ENCS3340 - Artificial Intelligence

Learning from Observations
Part 2

Supervised Learning

2 - Learning Decision Trees

Learning decision trees

Problem: **To Wait or not to Wait**: decide whether to wait for a table at a restaurant, based on the following attributes:

1. **Alternate:** is there an alternative restaurant nearby?
2. **Bar:** is there a comfortable bar area to wait in?
3. **Fri/Sat:** is today Friday or Saturday?
4. **Hungry:** are we hungry?
5. **Patrons:** number of people in the restaurant (None, Some, Full)
6. **Price:** price range ($, $$, $$$)
7. **Raining:** is it raining outside?
8. **Reservation:** have we made a reservation?
9. **Type:** kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate:** estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based Representations

- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table: T wait, F don't wait

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

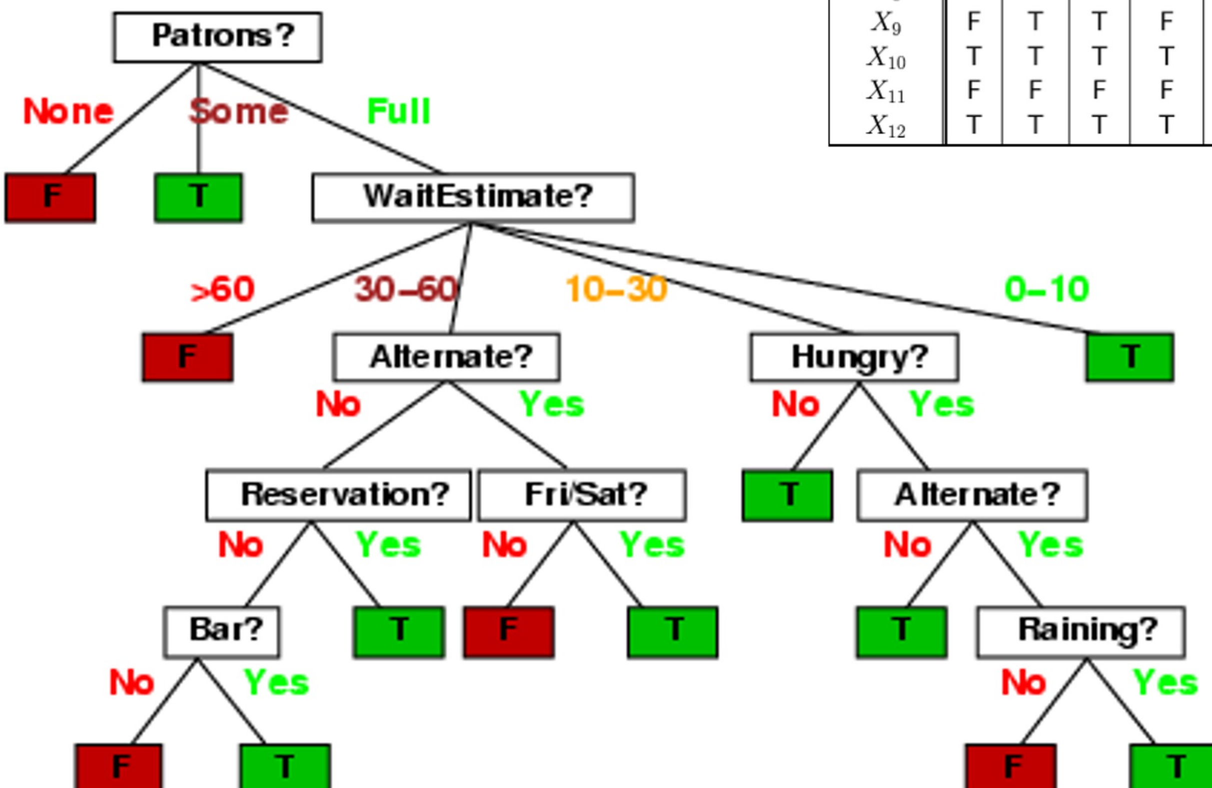
- We are learning Attribute **Wait**
- Classification of examples on **Wait** is positive (T) or negative (F)

Supervised Learning: Decision Trees

- **DT**: One possible representation for hypotheses
- E.g., here is the "true" tree

for deciding whether to wait:

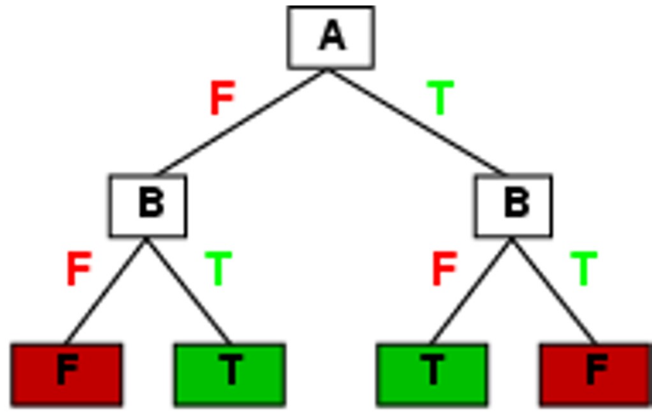
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T



Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

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



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples
- **Generally, DT is not unique for a set of data**
- Prefer to find more compact decision trees

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions (Value: True or False, 1 or 0)

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

- If variables are non_boolean: say 10 possibilities each: with n=6 such attributes, number of distinct values 10^n :number of distinct truth tables with 10^n rows = 10^{10^n}

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., *Hungry* $\wedge \neg$*Rain*)?

- Each attribute can be **in** (positive), **in** (negative), or **out**

  $\Rightarrow 3^n$ distinct conjunctive hypotheses for n attributes
- More expressive hypothesis space
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent with training set

    $\Rightarrow$ may get worse predictions

Decision tree learning

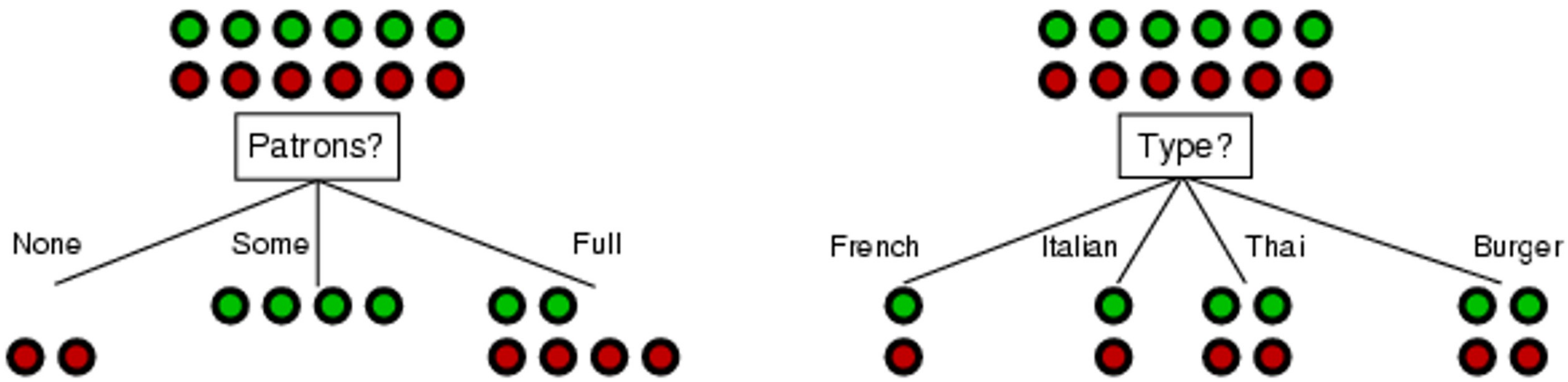
- Aim: find a **small** tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

- Which attribute to choose? **Most discriminating**?

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative“ for its values



- *Patrons?* is a better choice, Why?
- If we take Patrons: =none: **Red**, Some: **Green**, Full: **Red** (Why Red?). Errors: 2 out of 12=1/6 (on value=full).
- If we take Type: =French: **Red**, Italian: **Green**, Thai: **Red** , Burger: **Red** Errors: 6 out of 12=1/2 (on all values).

