Algorithms & Data Analysis

CISC 6950

Lecture 3

(these slides are based on the slides by Prof. F. Provost (Stern NYU) and E. Keogh (UC Riverside))

1

Data Mining: Terminology

Induction (aka ***learning, inductive learning, model induction***)

A process by which a model (or other pattern) is generalized from factual data.

Example:

By analyzing data on past credit customers who have and have not defaulted one can generalize what characterizes customers who are likely to default, as opposed to those who are not.

Attributes **Target**

Name	Balance	Age	Default
Mike	123,000	30	Yes
Mary	51,100	40	Yes
Bill	68,000	55	No
Jim	74,000	46	No
Mark	23,000	47	Yes
Anne	100,000	49	No

<u>**Pattern**</u>
<u>If</u> **Balance** >= 50K and **Age** > 45
<u>Then</u> **Default** = "no"
<u>Else</u> **Default** = "yes"

2

Data Mining: Terminology

A learner, inducer, induction algorithm

A method or algorithm used to generalize a model or pattern from a set of examples

Name	Balance	Age	Default
Mike	123,000	30	Yes
Mary	51,100	40	Yes
Bill	68,000	55	No
Jim	74,000	46	No
Mark	23,000	47	Yes
Anne	100,000	49	No

→ **Learner:** Induces a model from examples

<u>**Classification Model**</u>
<u>If</u> **Balance** >= 50K and **Age** > 45
<u>Then</u> **Default** = "no"
<u>Else</u> **Default** = "yes"

3

Data Mining: Terminology

Contrast: regression modeling (rather than classification)

Target Variable

Name	Balance	Age	Order $
Mike	123,000	30	183
Mary	51,100	40	131
Bill	68,000	55	178
Jim	74,000	46	166
Mark	23,000	47	117
Anne	100,000	49	198

→ **Learner:** Linear regression

<u>**Model**</u>
Amount = 0.002***Income**+2***Age**

4

Data Mining: Terminology

Supervised learning

- Model induction where the model describes a relationship between a set of independent attributes and <u>a predefined dependent attribute – the "target"</u>
- **AND,** the values for the target are available at induction time
- Most induction algorithms fall into the supervised learning category

Name	Balance	Age	Default
Mike	123,000	30	Yes
Mary	51,100	40	Yes
Bill	68,000	55	No
Jim	74,000	46	No
Mark	23,000	47	Yes
Anne	100,000	49	No

Training data, ***labeled*** data

→ **Supervised learning of a classification model**

<u>**Classification Model**</u>
<u>If</u> **Balance** >= 50K and **Age** > 45
<u>Then</u> **Default** = "no"
<u>Else</u> **Default** = "yes"

5

Why trees?

- Decision trees, or *classification trees*, are one of the most popular data mining tools (along with linear/logistic regression)
- They're:
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Almost all data mining packages include DTs
- They have advantages for model comprehensibility, which is important for:
  - model evaluation
  - communication to non-DM-savvy stakeholders

6

Information Gain as A Splitting Criteria

- Select the attribute with the highest information gain (**information gain is the expected reduction in entropy**).
- Entropy is a measure of the uncertainty associated with a random variable. In an information sense, is a measure of **unpredictability**.
- Assume there are two classes, P and N
  - Let the set of examples S contain p elements of class P and n elements of class N
  - The amount of information, needed to decide if an arbitrary example in S belongs to P or N is defined as

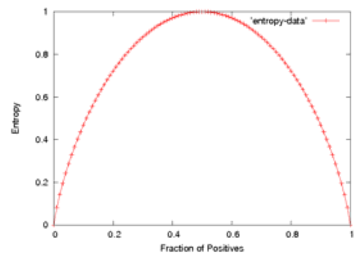
$$E(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

0 log(0) is defined as 0

7

Entropy Plot for Binary Classification

- The entropy is 0 if the outcome is ***certain*** → ***no unpredictability***.
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).



