GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 12: RANDOM FORESTS

0. DEMO SAMPLE PROJECTS
I. PROBABILITY
II. BAYES' THEOREM
III. EXAMPLE: BAYESIAN COIN FLIPS (OPTIONAL)
IV. NAIVE BAYES

Questions?

- DATA EXPLORATION
- SUPERVISED LEARNING: REGRESSION
- SUPERVISED LEARNING: CLASSIFICATION
- UNSUPERVISED LEARNING
- VARIOUS TOPICS

- LOGISTIC REGRESSION
- NAIVE BAYES
- RANDOM FORESTS (TODAY)
- SUPPORT VECTOR MACHINES
- COMPETITION

I. DECISION TREES
II. FITTING DECISION TREES
III. OBJECTIVE FUNCTIONS
IV. REGULARIZATION
V. ENSEMBLE METHODS

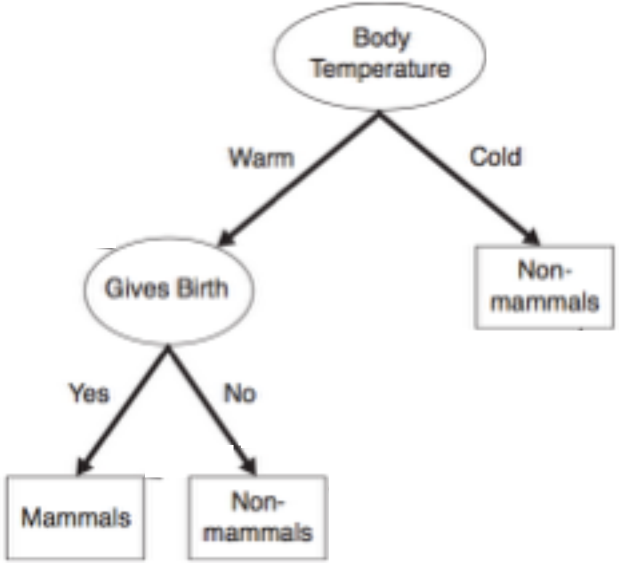
BAGGING BOOSTING RANDOM FORESTS

Ideas for your final project are due today!

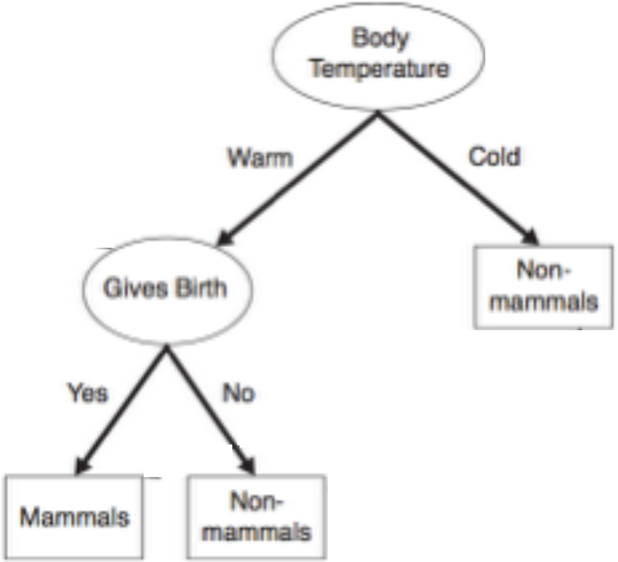
I. DECISION TREES

Q: What is a decision tree?

A decision tree for mammal classification...



A decision tree for mammal classification…

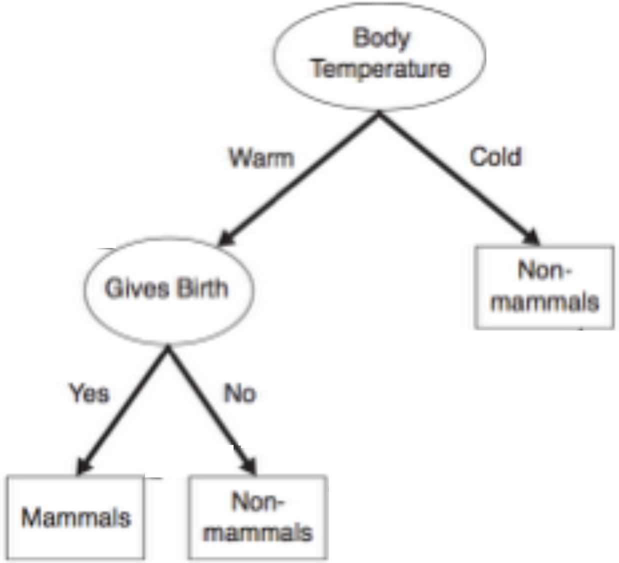


…may be an accurate way of describing the dataset

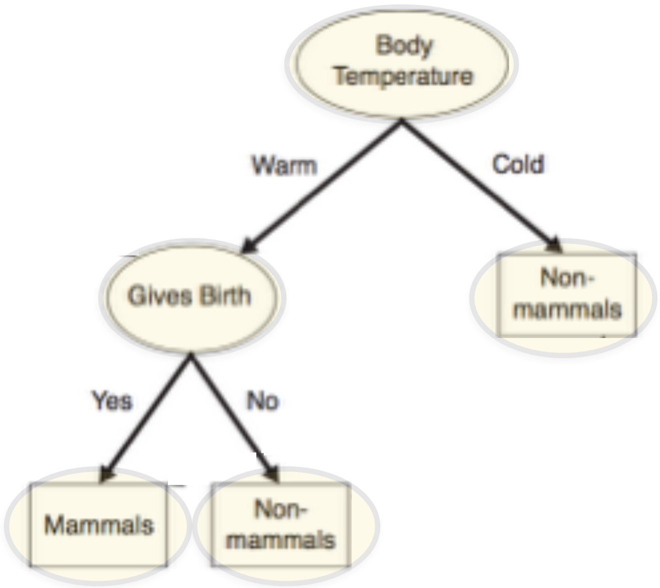
Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

Q: How is a decision tree represented?



Q: How is a decision tree represented?



nodes *represent questions ("test conditions")*

The top node of the tree is called the **root node**. *This node has 0 incoming edges, and 2+ outgoing edges.*

The top node of the tree is called the **root node**. *This node has 0 incoming edges, and 2+ outgoing edges.*

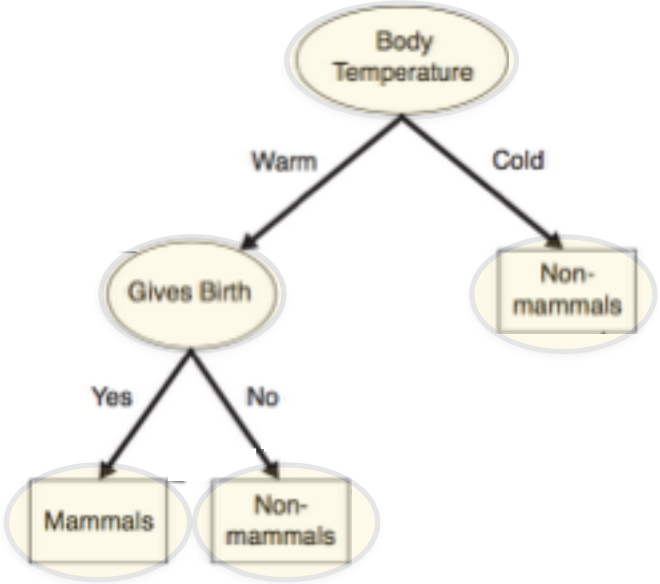
An **internal node** *has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.*

The top node of the tree is called the **root node**. *This node has 0 incoming edges, and 2+ outgoing edges.*

An **internal node** *has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.*

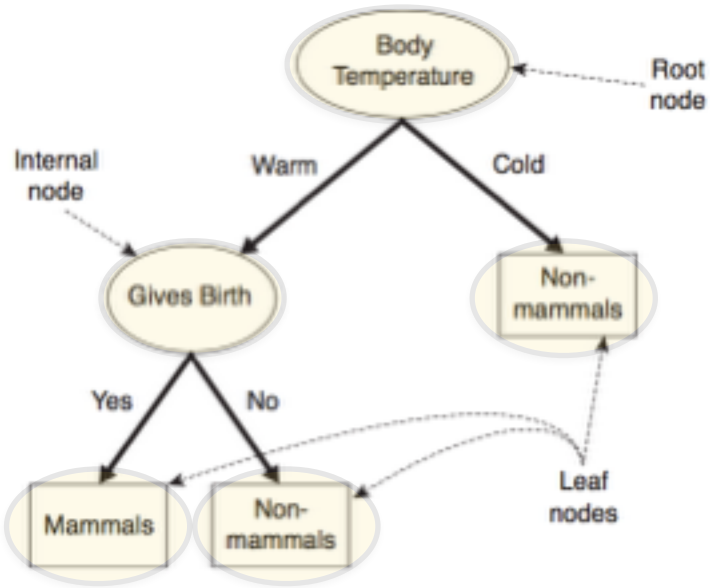
A **leaf node** *has 1 incoming edge and, 0 outgoing edges. Leaf nodes correspond to class labels.*

Q: How is a decision tree represented?