Using information theory

- To implement `Choose-Attribute` in the DTL algorithm
- Information Content (Entropy):

$$I(P(v_1), \dots , P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

- For a training set containing *p* positive examples and *n* negative examples:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Here: p=6, n=6; p/(n+p)=1/2; n/(n+p)=1/2; I(1/2,1/2)=1/2+1/2=1bit

If p=3, n=9; p/(n+p)=1/4; n/(n+p)=3/4; I(1/4,3/4)=1/4*2+3/4*.4=0.8bit

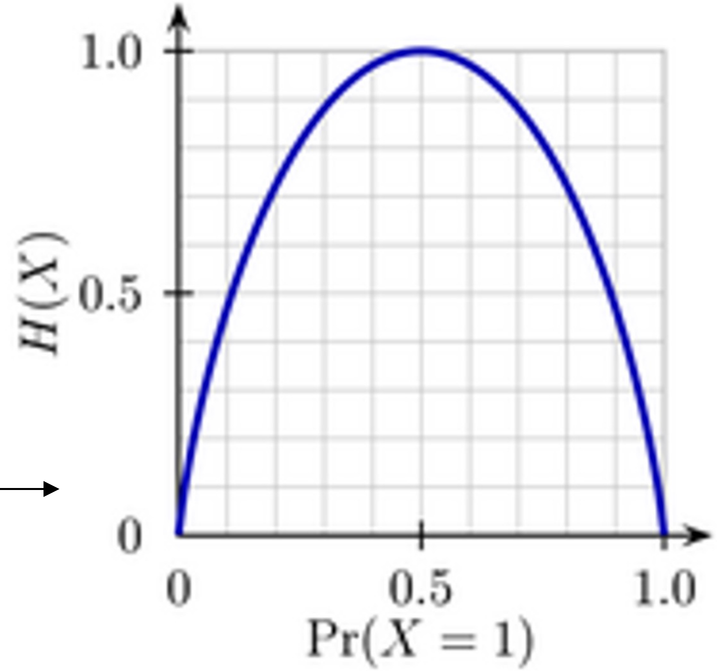
Using information theory

- Entropy measures the amount of uncertainty in a probability distribution:

Consider tossing a biased coin.
If you toss the coin VERY often,
the frequency of heads is, say, p,
and hence the frequency of tails is
1-p. (fair coin p=0.5).

The uncertainty in any actual outcome
is given by the entropy:

Note, the uncertainty is zero if p=0 or 1
and maximal if we have p=0.5.



Information gain

- A chosen attribute *A* divides the training set *E* into subsets $E_1, \ldots, E_v$ according to A values, where *A* has *v* distinct values.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- Defines how discriminating this Attribute is
- Information Gain (IG)/reduction in entropy from the attribute test:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- IG: entropy of the parent – weighted sum of entropy of children
- Defines difference (improvement) in discrimination between the **Learned attribute** and the **tested attribute**.
- Choose the attribute with the largest IG

A	B	C	D	E	Concept
				F	T
				T	F
				F	T
				F	T
				T	T
				T	F
				F	T
				F	T

Concept= True: error: 2/8

Entropy=0.8

E: F→True: T→Flase: Error: 1/8

Information gain: example

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=-\frac{p}{p+n}\log_2\frac{p}{p+n}-\frac{n}{p+n}\log_2\frac{n}{p+n}$$

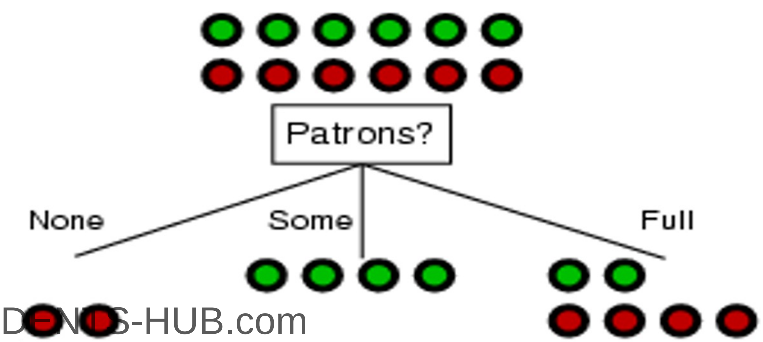
- For the training set, *p = n = 6, I(6/12, 6/12) = 1* bit
- Consider the attributes *Patrons* and *Type* (and others too):

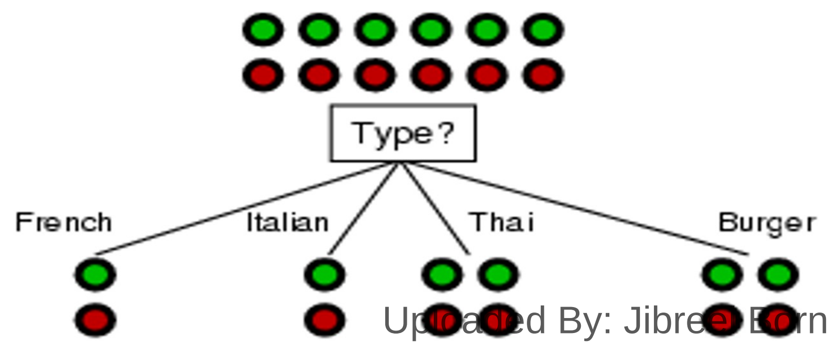
$$remainder(A)=\sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right) \qquad IG(A)=I\left(\frac{p}{p+n},\frac{n}{p+n}\right)-remainder(A)$$

$$IG(Patrons)=1-\left[\frac{2}{12}I(0,1)+\frac{4}{12}I(1,0)+\frac{6}{12}I\left(\frac{2}{6},\frac{4}{6}\right)\right]=.0541\text{ bits}$$

$$IG(Type)=1-\left[\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)+\frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)\right]=0\text{ bits}$$

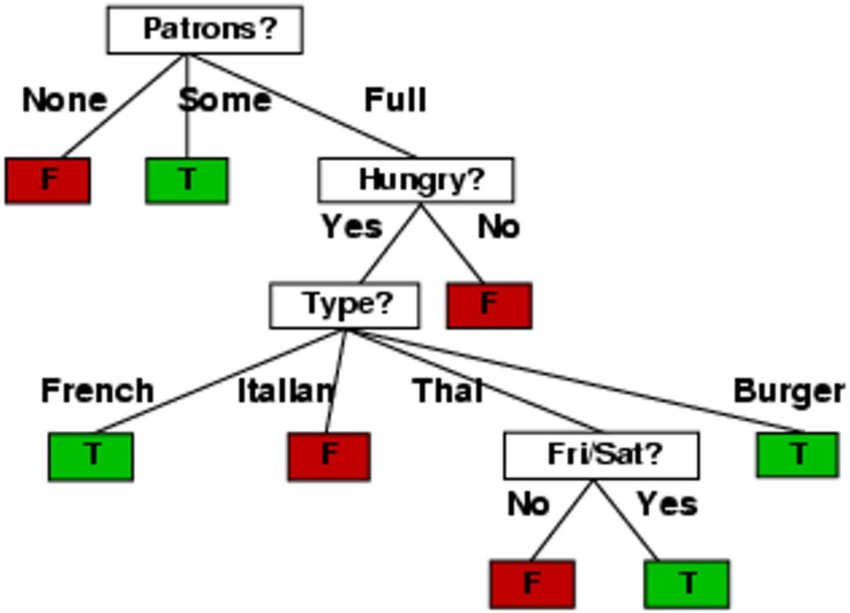
- *Do that for all 10. Assume Patrons* has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

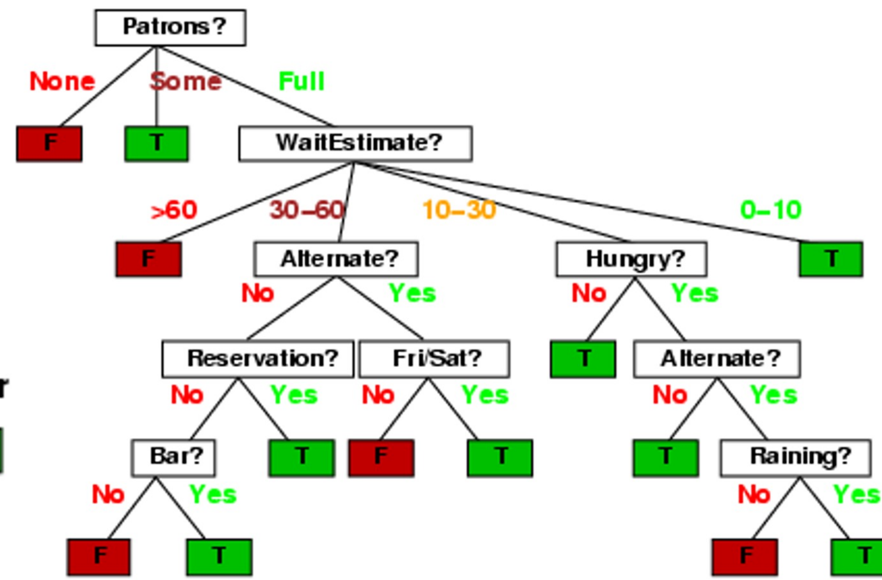




DT Example: contd.

