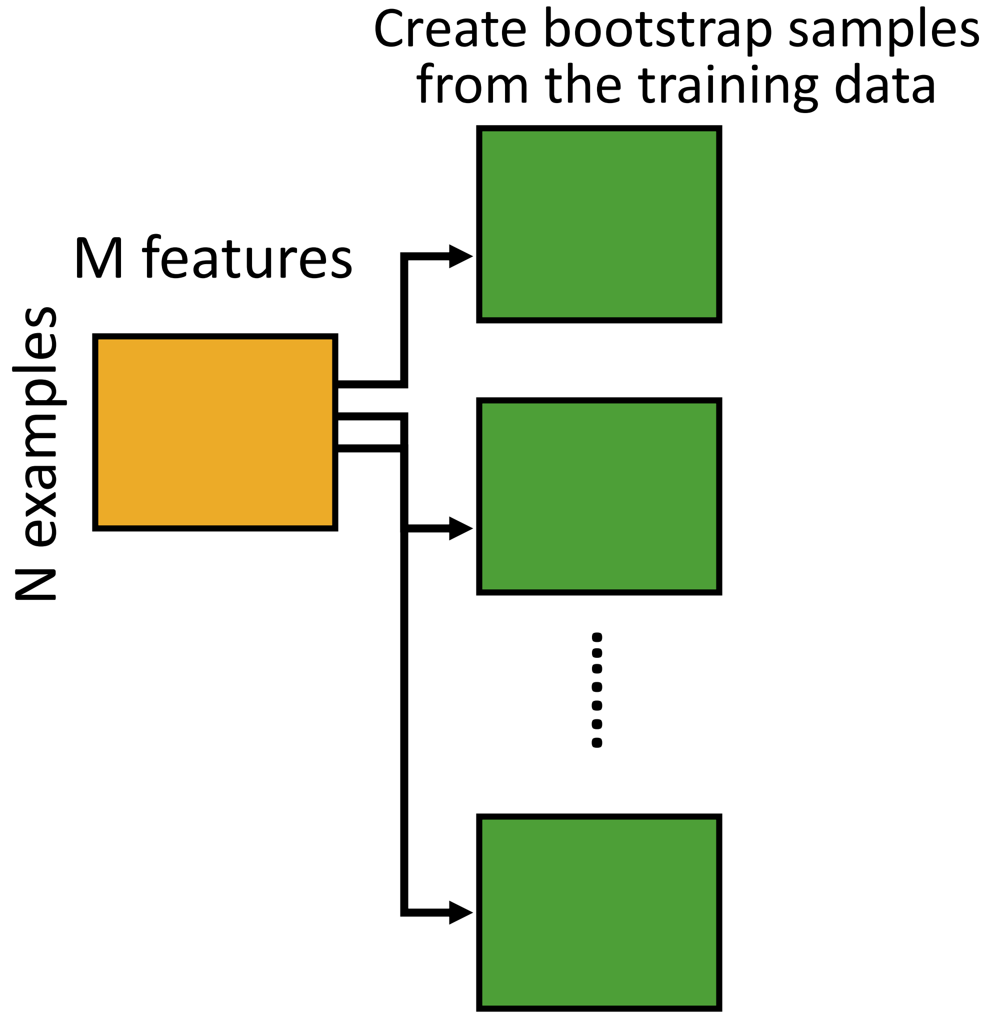
Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier

