10. Evaluation and Ensembles

Dongwoo Kim

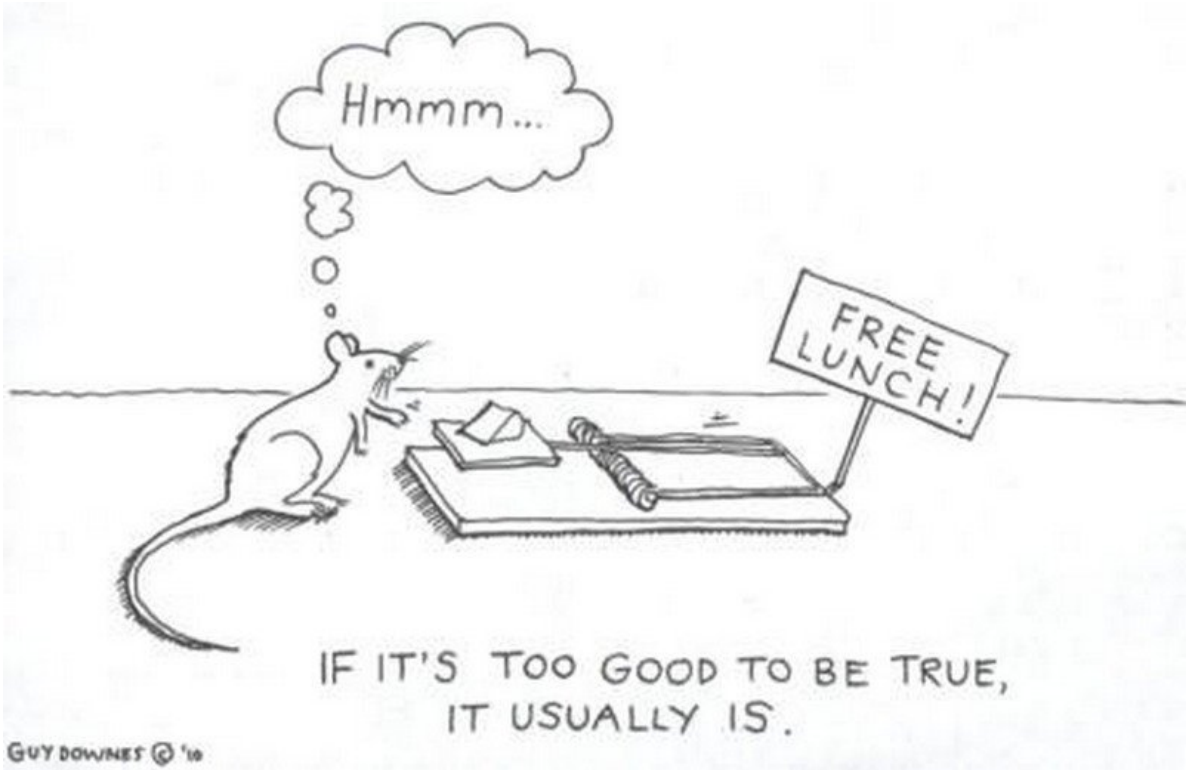
dongwookim.ac.kr

CSED515 - 2023 Spring

Table of Contents

No Free Lunch (Theorem)

We have studied several models so far, but which one to choose?

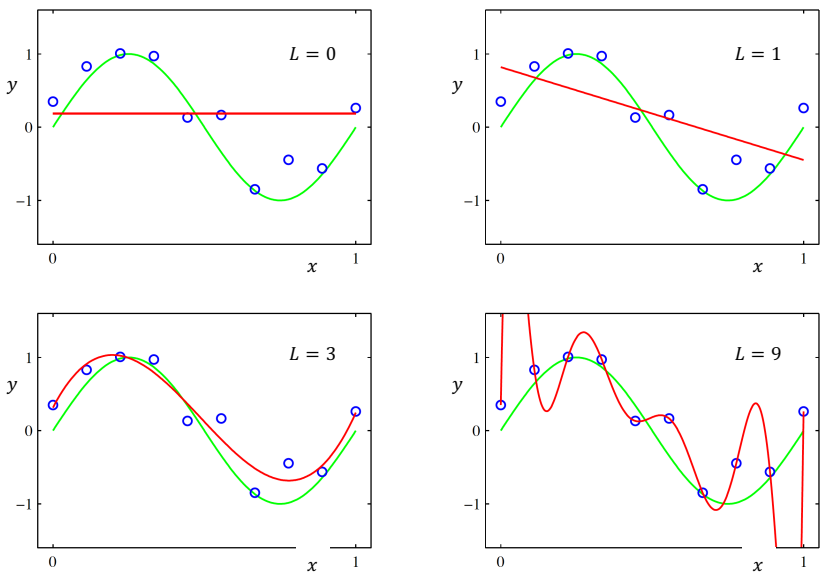


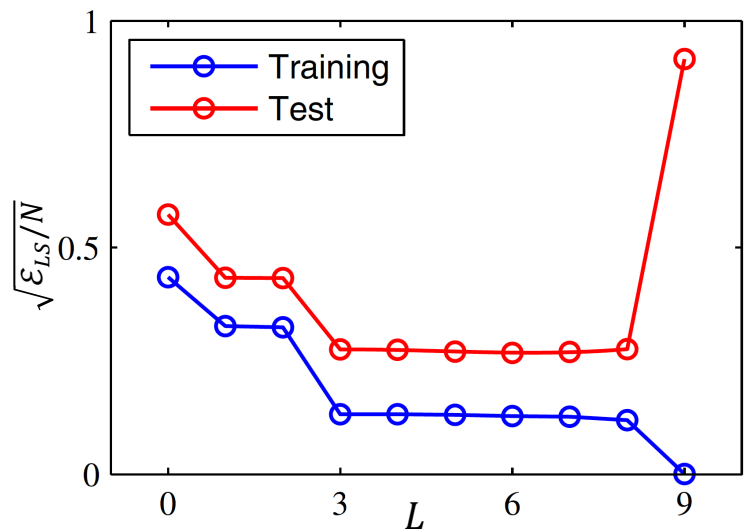
[No Free Lunch] There is no single learning algorithm that in any domain always induces the most accurate learner.

Generalization

When we talked about model overfitting, we evaluate the model performance on test set.

Generalization: model's ability to adapt properly to new, previously unseen data





Question: How can we evaluate the model performance fairly?

Test Set Revisited

How can we get an unbiased estimate of the accuracy of a learned model?