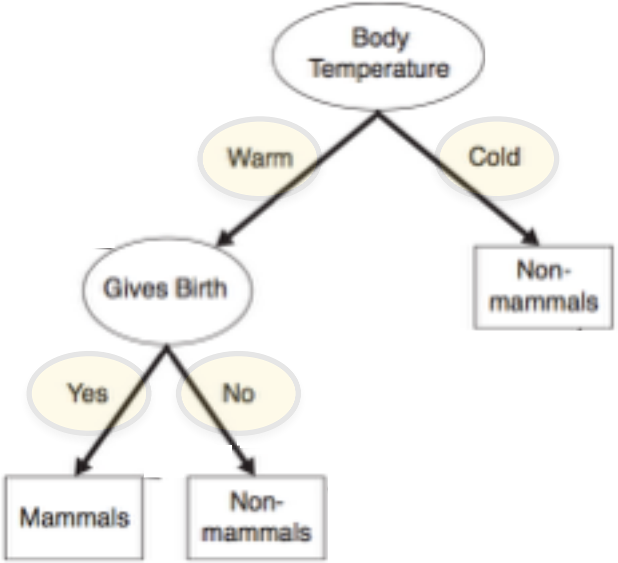
nodes *represent questions ("test conditions")*

Q: How is a decision tree represented?



nodes *represent questions ("test conditions")*

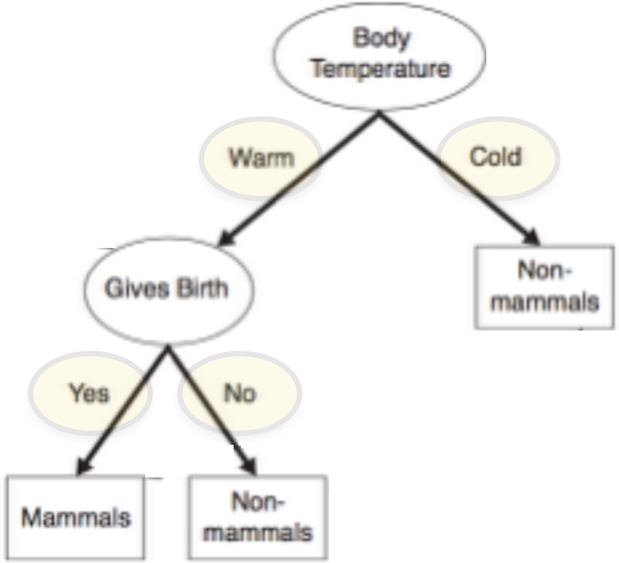
Q: How is a decision tree represented?



nodes *represent questions ("test conditions")*

edges *are the answers to these questions.*

Q: How is a decision tree represented?

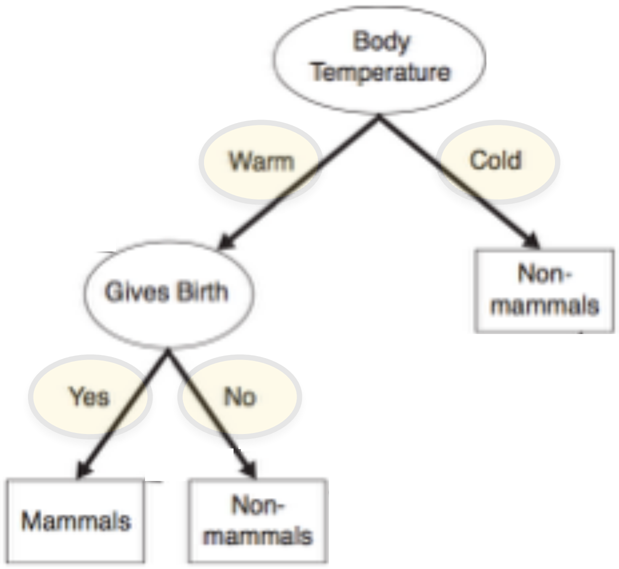


nodes *represent questions ("test conditions")*

edges *are the answers to these questions.*

In other words, a tree is a **directed acyclic graph***.*

Q: How is a decision tree represented?



nodes *represent questions ("test conditions")*

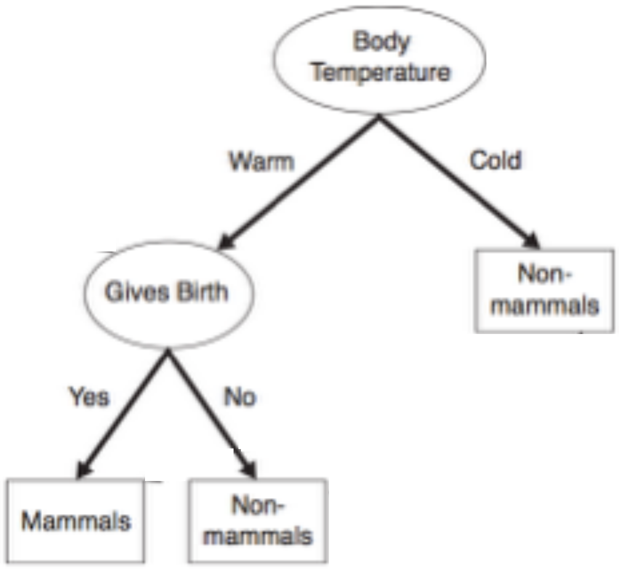
edges *are the answers to these questions.*

In other words, a tree is a **directed acyclic graph***.*

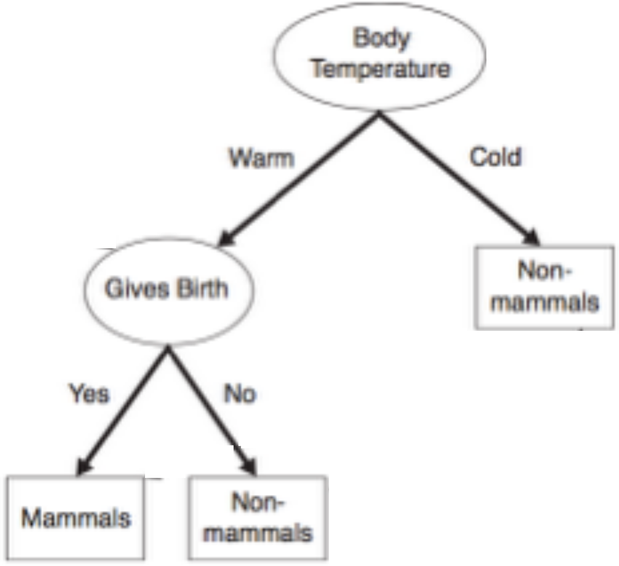
NOTE

The edges in the graph lead from a *parent node* to a *child node.*

Decision trees are a non-parametric hierarchical classification technique



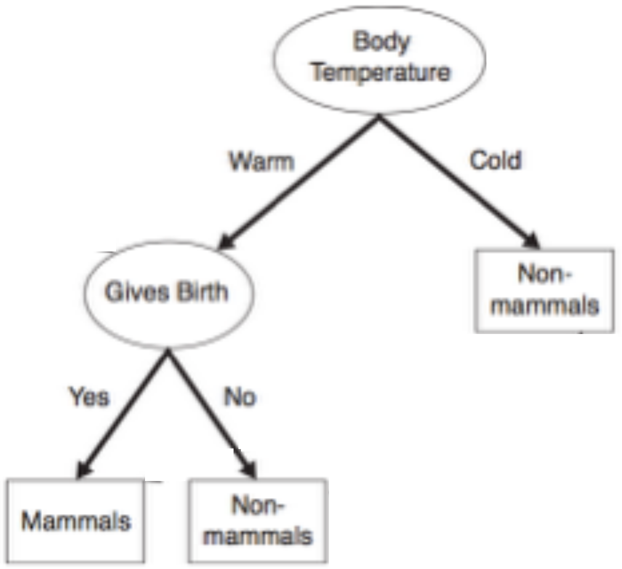
Decision trees are a non-parametric hierarchical classification technique



non-parametric

no parameters, no distribution assumptions

Decision trees are a non-parametric hierarchical classification technique



non-parametric

no parameters, no distribution assumptions

hierarchical

consists of a sequence of questions which yield a class label when applied to any record

II. FITTING DECISION TREES

Q: How do we build a decision tree?

