# Course 395: Machine Learning – Lectures

- Lecture 1-2: Concept Learning *(M. Pantic)*
- Lecture 3-4: Decision Trees & CBC Intro *(M. Pantic & S. Petridis)*
- Lecture 5-6: Evaluating Hypotheses *(S. Petridis)*
- Lecture 7-8: Artificial Neural Networks I *(S. Petridis)*
- Lecture 9-10: Artificial Neural Networks II *(S. Petridis)*
- Lecture 11-12: Instance Based Learning *(M. Pantic)*
- Lecture 13-14: Genetic Algorithms *(M. Pantic)*

# