HERIOT WATT UNIVERSITY

F20DL and F21DL: Part 2 Machine Learning
Lecture 5: Supervised Learning: Decision trees

Katya Komendantskaya

Weekly Feedback

HERIOT WATT UNIVERSITY

- Test 2: 52 % average: manual computations are mostly good, often Weka experiments either not run or run incorrectly (Use **Classes to clusters evaluation**)
- CW2: You are allowed to work with a smaller (reduced) data set if your computer struggles too much (this is explained in the spec, too).
  - Other options: increasing the heap size
  - Using CLI and automating the experiments
- CW2 is not a competition for accuracy!!! I will judge the **quality and quantity** of your work: programming, data and algorithm analysis, research into machine learning algorithms and settings. I want to see evidence of your learning and thinking while completing CW2.
- For detailed explanation of Weka output and options – see Youtube videos on Weka

So far

HERIOT WATT UNIVERSITY

... we covered:

- Bayesian Learning, Bayes Nets
- Learning was defined as Knowledge revision (in the form of prior/posterior probabilities)

So far

HERIOT WATT UNIVERSITY

... we covered:

- Bayesian Learning, Bayes Nets
- Learning was defined as Knowledge revision (in the form of prior/posterior probabilities)

- Unsupervised Learning, or clustering.
- Learning was about finding a good model (in the form of a *pval* function from features to clusters): learning as search

So far

... we covered:

- Bayesian Learning, Bayes Nets
- Learning was defined as Knowledge revision (in the form of prior/posterior probabilities)

- Unsupervised Learning, or clustering.
- Learning was about finding a good model (in the form of a *pval* function from features to clusters): learning as search

Today:

- We start Supervised Learning.
- This kind of learning is about finding a good hypothesis in the hypothesis space: now with the purpose of making (class) predictions.

Example Classification Data

Predicting whether an email will be read...

Training Examples:

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

New Examples:

e7	???	known	new	short	work
e8	???	unknown	new	short	work

We want to classify new examples on feature *Action* based on the examples' *Author*, *Thread*, *Length*, and *Where*.

Learning tasks

Learning tasks can be characterized by the feedback given to the learner.

- Unsupervised learning No classifications are given; the learner has to discover categories and regularities in the data.

Learning tasks

HERIOT WATT UNIVERSITY

Learning tasks can be characterized by the feedback given to the learner.

- Unsupervised learning No classifications are given; the learner has to discover categories and regularities in the data.
- Supervised learning What has to be learned is specified for each example.

Learning tasks

HERIOT WATT UNIVERSITY

Learning tasks can be characterized by the feedback given to the learner.

- Unsupervised learning No classifications are given; the learner has to discover categories and regularities in the data.
- Supervised learning What has to be learned is specified for each example.
- Reinforcement learning Feedback occurs after a sequence of actions.

Data

HERIOT WATT UNIVERSITY

- Data isn't perfect:
  - the features given are inadequate to predict the classification
  - there are examples with missing features
  - some of the features are assigned the wrong value

Data

HERIOT WATT UNIVERSITY

- Data isn't perfect:
  - the features given are inadequate to predict the classification
  - there are examples with missing features
  - some of the features are assigned the wrong value
- overfitting occurs when distinctions appear in the training data, but not in the unseen examples.

Supervised Learning

Given:

- a set of inputs features $X_1, \ldots, X_n$

Supervised Learning

HERIOT WATT UNIVERSITY

Given:

- a set of inputs features $X_1, \ldots, X_n$
- a set of target features $Y_1, \ldots, Y_k$

Supervised Learning

HERIOT WATT UNIVERSITY

Given:

- a set of inputs features $X_1, \ldots, X_n$
- a set of target features $Y_1, \ldots, Y_k$
- a set of training examples where the values for the input features and the target features are given for each example

Supervised Learning

HERIOT WATT UNIVERSITY

Given:

- a set of inputs features $X_1, \ldots, X_n$
- a set of target features $Y_1, \ldots, Y_k$
- a set of training examples where the values for the input features and the target features are given for each example
- a new example, where only the values for the input features are given

Supervised Learning

Given:

- a set of inputs features $X_1, \ldots, X_n$
- a set of target features $Y_1, \ldots, Y_k$
- a set of training examples where the values for the input features and the target features are given for each example
- a new example, where only the values for the input features are given

Task: predict the values for the target features for the new example.