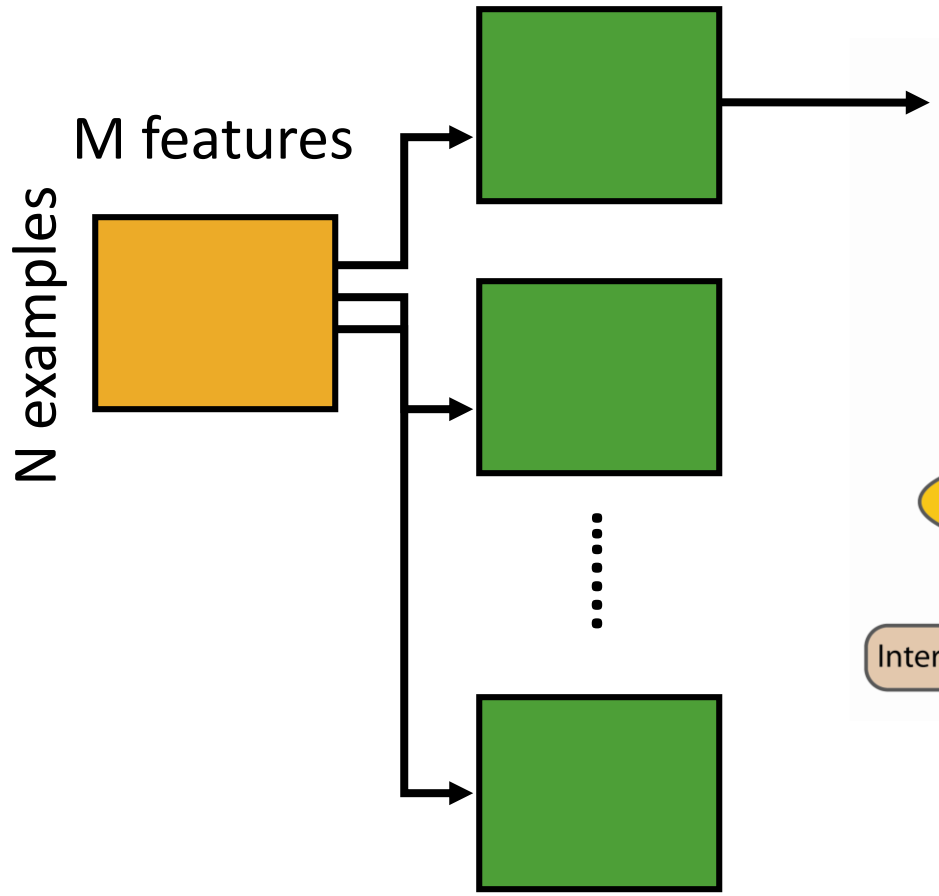
Training Data



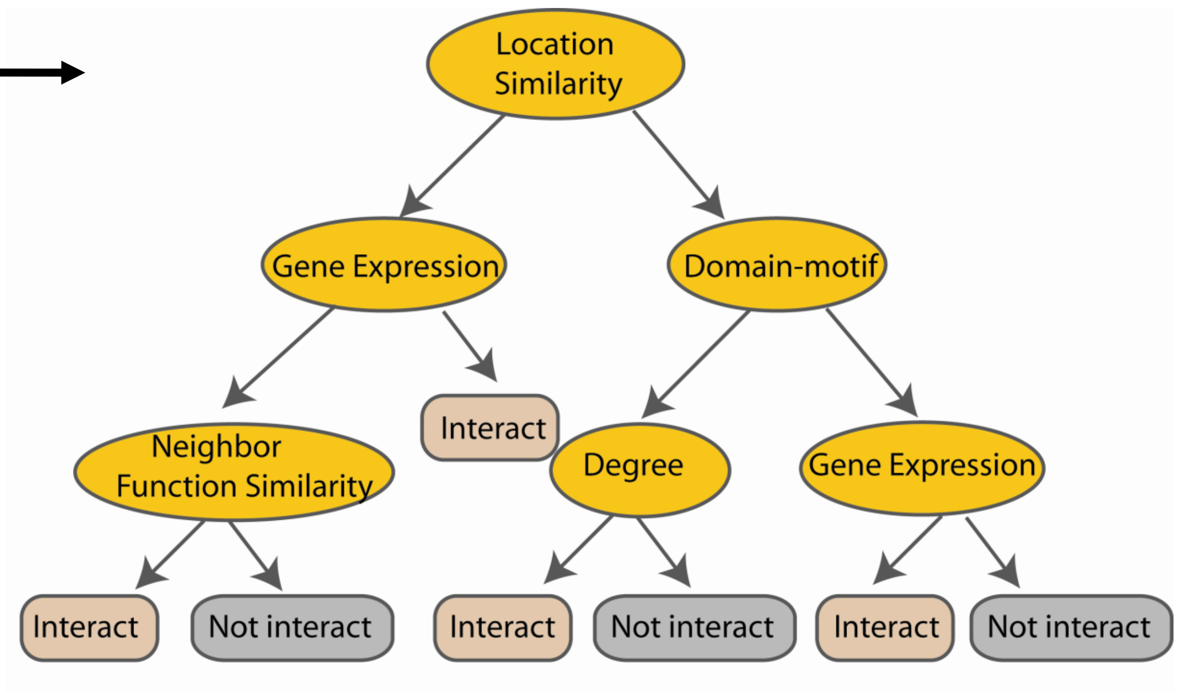
Random Forest Classifier



Random Forest Classifier

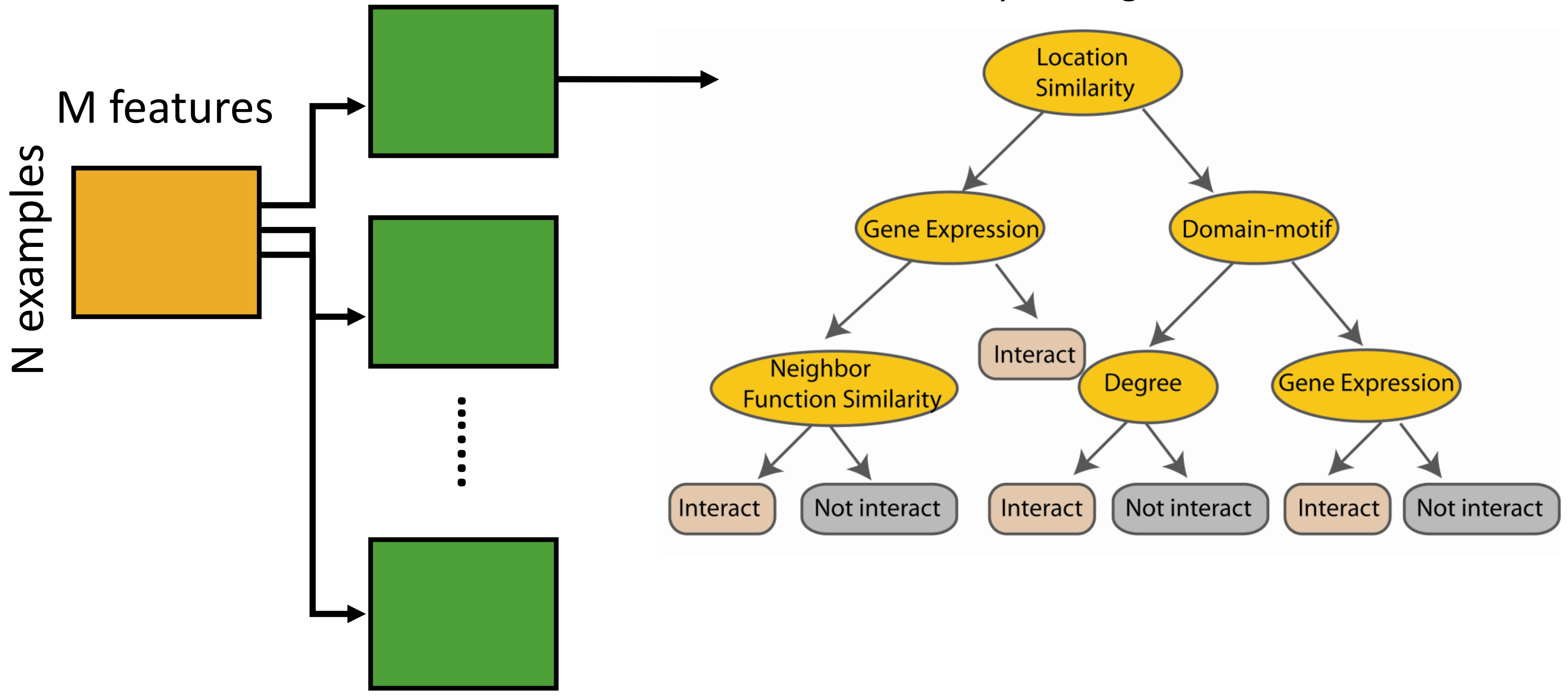


Construct a decision tree

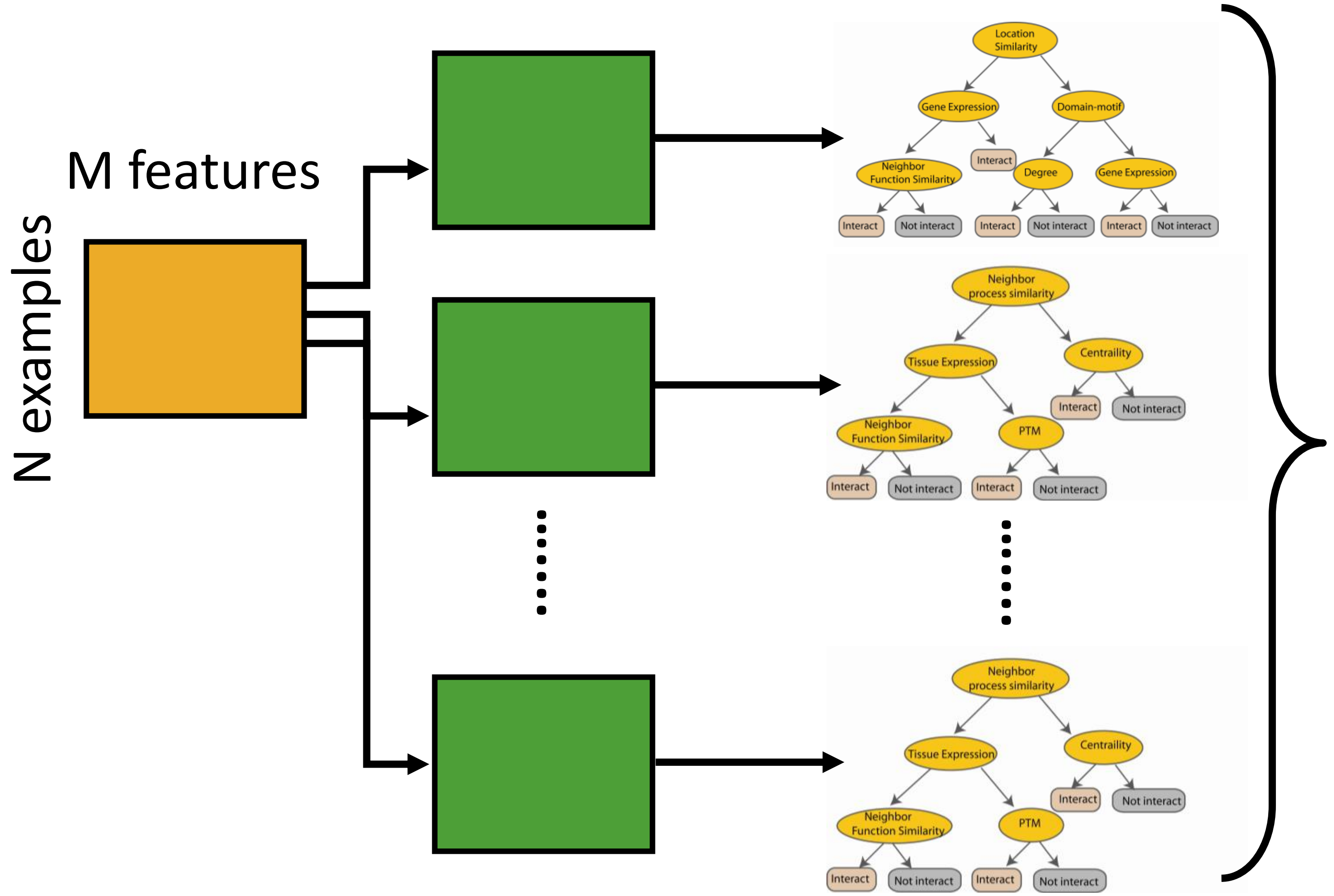


Random Forest Classifier

At each node in choosing the split feature
