Off-The-Shelf Classifiers

- A method that can be applied directly to data without requiring a great deal of time-consuming data preprocessing or careful tuning of the learning procedure

- Let's compare Perceptron, Logistic Regression, and LDA to ask which algorithms can serve as good off-the-shelf classifiers

Off-The-Shelf Criteria

- Natural handling of “mixed” data types
  - continuous, ordered-discrete, unordered-discrete
- Handling of missing values
- Robustness to outliers in input space
- Insensitive to monotone transformations of input features
- Computational scalability for large data sets
- Ability to deal with irrelevant inputs
- Ability to extract linear combinations of features
- Interpretability
- Predictive power

Handling Mixed Data Types with Numerical Classifiers

- Indicator Variables
  - sex: Convert to 0/1 variable
  - county-of-residence: Introduce a 0/1 variable for each county
- Ordered-discrete variables
  - example: {small, medium, large}
  - Treat as unordered
  - Treat as real-valued
    - Sometimes it is possible to measure the “distance” between discrete terms. For example, how often is one value mistaken for another? These distances can then be combined via multi-dimensional scaling to assign real values

Missing Values

- Two basic causes of missing values
  - Missing at random: independent errors cause features to be missing. Examples:
    - clouds prevent satellite from seeing the ground.
    - data transmission (wireless network) is lost from time-to-time
  - Missing for cause:
    - Results of a medical test are missing because physician decided not to perform it.
    - Very large or very small values fail to be recorded
    - Human subjects refuse to answer personal questions

Dealing with Missing Values

- Missing at Random
  - P(**x**,*y*) methods can still learn a model of P(**x**), even when some features are not measured.
  - The EM algorithm can be applied to fill in th emissing features with the most likely values for those features
  - A simpler approach is to replace each missing value by its average value or its most likely value
  - There are specialized methods for decision trees
- Missing for cause
  - The “first principles” approach is to model the causes of the missing data as additional hidden variables and then try to fit the combined model to the available data.
  - Another approach is to treat “missing” as a separate value for the feature
    - For discrete features, this is easy
    - For continuous features, we typically introduce an indicator feature that is 1 if the associated real-valued feature was observed and 0 if not.

Robust to Outliers in the Input Space

- Perceptron: Outliers can cause the algorithm to loop forever
- Logistic Regression: Outliers far from the decision boundary have little impact – robust!
- LDA/QDA: Outliers have a strong impact on the models of $P(\mathbf{x}|y)$ – not robust!

Remaining Criteria

- Monotone Scaling: All linear classifiers are sensitive to non-linear transformations of the inputs, because this may make the data less linearly separable
- Computational Scaling: All three methods scale well to large data sets.
- Irrelevant Inputs: In theory, all three methods will assign smalll weights to irrelevant inputs. In practice, LDA can crash because the Σ matrix becomes singular and cannot be inverted. This can be solved through a technique known as regularization (later!)
- Extract linear combinations of features: All three algorithms learn LTUs, which are linear combinations!
- Interpretability: All three models are fairly easy to interpret
- Predictive power: For small data sets, LDA and QDA often perform best. All three methods give good results.

Summary So Far

(we will add to this later)

Criterion	Perc	Logistic	LDA
Mixed data	no	no	no
Missing values	no	no	yes
Outliers	no	yes	no
Monotone transformations	no	no	no
Scalability	yes	yes	yes
Irrelevant inputs	no	no	no
Linear combinations	yes	yes	yes
Interpretable	yes	yes	yes
Accurate	yes	yes	yes

The Top Five Algorithms

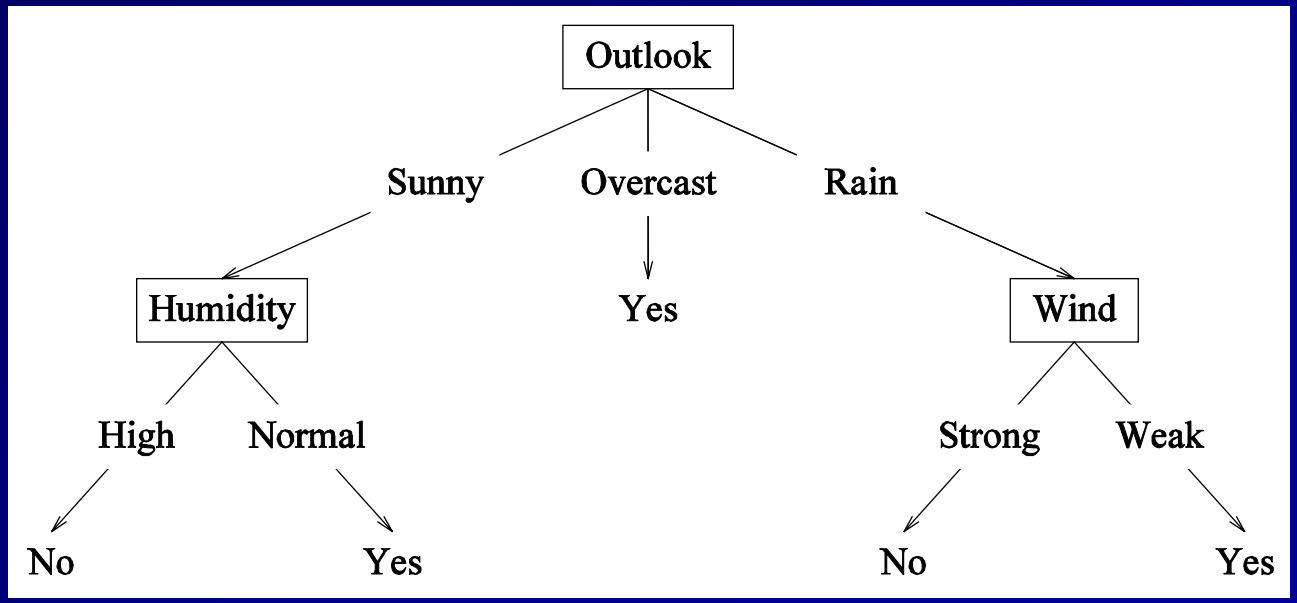
- Decision trees (C4.5)
- Neural networks (backpropagation)
- Probabilistic networks (Naïve Bayes; Mixture models)
- Support Vector Machines (SVMs)
- Nearest Neighbor Method

Learning Decision Trees

- Decision trees provide a very popular and efficient hypothesis space
  - Variable size: any boolean function can be represented
  - Deterministic
  - Discrete and Continuous Parameters
- Learning algorithms for decision trees can be described as
  - Constructive Search: The tree is built by adding nodes
  - Eager
  - Batch (although online algorithms do exist)

Decision Tree Hypothesis Space

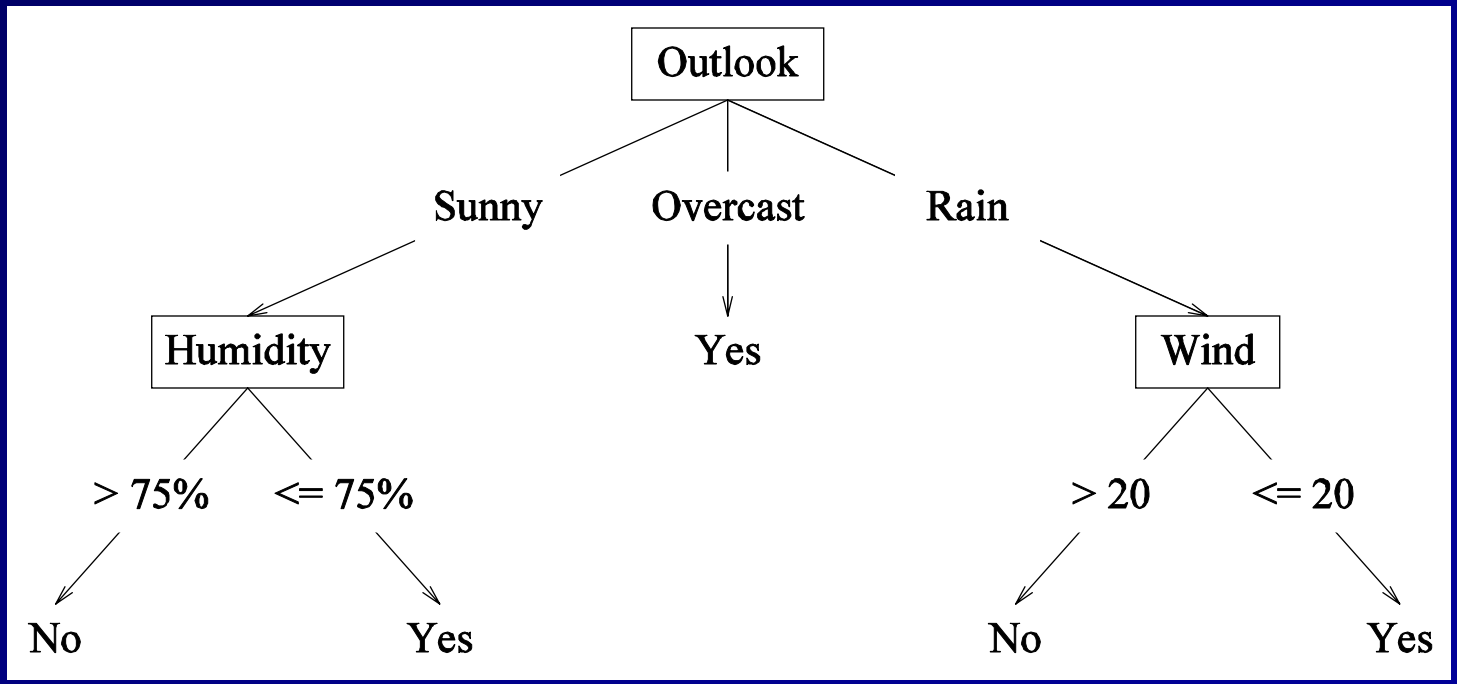
- Internal nodes: test the value of particular features x_j and branch according to the results of the test
- Leaf nodes: specify the class h(**x**)