Entropy of a 2-class problem with regard to the portion of one of the two groups

8

Information Gain

- Is the expected reduction in entropy caused by partitioning the examples according to this attribute.
- is the number of bits saved when encoding the target value of an arbitrary member of S, by knowing the value of attribute A.

9

Information Gain in Decision Tree Induction

- Assume that using attribute A, a current set will be partitioned into some number of child sets
- The encoding information that would be gained by branching on A
- Information gain = impurity(parent) - [impurity (children)]

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Note: entropy is at its minimum if the collection of objects is completely uniform

10

Continuous Attribute?

- Each non-leaf node is a test, its edge partitioning the attribute into subsets (easy for discrete attribute).
- For continuous attribute
  - Partition the continuous value of attribute A into a discrete set of intervals
  - Create a new boolean attribute A_c , looking for a threshold c,

$$A_c = \begin{cases} true & \text{if } A_c < c \\ false & \text{otherwise} \end{cases}$$

How to choose c ?

11

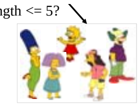
Person	Hair Length	Weight	Age	Class
Homer	0"	250	36	M
Marge	10"	150	34	F
Bart	2"	90	10	M
Lisa	6"	78	8	F
Maggie	4"	20	1	F
Abe	1"	170	70	M
Selma	8"	160	41	F
Otto	10"	180	38	M
Krusty	6"	200	45	M
Comic	8"	290	38	?

12

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Entropy(4**F**,5**M**) = -(4/9)log$_2$(4/9) - (5/9)log$_2$(5/9)
= 0.9911

yes — Hair Length <= 5? — no



Let us try splitting on *Hair length (whether it is <=5)*

Entropy(1**F**,3**M**) = -(1/4)log$_2$(1/4) - (3/4)log$_2$(3/4)
= **0.8113**

Entropy(3**F**,2**M**) = -(3/5)log$_2$(3/5) - (2/5)log$_2$(2/5)
= **0.9710**

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Hair Length <= 5) = 0.9911 – (4/9 * **0.8113** + 5/9 * **0.9710**) = **0.0911**

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Entropy(4**F**,5**M**) = -(4/9)log$_2$(4/9) - (5/9)log$_2$(5/9)
= 0.9911

yes — Weight <= 160? — no

Let us try splitting on *Weight (whether it is <=160)*

Entropy(4**F**,1**M**) = -(4/5)log$_2$(4/5) - (1/5)log$_2$(1/5)
= **0.7219**

Entropy(0**F**,4**M**) = -(0/4)log$_2$(0/4) - (4/4)log$_2$(4/4)
= **0**

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Weight <= 160) = 0.9911 – (5/9 * **0.7219** + 4/9 * **0**) = **0.5900**

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Entropy(4**F**,5**M**) = -(4/9)log$_2$(4/9) - (5/9)log$_2$(5/9)
= 0.9911

yes — age <= 40? — no

Let us try splitting on *Age (whether it is <=40)*

Entropy(3**F**,3**M**) = -(3/6)log$_2$(3/6) - (3/6)log$_2$(3/6)
= **1**

Entropy(1**F**,2**M**) = -(1/3)log$_2$(1/3) - (2/3)log$_2$(2/3)
= **0.9183**

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

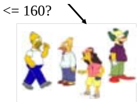
Gain(Age <= 40) = 0.9911 – (6/9 * **1** + 3/9 * **0.9183**) = **0.0183**

Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified… **RECURSION**!

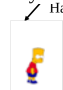
This time we find that we can split on *Hair length*, and we are done!

ID3 uses **information gain** to select the best attribute at each step in growing the tree

yes — Weight <= 160? — no



yes — Hair Length <= 2? — no



We don't need to keep the data around, just the test conditions.

How would these people be classified?

Weight <= 160?
- yes → **Hair Length <= 2?**
  - yes → **Male**
  - no → **Female**
- no → **Male**

It is trivial to convert Decision Trees to rules…

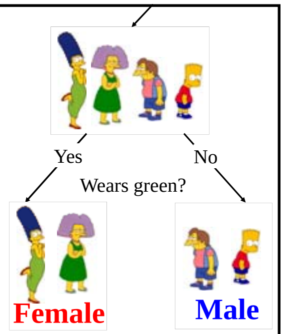
Weight <= 160?
- yes → **Hair Length <= 2?**
  - yes → **Male**
  - no → **Female**
- no → **Male**

Rules to Classify Males/Females

If *Weight* **greater than** 160, classify as **Male**
Elseif *Hair Length* **less than or equal** to 2, classify as **Male**
Else classify as **Female**

The worked examples we have seen were performed on small datasets. However with small datasets there is a great danger of overfitting the data…

When you have few data points, there are many possible splitting rules that perfectly classify the data, but will not generalize to future datasets.



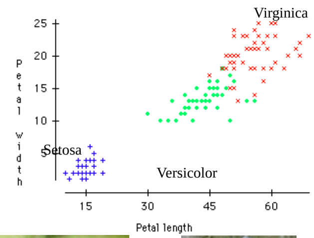
For example, the rule "Wears green?" perfectly classifies the data, so does "Mothers name is Jacqueline?", so does "Has blue shoes"…

A Famous Problem

R. A. Fisher's Iris Dataset.

- 3 classes
- 50 of each class

The task is to classify Iris plants into one of 3 varieties using the Petal Length and Petal Width.

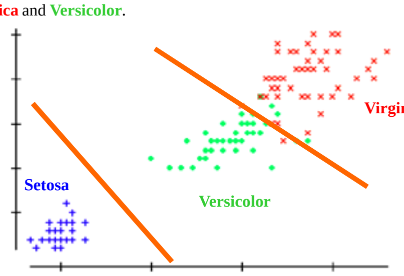


Iris Setosa

Iris Versicolor

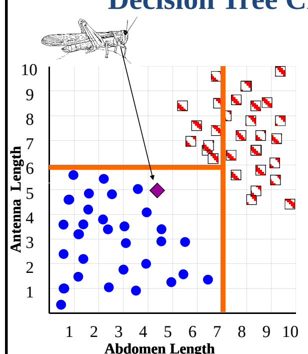
Iris Virginica

We can generalize the piecewise linear classifier to N classes, by fitting N-1 lines. In this case we first learned the line to (perfectly) discriminate between **Setosa** and **Virginica**/**Versicolor**, then we learned to approximately discriminate between **Virginica** and **Versicolor**.

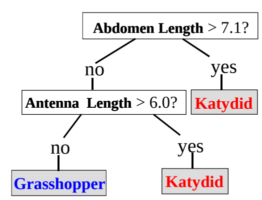


If petal width > 3.272 – (0.325 * petal length) **then** class = **Virginica**
Elseif petal width…

Decision Tree Classifier

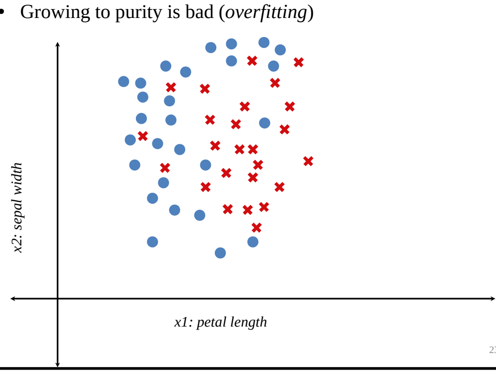


Ross Quinlan



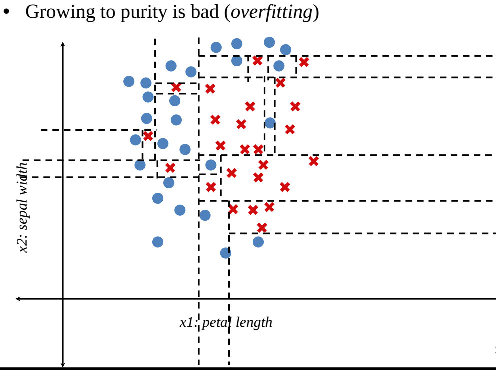
DTs in practice...