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → $O(2^n)$.

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → $O(2^n)$.

Q: How do we find a practical solution that works?

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → $O(2^n)$.

Q: How do we find a practical solution that works?

A: Use a **heuristic** *algorithm.*

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm***.*

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm***.*

This is a **greedy recursive** *algorithm that leads to a* **local optimum***.*

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm***.*

This is a **greedy recursive** *algorithm that leads to a* **local optimum***.*

greedy – *algorithm makes locally optimal decision at each step*
recursive – *splits task into subtasks, solves each the same way*
local optimum – *solution for a given neighborhood of points*

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called **purity**.

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called **purity**.

A partition is **100% pure** *when all of its records belong to a single class.*

Hunt's algorithm:

Hunt's algorithm:

1. *If all samples belong to class* **y**, *then* **t** *is a leaf node corresponding to class* **y**, *and you're done (100% purity)*

Hunt's algorithm:

1. *If all samples belong to class* **y**, *then* **t** *is a leaf node corresponding to class* **y**, *and you're done*
2. *Else, create a test condition to partition the records:* **t** *is now an internal node with outgoing edges to possible outcomes*

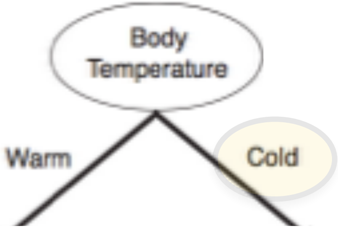
Hunt's algorithm:

1. *If all samples belong to class* **y**, *then* **t** *is a leaf node corresponding to class* **y**, *and you're done*
2. *Else, create a test condition to partition the records:* **t** *is now an internal node with outgoing edges to possible outcomes*
3. *Apply these steps to each child node*

Let's try an example

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

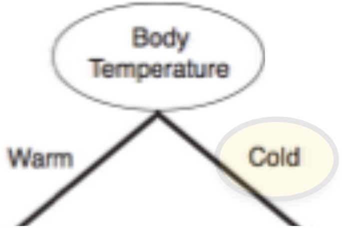
Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian



Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

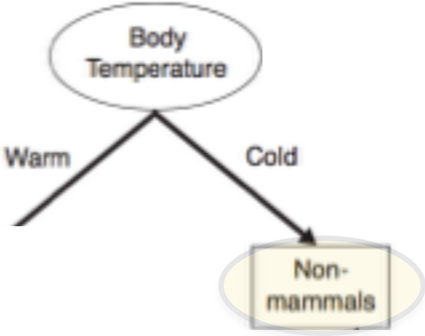
DECISION TREE CLASSIFIERS

This segment is 100% pure since all of its records belong to a single class (non-mammals)

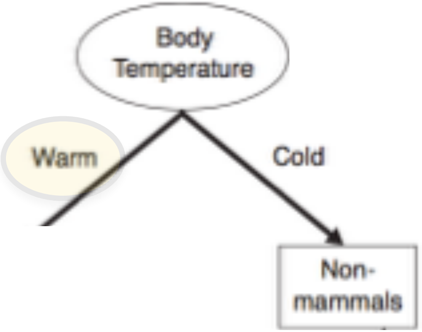


Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

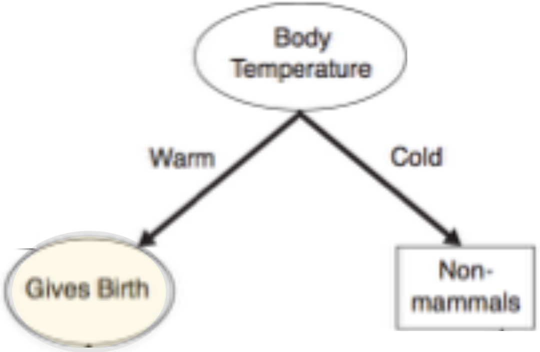
This segment is 100% pure since all of its records belong to a single class (non-mammals)



Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

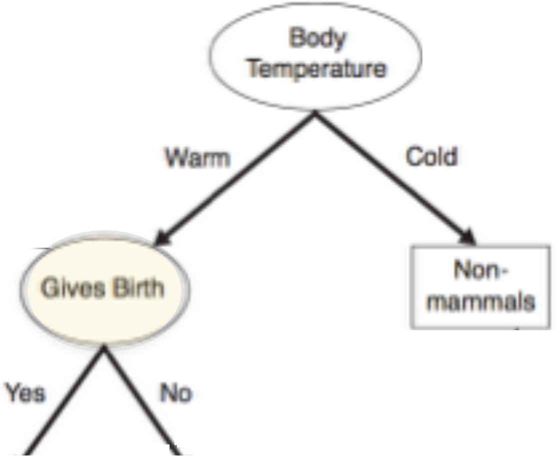


Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian



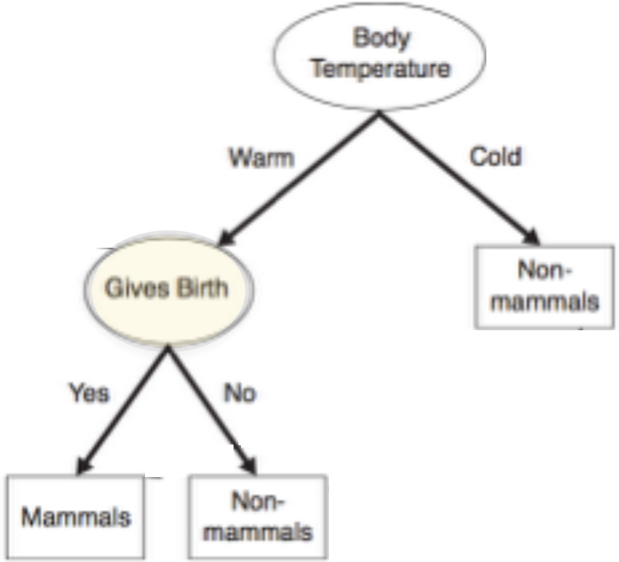
Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

DECISION TREE CLASSIFIERS

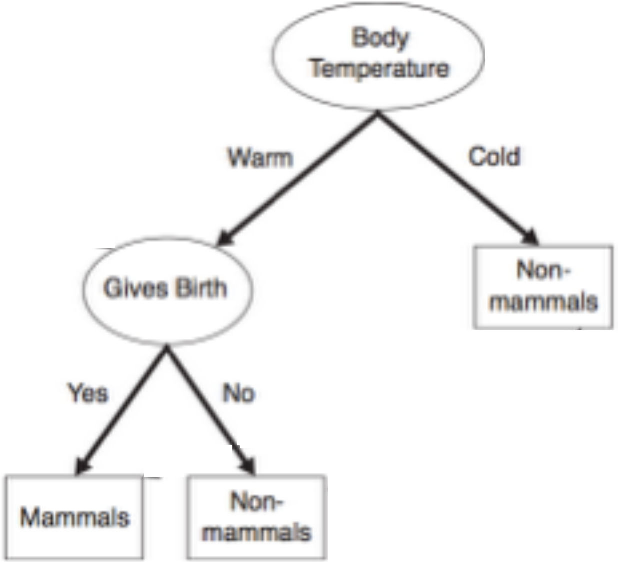


Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Both segments are 100% pure (mammals vs. non-mammals)