choose only among *m*<*M* features



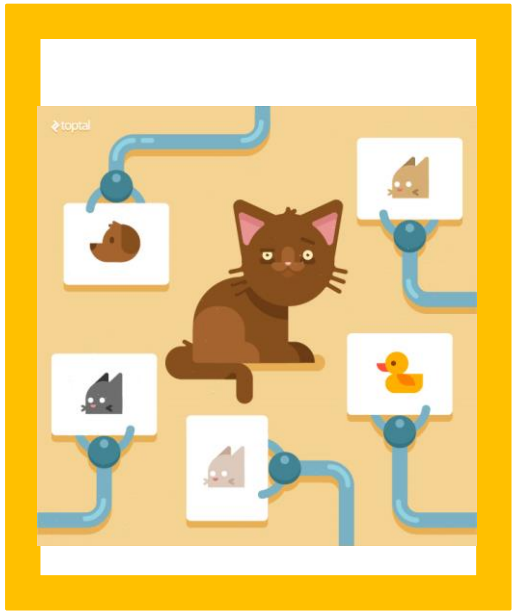
Random Forest Classifier



Take the majority vote

Random Forest: Summary. Questions?

- Extremely fast (to predict)!
  - RF is also fast to build. Avoiding cross-validation alone speeds training by over 10x.