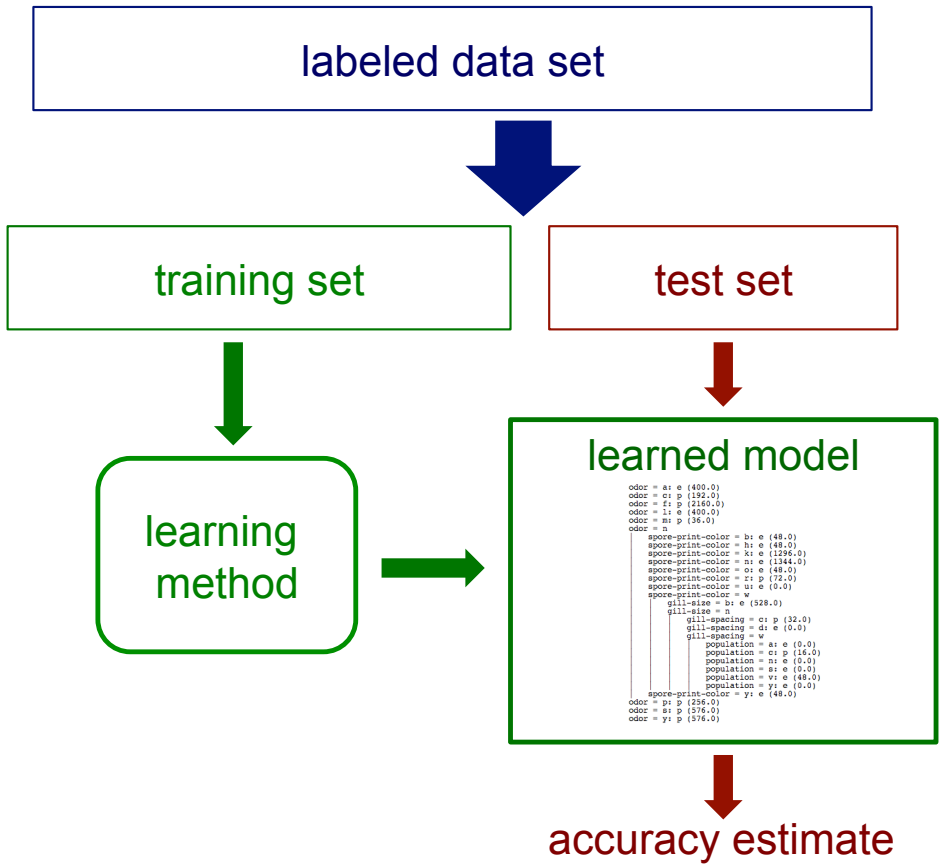


Figure from David Page

Test Set Revisited

How can we get an unbiased estimate of the accuracy of a learned model?

- When learning a model, you should pretend that you don't have the test data yet (it is "in the mail")*
- If the test-set labels influence the learned model in any way, accuracy estimates will be biased

* In some applications it is reasonable to assume that you have access to the feature vector (i.e. x) but not the y part of each test instance.

Learning Curve

How does the accuracy of a learning method change as a function of the training-set size?

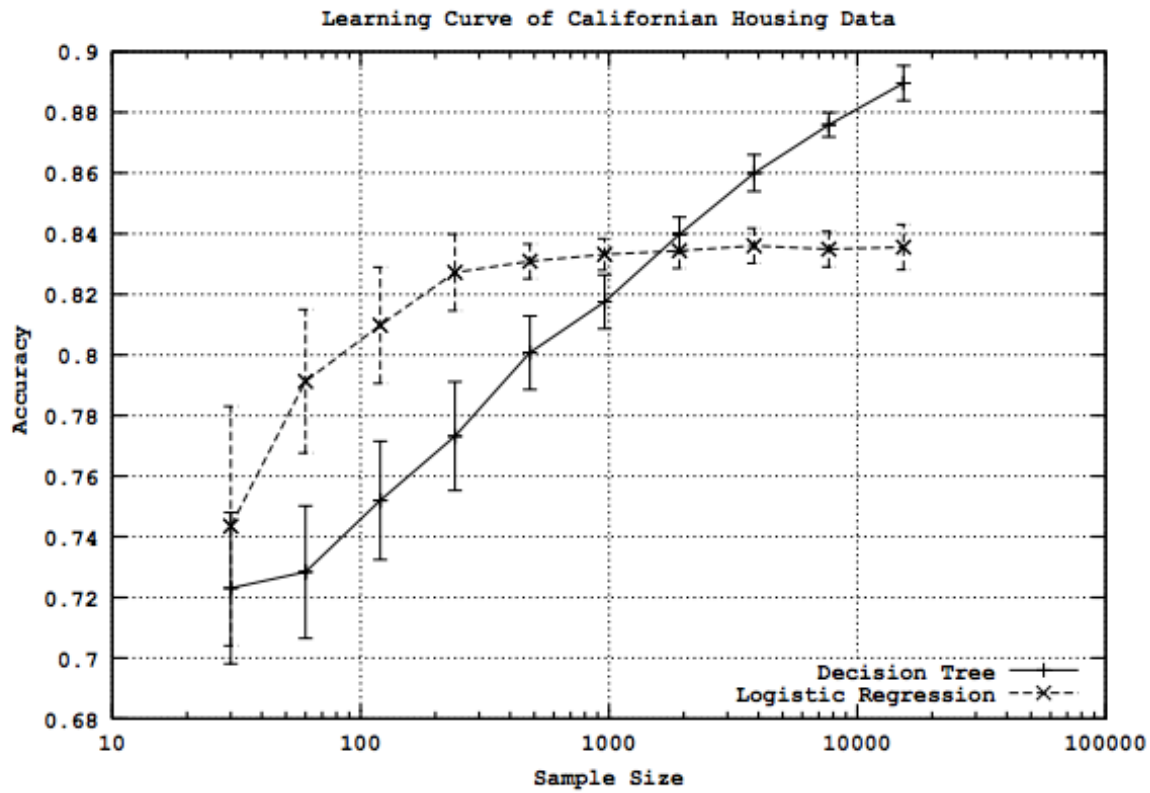


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

Learning Curve

Given training/test set partition

- For each sample size s on learning curve
  - (optionally) repeat n times
    - Randomly select s instances from training set
    - Learn model
    - Evaluate model on test set to determine accuracy a
    - Plot (s, a) or (s, avg. accuracy and error bars)

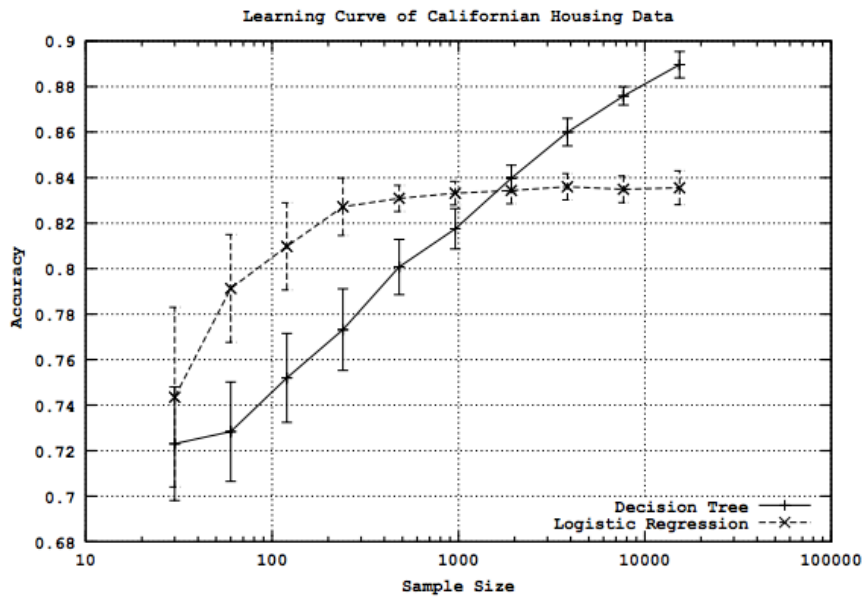


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

Validation (tuning) Sets

Suppose we want unbiased estimates of accuracy during the learning process (e.g. number of layers in MLP)?