Supervised Learning

Given:

- a set of inputs features $X_1, \ldots, X_n$
- a set of target features $Y_1, \ldots, Y_k$
- a set of training examples where the values for the input features and the target features are given for each example
- a new example, where only the values for the input features are given

Task: predict the values for the target features for the new example.

- classification when the Y_i are discrete
- regression when the Y_i are continuous

Evaluating Predictions

Suppose we want to make a prediction of a value for a target feature on example e:

Evaluating Predictions

Suppose we want to make a prediction of a value for a target feature on example e:

- $val(e, Y)$ is the observed value of target feature Y on example e.

Evaluating Predictions

Suppose we want to make a prediction of a value for a target feature on example e:

- $val(e, Y)$ is the observed value of target feature Y on example e.
- $pval(e, Y)$ is the predicted value of target feature Y on example e.

Evaluating Predictions

HERIOT WATT UNIVERSITY

Suppose we want to make a prediction of a value for a target feature on example e:

- $val(e, Y)$ is the observed value of target feature Y on example e.
- $pval(e, Y)$ is the predicted value of target feature Y on example e.
- The error of the prediction is a measure of how close $pval(e, Y)$ is to $val(e, Y)$.

Evaluating Predictions

Suppose we want to make a prediction of a value for a target feature on example e:

- $val(e, Y)$ is the observed value of target feature Y on example e.
- $pval(e, Y)$ is the predicted value of target feature Y on example e.
- The error of the prediction is a measure of how close $pval(e, Y)$ is to $val(e, Y)$.
- There are many possible errors that could be measured.

Evaluating Predictions

HERIOT WATT UNIVERSITY

Suppose we want to make a prediction of a value for a target feature on example e:

- $val(e, Y)$ is the observed value of target feature Y on example e.
- $pval(e, Y)$ is the predicted value of target feature Y on example e.
- The error of the prediction is a measure of how close $pval(e, Y)$ is to $val(e, Y)$.
- There are many possible errors that could be measured.

Sometimes $pval(e, Y)$ can be a real number even though $val(e, Y)$ can only have a few values.

Most common error rates

Given E – a set of examples, and T – a set of target features

- absolute error
  $\sum_{e \in E} \sum_{Y \in T} |val(e, Y) - pval(e, Y)|$ - summed over all examples

Most common error rates

Given E – a set of examples, and T – a set of target features

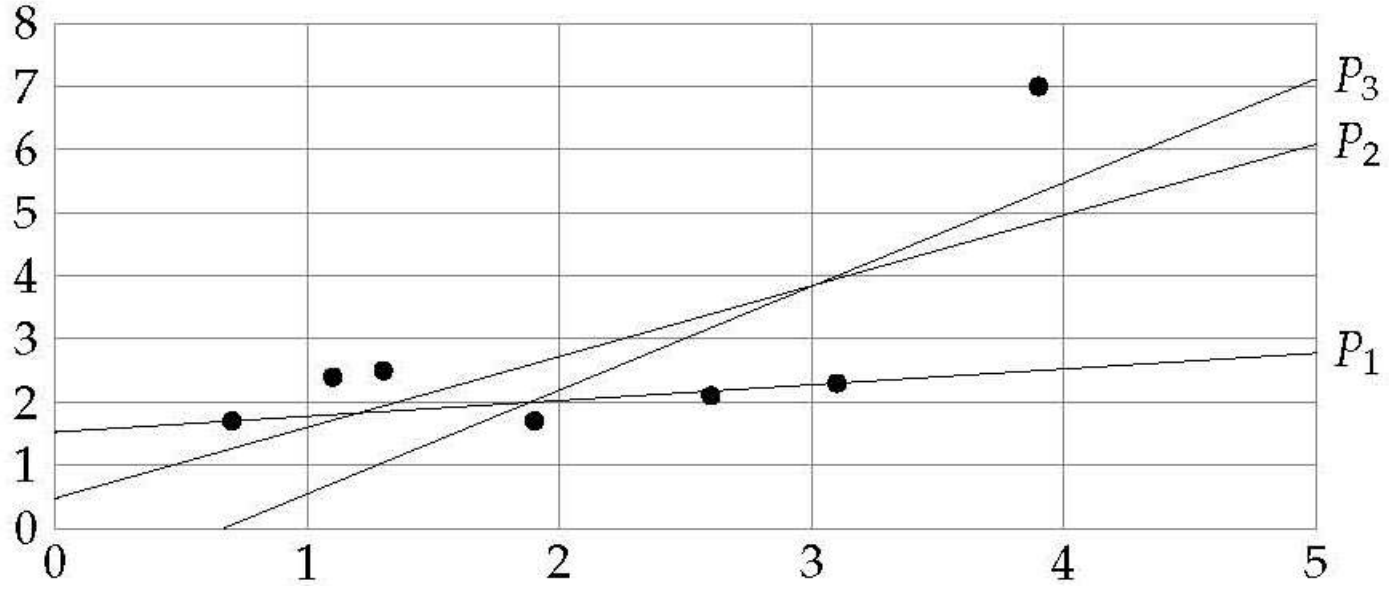
- absolute error
  $\sum_{e \in E} \sum_{Y \in T} |val(e, Y) - pval(e, Y)|$ - summed over all examples
- sum of squares error
  $\sum_{e \in E} \sum_{Y \in T} (val(e, Y) - pval(e, Y))^2$ – summed over all examples

Most common error rates

Given E – a set of examples, and T – a set of target features

- absolute error
  $\sum_{e \in E} \sum_{Y \in T} |val(e, Y) - pval(e, Y)|$ - summed over all examples
- sum of squares error
  $\sum_{e \in E} \sum_{Y \in T} (val(e, Y) - pval(e, Y))^2$ – summed over all examples
- worst-case error $max_{e \in E} max_{Y \in T} |val(e, Y) - pval(e, Y)|$

The effects of different Error estimation:



- P_1 minimizes the absolute error,
- P_2 minimizes the sum-of-squares error,
- P_3 minimizes the worst-case error.

Training and Test Sets

HERIOT WATT UNIVERSITY

To evaluate how well a learner will work on future predictions, we divide the examples into:

- training examples that are used to train the learner
- test examples that are used to evaluate the learner

...these must be kept separate.

Basic Models for Supervised Learning

HERIOT WATT UNIVERSITY

Many learning algorithms can be seen as deriving from:

- Bayesian classifiers
- decision trees
- linear (and non-linear) classifiers – of which Neural nets are a part

Learning Decision Trees

HERIOT WATT UNIVERSITY

- Representation is a decision tree.
- Bias is towards simple decision trees.
- Search through the space of decision trees, from simple decision trees to more complex ones.

Intuition

Let us start with a motivating example and construct a decision tree on the board.

Example on the board

HERIOT WATT UNIVERSITY

Training Examples:

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work
e7	skips	unknown	old	short	work
e8	reads	unknown	new	short	work
e9	skips	known	old	long	home
e10	skips	known	new	long	work
e11	skips	unknown	old	short	home
e12	skips	known	new	long	work
e13	reads	known	old	short	home
e14	reads	known	new	short	work
e15	reads	known	new	short	home
e16	reads	known	old	short	work
e17	reads	known	new	short	home
e18	reads	unknown	new	short	work

Example on the board

HERIOT WATT UNIVERSITY

	Action	Author	Thread	Length	Where
e1					
e2	reads	unknown	new	short	work
e3					
e4					
e5	reads	known	new	short	home
e6					
e7	skips	unknown	old	short	work
e8	reads	unknown	new	short	work
e9					
e10					
e11	skips	unknown	old	short	home
e12					
e13	reads	known	old	short	home
e14	reads	known	new	short	work
e15	reads	known	new	short	home
e16	reads	known	old	short	work
e17	reads	known	new	short	home
e18	reads	unknown	new	short	work

Example on the board

HERIOT WATT UNIVERSITY

	Action	Author	Thread	Where
e1				
e2	reads	unknown	new	work
e3				
e4				
e5	reads	known	new	home
e6				
e7	skips	unknown	old	work
e8	reads	unknown	new	work
e9				
e10				
e11	skips	unknown	old	home
e12				
e13	reads	known	old	home
e14	reads	known	new	work
e15	reads	known	new	home
e16	reads	known	old	work
e17	reads	known	new	home
e18	reads	unknown	new	work