- Decision tree learned from the 12 examples: tested 10 attributes, Started with Patrons, Then tested 9: Hungry,...
- Always Stop when leaves are pure (all T/all F here)





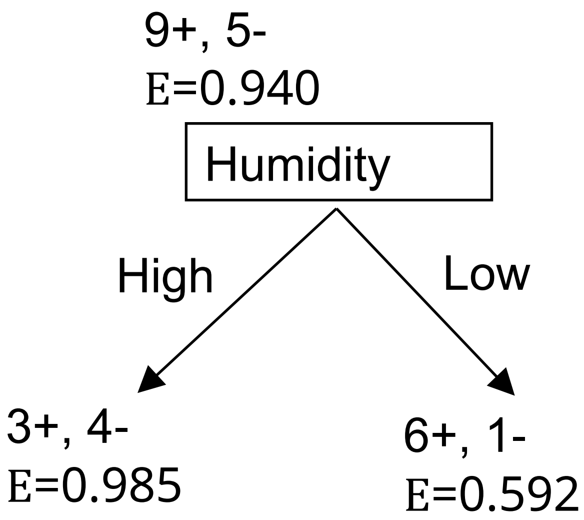
- Substantially simpler than “true” tree. Less leaves also.
- a more complex hypothesis isn’t justified by small amount of data

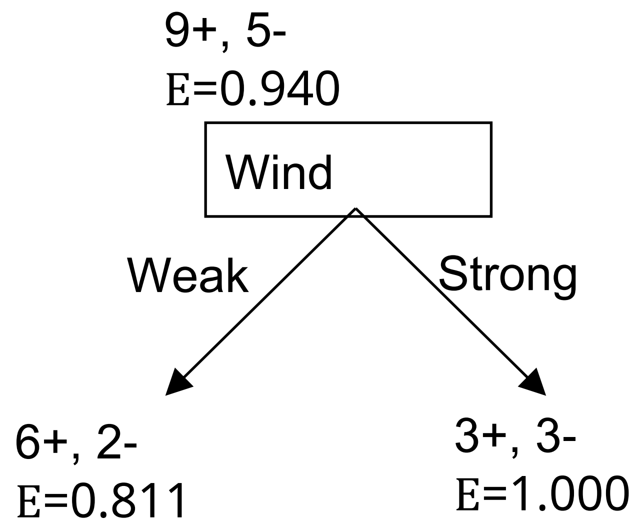
DT Example2: To Play or Not to Play

- 14 examples, 4 attributes, Yes/No Concept

Day	Outlook	Temp	Humidity	Wind	**Tennis ?**
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*

Determine the Root Attribute





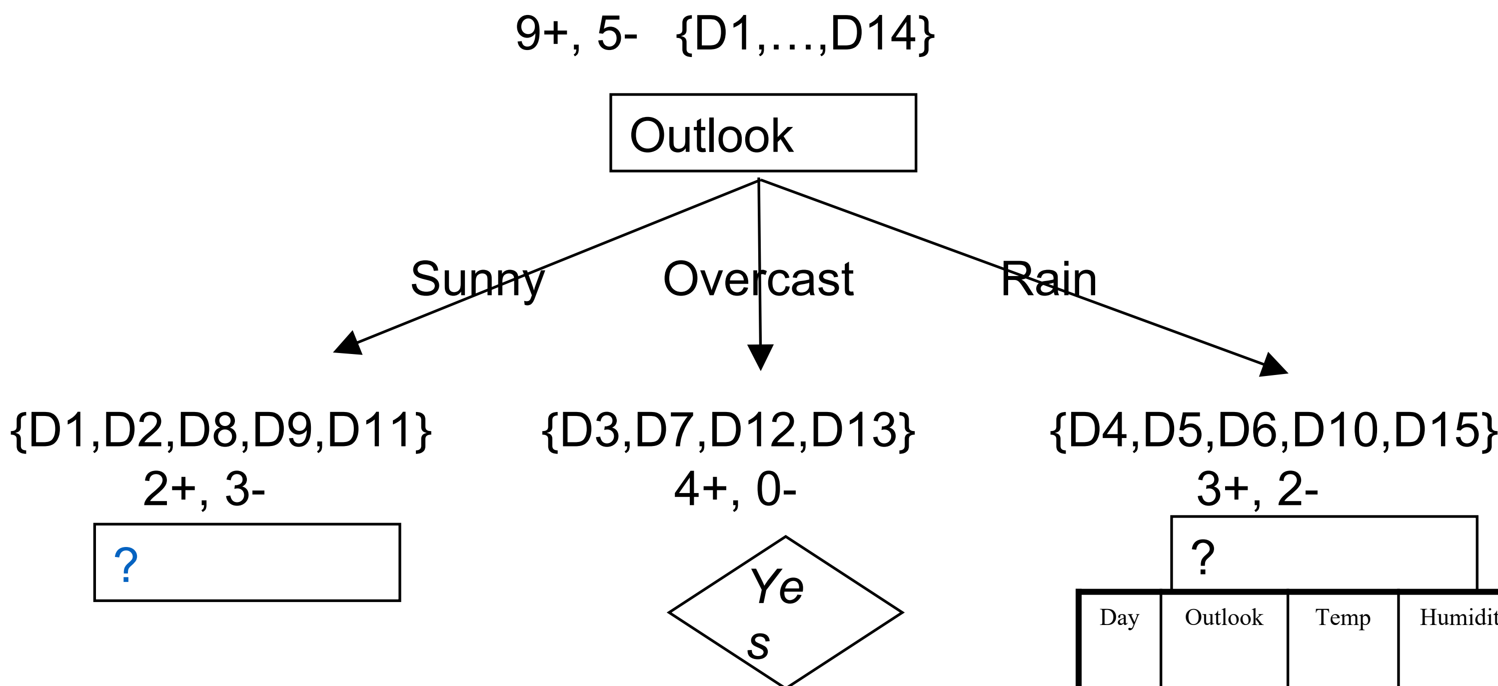
Gain (S, Humidity) = 0.151

Gain (S, Wind) = 0.048

Gain (S, Outlook) = 0.246

Gain (S, Temp) = 0.029

Sort the Training Examples



Day	Outlook	Temp	Humidity	Wind	Tennis?
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*