- Features: Outlook (x_1), Temperature (x_2), Humidity (x_3), and Wind (x_4)
- **x** = (sunny, hot, high, strong) will be classified as No.

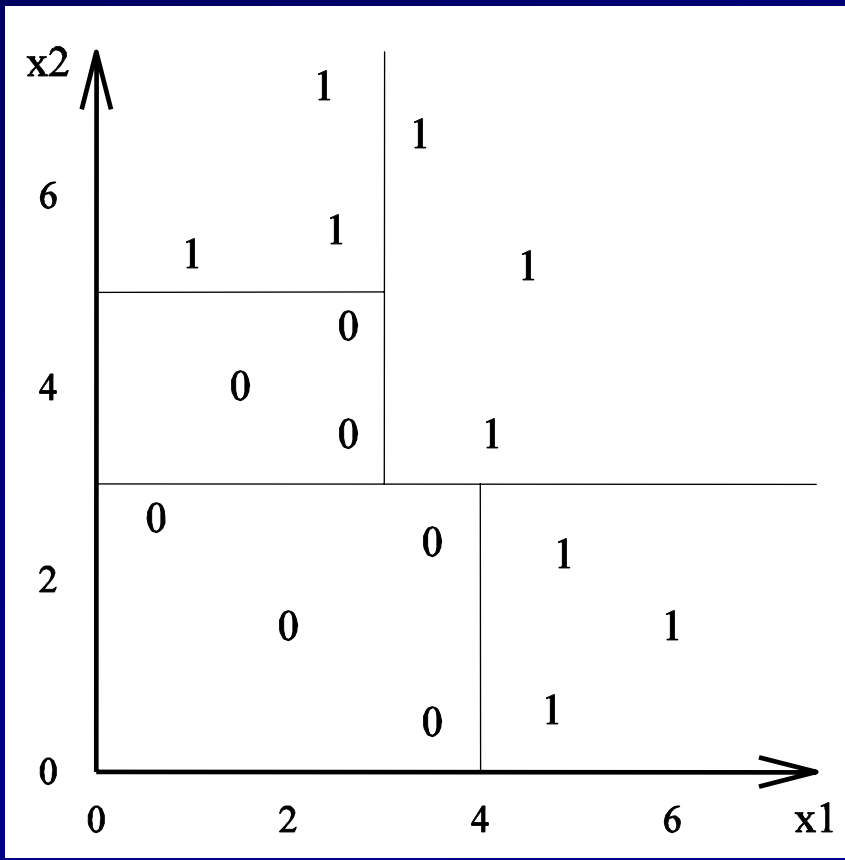
Decision Tree Hypothesis Space (2)

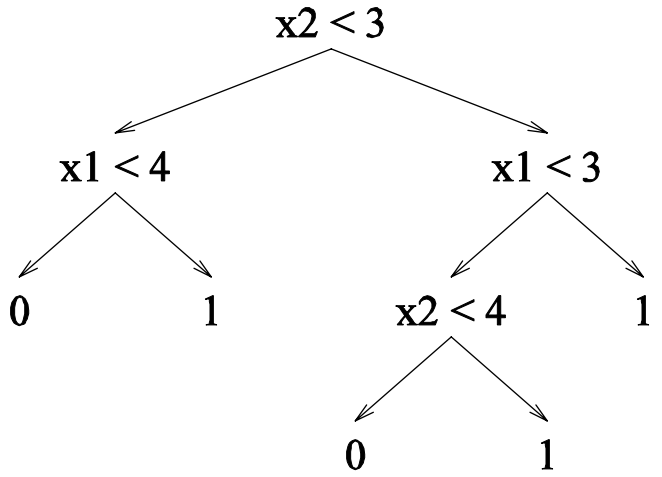
- If the features are continuous, internal nodes may test the value of a feature against a threshold