Example on the board

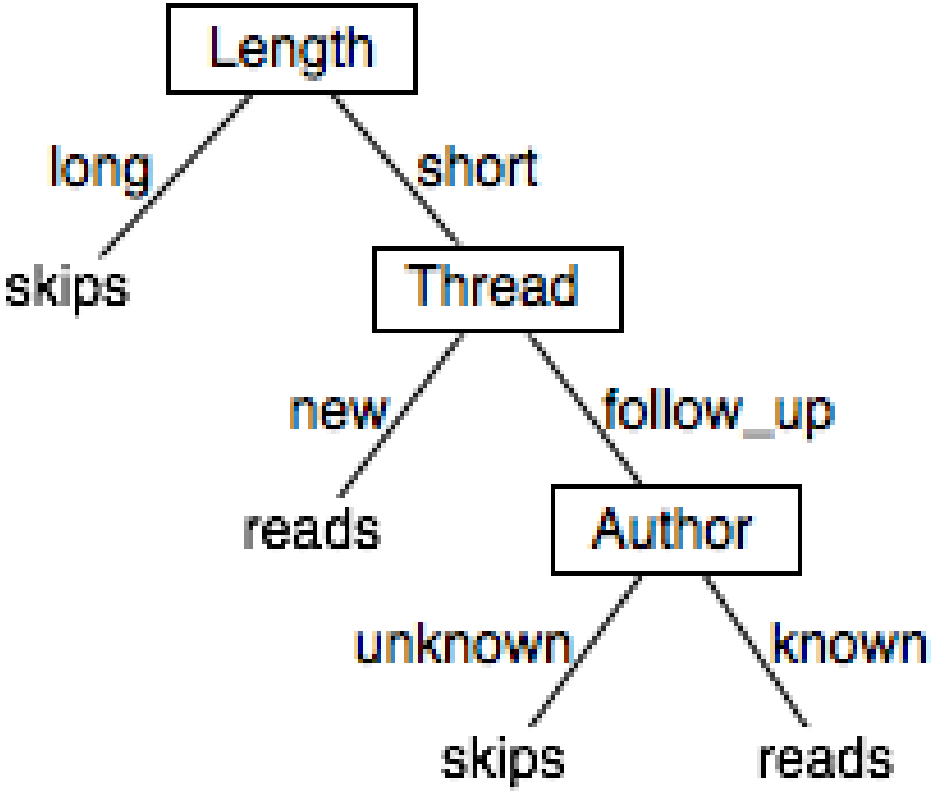
HERIOT WATT UNIVERSITY

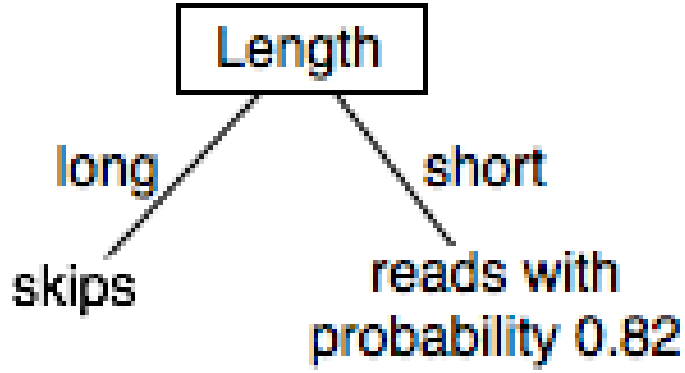
	Action	Author	Thread	Where
e1				
e2				
e3				
e4				
e5				
e6				
e7	skips	unknown	old	work
e8				
e9				
e10				
e11	skips	unknown	old	home
e12				
e13	reads	known	old	home
e14				
e15				
e16	reads	known	old	work
e17				
e18				

Example on the board

	Action	Author	Where
e1			
e2			
e3			
e4			
e5			
e6			
e7	skips	unknown	work
e8			
e9			
e10			
e11	skips	unknown	home
e12			
e13	reads	known	home
e14			
e15			
e16	reads	known	work
e17			
e18			

Example Decision Trees





Conclusions from this experiment

HERIOT WATT UNIVERSITY

- It is a simple iterative algorithm, with termination condition given by...

Conclusions from this experiment

HERIOT WATT UNIVERSITY

- It is a simple iterative algorithm, with termination condition given by... reducing the number of attributes at each run
- Choice of nodes/attributes is important
- It determines tree architecture and size, as well as accuracy

Conclusions from this experiment

HERIOT WATT UNIVERSITY

- It is a simple iterative algorithm, with termination condition given by... reducing the number of attributes at each run
- Choice of nodes/attributes is important
- It determines tree architecture and size, as well as accuracy
- Some features may be redundant and will not appear in the tree at all (which is good!)