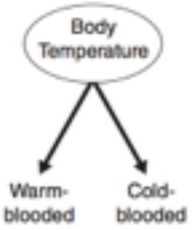


Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

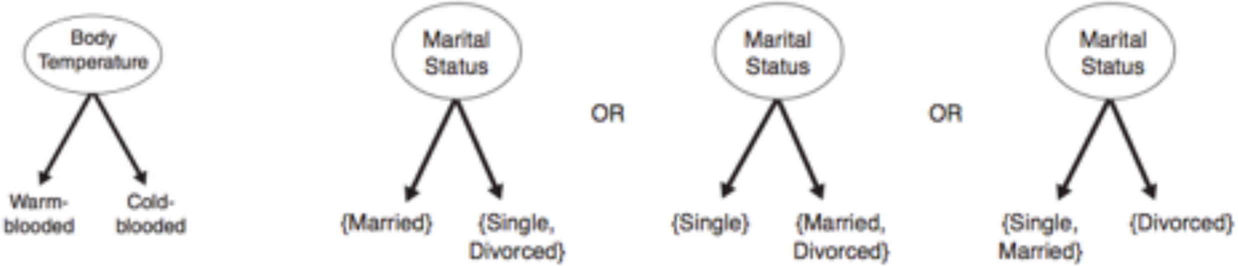


Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

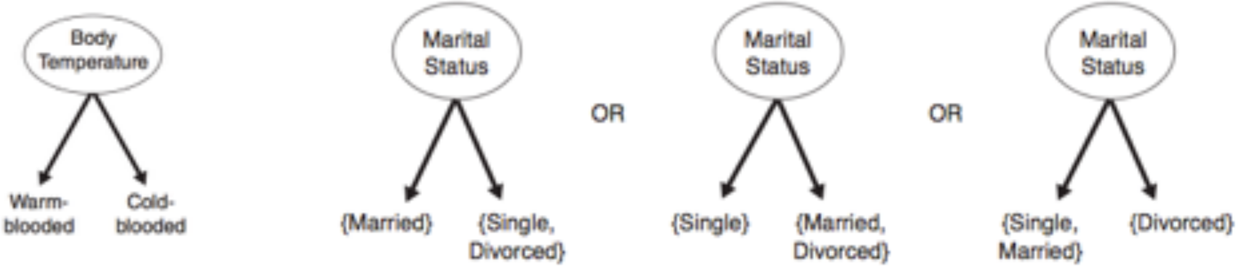
Splits can be binary



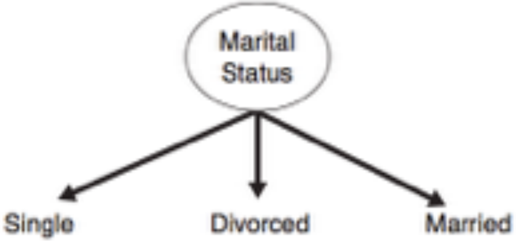
Splits can be binary *...also for features with more than 2 categories*



Splits can be binary

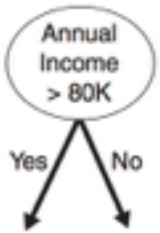


...or multiway

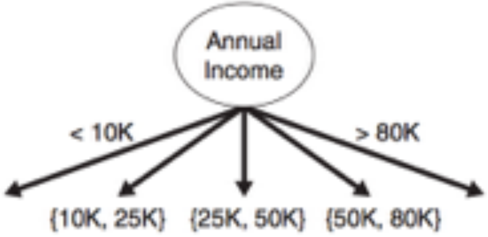


Splits can be binary

...same applies to continuous features



...or multiway



Q: How do we determine the best split?

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the **highest possible purity**.

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the **highest possible purity***.*

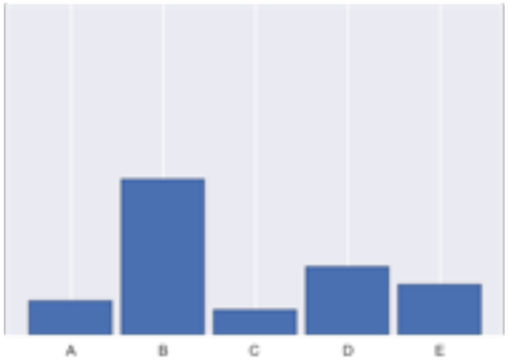
We need an objective function to optimize!

III. OBJECTIVE FUNCTIONS

Q: How do we measure purity?

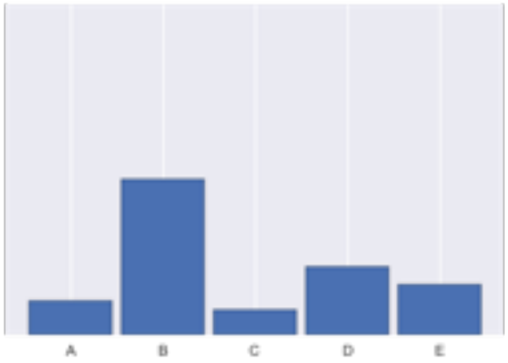
Q: How do we measure purity?

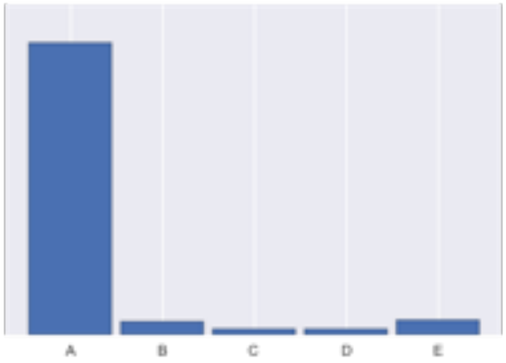
A: We can look at the distribution of class labels



Q: How do we measure purity?

A: We can look at the distribution of class labels

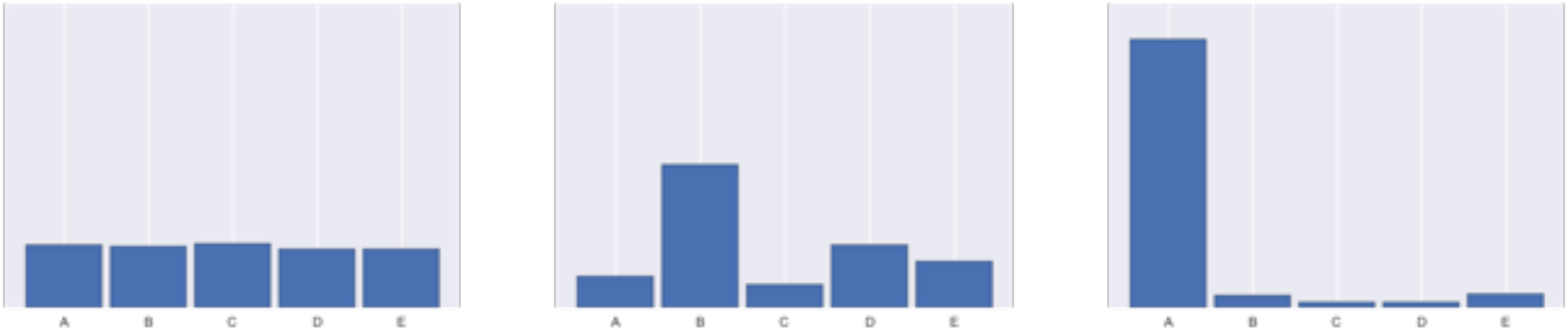




Very pure: almost all samples belong to the same class

Q: How do we measure purity?

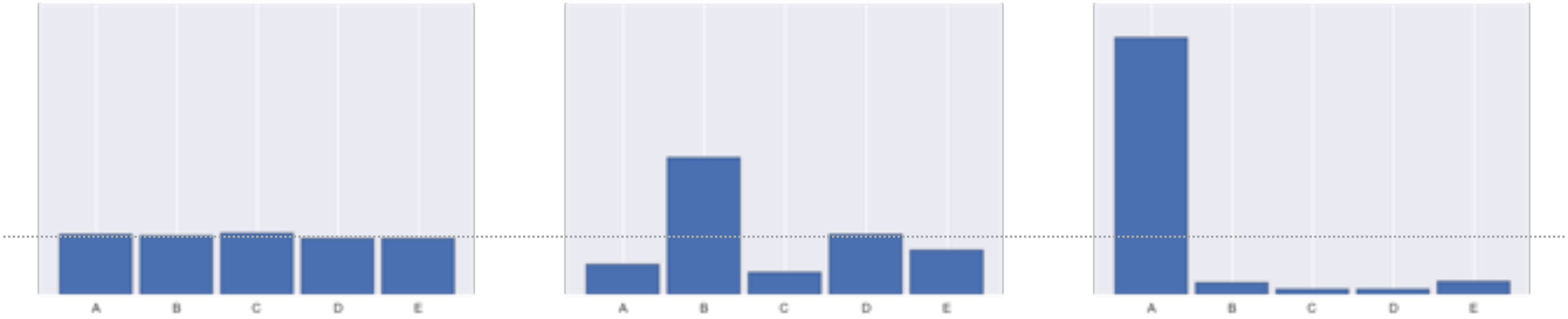
A: We can look at the distribution of class labels



Not pure at all: almost all classes are equally represented

Q: How do we measure purity?