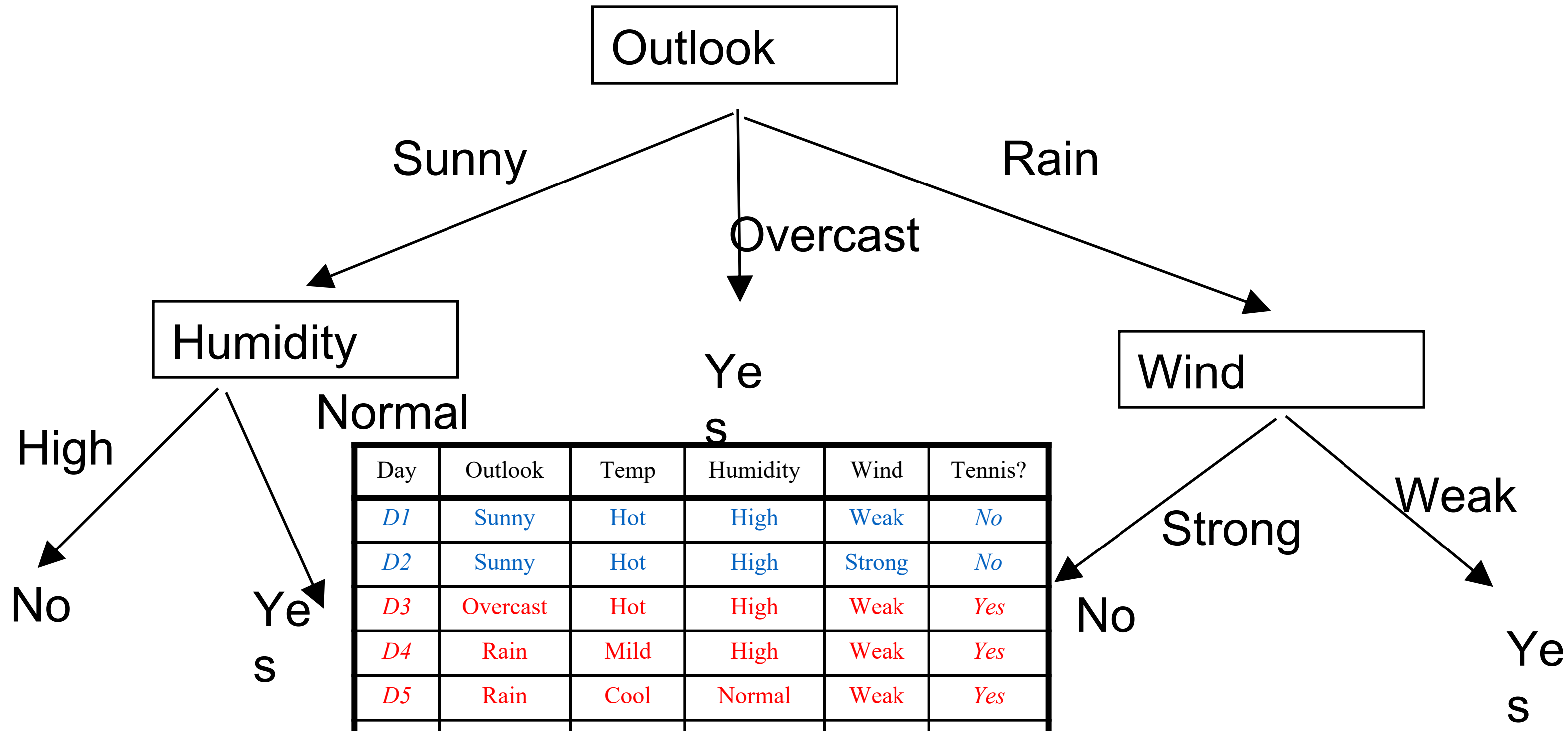
S_{sunny}= {D1,D2,D8,D9,D11}

Gain (S_{sunny}, Humidity) = .970

Gain (S_{sunny}, Temp) = .570

Gain (S_{sunny}, Wind) = .019

Final Decision Tree for Example



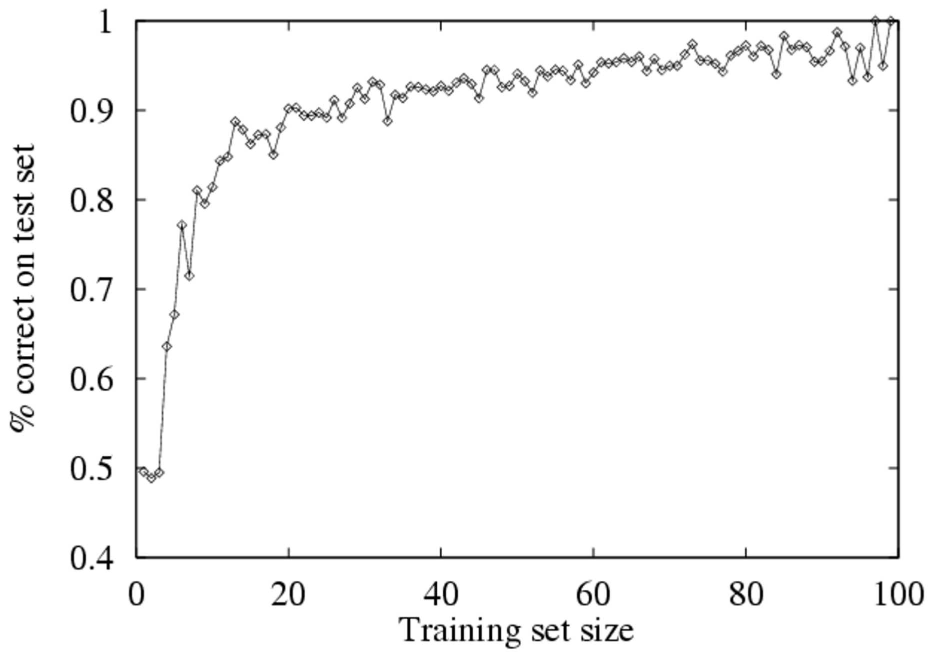
Day	Outlook	Temp	Humidity	Wind	Tennis?
D1	Sunny	Hot	High	Weak	*No*
D2	Sunny	Hot	High	Strong	*No*
D3	Overcast	Hot	High	Weak	*Yes*
D4	Rain	Mild	High	Weak	*Yes*
D5	Rain	Cool	Normal	Weak	*Yes*
D6	Rain	Cool	Normal	Strong	*No*
D7	Overcast	Cool	Normal	Strong	*Yes*
D8	Sunny	Mild	High	Weak	*No*
D9	Sunny	Cool	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*

Performance measurement

- How do we know that $h \approx f$?
  - Use theorems of computational/statistical learning theory
  - Try h on a new test set of examples

    (use same distribution over example space as training set)

Learning curve = % correct on test set as a function of training set size



Evaluation methods

- Holdout set: the available dataset *D* is divided into two disjoint subsets,
  - the *training set* D_{train} (for learning a model)
  - the *testing set* D_{test} (for testing the model)
- Important: training set should not be used in testing and the test set should not be used in learning.
  - unseen test set provides a unbiased estimate of accuracy.
- The test set is also called the holdout set (the examples in the original dataset *D* are all labeled with classes).
- This method is mainly used when the dataset *D* is large

Given 120 examples: Holdout: 25%:75% (30:90) or 50%:50% (60:60) or **34%:66% (40:80) usual** **[one test, random]**

Evaluation methods (Cont.)

- n-fold cross-validation: the evaluation data is partitioned into n equal-size disjoint subsets.
- Use each subset as the test set and combine the rest *n-1* subsets as the training set to learn a classifier.
- The procedure is run *n* times, which give *n* accuracies.
- The final estimated accuracy of learning is the average of the *n* accuracies.
- *10-fold* and *5-fold* cross-validations are commonly used.
- This method is used when the available data is not large.

Consider our 12 example: 6 fold: **6x2**:

S1: Test {1,2}, Training{3,4,…12}, S2: Test {3,4}, Training{1,2,5,6,…12}, S3: Test {5,6}, Training{1-4,7,8-12}, S4: Test {7,8}, Training{1-6,9-12},

S5: Test {9,10}, Training{1-8,11,12}, S6: Test {11,12}, Training{1-10}.

- Each can have: 100% success, 0% success, 50% success: average=?
- Order can vary! 6 runs, 12 tests, 8 correct: success=8/12

Evaluation methods (Cont.)

- Leave-one-out cross-validation: This method is used when the dataset is very small.
- It is a special case of cross-validation.
- Each fold of the cross validation has only a single test example and all the rest data is used in training.
- If the original data has m examples, this is m-fold cross-validation.

Given total of 12 examples: **[12 tests, 12 runs]: each success or fail: 8 fails: accuracy: 4/12**

Evaluation methods (Cont.)

- Validation set: the available data is divided into three subsets,
  - a training set,
  - validation set and
  - a test set
- A validation set is used frequently for estimating parameters in learning algorithms.
- In such cases, the values that give the best accuracy on the validation set are used as the final parameter values.
- cross-validation can be used for parameter estimating as well

Evaluation: Classification measures

- Accuracy is only one measure (error = 1-accuracy).
- Accuracy is not suitable in some applications.
- In text mining, we may only be interested in the documents of a particular topic, which are only a small portion of a big document collection.
- In classification involving skewed or highly imbalance data, e.g., network intrusion and financial fraud detection, we are interested only in the minority class.
  - High accuracy does not mean any intrusion is detected.
  - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the positive class, and the rest negative classes.

Evaluation: Classification measures (Cont.)

- Accuracy is about correct answers.
- Go with the majority class to get it high: very high if negative class dominates: Web Search
- 1000K items, only 1K positive: All negative! **99.9%**

Evaluation: Beyond Accuracy

- So need better measure: we classify to Positive (our class) and Negative (the rest):
- Some of the declared positive are really positive (TP√) and some are not (FP)
- Some of the declared negative are really negative (TN√) and some are not (FN)
- Really (actual) positive= TP√ + FN
- Really (actual) negative= TN√ + FP

Evaluation

- Database with 10 localities: 6 cities and 4 villages:
  Jaffa, Ramallah, Ram, Jerusalem, Hebron,
  Abu Qash, Birzeit, Surda, Jifna, Gaza
- Query: List **villages** in Palestine: RED: Actual Positive, Green?
- **Answer (Positive):** Abu Qash, Birzeit, Surda, Jifna, Gaza
  TP FP TP TP FP
- **Not Village (negative) by default: implicit not listed!:**
  Jaffa, Ramallah, Ram, Jerusalem, Hebron,
  TN TN FN TN TN
- How many errors made? 3/10
- If we always return all as cities: error =4/10
- If we always return all as Villages: error =6/10 (majority rule)
- What if: 1000 localities, all villages but only 10 cities!

Evaluation: **Precision** and **recall** measures

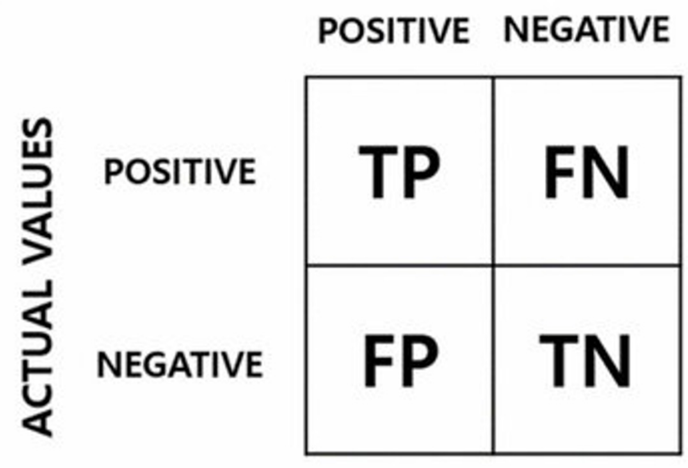
- Used in information retrieval and text classification.
- We use a confusion matrix to introduce them.

Where,

- ➔ TP: **true positive**, the number of correct classifications of the positive examples
- ➔ FN: **false negative**, the number of incorrect classifications of the positive examples
- ➔ FP: **false positive**, the number of incorrect classifications of the negative examples
- ➔ TN: **true negative**, the number of correct classifications of the negative examples

		Predicted Class P	Predicted Class N
Actual Class	P	True Positives (TP)	False Negatives (FN)
Actual Class	N	False Positives (FP)	True Negatives (TN)

Evaluation: **precision** and **recall** measures (Cont.)



$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

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

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Precision *p* is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.