  - Fully parallelizable … to make training and testing even faster!
- Automatic predictor selection from large number of candidates
- Resistance to over training
- Ability to handle data without preprocessing
  - Data does not need to be rescaled, transformed, or modified
  - Resistant to outliers



Other Ensemble Methods

Other solutions from a group of solvers

Ensemble Clustering

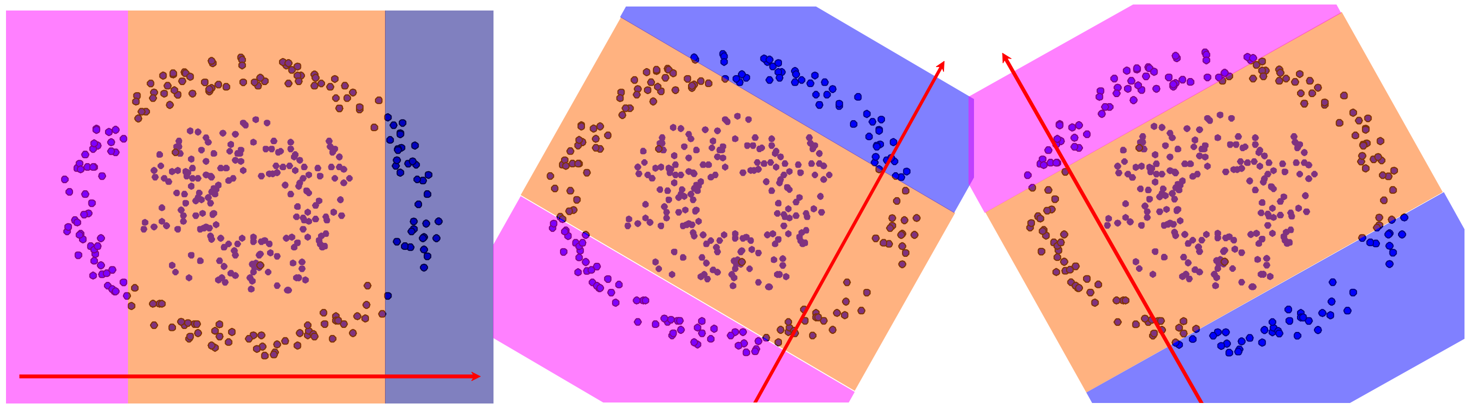
- Generate a set of weak clustering
  - Could be done efficiently
  - Need not be too accurate on number of clusters
- Combine the clusterings together
  - How to generate consensus ?
  - Can give effective ways to determine number of clusters