A: We can look at the distribution of class labels



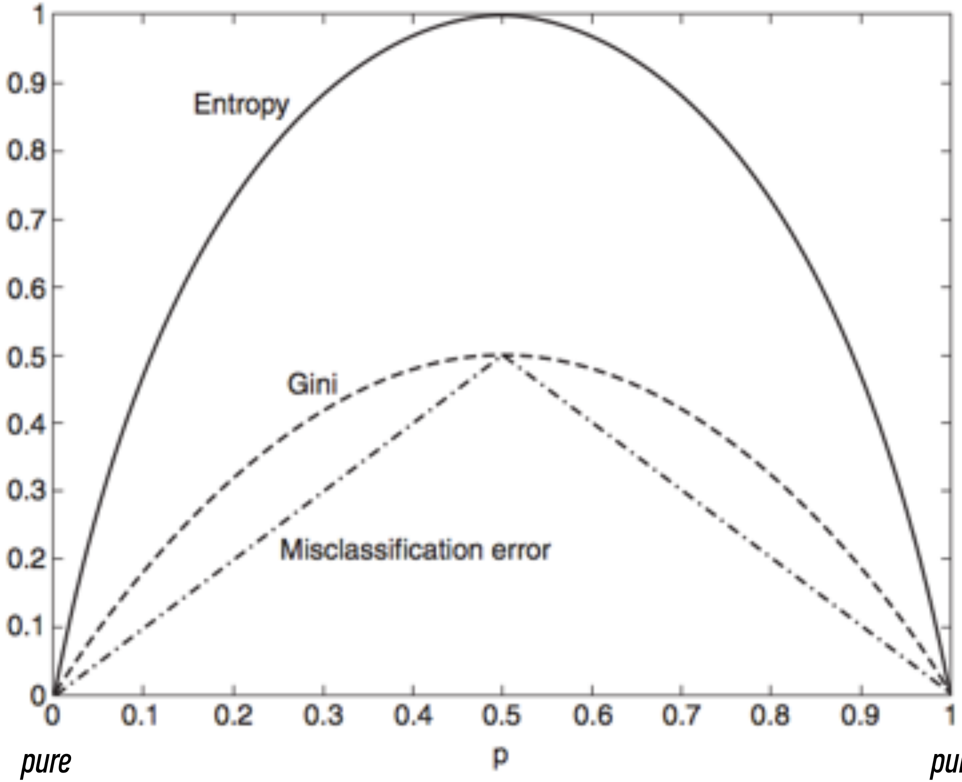
How far is the distribution away from the uniform distribution?

We have several metrics we could choose:

$$\begin{aligned}\text{Entropy}(t) &= -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t), \\ \text{Gini}(t) &= 1-\sum_{i=0}^{c-1}[p(i|t)]^2, \\ \text{Classification error}(t) &= 1-\max_i[p(i|t)],\end{aligned}$$

where $p(i|t)$ *is the fraction of records labeled i at node t*

For a binary classifier, each measure achieves its maximum at 0.5, and its minimum at 0 and 1.



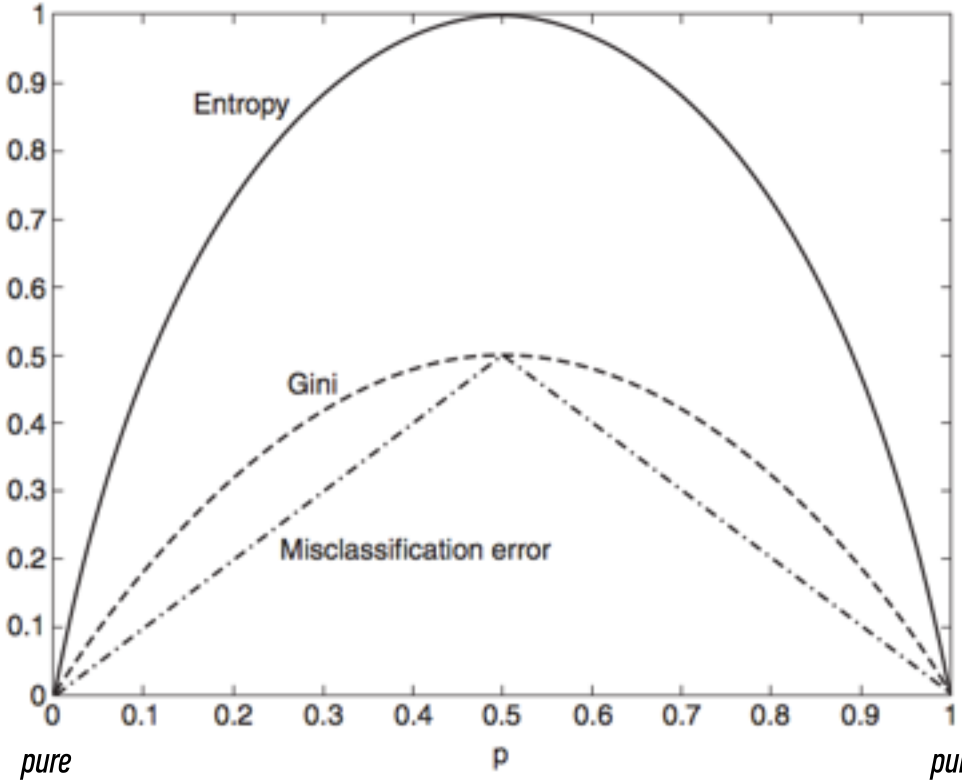
pure (all negative)

pure (all postive)

For a binary classifier, each measure achieves its maximum at 0.5, and its minimum at 0 and 1.

NOTE

Despite consistency, different measures may create different splits.



pure (all negative) *pure* (all postive)

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

We still need to look at impurity before & after the split.

We can make this comparison using the **gain***:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I is the impurity measure, N_j denotes the number of records at child node j, and N denotes the number of records at the parent node.)

We can make this comparison using the **gain**:

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I is the impurity measure, N_j denotes the number of records at child node j, and N denotes the number of records at the parent node.)

When I is the entropy, this quantity is called the **information gain**.

Having chosen an objective function, we could now create a decision tree by walking through all features, considering each split, and creating nodes for the split with the highest gain.

Having chosen an objective function, we could now create a decision tree by walking through all features, considering each split, and creating nodes for the split with the highest gain.

But there's one big issue with this…

which one?

IV. REGULARIZATION
(PREVENTING OVERFITTING)

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only. (e.g., the CART algorithm)

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only. (e.g., the CART algorithm)

Another way is to use a splitting criterion which explicitly penalizes the number of outcomes. (e.g., C4.5)

Still, only using binary splits, if we only stop splitting when all samples belong to the same class (or when all samples have identical features), we would likely overfit.

Still, only using binary splits, if we only stop splitting when all samples belong to the same class (or when all samples have identical features), we would likely overfit.

One possibility is **pre-pruning***, which involves setting a minimum gain, and stopping when no split achieves this threshold.*

Still, only using binary splits, if we only stop splitting when all samples belong to the same class (or when all samples have identical features), we would likely overfit.

One possibility is **pre-pruning***, which involves setting a minimum gain, and stopping when no split achieves this threshold.*

This prevents overfitting, but is difficult to calibrate in practice.