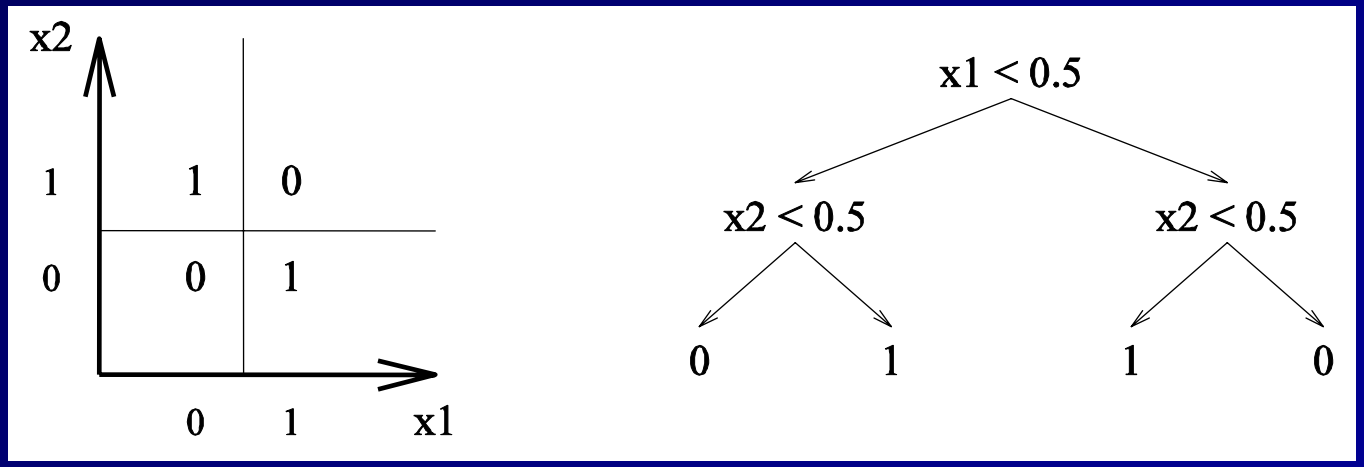
Decision Tree Decision Boundaries

- Decision Trees divide the feature space into axis-parallel rectangles and label each rectangle with one of the K classes





Decision Trees Can Represent Any Boolean Function



- In the worst case, exponentially many nodes will be needed, however

Decision Trees Provide Variable-Sized Hypothesis Space

- As the number of nodes (or depth) of tree increases, the hypothesis space grows
  - Depth 1 (“decision stump”) can represent any boolean function of one feature
  - Depth 2: Any boolean function of two features and some boolean functions involving three features:
    - $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$

Objective Function

- Let h be a decision tree
- Define our objective function to be the number of misclassification errors on the training data:

$$J(h) = |\{ (\mathbf{x},y) \in S : h(\mathbf{x}) \neq y \}|$$

- Find h that minimizes $J(h)$
  - Solution: Just create a decision tree with one path from root to leaf for each training example
  - Bug: Such a tree would just memorize the training data. It would not generalize to new data points
  - Solution 2: Find the smallest tree h that minimizes $J(h)$.
  - Bug 2: This is NP-Hard
  - Solution 3: Use a greedy approximation

Learning Algorithm for Decision Trees

GrowTree(S)
if ($y = 0$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(0)
else if ($y = 1$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(1)
else
    choose best attribute x_j
    $S_0 :=$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
    $S_1 :=$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
    if $S_0 = \emptyset$ **return** new leaf(majority(S));
    else if $S_1 = \emptyset$ **return** new leaf(majority(S));
    else return new node(x_j, GrowTree(S_0), GrowTree(S_1))

Choosing the Best Attribute (Method 1)

- Perform 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data

ChooseBestAttribute(S)

choose j to minimize J_j, computed as follows:

$S_0 :=$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;

$S_1 :=$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;