Generation of Ensemble by Weak Clusterings

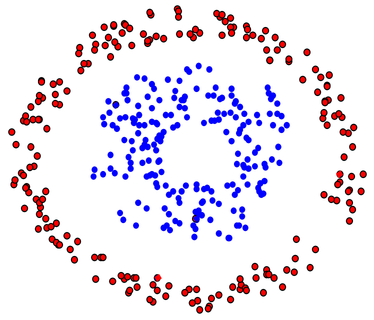
- Weak clustering is defined as a partition that is only slightly better than a random partition of the data
- Weak clusterings can be generated efficiently compared to sophisticated clustering algorithms
- Weak clusterings can be obtained by (i) clustering in low dimensional projections of data, (ii) by random “cuts” of the data, or (iii) using sub-samples of data

Ensemble Generation by Random Projections

- Random subspaces provide us with different views of the multidimensional data; each random subspace can be of very low dimensionality (e.g., 1-D)
- Clustering in 1-D space is computationally inexpensive and can be implemented by k-means algorithm



Different 3-cluster partitions of 2-dim data resulting from projections onto random lines



Concentric circular clusters can be perfectly detected by an ensemble of 50-100 partitions

Co-association As Consensus Function

- Similarity between objects can be estimated by the number of clusters shared by two objects in all the partitions of an ensemble
- This similarity definition expresses the strength of co-association of n objects by an n x n matrix

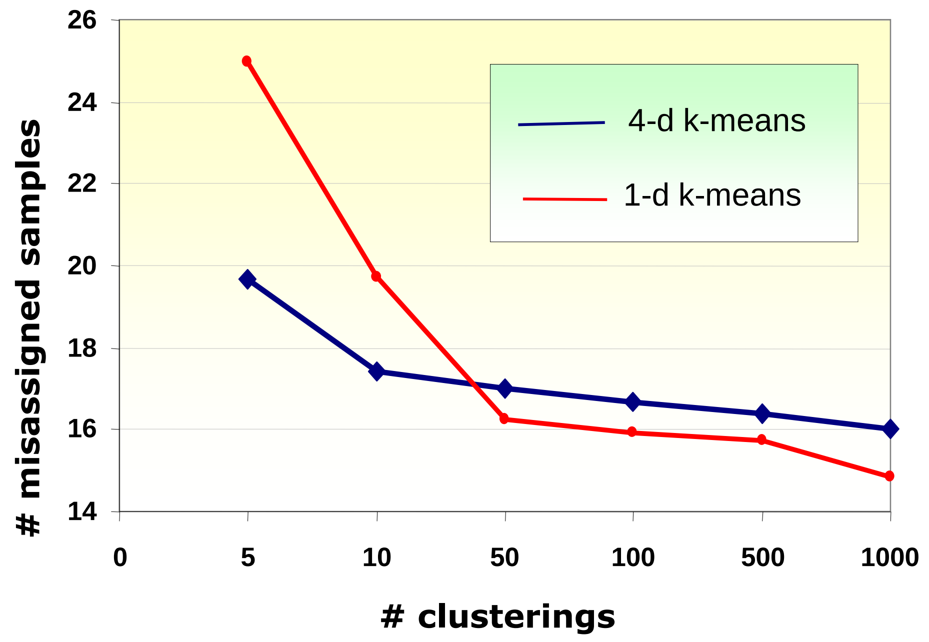
$$C_{ij} = C(x_i, x_j) = \frac{1}{N} \sum_{k=1}^{N} I(\pi_k(x_i) = \pi_k(x_j))$$

  - x_i: the *i*-th pattern; $\pi_k(x_i)$: cluster label of xi in the *k*-th partition; I(): Indicator function; *N* = no. of different partitions
- This consensus function eliminates the need for solving the label correspondence problem

Results for Ensembles of Random Projections

"Galaxy/Star" data (4600 points in 14 dimensions, 2 classes)