Recall *r* is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

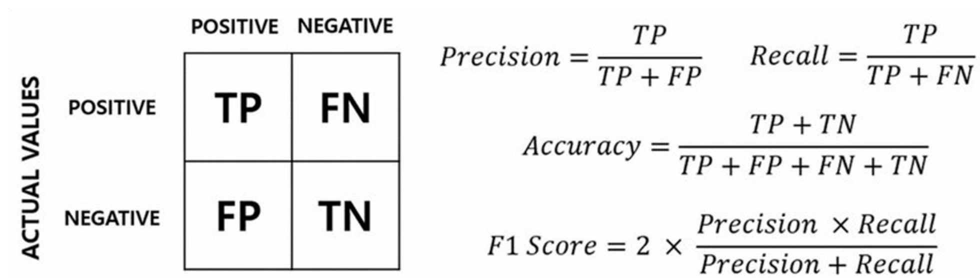
Accuracy *Acc* is the number of correctly classified **positive and negative** examples divided by the total number of examples in the test set.

Evaluation: An example

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

- This confusion matrix gives
  - precision p = 100%
  - recall r = 1%

  because we only classified one positive example correctly and no negative examples wrongly.



- Note: precision and recall only measure classification on the positive class.
- Accuracy= (TP+TN)/(TP+FP+TN+FN)=(1+1000)/1100=0.90

Evaluation: An example (Cont.)

- Can have high accuracy and recall separately easily:
  - Declare all positive: 100% r, low p
  - Declare all negative: 0 recall, High p

  Need a composite measure:

  F measure: 1/F=1/2(1/p+1/r) =1/2((p+r)/p*r) =(p+r)/2p*r **F=2*p*r/(p+r)**

Evaluation: **F1**-measure

- F1-value (also called F1-score)
- It is hard to compare two classifiers using two measures. F1-score combines precision and recall into one measure

$$\textbf{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-score is the harmonic mean of precision and recall

$$\textbf{F1 Score} = \frac{2}{\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)}$$

- The harmonic mean of two numbers tend to be close to the smaller of the two
- For F1-value to be large, both p and r must be large

Examples

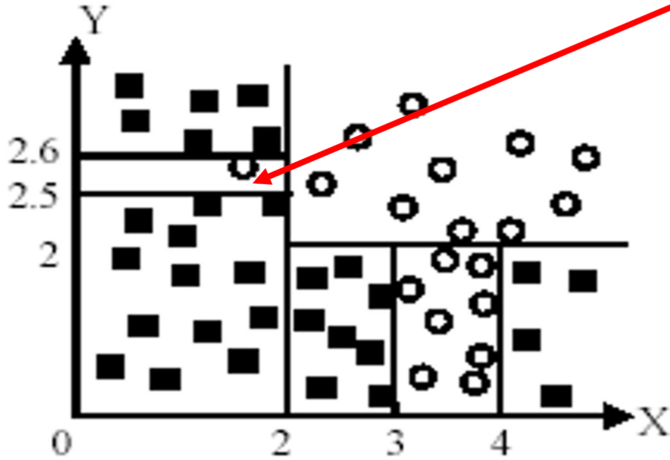
- p=1, r=0, F1=0/1=0
- p=0, r=1, F1=0/1=0
- p=1, r=1, F1=2/2=1
- p=1/2, r=1/2, F1=(1/2)/1=1/2
- p=0.8, r=0.8, F1=1.28/1.6=0.8
- p=0.2, r=0.8, F1=0.32/1=0.32
- p=0.2, r=0.1, F1=.04/0.3=0.13

Avoid overfitting in classification

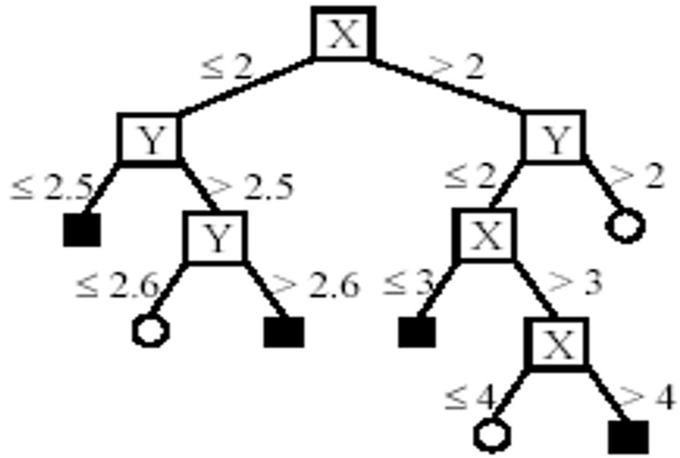
- Ideal goal of classification: Find the simplest decision tree that fits the data and **generalizes** to unseen data
- Overfitting: A tree may overfit the training data
    - **Good** accuracy on training data, poor on test data
    - Symptoms: tree too **deep** and too **many** branches, some may reflect anomalies due to noise or outliers
    - Overfitting results in decision trees that are more complex than necessary
    - Trade-off: full consistency for compactness
        - Larger decision trees can be more consistent
        - Smaller decision trees generalize better