Alternatively we build the full tree, and then **prune** *it afterwards.*

Alternatively we build the full tree, and then **prune** *it afterwards.*

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

Alternatively we build the full tree, and then **prune** *it afterwards.*

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

Complicated subtrees can be replaced either with

- *a single node (called* **subtree replacement***)*

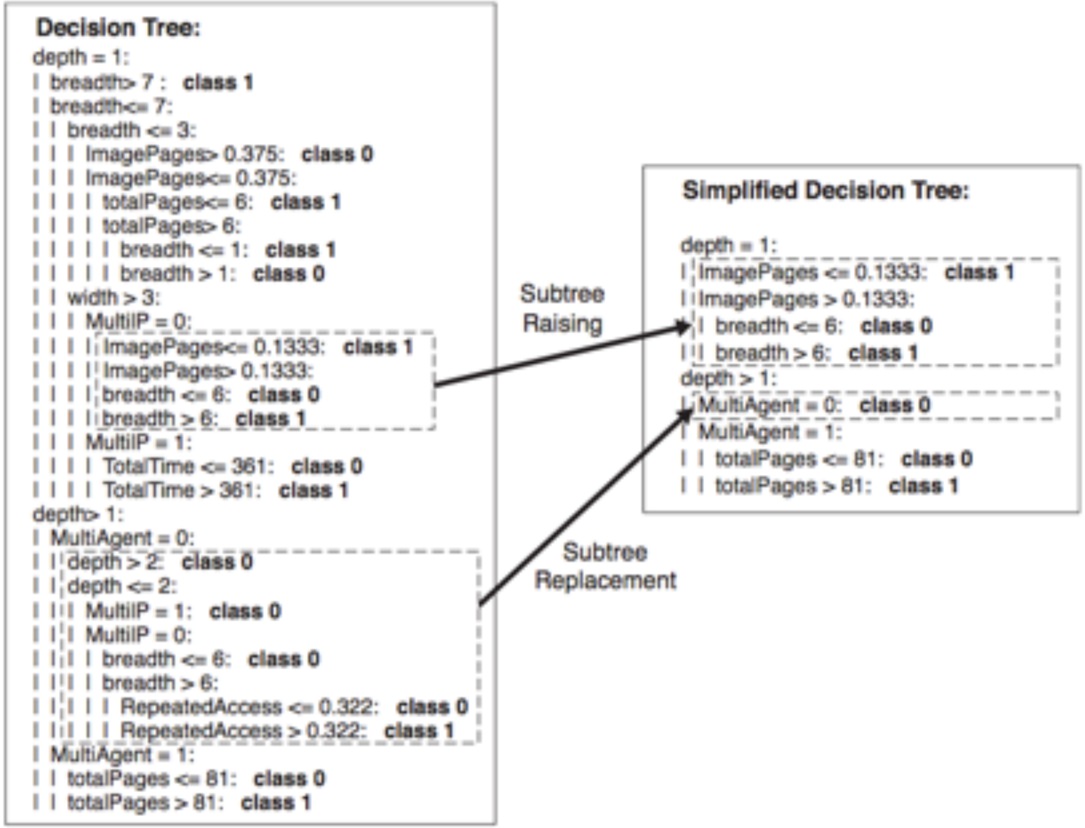
Alternatively we build the full tree, and then **prune** *it afterwards.*

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

Complicated subtrees can be replaced either with

- *a single node (called* **subtree replacement***), or*
- *with a simpler subtree (***subtree raising***).*

DECISION TREE CLASSIFIERS



Another, very powerful method to prevent overfitting is using **ensemble methods**, *like* **bagging** *and* **boosting**.

V. ENSEMBLE TECHNIQUES

	continuous	categorical
supervised	regression	classification
unsupervised	dimension reduction	clustering

Ensemble techniques *are methods of improving classification accuracy by aggregating predictions over several* **base classifiers***.*

Ensemble techniques *are methods of improving classification accuracy by aggregating predictions over several* **base classifiers***.*

Ensembles are often much more accurate than the base classifiers that compose them.

Ensemble techniques *are methods of improving classification accuracy by aggregating predictions over several* **base classifiers***.*

Ensembles are often much more accurate than the base classifiers that compose them.

The idea is that the poor predictions cancel each other out, while the strong ones remain

Ensemble techniques *are methods of improving classification accuracy by aggregating predictions over several* **base classifiers**.

Ensembles are often much more accurate than the base classifiers that compose them.

The idea is that the poor predictions cancel each other out, while the strong ones remain

NOTE

Base classifiers and ensemble classifiers are sometimes called ***weak learners*** and ***strong learners***.

An ensemble classifier can only outperform a single base classifier if the following conditions are met:

An ensemble classifier can only outperform a single base classifier if the following conditions are met:

1. *the base classifier must be **accurate***
   they must outperform random guessing

An ensemble classifier can only outperform a single base classifier if the following conditions are met:

1. *the base classifier must be **accurate***
   they must outperform random guessing

2. *the base classifier must be **diverse***
   their misclassifications must occur on different training examples

An ensemble classifier can mitigate three kinds of common problems in supervised learning

Statistical *Computational* *Representational*

An ensemble classifier can mitigate three kinds of common problems in supervised learning

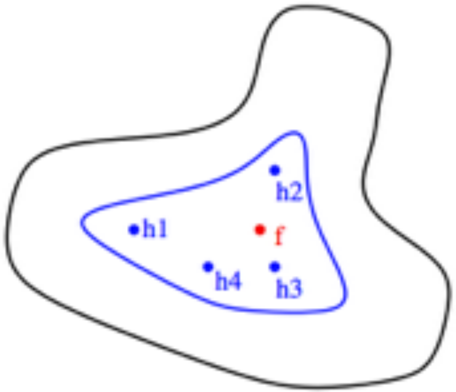
Statistical Computational Representational

Little data (or many features) cause the classifier to overfit.

An ensemble can mitigate this problem by "averaging out" base classifier predictions.

Statistical Computational Representational

Little data (or many features) cause the classifier to overfit.



source: http://www.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf

Statistical

Computational

Representational

Little data (or many features) cause the classifier to overfit.

It may be computationally hard to find the best classifier.

For example, some classifiers require an exhaustive search of all possibilities, which is very expensive (NP-complete).
(e.g. decision trees)

Statistical

Little data (or many features) cause the classifier to overfit.

Computational

It may be computationally hard to find the best classifier.

For example, some classifiers require an exhaustive search of all possibilities, which is very expensive (NP-complete).

(e.g. decision trees)

Representational

NOTE

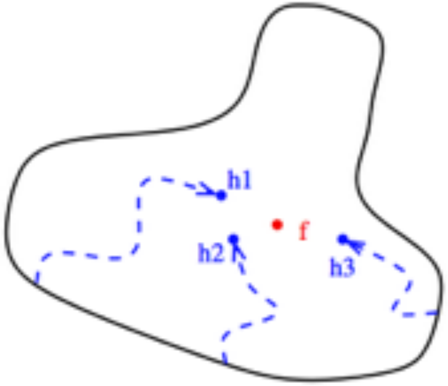
Recall that this is why we used a *heuristic algorithm* (greedy search).

Statistical

Little data (or many features) cause the classifier to overfit.

Computational

It may be computationally hard to find the best classifier.



Different starting points provide better results than a single base classifier

Representational

source: http://www.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf

Statistical

Little data (or many features) cause the classifier to overfit.

Computational

It may be computationally hard to find the best classifier.

Representational

The ideal classifier is impossible to express in the chosen model

An ensemble can express more complex structures than a single base classifier.

Statistical

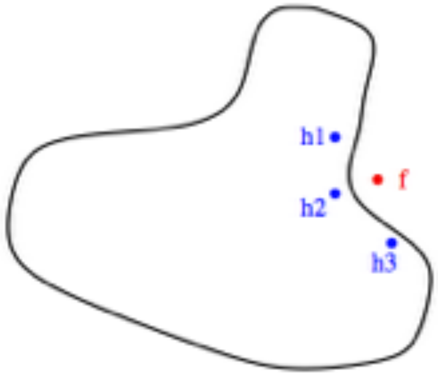
Little data (or many features) cause the classifier to overfit.

Computational

It may be computationally hard to find the best classifier.