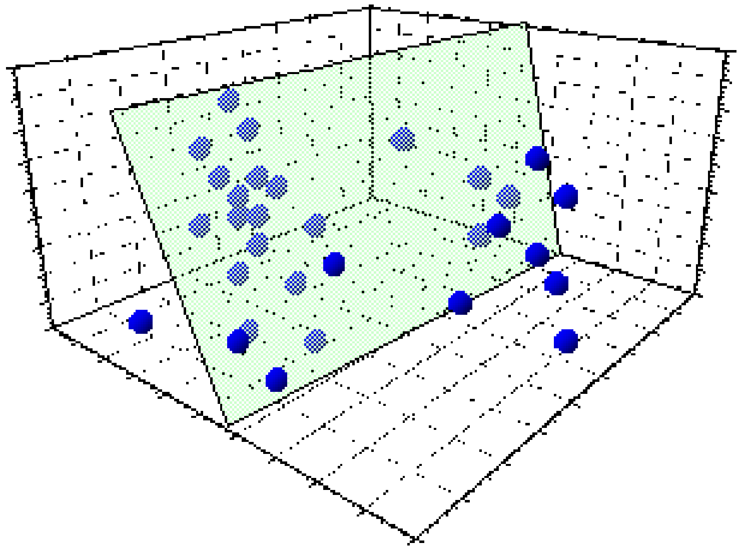
H, # of components	k, # of cl. in component	Type of Consensus Function: Hypergraph methods HGPA	MCLA	Median partition, QMI k-means
5	2	49.7	20.0	20.4
10	2	49.7	23.5	21.1
20	2	49.7	21.0	18.0
5	3	49.7	22.0	21.7
10	3	49.7	17.7	13.7
20	3	49.7	15.8	13.3
5	4	49.7	19.7	16.7
10	4	49.7	16.9	15.5
20	4	49.7	14.1	13.2
5	5	49.7	22.0	22.5
10	5	49.7	17.7	17.4
20	5	49.6	15.2	12.9



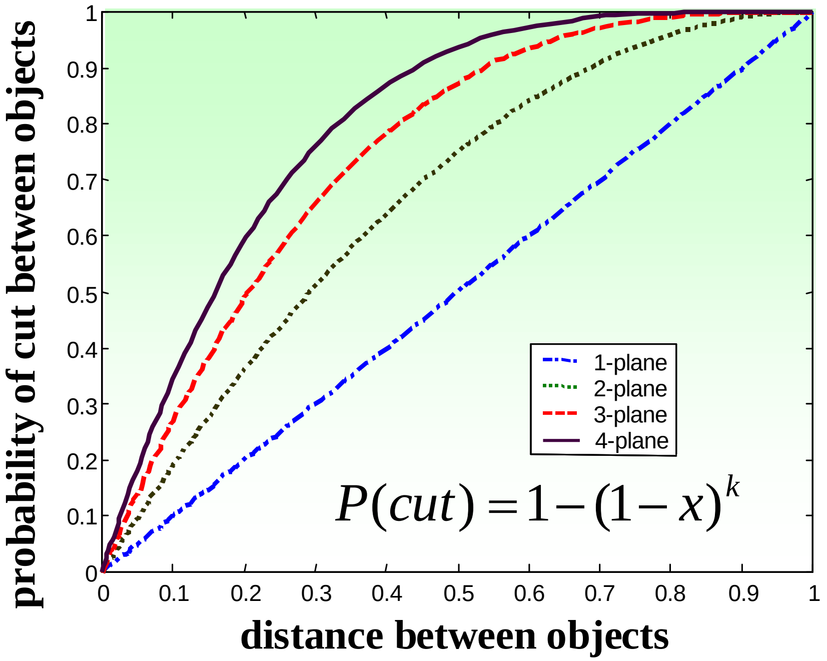
Ensemble finds novel and better clustering solutions compare with regular *k*-means that has more than 30% error rate, on average, for Galaxy data

Partitions by Random Cuts

- This approach pushes the notion of the weak clustering to the extreme.
- Data set is cut by random hyperplanes. Points separated by hyperplanes are declared to be in different clusters