$y_0 :=$ the most common value of y in S_0

$y_1 :=$ the most common value of y in S_1

$J_0 :=$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$

$J_1 :=$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$

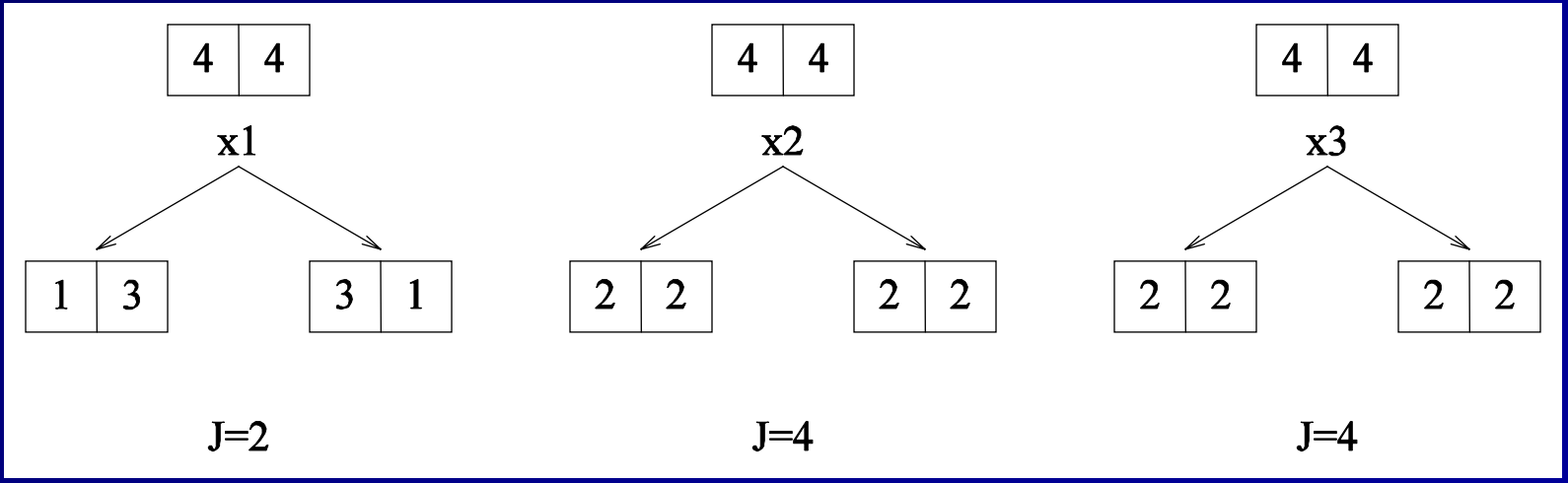
$J_j := J_0 + J_1$ (total errors if we split on this feature)

return j

Choosing the Best Attribute
An Example

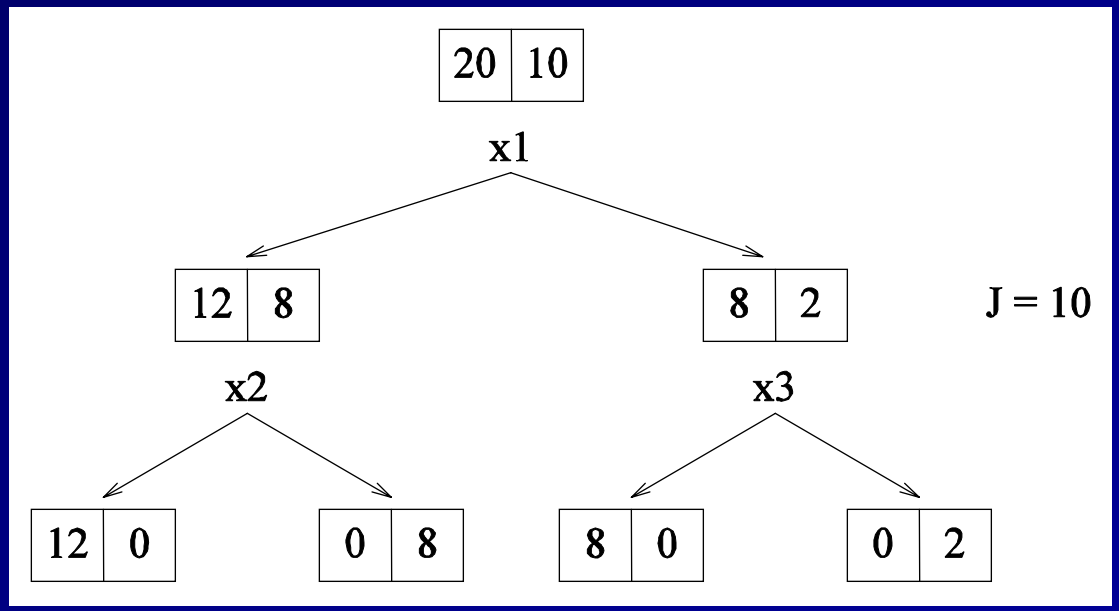
Training Examples

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Choosing the Best Attribute (3)

- Unfortunately, this measure does not always work well, because it does not detect cases where we are making "progress" toward a good tree