- Growing to purity is bad (*overfitting*)



DTs in practice...

- Growing to purity is bad (*overfitting*)



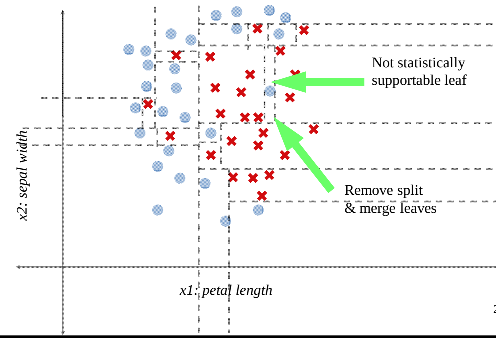
DTs in practice...

- Growing to purity is bad (*overfitting*)
  - Terminate growth early
  - Grow to purity, then *prune* back

25

DTs in practice...

- Growing to purity is bad (*overfitting*)



26

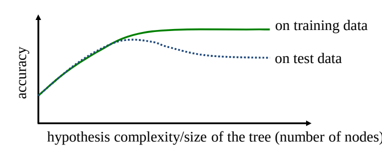
Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples

27

Overfitting

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization to unseen data.
  - There may be noise in the training data that the tree is erroneously fitting.
  - The algorithm may be making poor decisions towards the leaves of the tree that are based on very little data and may not reflect reliable trends.
- A hypothesis, *h*, is said to overfit the training data is there exists another hypothesis which, *h´*, such that *h* has less error than *h´* on the training data but greater error on independent test data.



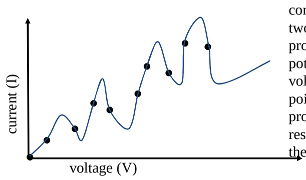
28

Overfitting Example

In electrical circuits, Ohm's law states that the current through a conductor between two points is directly proportional to the potential difference or voltage across the two points, and inversely proportional to the resistance between them.

Experimentally measure 10 points

Fit a curve to the Resulting data.



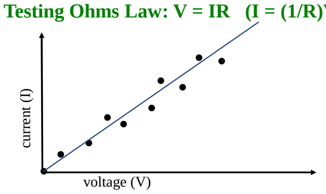
Perfect fit to training data with an 9th degree polynomial (can fit *n* points exactly with an *n*-1 degree polynomial)

Ohm was wrong, we have found a more accurate function!

29

Overfitting Example

Testing Ohms Law: V = IR (I = (1/R)V)



Better generalization with a linear function that fits training data less accurately.

30

How to avoid overfitting?

1. Stop growing the tree before it reaches the point where it perfectly classifies the training data.
   - Such estimation is difficult
   - **Prepruning**: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
2. Allow the tree to overfit the data, and then post-prune the tree
   - Is used
   - **Postpruning**: Grow the full tree, then remove subtrees that do not have sufficient evidence.

Although first approach is more direct, second approach found more successful in practice: because difficult to estimate when to stop

Both need a criterion to determine final tree size

31

Criterion to Determine Correct Tree Size

1. Training and Validation Set Approach:
   - Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
2. Use all available data for training,
   - but apply a statistical test (Chi-square test) to estimate whether expanding (or pruning) a particular node is likely to produce an improvement.
3. Use an explicit measure of the complexity
   - for encoding the training examples and the decision tree,
   - halting growth when this encoding size is minimized.