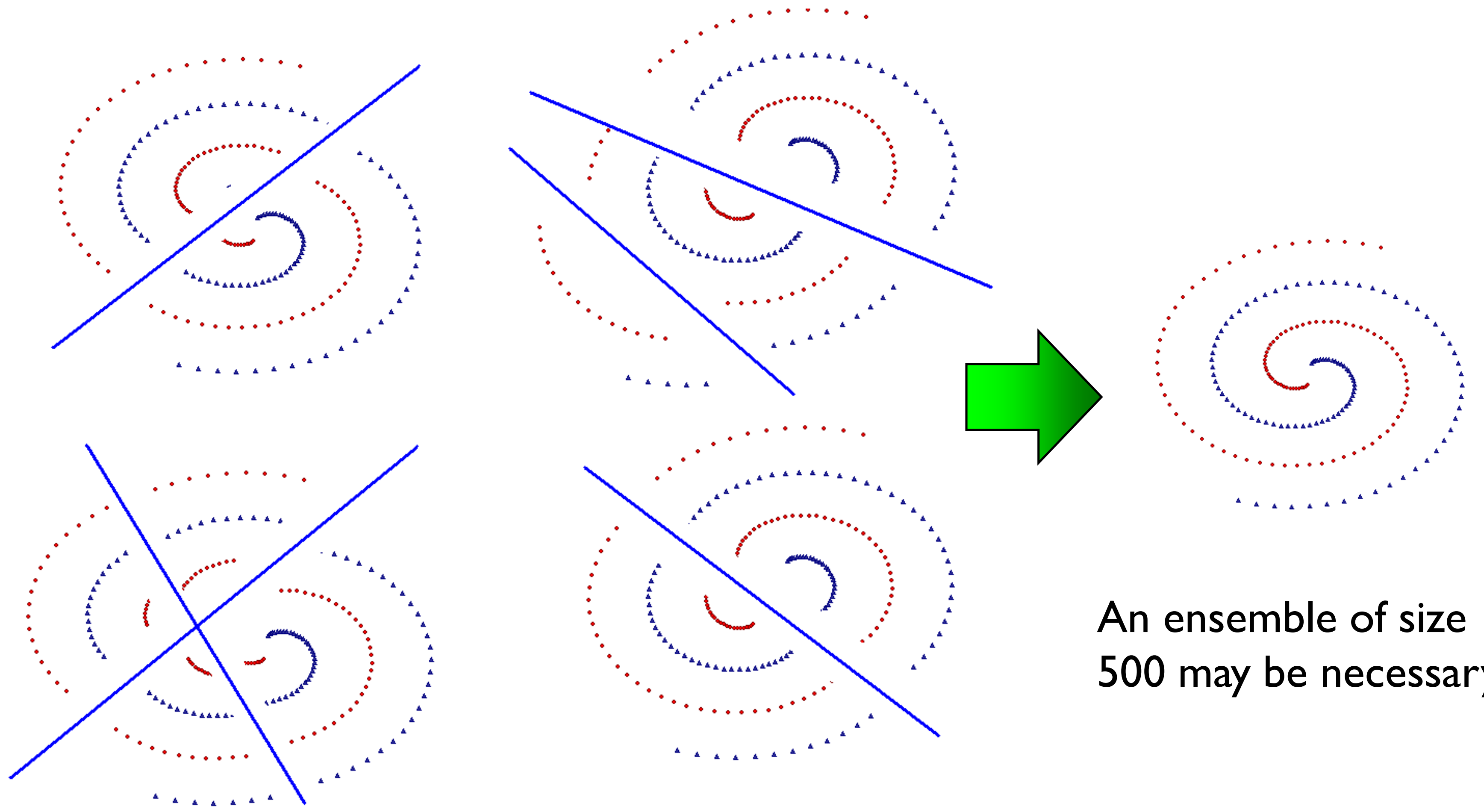
Even random cuts can uncover inter-pattern similarity values and provide relevant cluster information



Probability of separating two patterns using different numbers of random cuts

Results for Ensemble of Random Cuts

We can correctly identify two spirals by combining partitions resulting from random cuts using co-association consensus function with SL



An ensemble of size 500 may be necessary

Summary: Ensemble Methods

- A group of weak learners do better than an expert
- Generate a set of solutions
  - Ensure the solutions are weak
    - Linear models, simple decision trees, projecting on a line, etc.
  - Ensure the solutions are diverse
    - Could be done through data sampling, feature dropping, etc.
- Combine the solutions
  - Intelligent combination that weighs solutions by their performance
- Effective approach in a variety of Machine Learning problems
- Ensemble methods are often highly parallelizable
- Using an ensemble of simple solutions tend to reduce overfitting

Questions?