Decision trees: high-level summary

HERIOT WATT UNIVERSITY

A (binary) decision tree (for a particular output feature) is a tree where:

- Each nonleaf node is labeled with a test (function of input features).
- The arcs out of a node labeled with values for the test.
- The leaves of the tree are labeled with point prediction of the output feature.

Now make predictions: on the board

HERIOT WATT UNIVERSITY

Training Examples:

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
⋮	⋮	⋮	⋮	⋮	⋮

New Examples:

e19	???	known	new	short	work
e20	???	unknown	new	short	work

We want to classify new examples on feature *Action* based on the examples' *Author*, *Thread*, *Length*, and *Where*.

Issues in decision-tree learning

HERIOT WATT UNIVERSITY

- Given some training examples, which decision tree should be generated?

Issues in decision-tree learning

HERIOT WATT UNIVERSITY

- Given some training examples, which decision tree should be generated?
- A decision tree can represent any discrete function of the input features.

Issues in decision-tree learning

HERIOT WATT UNIVERSITY

- Given some training examples, which decision tree should be generated?
- A decision tree can represent any discrete function of the input features.
- You need a bias. For example, prefer the smallest tree. Least depth? Fewest nodes? Which trees are the best predictors of unseen data?

Issues in decision-tree learning

HERIOT WATT UNIVERSITY

- Given some training examples, which decision tree should be generated?
- A decision tree can represent any discrete function of the input features.
- You need a bias. For example, prefer the smallest tree. Least depth? Fewest nodes? Which trees are the best predictors of unseen data?
- How should you go about building a decision tree? The space of decision trees is too big for systematic search for the smallest decision tree.

Algorithm: general idea

HERIOT WATT UNIVERSITY

- The input is a set of input features, a target feature and, a set of training examples.

Algorithm: general idea

HERIOT WATT UNIVERSITY

- The input is a set of input features, a target feature and, a set of training examples.
- Either:
  - Stop and return a value for the target feature or a distribution over target feature values

Algorithm: general idea

HERIOT WATT UNIVERSITY

- The input is a set of input features, a target feature and, a set of training examples.
- Either:
  - Stop and return a value for the target feature or a distribution over target feature values
  - Choose a test (e.g. an input feature) to split on. For each value of the test, build a subtree for those examples with this value for the test.

Algorithm 1: Learning a tree

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeLearner*(X, Y, E)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of training examples

Algorithm 1: Learning a tree

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeLearner*(X, Y, E)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of training examples
6: **Output**
7: decision tree

Algorithm 1: Learning a tree

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeLearner*(X, Y, E)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of training examples
6: **Output**
7: decision tree
8: **if** stopping criterion is true **then**
9: **return** *pointEstimate*(Y, E)
10: **else**

Algorithm 1: Learning a tree

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeLearner*(X, Y, E)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of training examples
6: **Output**
7: decision tree
8: **if** stopping criterion is true **then**
9: **return** *pointEstimate*(Y, E)
10: **else**
11: Select feature $X_i \in X$ with domain $\{v_1, v_2\}$
12: let $E_1 = \{e \in E : val(e, X_i) = v_1\}$
13: let $T_1 = DecisionTreeLearner(X/\{X_i\}, Y, E_1)$

Algorithm 1: Learning a tree

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeLearner*(X, Y, E)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of training examples
6: **Output**
7: decision tree
8: **if** stopping criterion is true **then**
9: **return** *pointEstimate*(Y, E)
10: **else**
11: Select feature $X_i \in X$ with domain $\{v_1, v_2\}$
12: let $E_1 = \{e \in E : val(e, X_i) = v_1\}$
13: let $T_1 = DecisionTreeLearner(X/\{X_i\}, Y, E_1)$
14: let $E_2 = \{e \in E : val(e, X_i) = v_2\}$
15: let $T_2 = DecisionTreeLearner(X/\{X_i\}, Y, E_2)$

Algorithm 1: Learning a tree

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeLearner*(X, Y, E)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of training examples
6: **Output**
7: decision tree
8: **if** stopping criterion is true **then**
9: **return** *pointEstimate*(Y, E)
10: **else**
11: Select feature $X_i \in X$ with domain $\{v_1, v_2\}$
12: let $E_1 = \{e \in E : val(e, X_i) = v_1\}$
13: let $T_1 = DecisionTreeLearner(X/\{X_i\}, Y, E_1)$
14: let $E_2 = \{e \in E : val(e, X_i) = v_2\}$
15: let $T_2 = DecisionTreeLearner(X/\{X_i\}, Y, E_2)$
16: **return** $\langle X_i = v_1, T_1, T_2 \rangle$