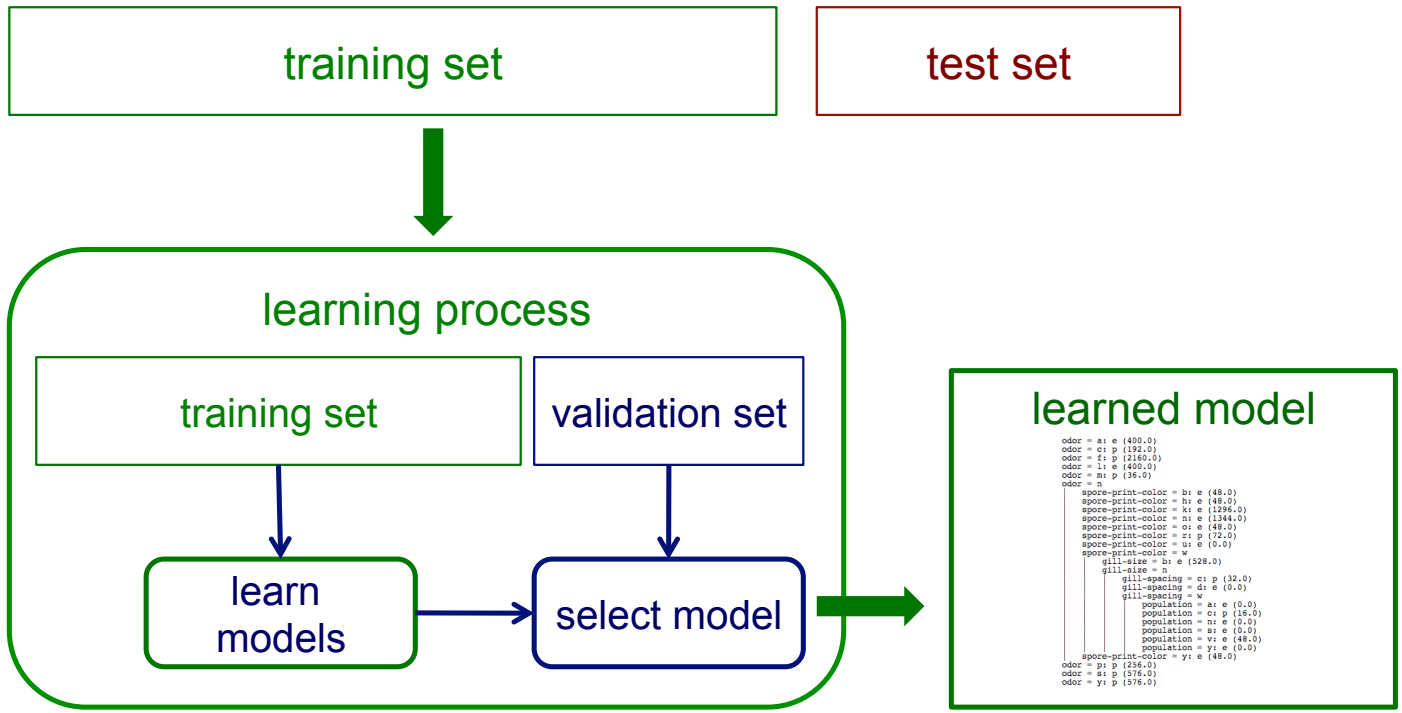


Figure from David Page

Limitations of using a single training/test partition

- We may not have enough data to make sufficiently large training and test sets
  - A larger test set gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  - But... a larger training set will be more representative of how much data we actually have for learning process
- A single training set doesn't tell us how sensitive accuracy is to a particular training sample

Random Resampling

We can address the second issue by repeatedly randomly partitioning the available data into training and set sets.

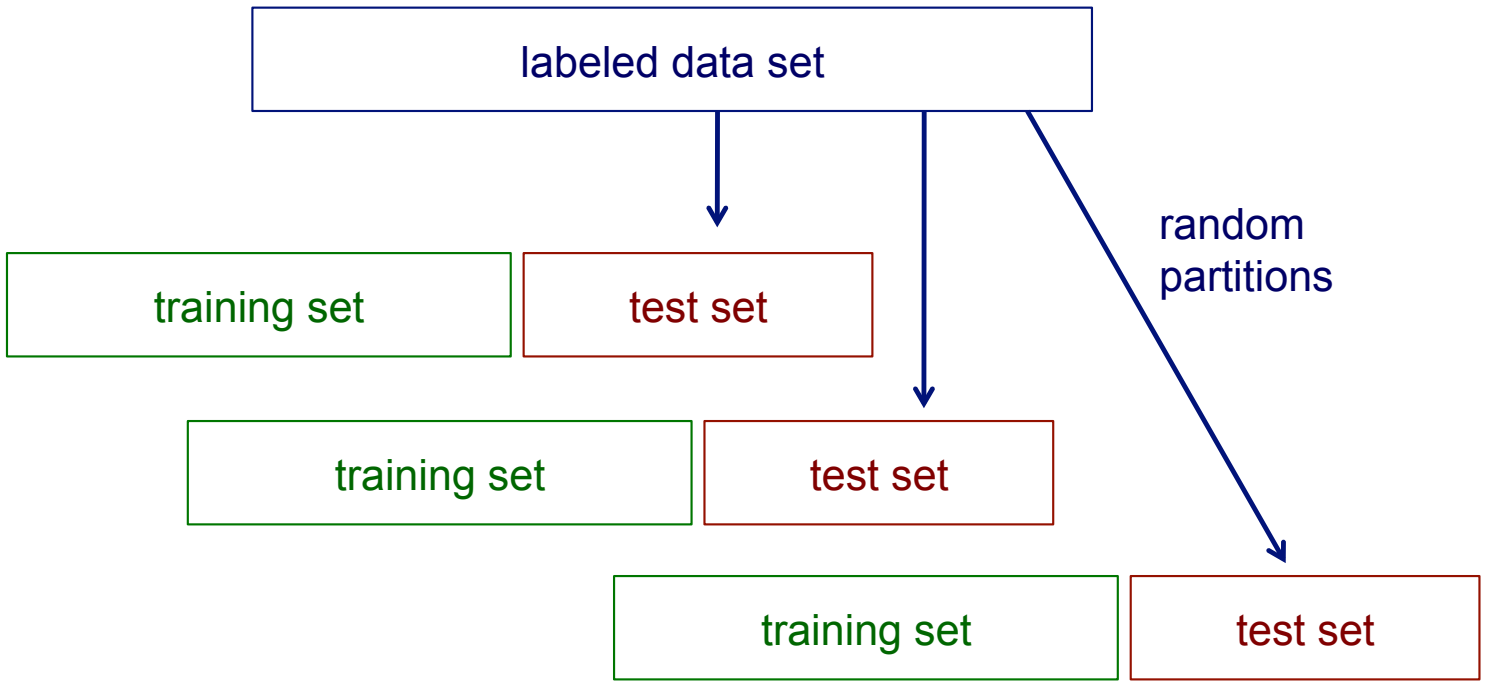
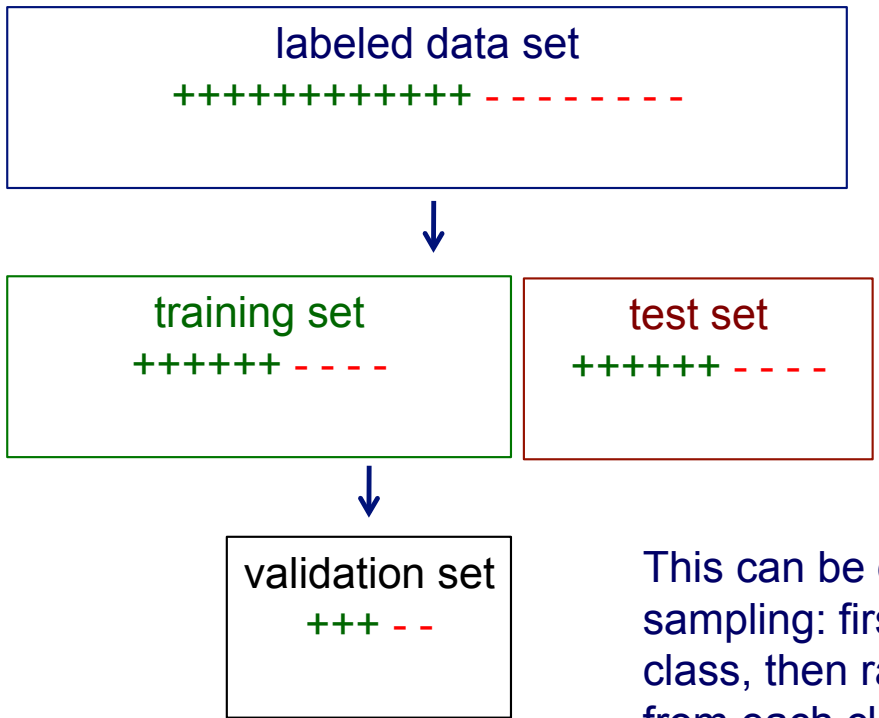


Figure from David Page

Stratified sampling

When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set.