A Better Heuristic from Information Theory

- Let *V* be a random variable with the following probability distribution

P(V = 0)	*P(V = 1)*
0.2	0.8

- The surprise *S(V=v)* of each value of *V* is defined to be

$$S(V=v) = -\log_2 P(V = v)$$

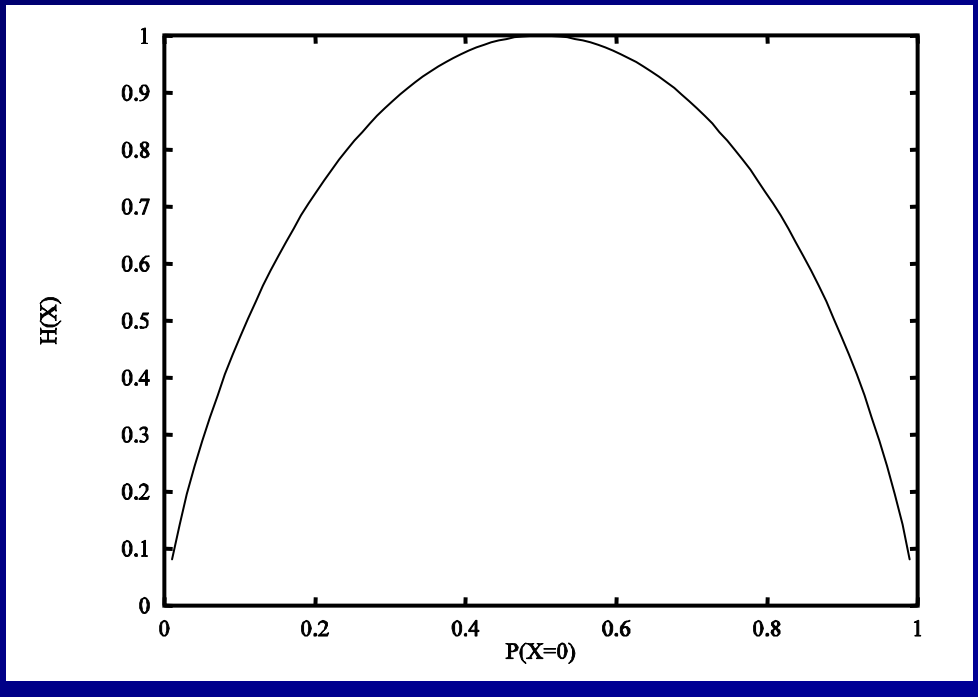
- An event with probability 1 has zero surprise
- An event with probability 0 has infinite surprise
- The surprise is equal to the asymptotic number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results. Hence, this is also called the description length of *V*.

Entropy

- The entropy if *V*, denoted *H(V)*, is defined as

$$H(V) = \sum_{v=0}^{1} -P(V = v) log_2 P(V = v)$$

- This is the average surprise describing the result of one trial of *V* (one coin toss). It can be viewed as a measure of uncertainty