An example

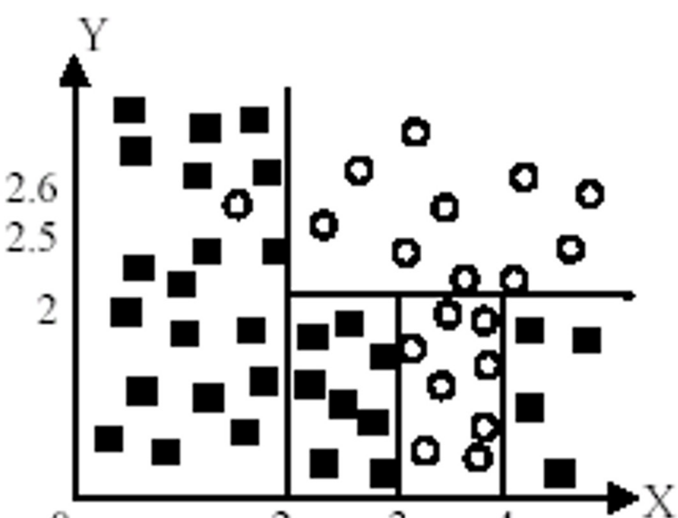
Likely to overfit the data

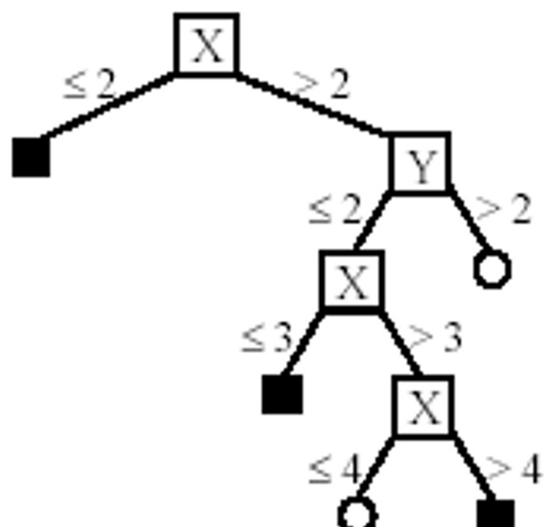


(A) A partition of the data space



(B). The decision tree



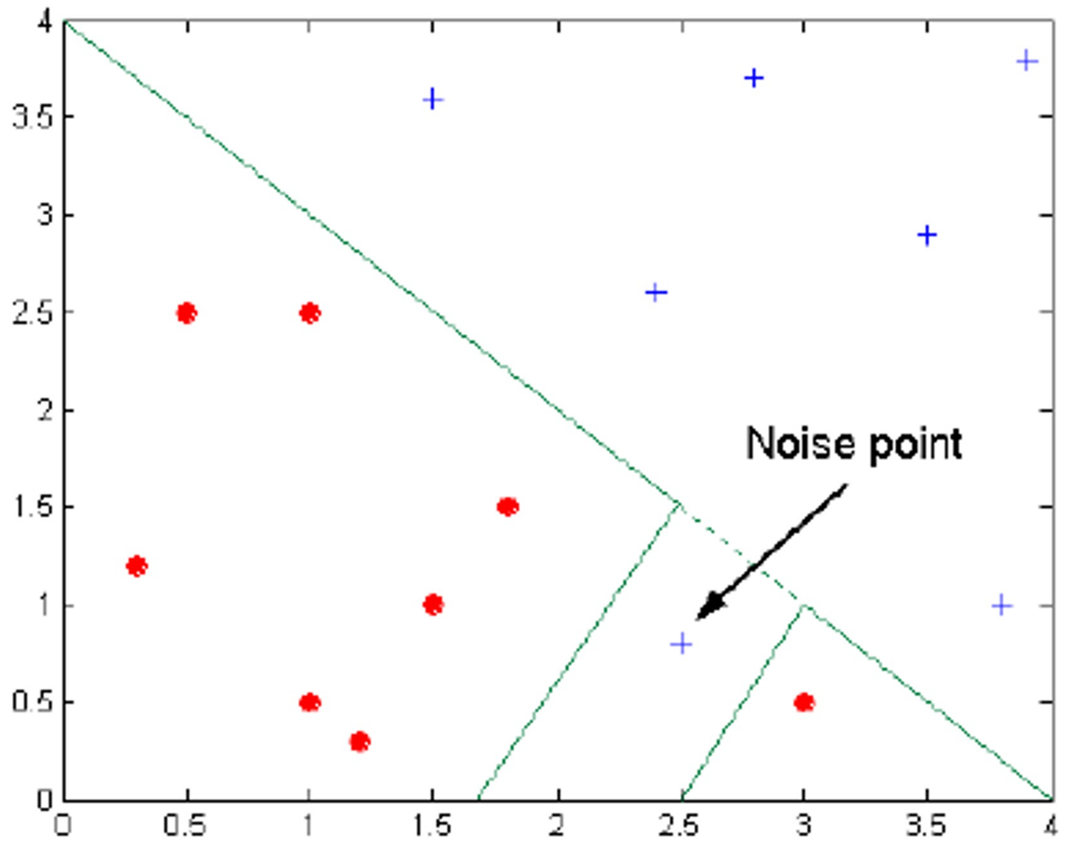


Simple Boolean Example

#	A	B	C	D	F
1	0	0	1	0	**T**
2	0	0	0	1	**T**
3	0	0	1	1	**T**
4	0	0	0	0	**T□ F**
5	1	1	1	1	**F**
6	1	1	0	1	**F**
7	1	1	1	0	**F□T**

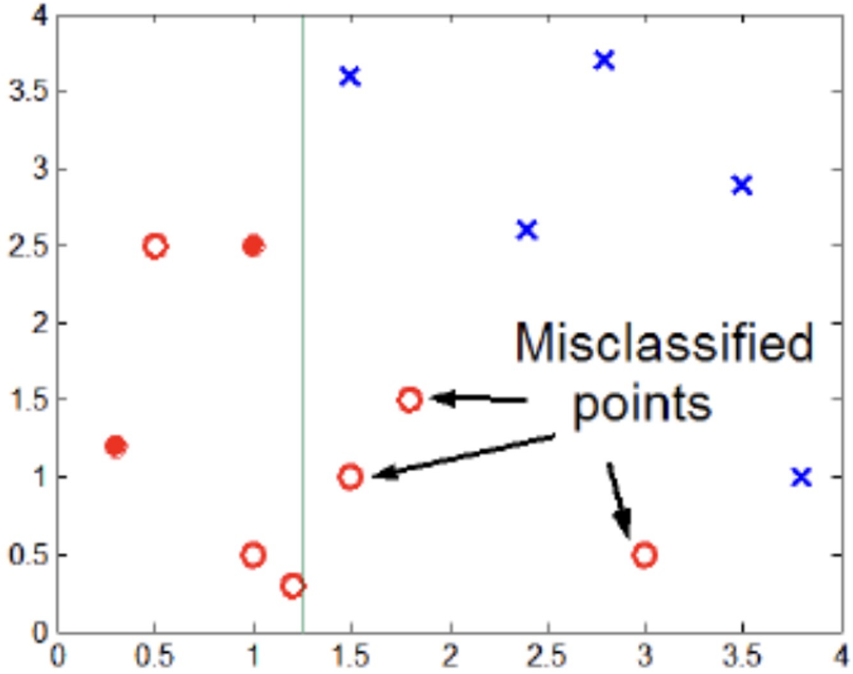
- The function=F=A’
- Change 4 to F: F= A’ and B’
- Change 7 to T: F= ??

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples

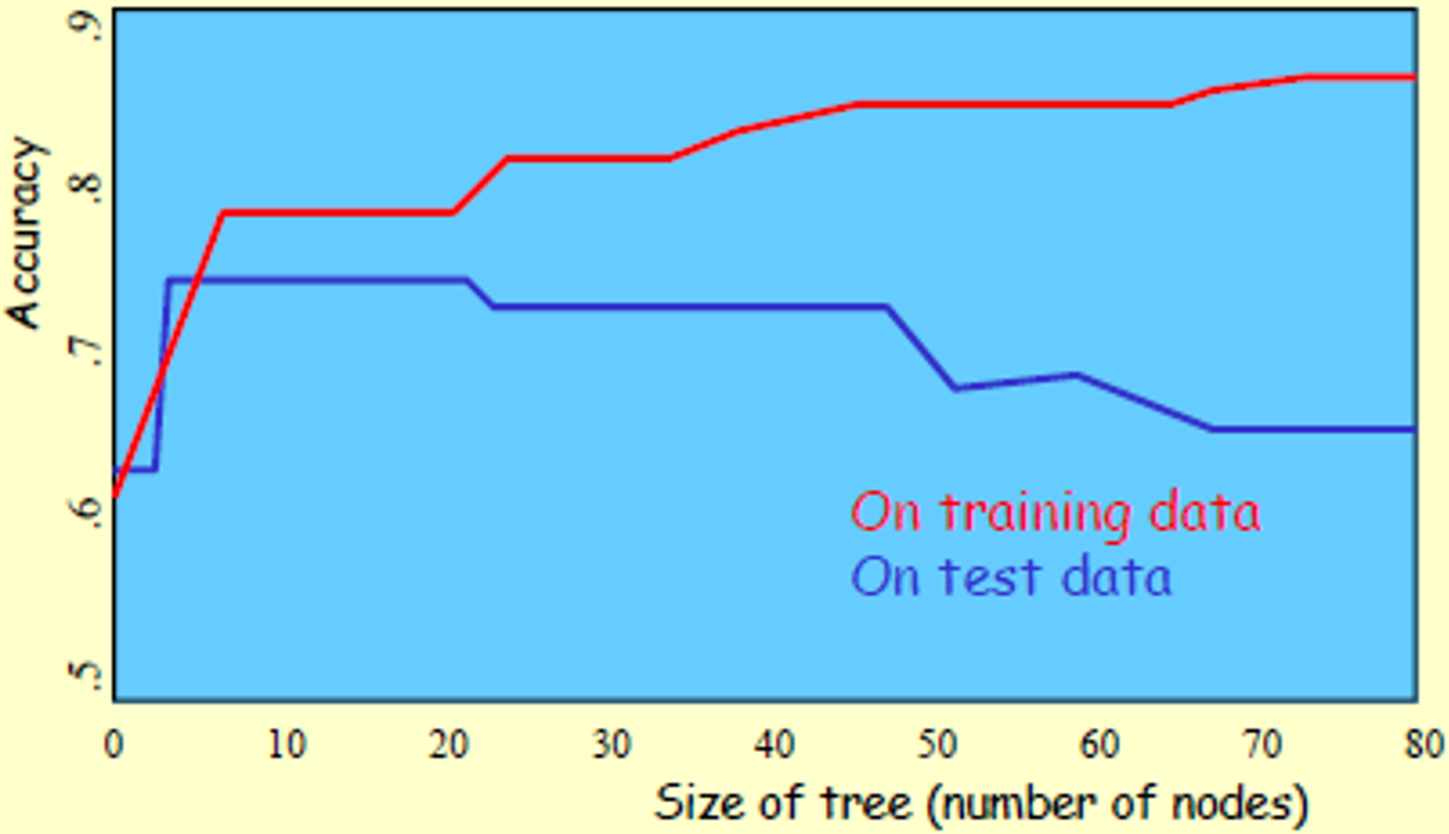


Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Overfitting and accuracy

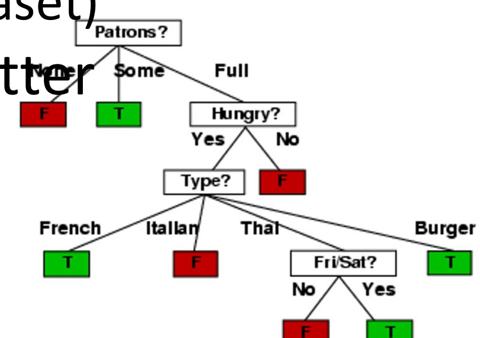
- Typical relation between tree size and accuracy:



Pruning to avoid overfitting

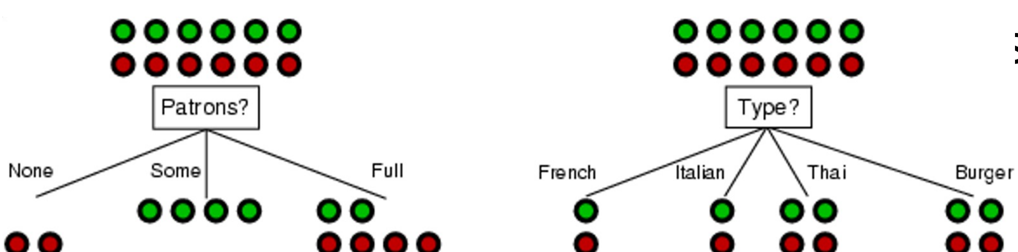
- **Prepruning**: Stop growing the tree when there is not enough data to make reliable decisions or when the examples are acceptably homogenous (ID3)
  - Do not split a node if this would result in the goodness measure falling below a threshold (e.g. InfoGain)
  - Difficult to choose an appropriate threshold
  - Since we use a hill-climbing search, looking only one step ahead, pre-pruning might stop too early.