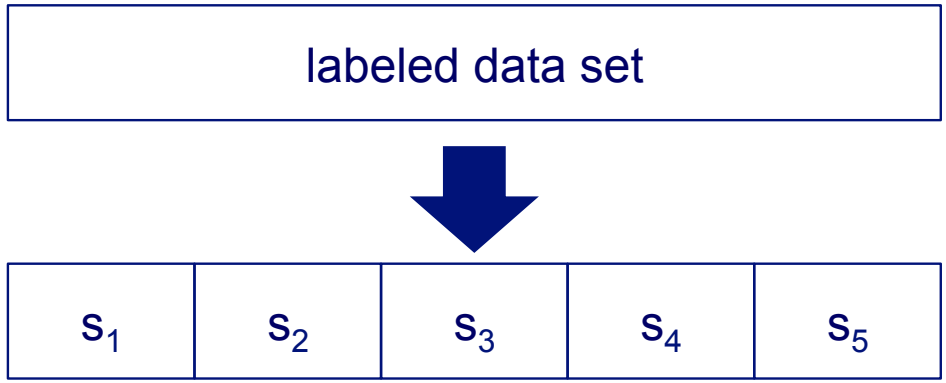


This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

Figure from David Page

Cross Validation

partition data into *n* subsamples



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Figure from David Page

Cross Validation Example

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

accuracy = 73/100 = 73%

Figure from David Page

Cross Validation

- 10 -fold cross validation is common, but smaller values of n are often used when learning takes a lot of time.
- In leave-one-out cross validation, $n = \#$ instances.
- In stratified cross validation, stratified sampling is used when partitioning the data.
- CV makes efficient use of the available data for testing.
- Note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a learning method as opposed to an individual learned model.

Internal cross validation

Instead of a single validation set, we can use cross-validation within a training set to select a model (e.g. to choose the best number of hidden units in MLP).

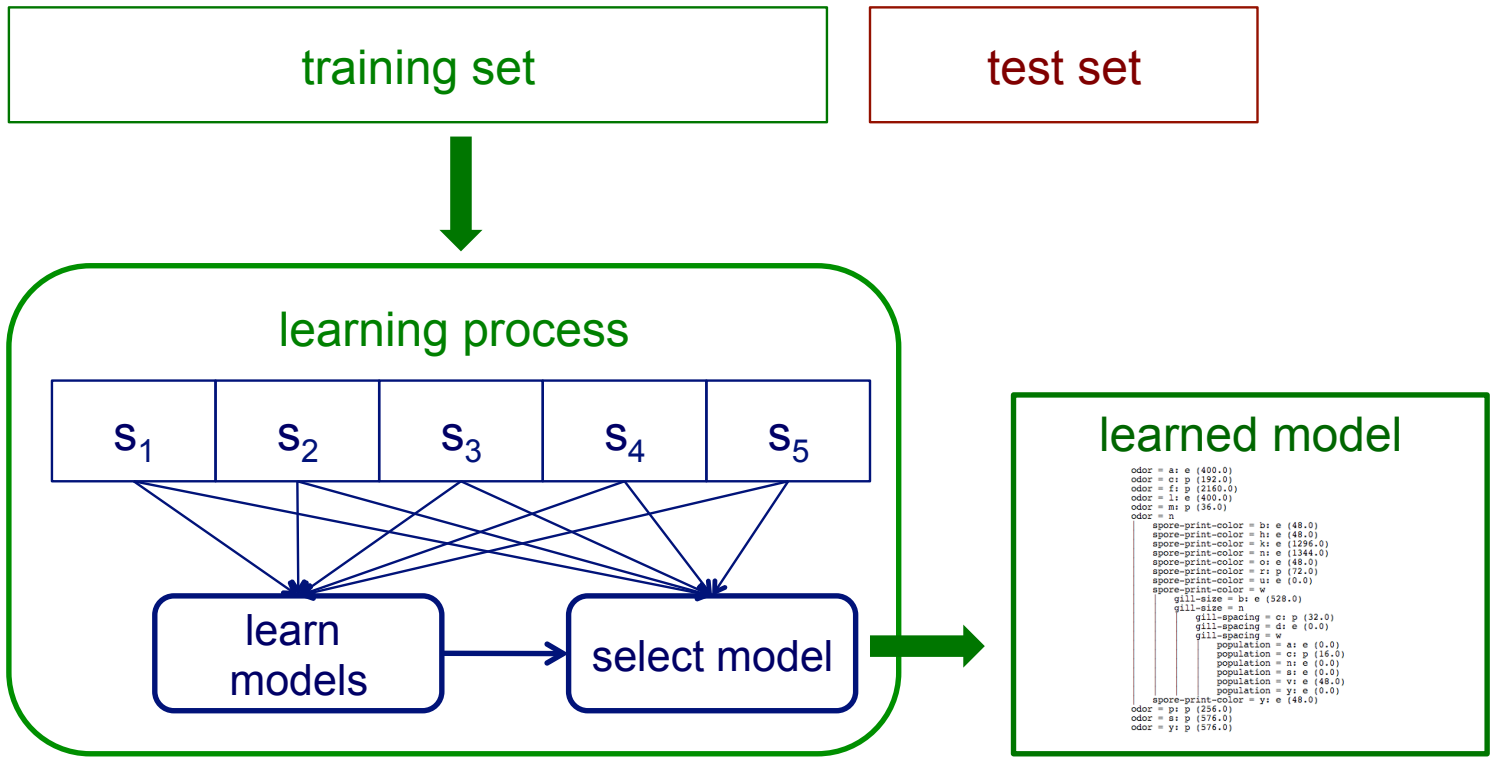


Figure from David Page

Example: using internal cross validation in MLP

Instead of a single validation set, we can use cross-validation within a training set to select a model (e.g. to choose the best number of layers in MLP).

Given a training set

1. partition training set into n folds, $s_1 \ldots s_n$
2. for each value of ℓ considered for $i = 1$ to n learn ℓ-layer MLP using all folds but s_i evaluate accuracy on s_i
3. select ℓ that resulted in best accuracy for $s_1 \ldots s_n$
4. learn model using entire training set and selected k

The steps are run independently for each training set (i.e. if we're using 10-fold CV to measure the overall accuracy, then the procedure would be executed 10 times)

Table of Contents

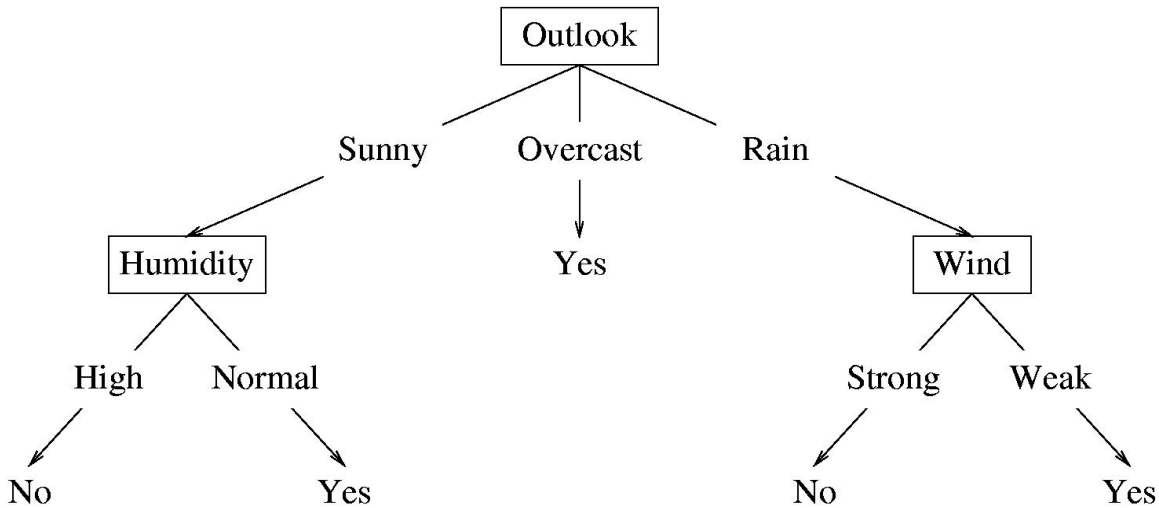
Model Ensemble

We have studied multiple models so far. Is there a way to combine multiple models to increase performance?