Representational

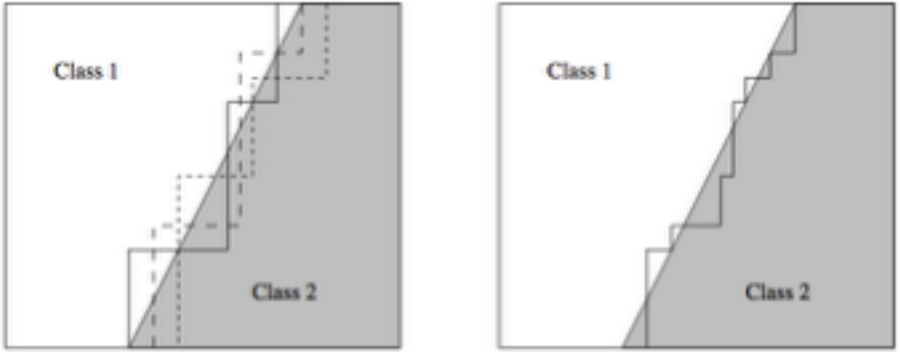
The ideal classifier is impossible to express in the chosen model



source: http://www.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf

Representational

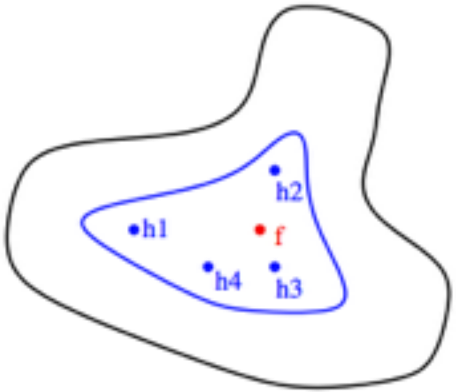
The ideal classifier is impossible to express in the chosen model



For example, a decision tree with limited depth can only represent a small number of rectilinear segments. It is therefore a bad model for data with a diagonal decision boundary. However, it may be still be possible to approximate the boundary using ensemble methods.

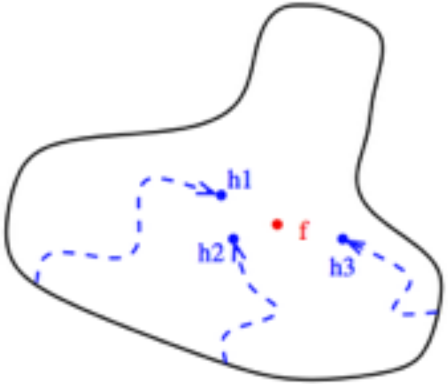
Statistical

Little data (or many features) cause the classifier to overfit.



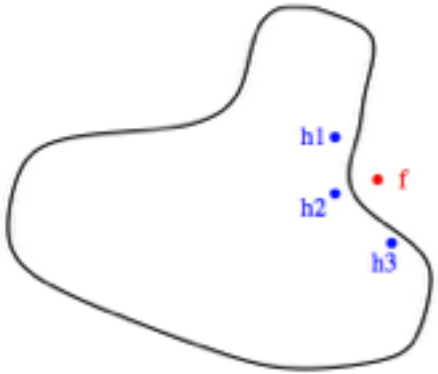
Computational

It may be computationally hard to find the best classifier.



Representational

The ideal classifier is impossible to express in the chosen model



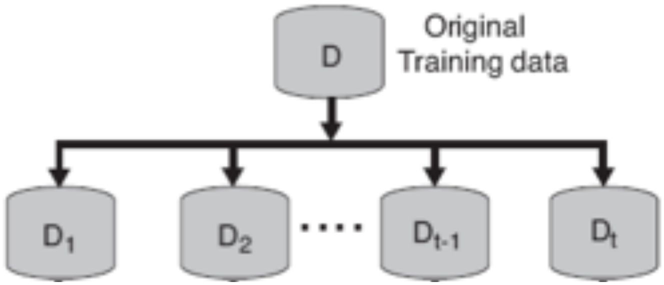
source: http://www.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf

V. ENSEMBLE TECHNIQUES – BAGGING

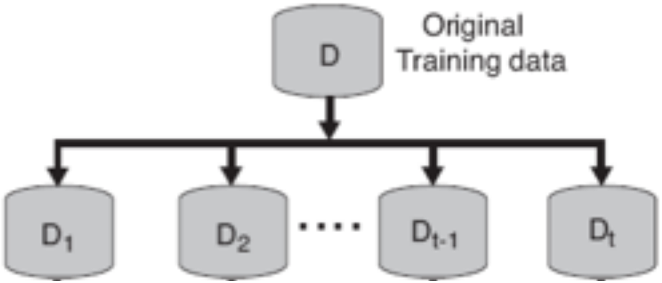
Bagging *is short for bootstrap aggregating.*

Bagging *is short for bootstrap aggregating.*

How does bagging work?



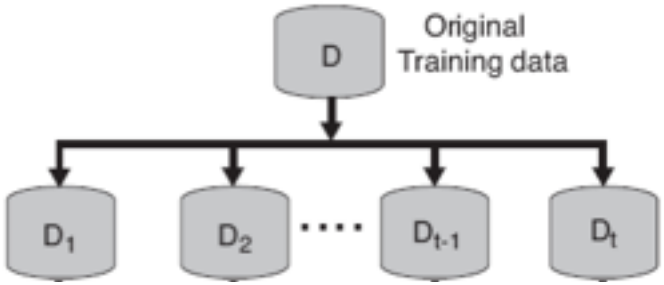
1. *Split your data into t different sets of the same size (sampling with replacement)*



1. *Split your data into t different sets of the same size (sampling with replacement)*

NOTE

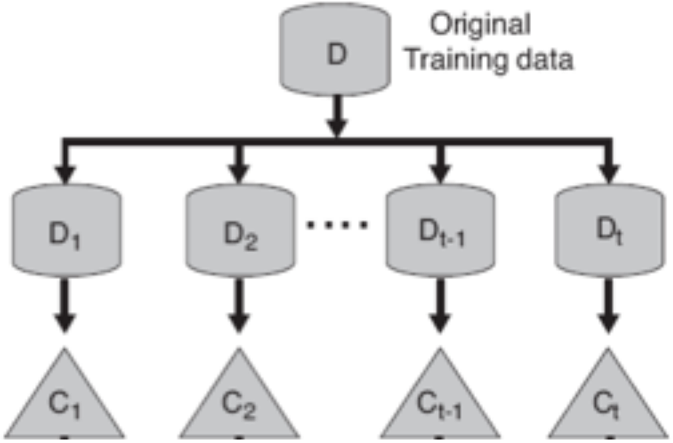
This procedure is called a ***bootstrap***



1. *Split your data into t different sets of the same size (sampling with replacement)*

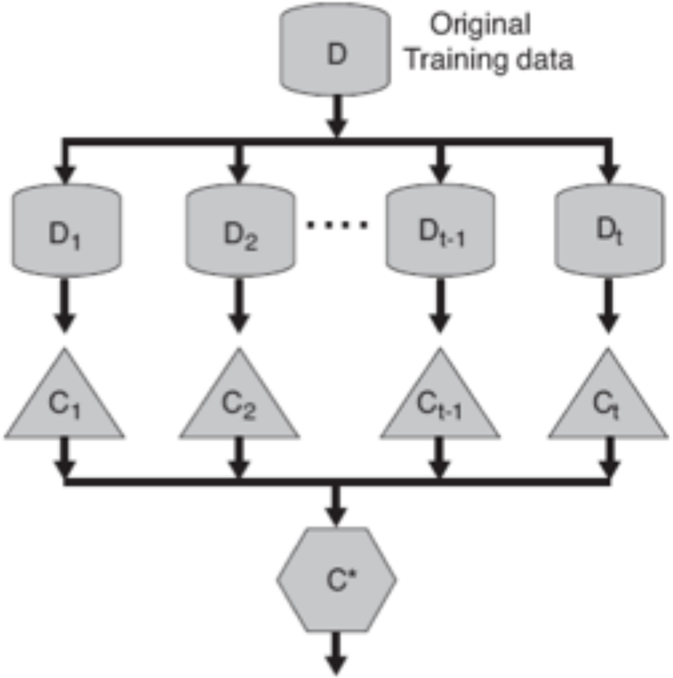
NOTE

Resampling means that some training records may appear in a sample more than once, or even not at all.



1. *Split your data into t different sets of the same size (sampling with replacement)*

2. *Train t base classifiers on each dataset*



1. *Split your data into t different sets of the same size (sampling with replacement)*
2. *Train t base classifiers on each dataset*
3. *Take majority vote*

Bagging reduces the variance (overfitting) by aggregating multiple base classifiers together.

Bagging reduces the variance (overfitting) by aggregating multiple base classifiers together.

If the base classifiers are under-fit, however, then the ensemble error is primarily due to base classifier bias, and bagging may not be effective.