Algorithm 2: Making predictions

1: **Algorithm** *DecisionTreeClassfy*(e, X, Y, DT)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: e: example to classify
6: DT: decision tree

Algorithm 2: Making predictions

1: **Algorithm** *DecisionTreeClassfy*(e, X, Y, DT)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: e: example to classify
6: DT: decision tree
7: **Output**
8: prediction on Y for example e

Algorithm 2: Making predictions

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeClassfy*(e, X, Y, DT)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: e: example to classify
6: DT: decision tree
7: **Output**
8: prediction on Y for example e
7: **Local**
9: S subtree of DT, $S := DT$

Algorithm 2: Making predictions

HERIOT WATT UNIVERSITY

1: **Algorithm** *DecisionTreeClassfy*(e, X, Y, DT)
2: **Inputs**
3: X: set of features $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: e: example to classify
6: DT: decision tree
7: **Output**
8: prediction on Y for example e
7: **Local**
9: S subtree of DT, $S := DT$
10: **while** S is an internal node of the form $\langle X_i = v, T_1, T_2 \rangle$ **do**
11: **if** $val(e, X_i) = v$ **then**
12: $S := T_1$
13: **else**
14: $S := T_2$
15: **return** S

Choices in implementing the algorithm

HERIOT WATT UNIVERSITY

- When to stop:

Choices in implementing the algorithm

HERIOT WATT UNIVERSITY

- When to stop:
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split

Choices in implementing the algorithm

HERIOT WATT UNIVERSITY

- When to stop:
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split
- Which test to split on isn't defined. Often one uses myopic split: which single split gives smallest error.
- With multi-valued features, the test can be to split on all values or split values into half. More complex tests are possible.

Example Classification Data

HERIOT WATT UNIVERSITY

Training Examples:

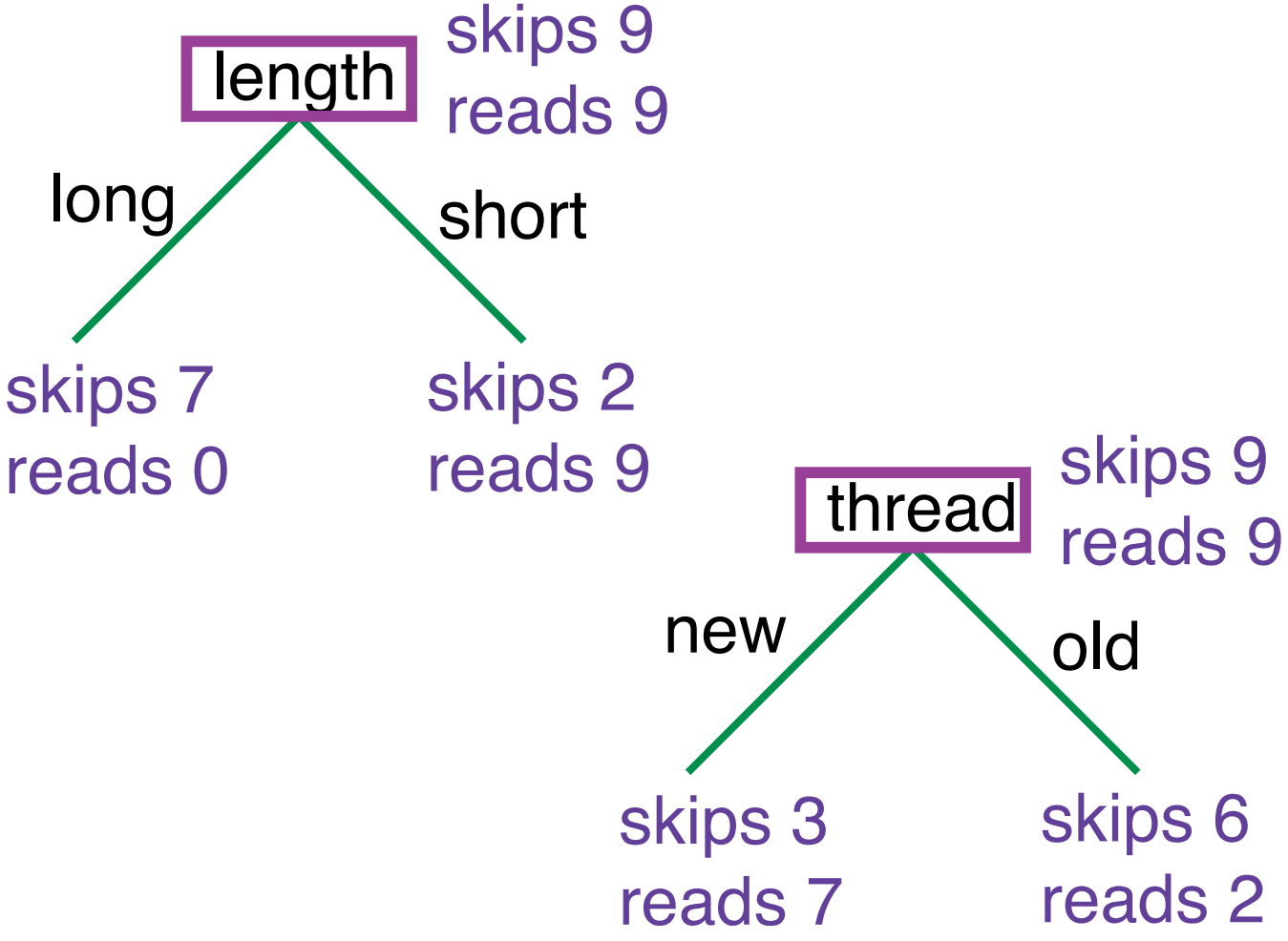
	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