- Different models (SVM vs MLP)
- Different hyperparameters (1-hidden vs 2-hidden)
- Different input representations (input dimensionality reduction)
- Different training sets

We will talk about AdaBoost and Decision Tree.

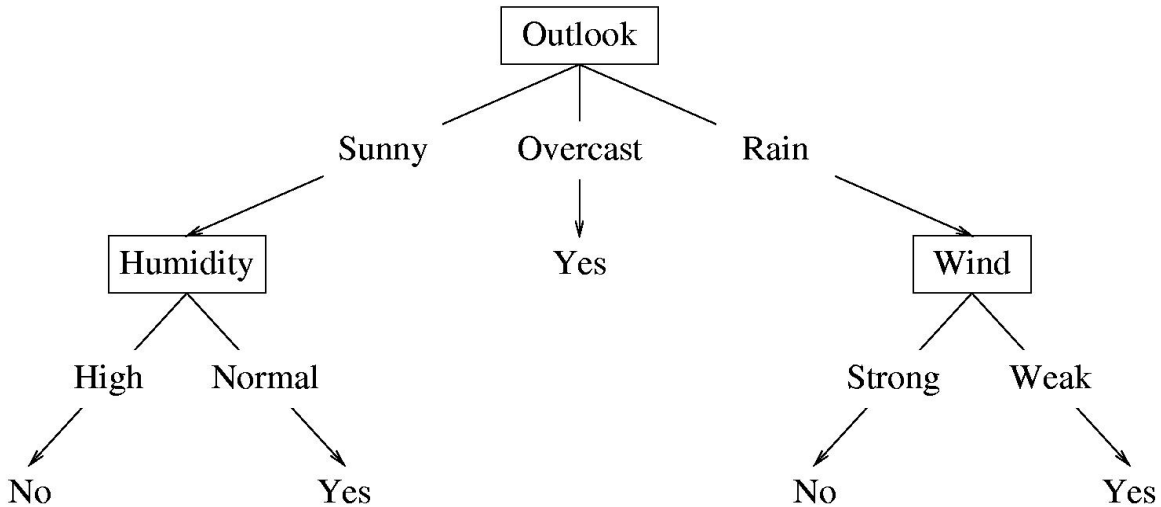
Decision Tree



Class label denotes whether a tennis game was played

- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y

Decision Tree

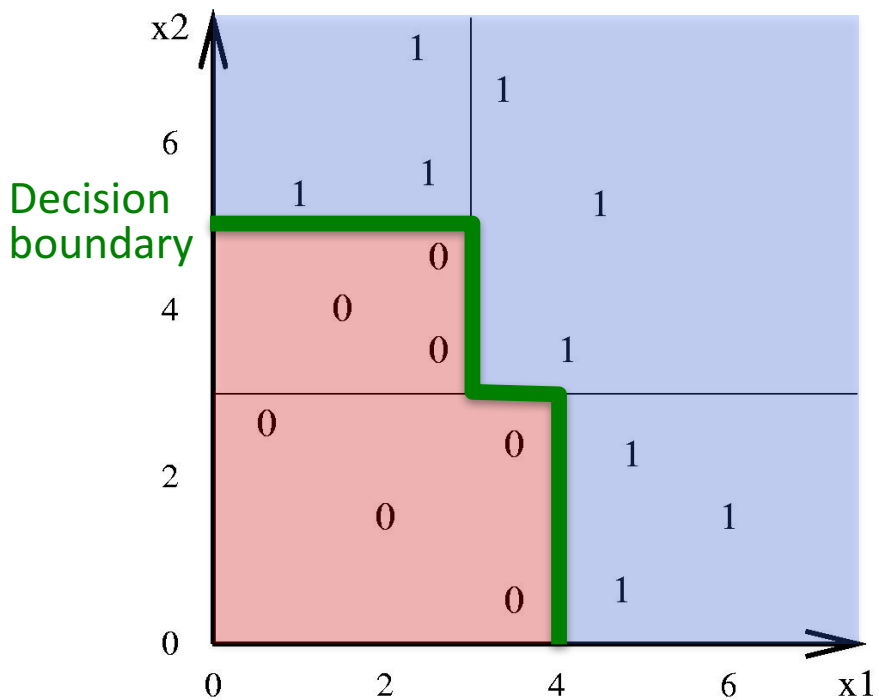


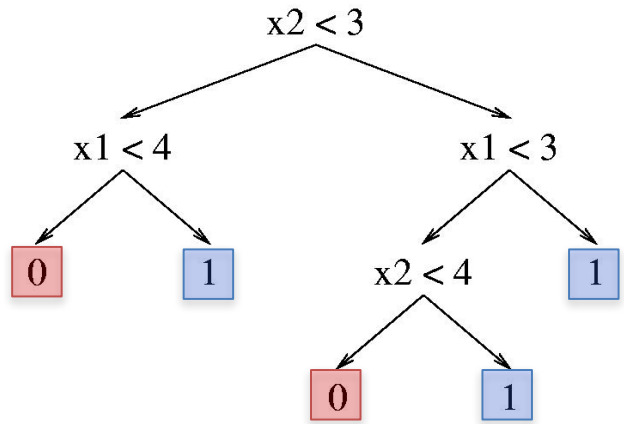
Class label denotes whether a tennis game was played

- Starting from the root, the leaf node at the end of sequence of decisions in tree reveals the result
- What prediction would we make for <outlook=sunny, temperature=hot, humidity=high, wind=weak>?

Decision Tree - Decision Boundary

- Decision trees divide the feature space into axis parallel (hyper-)rectangles
- Each rectangular region is labeled with one label - or a probability distribution over labels

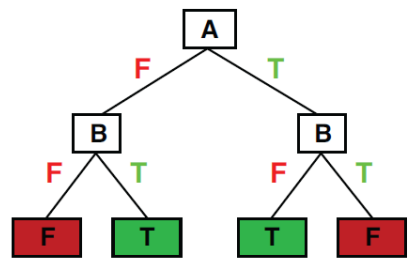




Expressiveness

- Given a particular space of functions, you may not be able to represent everything
- What functions can decision trees represent?
- Decision trees can represent any function of the input attributes!
- Boolean operations (and, or, xor, etc.)?
  - Yes!
- All boolean functions?
  - Yes!

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



(Figure from Stuart Russell)

How to learn/construct the decisions of such a tree?

Basic Algorithm for Top-Down Learning of Decision Trees

Starting from a root node of the decision tree (ID3 by Quinlan)

1. A $\leftarrow$ the "best" decision attribute for the next node.
2. Assign A as decision attribute for node.
3. For each value of A, create a new descendant of node.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop. Else, recurse over new leaf nodes.