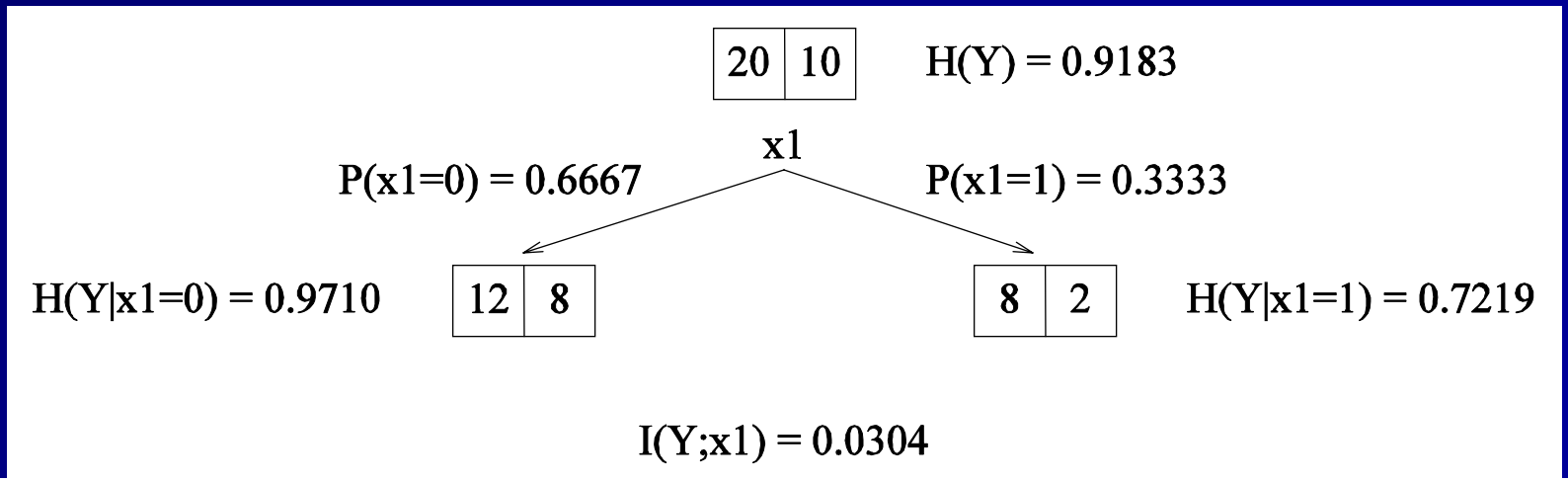


Mutual Information

- Consider two random variables *A* and *B* that are not necessarily independent. The mutual information between *A* and *B* is the amount of information we learn about *B* by knowing the value of *A* (and vice versa – it is symmetric). It is computed as follows:

$$I(A;B) = H(B) - \sum_a P(A=a) \cdot H(B|A=a)$$

- Consider the class *y* of each training example and the value of feature x_1 to be random variables. The mutual information quantifies how much x_1 tells us about *y*.



Choosing the Best Attribute (Method 2)

- Choose the attribute x_j that has the highest mutual information with y.

$$\begin{aligned}\arg\max_j I(x_j; y) &= H(y) - \sum_v P(x_j = v) H(y|x_j = v) \\ &= \arg\min_j \sum_v P(x_j = v) H(y|x_j = v)\end{aligned}$$

- Define $\tilde{J}(j)$ to be the expected remaining uncertainty about y after testing x_j

$$\tilde{J}(j) = \sum_v P(x_j = v) H(y|x_j = v)$$

Choosing the Best Attribute (Method 2)

ChooseBestAttribute(S)

choose j to minimize $\tilde{J}_j$, computed as follows:

$S_0 :=$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;

$S_1 :=$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;

$p_0 := |S_0|/|S|$;

$n_0 := |S_0|$;

$n_{0,y} :=$ number of examples in S_0 with class y

$p_{0,y} := n_{0,y}/n_0$ probability of examples from class y in S_0;

$H(y|x_j = 0) := -\sum_y p_{0,y} \log p_{0,y}$;

compute p_1 and $H(y|x_j = 1)$ in the same way

$\tilde{J}_j := p_0 H(y|x_j = 0) + p_1 H(y|x_j = 1)$

return j

Non-Boolean Features

- Multiple discrete values
  - Method 1: Construct multiway split
  - Method 2: Test for one value versus all of the others
  - Method 3: Group the values into two disjoint sets and test one set against the other
- Real-valued variables
  - Test the variable against a threshold
- In all cases, mutual information can be computed to choose the best split

Efficient Algorithm for Real-Valued Features

- To compute the best threshold θ_j for attribute *j*
  - Sort the examples according to x_{ij}.
    - Let θ be the smallest observed x_{ij} value
    - Let n_{0L}:=0 and n_{1L}:=0 be the number of examples from class *y*=0 and *y*=1 such that $x_{ij} < \theta$
    - Let n_{0R} := N_0 and n_{1R} := N_1 be the number of examples from class *y*=0 and *y*=1 such that $x_{ij} \geq \theta$
  - Increase θ
    - Let y_i be the class of the next instance
      - if y_i = 0, then n_{0L}++ and n_{0R}--
      - else n_{1L}++ and n_{1R}—
    - Compute $J(\theta)$ from n_{0L} , n_{1L} , n_{0R} , and n_{1R} .
    - Remember the smallest value of J and the corresponding θ

Real-Valued Features

- Mutual information of $\theta = 1.2$ is 0.2294

y_i	0	0	1	0	1	1	0	1	1
x_{ij}	0.2	0.4	0.7	1.1	1.3	1.7	1.9	2.4	2.9

$n_{0,L} = 3$	$n_{0,R} = 1$
$n_{1,L} = 1$	$n_{1,R} = 4$

- Mutual information only needs to be computed at points between examples from different classes