- **Postpruning**: Grow the full tree, then remove nodes for which there is not sufficient evidence (C4.5)
  - Replace a split (subtree) with a leaf if the *predicted validation error is* no worse than the more complex tree (use dataset)



- Prepruning easier, but postpruning works better

stop



Decision Trees: the good and the bad

- Advantages:
  - Easy to understand (Doctors love them!)
  - Easy to generate rules

- Disadvantages:
  - May suffer from overfitting.
  - Classifies by rectangular partitioning (so does not handle correlated features very well).
  - Can be quite large – pruning is necessary.
  - Does not handle streaming data easily

Supervised Learning

3 - Naïve Bayes Classifier

Thomas Bayes
1702 - 1761

Why Probabilities

Kolmogorov showed that three simple axioms lead to the rules of probability theory

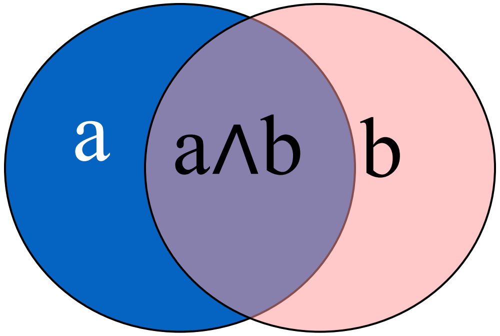
1. All probabilities are between 0 and 1:

   $0 \leq P(a) \leq 1$

1. Valid propositions (tautologies) have probability 1, and unsatisfiable propositions have probability 0:

   $P(true) = 1$; $P(false) = 0$

1. The probability of a disjunction is given by: $P(a \lor b) = P(a) + P(b) - P(a \land b)$



Probability theory

- **Random variables**
  - Domain
- **Atomic event**: complete specification of state
- **Prior probability**: degree of belief without evidence
- **Joint probability**: matrix of combined probabilities of a set of variables

- Alarm, Burglary, Earthquake
  - Boolean, discrete, continuous
- Alarm=T∧Burglary=T∧Earthquake=F
  alarm ∧ burglary ∧ ¬earthquake
- P(Burglary) = 0.1
  P(Alarm) = 0.19
  P(earthquake) = 0.000003
- P(Alarm, Burglary) =

	alarm	¬alarm
burglary	.09	.01
¬burglary	.1	.8

Probability Cont.

	alarm	¬alarm
burglary	.09	.01
¬burglary	.1	.8

- **Conditional probability**: prob. of effect given causes
- **Computing conditional probs**: a given b (evidence)
  - $P(a \mid b) = P(a \wedge b) / P(b)$
  - P(b): **normalizing** constant-known
- **Product rule**:
  - $P(a \wedge b) = P(a \mid b) * P(b)$
  - $P(a \wedge b) = P(b \mid a) * P(a)$
  - $P(a \mid b) * P(b) = P(b \mid a) * P(a)$
  - **$P(a \mid b) = P(b \mid a) * P(a) / P(b)$**

- P(burglary | alarm) = .47
  P(alarm | burglary) = .9
- P(burglary | alarm) =
  P(burglary ∧ alarm) / P(alarm)
  = .09/.19 = .47
- P(burglary ∧ alarm) =
  P(burglary | alarm) * P(alarm)
  = .47 * .19 = .09
- P(alarm) = P(alarm ∧ burglary) + P(alarm ∧ ¬burglary) = .09+.1 = .19
- P(burglary) = P(alarm ∧ burglary) + P(¬ alarm ∧burglary) = .09+.01 = .1

Probability Basics

- **Prior**, **conditional** and **joint** probability for random variables
  - Prior probability: $P(X)$
  - Conditional probability: $P(X_1 | X_2), P(X_2 | X_1)$
  - Joint probability: $\mathbf{X} = (X_1, X_2), P(\mathbf{X}) = P(X_1, X_2)$
  - Relationship: $P(X_1, X_2) = P(X_2 | X_1)P(X_1) = P(X_1 | X_2)P(X_2)$
  - Independence: $P(X_2 | X_1) = P(X_2), P(X_1 | X_2) = P(X_1), P(X_1, X_2) = P(X_1)P(X_2)$

  Bayesian Rule: *Recall*: P(a ∧ b) = P(a | b) * P(b), P(a ∧ b) = P(b | a) * P(a)
- P(a | b) * P(b) = P(b | a) * P(a); P(a | b) = P(b | a) * P(a)/ * P(b); a=C, b=X,

$$P(C | \mathbf{X}) = \frac{P(\mathbf{X} | C)P(C)}{P(\mathbf{X})} \qquad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

- P(Swede|Blonde)=P(Blonde|Swede)*P(Swede)/P(Blonde)

  Hi (relatively) 70% swedes blonde .008 swedes 10% Blonde

  =0.7*0.008/0.1=0.7*0.08=0.056

The Blonde's Passport!

$$P(C/\mathbf{X}) = \frac{P(\mathbf{X}/C)P(C)}{P(\mathbf{X})}$$

$P(a \wedge b) = P(a \mid b) * P(b) = P(b \mid a) * P(a)$ □

P(a|b)=P(b|a)P(a)/p(b) Let C-Class (country), X-Observation-Blonde

You saw a **Blonde** : what is the **nationality**? Choices: SW, S**A**, US

- 10% of world population are **Blonde** **P(Blonde)=0.1**

70% Swedes: **Blonde;** .008 world population are Swedes: P(Swede)=.008

P(Swede| Blonde)=P(Blonde |Swede)*P(Swede)/P(Blonde)

=0.7 *0.008 /0.1= **0.056** **Relatively Hi!**

10% S**A** : **Blonde**, 0.016 world population are S**A**

P(SA|Blonde)=P(Blonde|SA)*P(SA)/P(Blonde)

= 0.1 *0.016/0.1=0.1*0.16=**0.016** **Relatively low**

50% USA: **Blonde**, 0.03 world population are USA

P(USA|Blonde)=P(Blonde|USA)*P(USA)/P(Blonde)

=0.5*0.03/0.1=0.5*0.3=**0.15**

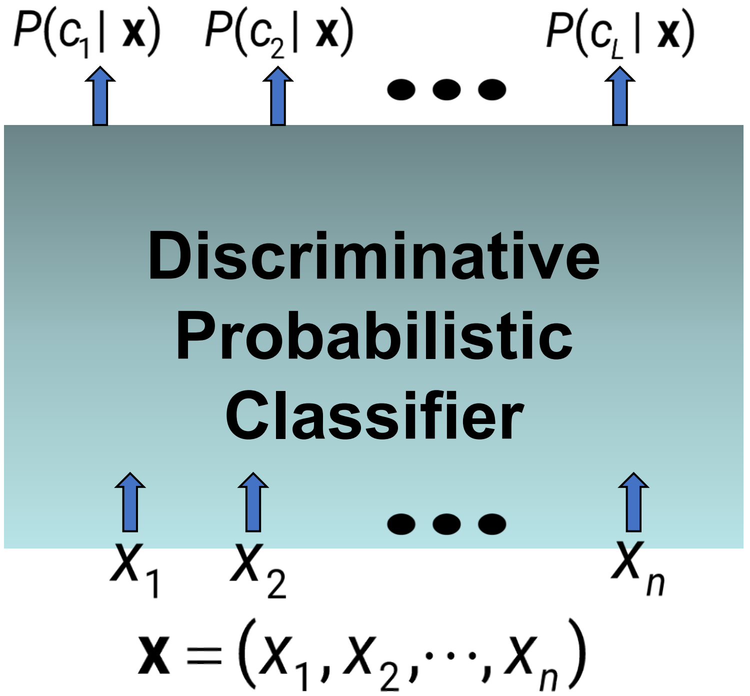
Real Hi

What if Blonde, Height, Language,…? Do we need to divide by 0.1

Probabilistic Classification

- Establishing a probabilistic model for classification
    - **Discriminative model**