New Examples:

e7	???	known	new	short	work
e8	???	unknown	new	short	work

We want to classify new examples on feature *Action* based on the examples' *Author*, *Thread*, *Length*, and *Where*.

Example: possible splits



Handling Overfitting

HERIOT WATT UNIVERSITY

- This algorithm can overfit the data. This occurs when

Handling Overfitting

HERIOT WATT UNIVERSITY

- This algorithm can overfit the data.
  This occurs when noise and correlations present in the training set are not reflected in the data as a whole.
- To handle overfitting:
  - restrict the splitting, and split only when the split is useful.
  - allow unrestricted splitting and prune the resulting tree where it makes unwarranted distinctions.
  - learn multiple trees and average them.

You are ready to practice Weka Decision tree tools

As always – check related Chapters in the Course textbook:
Section 4.3, pp.99-108; Section 11.4, 454-457

Test 3 – 3 %

HERIOT WATT UNIVERSITY

1. Construct a decision tree for the "Small Emotion Recognition set" (manually).

- The splitting criterion is: take each attribute in order of its appearance in the data set.
- The terminating condition is: all examples are classified in the same class
- Left branches should stand for "White", and right branches should stand for "Black".
- Be ready to answer questions about the tree's architecture.

Test 3

HERIOT WATT UNIVERSITY

2 Use this tree and the tests:

1. Test 1: a Happy face with noise:
   `Black , Black , Black , White , ???`
2. Test 2: a Sad face with mustache:
   `White, Black, Black, Black, ???`

- Make predictions

Test 3

3 Construct a decision tree for the "Small Emotion Recognition set" (manually).

- This time, use a **myopic** splitting criterion: always choose the attribute splitting of which allows to firmly classify some instances.
- If such does not exist, or several such exist, choose the leftmost attribute.
- All other settings are the same.
- Be ready to answer questions about the tree's architecture.

4 Perform the same tests on this new tree
Test 1: `Black , Black , Black , White , ???`
Test 2: `White, Black, Black, Black, ???`

Test 3

HERIOT WATT UNIVERSITY

5 Make conclusions about accuracy, size, performance of these two trees; redundant and most correlating attributes.

Test 3

HERIOT WATT UNIVERSITY

6 Decisions tree in Weka – Algorithm J48

- Load the given "small emotion recognition"set. Load the test set that includes the above examples, marked as Happy and Sad, respectively.
- For running J48 in Weka, set the following options:
- do binary splits (binary splits = True)
- do not collapse tree (collapseTRee = false)
- minimum number of Objects = 1
- save Instance data
- seed 1
- construct unpruned tree
- All set, you should have the configuration:

Test 3

HERIOT WATT UNIVERSITY

7 Repeat Weka experiment with Algorithm J48 :

- Use the same options but set
- minimum number of Objects = 2
- Visualise this tree, be ready to answer questions about its architecture and accuracy on the given tests