Bagging reduces the variance (overfitting) by aggregating multiple base classifiers together.

If the base classifiers are under-fit, however, then the ensemble error is primarily due to base classifier bias, and bagging may not be effective.

Because of the bootstrap sampling of training data,
bagging is not very susceptible to overfitting.

V. ENSEMBLE TECHNIQUES – BOOSTING

Boosting *is similar to bagging:*

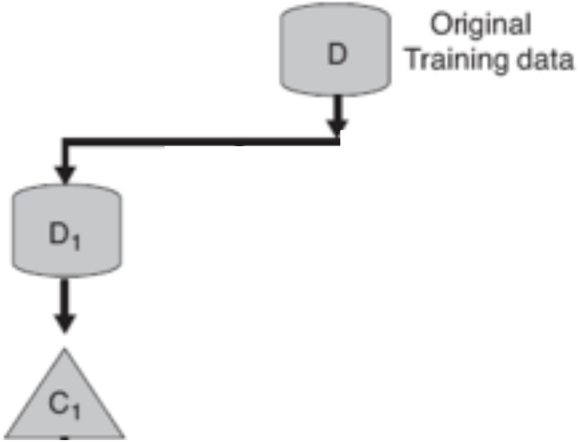
Instead of selecting data points randomly with the bootstrap,
we now favor the **misclassified samples***.*

Boosting *is similar to bagging:*

Instead of selecting data points randomly with the bootstrap, we now favor the **misclassified samples***.*

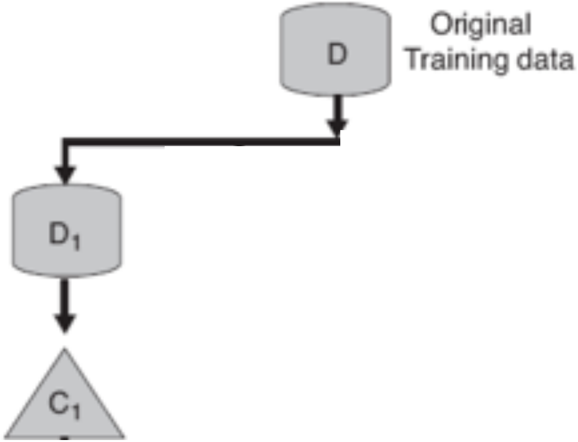
The first iteration uses uniform weights (like bagging). In subsequent iterations, the weights are adjusted to emphasize records that were misclassified in previous iterations.

Initialize the weights of your samples



Initialize the weights of your samples

1. *Resample your data with respect to the weights and train your base classifier*



Initialize the weights of your samples

1. *Resample your data with respect to the weights and train your base classifier*

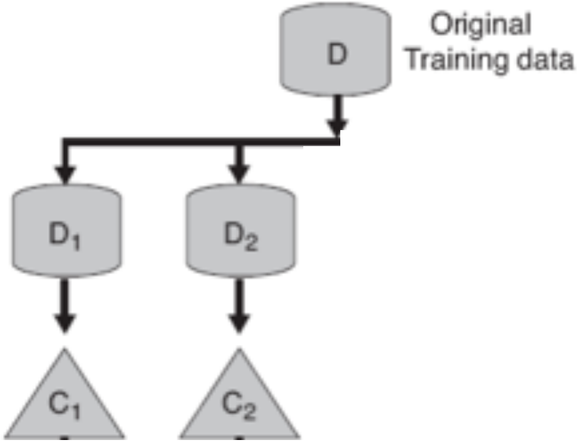
2. *Increase weights of misclassified samples*

$$\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$$

sum of weights for misclassified examples

$$D_{t+1}(i) = \frac{\epsilon_t}{1-\epsilon_t} D_t(i)$$

odds of misclassifying

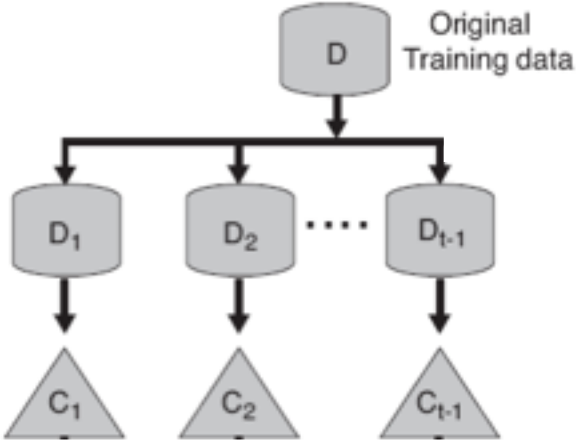


Initialize the weights of your samples

1. *Resample your data with respect to the weights and train your base classifier*

2. *Increase weights of misclassified samples*

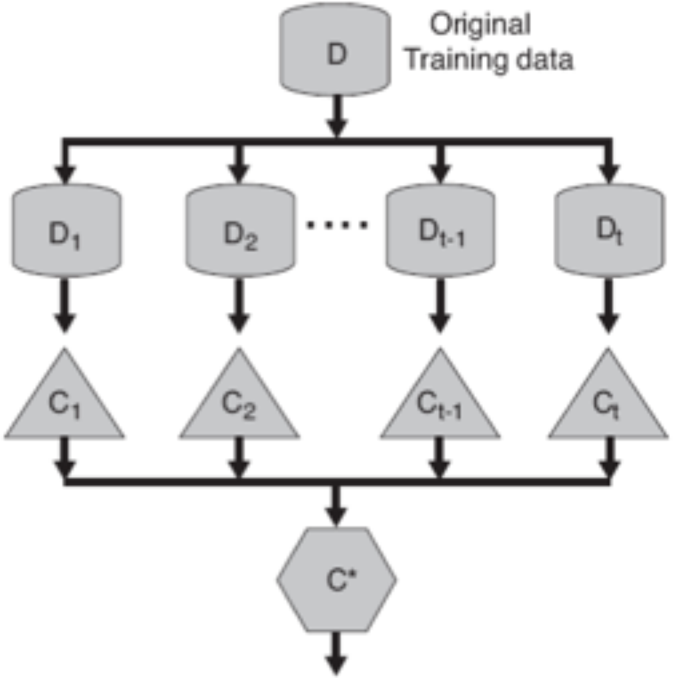
Repeat…



Initialize the weights of your samples

1. *Resample your data with respect to the weights and train your base classifier*
2. *Increase weights of misclassified samples*

Repeat…



Initialize the weights of your samples

1. *Resample your data with respect to the weights and train your base classifier*
2. *Increase weights of misclassified samples*

   Repeat…

3. *Take majority vote (possibly weighted)*

Like in bagging, sampling is done with replacement, and as a result some records may not appear in a given training sample.

Like in bagging, sampling is done with replacement, and as a result some records may not appear in a given training sample.

These omitted records will likely be misclassified, and given greater weight in subsequent iterations once the sampling distribution is updated.

Like in bagging, sampling is done with replacement, and as a result some records may not appear in a given training sample.

These omitted records will likely be misclassified, and given greater weight in subsequent iterations once the sampling distribution is updated.

So even if a record is left out at one stage, it will be emphasized later.

Updating the sampling distribution and forming an ensemble prediction leads to a nonlinear combination of the base classifiers.

Updating the sampling distribution and forming an ensemble prediction leads to a nonlinear combination of the base classifiers.

The base classifiers focus more and more closely on records that are difficult to classify as the sequence of iterations progresses.

Updating the sampling distribution and forming an ensemble prediction leads to a nonlinear combination of the base classifiers.

The base classifiers focus more and more closely on records that are difficult to classify as the sequence of iterations progresses.

Thus they're faced with progressively more difficult learning problems.

V. ENSEMBLE TECHNIQUES – RANDOM FORESTS

A **random forest** *is an ensemble of decision trees where each base classifier is grown using a random effect.*

A **random forest** *is an ensemble of decision trees where each base classifier is grown using a random effect.*

Each tree is grown on a bootstrapped dataset (i.e., bagging)

A **random forest** *is an ensemble of decision trees where each base classifier is grown using a random effect.*

Each tree is grown on a bootstrapped dataset (i.e., bagging)

But at each split, only a limited number of random features are considered. e.g., generally sqrt(n_features)

Random forests are about as accurate as AdaBoost, more robust to noise, and can also have better runtime than other ensemble methods (since the feature space is reduced in some cases).

DISCUSSION