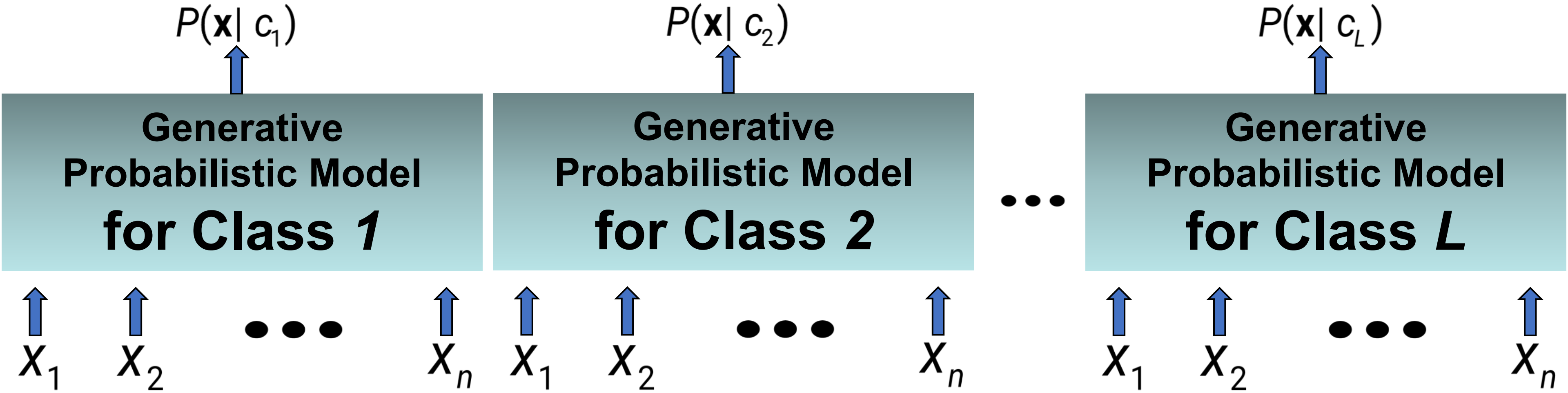
$$\boxed{P(C|\mathbf{X}) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)}$$



Probabilistic Classification

- Establishing a probabilistic model for classification (cont.)
    - **Generative model**

$$\boxed{P(\mathbf{X}|C) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)}$$



$$\mathbf{x} = (x_1, x_2, \cdots, x_n)$$

Probabilistic Classification

- MAP classification rule
  - **MAP**: **M**aximum **A** **P**osteriori
  - Assign example ***x*** to class (category, concept) ***c**** if

$$P(C = c^* | \mathbf{X} = \mathbf{x}) > P(C = c | \mathbf{X} = \mathbf{x}) \quad c \neq c^*,\ c = c_1, \cdots, c_L$$

- Generative classification with the MAP rule
  - Apply Bayesian rule to convert them into posterior probabilities

$$P(C = c_i | \mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x} | C = c_i)P(C = c_i)}{P(\mathbf{X} = \mathbf{x})}$$

$$\propto P(\mathbf{X} = \mathbf{x} | C = c_i)P(C = c_i)$$

$$\text{for } i = 1, 2, \cdots, L$$

  - Then apply the MAP rule

Naïve Bayes

- Bayes classification

$$P(C|\mathbf{X}) \propto P(\mathbf{X}|C)P(C) = P(X_1, \cdots, X_n|C)P(C)$$

Difficulty: learning the joint probability $P(X_1, \cdots, X_n|C)$

- Naïve Bayes classification
  - Assumption that all input features are conditionally **independent**!
  - **Recall:** $P(X_1, X_2) = P(X_2|X_1)P(X_1) = P(X_1|X_2)P(X_2)$

$$\begin{aligned} P(X_1, X_2, \cdots, X_n|C) &= P(X_1|X_2, \cdots, X_n, C)P(X_2, \cdots, X_n|C) \\ &= P(X_1|C)P(X_2, \cdots, X_n|C) \\ &= P(X_1|C)P(X_2|C)\cdots P(X_n|C) \end{aligned}$$

  - MAP classification rule: for

$$[P(x_1|c^*)\cdots P(x_n|c^*)]P(c^*) > [P(x_1|c)\cdots P(x_n|c)]P(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Naïve Bayes

- Algorithm: Discrete-Valued Features
  - Learning Phase: Given a training set **S** of F features and L classes,

For each target value of c_i $(c_i = c_1, \cdots, c_L)$

$\hat{P}(C = c_i) \leftarrow$ estimate $P(C = c_i)$ with examples in $\mathbf{S}$;

For every feature value x_{jk} of each feature X_j $(j = 1, \cdots, F; k = 1, \cdots, N_j)$

$\hat{P}(X_j = x_{jk} | C = c_i) \leftarrow$ estimate $P(X_j = x_{jk} | C = c_i)$ with examples in $\mathbf{S}$;

Output: $F * L$ conditional probabilistic (generative) models

  - Test Phase: Given an unknown instance $\mathbf{X}' = (a'_1, \cdots, a'_n)$

"Look up tables" to assign the label c^* to $\mathbf{X}'$ if

$$[\hat{P}(a'_1 | c^*) \cdots \hat{P}(a'_n | c^*)]\hat{P}(c^*) > [\hat{P}(a'_1 | c) \cdots \hat{P}(a'_n | c)]\hat{P}(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Naïve Bayes

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

We want to find the class (*Class*) given the features (*feature*)

$$P(Class \mid feature)$$

Using Bayes rule

$$P(Class \mid feature) = \frac{P(feature \mid Class) \times P(Class)}{P(feature)}$$

From the table, we will count the number of events for the **class** and each **attribute/class** combination

Feature is a vector!!! Not only **blonde** as earlier, instead:
outlook, humidity, temperature, wind

Note: When selecting examples: we have control over class, not care about individual attribute values!!!

Example

- Example: Play Tennis:
- Our class:
- C1:Play= *yes*
- C2:Play= *No*

- P(*Play=yes*)= 9/14
- P(*Play=No*)= 5/14

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Example (Cont.)

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny*	Hot	High	Weak	No
Sunny*	Hot	High	Strong	No
Overcast**	Hot	High	Weak	Yes
Rain***	Mild	High	Weak	Yes
Rain***	Cool	Normal	Weak	Yes
Rain**	Cool	Normal	Strong	No
Overcast**	Cool	Normal	Strong	Yes
Sunny*	Mild	High	Weak	No
Sunny*	Cool	Normal	Weak	Yes
Rain***	Mild	Normal	Weak	Yes
Sunny*	Mild	Normal	Strong	Yes
Overcast**	Mild	High	Strong	Yes
Overcast**	Hot	Normal	Weak	Yes
Rain**	Mild	High	Strong	No

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Example (Cont.)

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Example (Cont.)

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Example (Cont.)

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

Example (Cont.) - Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

P(Play=*Yes*) = 9/14

P(Play=*No*) = 5/14

P(Wind=Strong/Play=*No*) = 3/5

P(Outlook=Sunny/Play=*Yes*) = 2/9

What if we have 4 classes: C0-C3? [play = yes/no/maybe/NoRec

What if we had 10 attributes: A0-A9? [Add: sick, court_ready, fee,..]

Example (Cont.) - Test Phase

- Given a new instance, predict its label:

  x′=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

  We compute: (Remember: Naiveté: independence):

  P(Play=Yes|Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)=
  P(Play=Yes|Outlook=*Sunny*)*P(Play=Yes|Temperature=*Cool*)*
  P(Play=Yes|Humidity=*High*)*P(Play=Yes|Wind=*Strong*)

P(Play=No|Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)=
P(Play=No|Outlook=*Sunny*)*P(Play=No|Temperature=*Cool*)*
P(Play=No|Humidity=*High*)*P(Play=No|Wind=*Strong*)

$$P(C|\mathbf{X}) \propto P(\mathbf{X}|C)P(C) = P(X_1,\cdots,X_n|C)P(C)$$

P(*Yes*|**x**′) ≈ [**P(**Outlook=***Sunny***|**Play=*Yes*)*P(**Temperature=***Cool***|**Play=*Yes*)***
P(Humidity=***High***|**Play=*Yes*)* P(**Wind=***Strong***|**Play=*Yes*)]*P(Play=*Yes*)**

P(*No*|**x**′) ≈ **[P(*Sunny*|Play=N*o*) *P(*Cool*|Play=N*o*)*P(*High*|Play=*No*)***
P(*Strong*|Play=*No*)]*P(Play=*No*)

Example (Cont.) - Test Phase

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

- Given a new instance, predict its label

  x′=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

- Look up tables achieved in the learning phrase

Temperature	**Play=*Yes***	Play=*No*
Hot	**2/9**	2/5
Mild	**4/9**	2/5
Cool	**3/9**	1/5

P(Outlook=*Sunny*|Play=*Yes*) = 2/9

P(Temperature=*Cool*|Play=*Yes*) = 3/9

P(Huminity=*High*|Play=*Yes*) = 3/9

P(Wind=*Strong*|Play=*Yes*) = 3/9

P(Play=*Yes*) = 9/14

P(Outlook=*Sunny*|Play=*No*) = 3/5

P(Temperature=*Cool*|Play==*No*) = 1/5

P(Huminity=*High*|Play=*No*) = 4/5

P(Wind=*Strong*|Play=*No*) = 3/5

P(Play=***No***) = 5/14

Wind	**Play=*Yes***	Play=*No*
Strong	**3/9**	3/5
Weak	**6/9**	2/5

P(*Yes*|**x**′) ≈ [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|**x**′) ≈ [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = **0.0206**

Given the fact P(*Yes*|**x**′) < P(*No*|**x**′), we label **x**′ to be "*No*".

Advantages/Disadvantages of Naïve Bayes

- Advantages:
  - Fast to train (single scan). Fast to classify
  - Not sensitive to irrelevant features
  - Handles real and discrete data
  - Handles streaming data well
- Disadvantages:
  - Assumes independence of features