How do we choose which attribute is best?

Choosing the Best Attribute

Key problem: choosing which attribute to split a given set of examples

Some possibilities are:

- Random: Select any attribute at random
- Least-Values: Choose the attribute with the smallest number of possible values
- Most-Values: Choose the attribute with the largest number of possible values
- Max-Gain: Choose the attribute that has the largest expected information gain
  - i.e., attribute that results in smallest expected size of subtrees rooted at its children rooted at its children

Iterative Dichotomiser 3 (ID3) algorithm uses the Max-Gain method of selecting the best attribute.

Information Gain

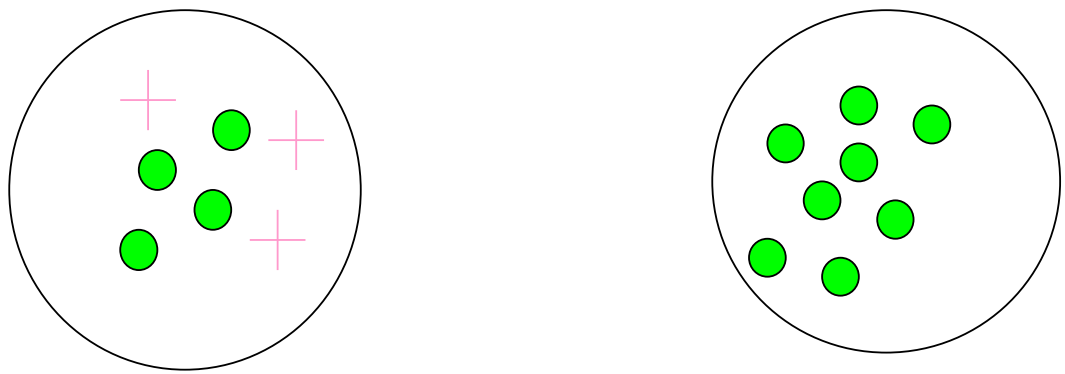
Which test is more informative?

(Figure from Pedro Domingos)

Information Gain

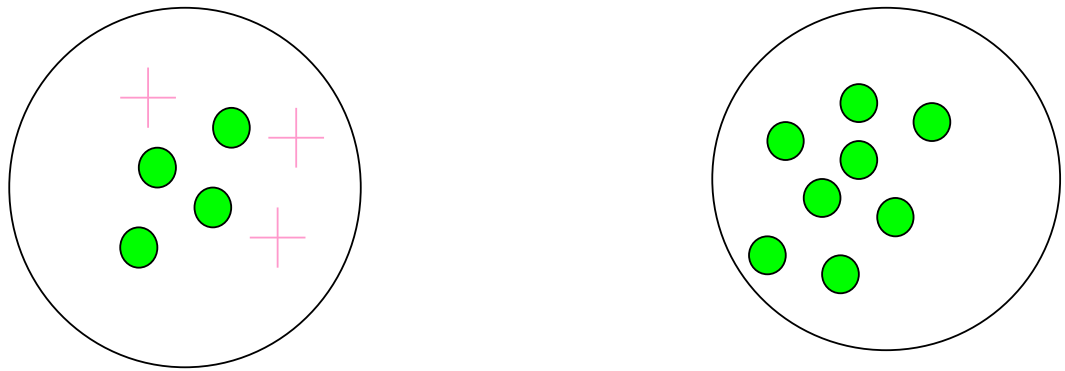
Impurity/Entropy (informal)

– Measures the level of impurity in a group of examples



(Figure from Pedro Domingos)

Entropy: a common way to measure impurity



(Figure from Pedro Domingos)

- Entropy = $\sum_i -p_i \log_2 p_i$
  - p_i is the probability of class i
  - Sample entropy: Compute it with the proportion of class i in a node
- Entropy comes from information theory. The higher the entropy the more the information content.

Metric for Measuring Best Split: Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.
- Information gain = entropy(parent) – [average entropy(children)]
  - Information gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting

Metric for Measuring Best Split: Information Gain

Information gain (informal) = entropy(parent) – [average entropy(children)]

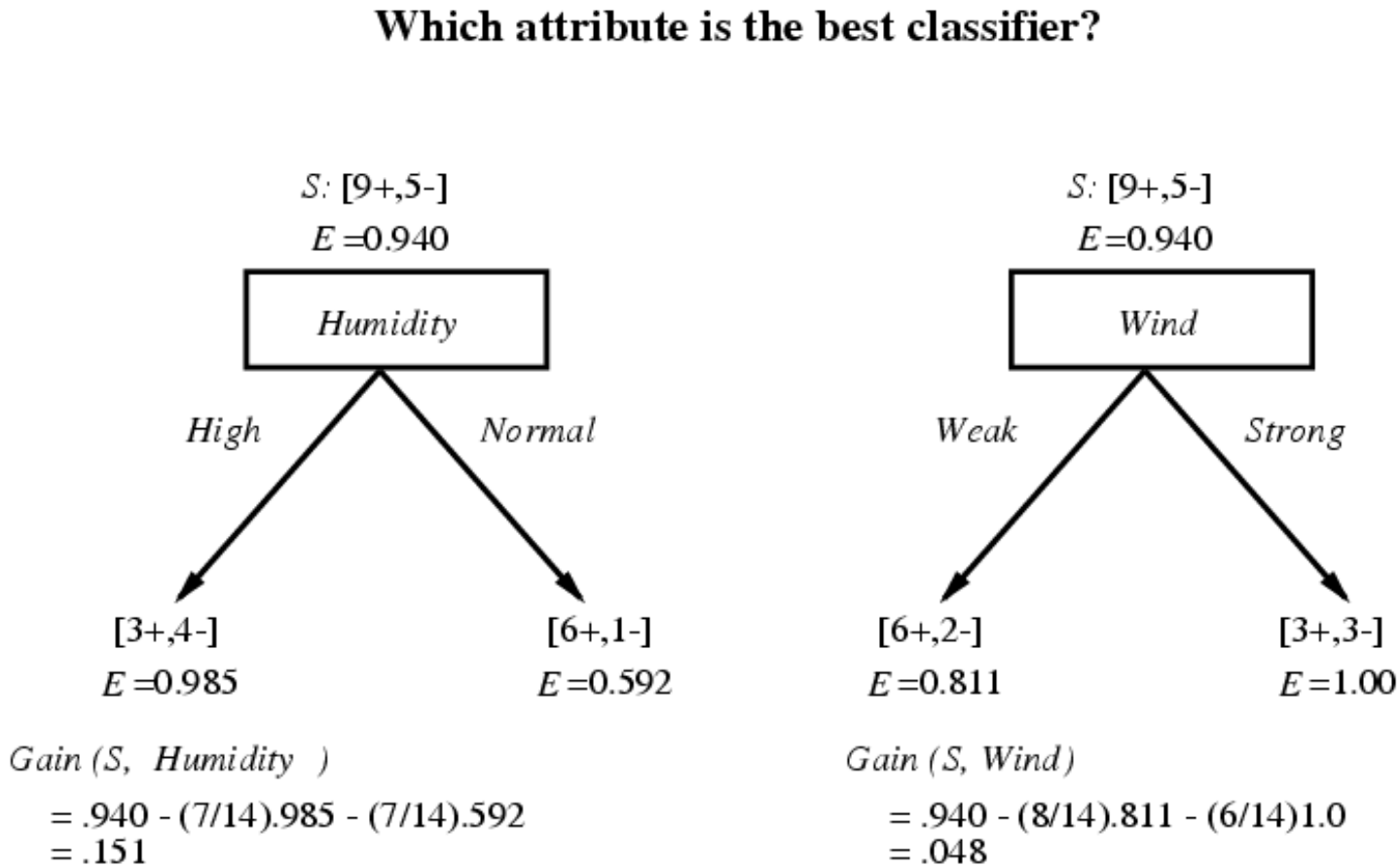
Information gain (IG) of dataset $\mathcal{D}$ at parent node, and dataset $\mathcal{D}_j$ at j-th child based on split function is defined:

$$IG(\mathcal{D}, f) = I(\mathcal{D}) - \sum_{j=1}^{J} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} I(\mathcal{D}_j)$$

where $p(c|\mathcal{D})$ is the portion of class c in $\mathcal{D}$, and candidates of $I(\mathcal{D})$ is

$$I(\mathcal{D}) = -\sum_{c=1}^{C} p(c|\mathcal{D}) \log p(c|\mathcal{D})$$

Example



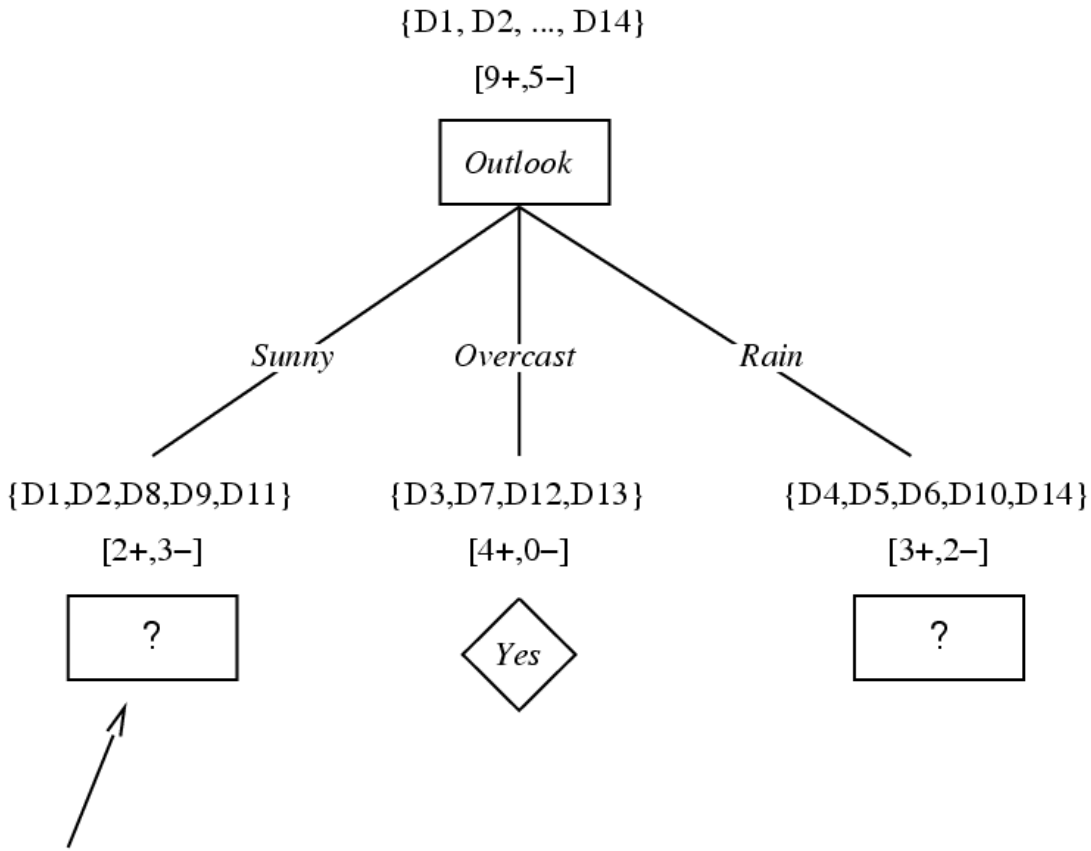
(Figure from Tom Mitchell)

Example: Training Set

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

(Figure from Tom Mitchell)

Example: Test Attribute



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Slide by Tom Mitchell

Avoiding Overfitting in Decision Trees

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Acquire more training data
- Remove irrelevant attributes (manual process - not always possible)
- Grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure (heuristic: simpler is better)