Training and Validation Set Approach

- Training Set:
  - used to form learned hypotheses
- Validation Set:
  - used to evaluate the accuracy of this hypothesis over subsequent data
  - also, evaluate impact of pruning hypothesis
- Philosophy:
  - Validation set is unlikely to exhibit same random fluctuations as Training set
  - check against over-fitting

Validation Set

- Provides a safety check against overfitting spurious characteristics of data
- Needs to be large enough to provide a statistically significant sample of instances
- Typically validation set is one half size of training set

How to use validation set to Prune

- Consider each node of the decision nodes in the tree to be candidates for pruning
- Pruning a decision tree consists of
  - removing a sub-tree rooted at the node
  - making it a leaf node
  - assigning it the most common classification of the training examples affiliated with that node.
- **Reduced Error Pruning**: Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set.

Reduced Error Pruning Properties

- When pruning begins tree is at maximum size and lowest accuracy over test set
- As pruning proceeds no of nodes is reduced and accuracy over test set increases
- **Disadvantage**: when data is limited, no of samples available for training is further reduced
  - Rule post-pruning is one approach
  - Alternatively, partition available data several times in multiple ways and then average the results

Pre-pruning

- Stop growing the tree when a node is reached which yields the accuracy above a pre-defined *threshold*
- (instead of 100% as in the basic version of ID3).
- e.g. *Chi-square pruning*: only split a node if the deviation in the data is statistically significant.

Post-pruning

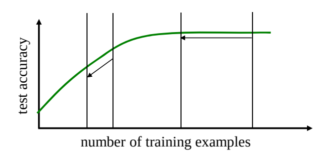
- First create a full tree which may overfit in the first pass; then prune unnecessary subtrees in the second step. Often works better than pre-pruning.

Two post-pruning methods

- **Reduced error pruning**
  - Prune some subtrees (which yield accuracy no worse than before) by:
    - making them a leaf node and assign the majority classification -- ***subtree replacement***
    - removing an internal node -- ***subtree raising*** (used in C4.5)
- **Rule post-pruning**
  - Convert the tree to a set of rules.
  - Remove preconditions of rules which yield accuracy no worse than before -- simplify rules.
  - Order the resulting rules according to the accuracy.

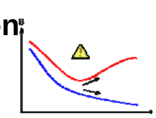
Issues with Reduced Error Pruning

- The problem with this approach is that it potentially "wastes" training data on the validation set.
- Severity of this problem depends where we are on the learning curve:



40

Holdout validation

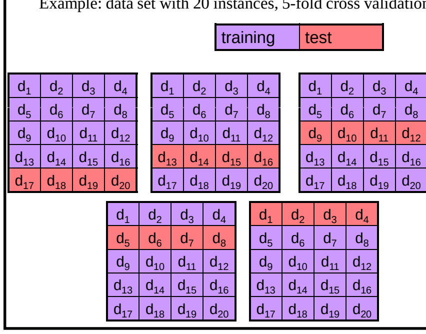


- We are interested in **generalization** – the performance on **data not used for training**
- Given only one data set, we hold out some data for evaluation
  - holdout set for final evaluation is called the test set
- Accuracy on training data is sometimes called "in-sample" accuracy, vs. "out-of-sample" accuracy on test data

41

Cross Validation

Example: data set with 20 instances, 5-fold cross validation



compute error rate for each fold →
then compute average error rate

42

Advantages/Disadvantages of Decision Trees

- 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

43

Additional Decision Tree Issues

- Better splitting criteria
  - Information gain prefers features with many values.
- Continuous features
- Predicting a real-valued function (regression trees)
- Missing feature values
- Features with costs
- Misclassification costs
- Incremental learning
  - ID4
  - ID5
- Mining large databases that do not fit in main memory

44

- Continuous values → nominal

- J48 is a re-implementation of C4.5 release 8 (hence the name J48) in Java. A lot of time has been spent getting the same results as the original C4.5. J48 implements both C4.5's confidence-based post- pruning (default) and sub-tree raising.