Handling Missing Values: Proportional Distribution

- Attach a weight w_i to each example $(\mathbf{x}_i, y_i)$.
  - At the root of the tree, all examples have a weight of 1.0
- Modify all mutual information computations to use weights instead of counts
- When considering a test on attribute *j*, only consider those examples for which x_{ij} is not missing
- When splitting the examples on attribute *j*:
  - Let p_L be the probability that a non-missing example is sent to the left child and p_R be the probability that it is sent to the right child
  - For each example $(\mathbf{x}_i, y_i)$ that is missing attribute *j*, sent it to both children. Send it to the left child with weight $w_i := w_i \cdot p_L$ and to the right child with weight $w_i := w_i \cdot p_R$
- When classifying an example that is missing attribute *j*:
  - Send it down the left subtree. Let $P(\hat{y}_L|\mathbf{x})$ be the resulting prediction
  - Send it down the right subtree. Let $P(\hat{y}_R|\mathbf{x})$ be the resulting prediction
  - Return $p_L \cdot P(\hat{y}_L|\mathbf{x}) + p_R \cdot P(\hat{y}_R|\mathbf{x})$

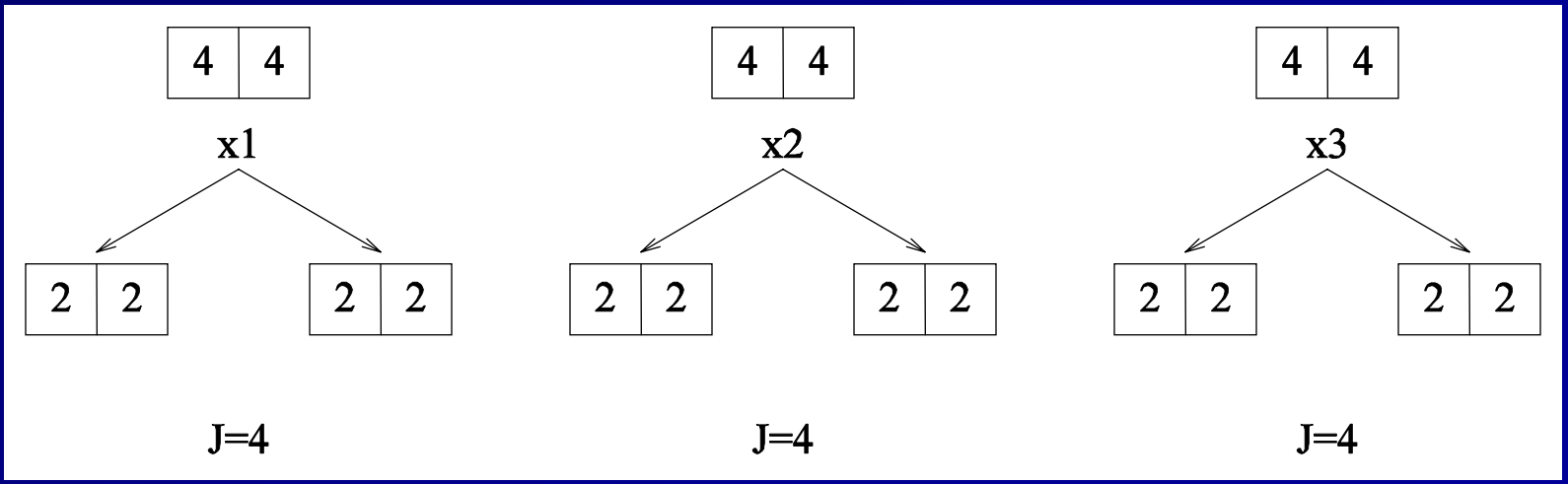
Handling Missing Values: Surrogate Splits

- Choose an attribute *j* and a splitting threshold θ_j using all examples for which x_{ij} is not missing
  - Let u_i be a variable that is 0 if $(\mathbf{x}_i, y_i)$ is sent to the left subtree and 1 if $(\mathbf{x}_i, y_i)$ is sent to the right subtree
  - For each remaining attribute *q*, find the splitting threshold θ_q that best predicts u_i. Sort these by their predictive power and store them in node x_j of the decision tree
- When classifying a new data point $(\mathbf{x}, y)$ that is missing x_j, go through the list of surrogate splits until one is found that is not missing in $\mathbf{x}$. Use x_q and θ_q to decide which child to send $\mathbf{x}$ to.

Failure of Greedy Approximation

- Greedy heuristics cannot distinguish random noise from XOR

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



Decision Tree Evaluation

Criterion	Perc	Logistic	LDA	Trees
Mixed data	no	no	no	yes
Missing values	no	no	yes	yes
Outliers	no	yes	no	yes
Monotone transformations	no	no	no	yes
Scalability	yes	yes	yes	yes
Irrelevant inputs	no	no	no	somewhat
Linear combinations	yes	yes	yes	no
Interpretable	yes	yes	yes	yes
Accurate	yes	yes	yes	no

Decision Tree Summary

- Hypothesis Space
  - variable size (contains all functions)
  - deterministic
  - discrete and continuous parameters
- Search Algorithm
  - constructive search
  - eager
  - batch