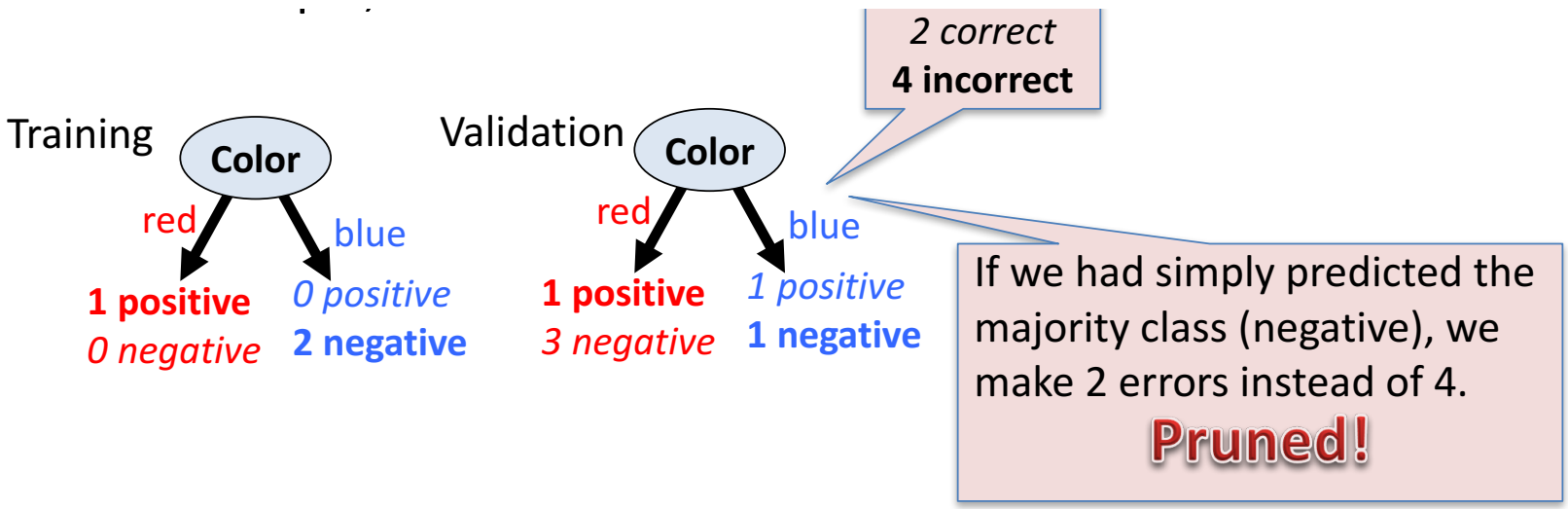
Reduced-Error Pruning

Split training data further into *training* and *validation* sets

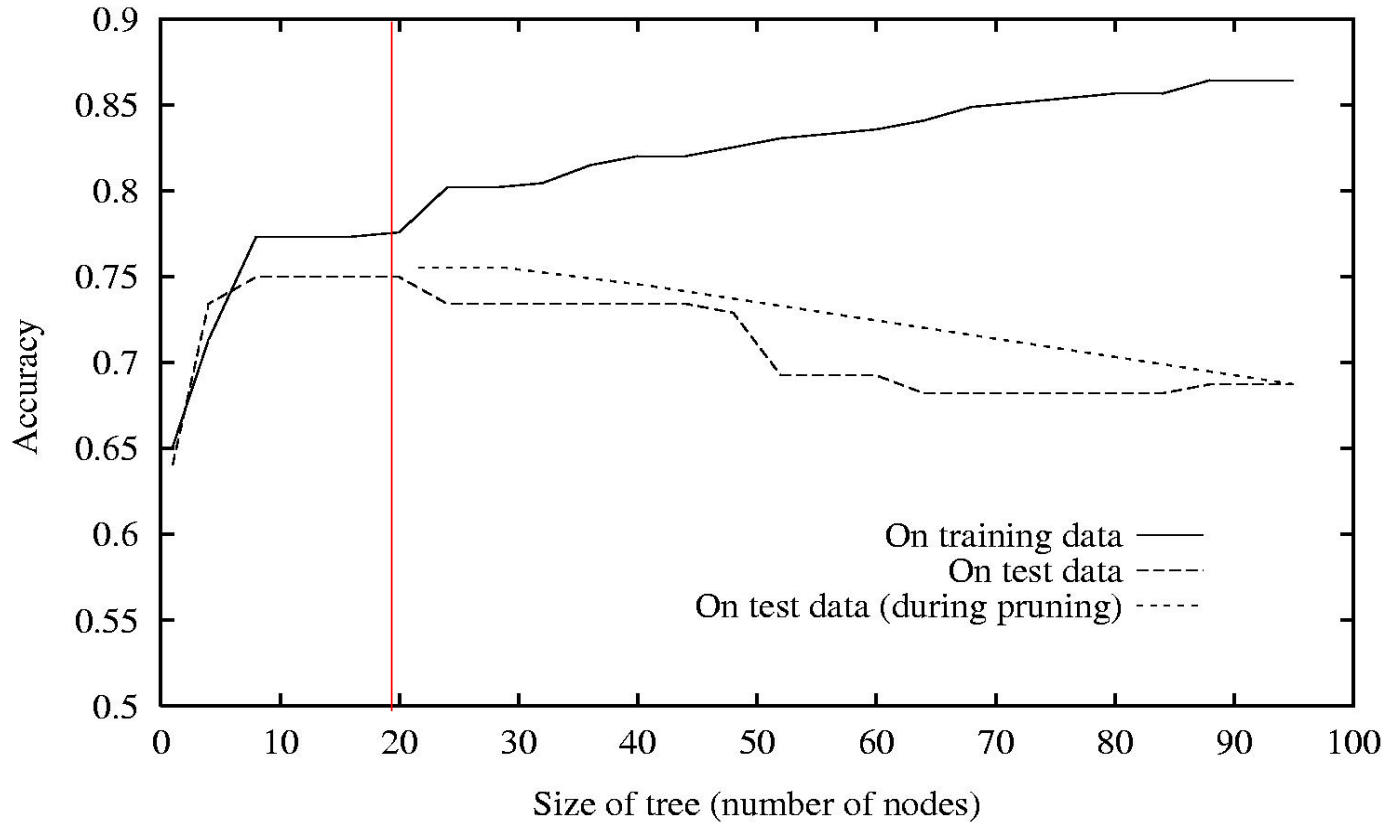
Grow tree based on training set

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the node that most improves validation set accuracy



Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

Summary - Decision Tree

- Widely used in practice
- Strengths include
  - Fast and simple to implement
  - Can convert to rules
  - Handles noisy data
- Weaknesses include
  - Univariate splits/partitioning using only one attribute at a time — limits types of possible trees
  - Large decision trees may be hard to understand
  - Requires fixed-length feature vectors
  - Non-incremental (i.e., batch method)

Adaptive Boosting (AdaBoost) [Schapire 90]

Given dataset $\mathcal{D} = (x^{(i)}, y^{(i)})_{i=1,2,\ldots,N}$ where $x^{(i)} \in \mathcal{X}$ and $y^{(i)} \in \mathcal{Y} = \{-1, 1\}$

- Consider a set $\mathcal{F}$ of weak classifier $f_t : \mathcal{X} \mapsto \mathcal{Y}$ such that

$$f_t(x) = \begin{cases} -1 & \text{if } x < \theta_t \\ 1 & \text{otherwise} \end{cases}$$

  which is called *decision stump*, c.f., one can consider other weak classifiers as well

- Output: $\text{sign}(F_T(x))$ where

$$F_T(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

- How to build F_T? (find θ_ts?)

AdaBoost: An Iterative Method

Note that

$$F_t(x) = F_{t-1}(x) + \alpha_t f_t(x)$$

AdaBoost provides an iterative method to select $f_t \in \mathcal{F}$ and α_t given F_{t-1}

Then how can we choose f_t and α_t?

Let's define the error of F_t to be

$$E(F_t) = \sum_i \exp\left(-y^{(i)} F_t(x^{(i)})\right)$$

Let's also define factors $\gamma_t^{(i)}$:

$$\gamma_1^{(i)} = 1 \quad \forall i; \qquad \gamma_t^{(i)} = \exp\left(-y^{(i)} F_{t-1}(x^{(i)})\right)$$

Consequently,

$$\begin{aligned} E(F_t) &= \sum_i \gamma_t^{(i)} \exp\left(-y^{(i)} \alpha_t f_t(x^{(i)})\right) = \sum_{i: y^{(i)} = f_t(x^{(i)})} \gamma_t^{(i)} e^{-\alpha_t} + \sum_{i: y^{(i)} \neq f_t(x^{(i)})} \gamma_t^{(i)} e^{\alpha_t} \\ &= \sum_i \gamma_t^{(i)} e^{-\alpha_t} + \sum_{i: y^{(i)} \neq f_t(x^{(i)})} \gamma_t^{(i)} (e^{\alpha_t} - e^{-\alpha_t}) \end{aligned}$$

Therefore, pick f_t with lowest weighted error $\sum_{i: y^{(i)} \neq f_t(x^{(i)})} \gamma_t^{(i)}$

$\Rightarrow$ Adaboost wants f_t incorrectly classify an instance that is likely to be classified correctly in previous iterations.

How to pick α_t?

$$\frac{dE(F_t)}{d\alpha_t} = 0$$

Result:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \qquad \text{where } \epsilon_t = \frac{\sum_{i:y^{(i)} \neq f_t(x^{(i)})} \gamma_t^{(i)}}{\sum_i \gamma_t^{(i)}}$$

Algorithm: AdaBoost for binary classification

- Initialize $\gamma_1^{(i)} = 1$ for all $i = 1, .., N$
- Iterate for $t = 1, ..., T$
  - Pick $f_t \in \mathcal{F}$ which minimizes weighted error $\sum_{i:y^{(i)} \neq f_t(x^{(i)})} \gamma_t^{(i)}$.
  - Calculate error rate ϵ_t and weight α_t
  - Improve F_{t-1} to $F_t = F_{t-1} + \alpha_t f_t$
  - Update $\gamma_{t+1}^{(i)} = \exp(-y^{(i)} F_t(x^{(i)}))$

General Observations on AdaBoost

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \qquad \text{where } \epsilon_t = \frac{\sum_{i:y^{(i)} \neq f_t(x^{(i)})} \gamma_t^{(i)}}{\sum_i \gamma_t^{(i)}}$$

- If $\epsilon_t > 1/2$ then $\alpha_t < 0$, which is appropriate since switching the sign on the result of the corresponding hypothesis would give error rate less than $1/2$.
- If $\epsilon_t = 0$ then $\alpha_t = \infty$, which corresponds to the result that the ensemble should consist of that single hypothesis alone.
- If $\epsilon_t = 1/2$ then $\alpha_t = 0$, which means
  - the corresponding hypothesis contributes nothing to the ensemble.
  - the data weights will remain unchanged in every subsequent round.
- When $\epsilon_t = 1/2$ occurs we conclude that no ensemble of weak hypotheses from the given class can fit this training data.

Summary

1. Fair comparison between models
   - Training and Test Sets
   - Validation Sets
   - Cross Validation

2. Combining simple models: AdaBoost, decision tree learning
   - Decision Trees
   - AdaBoost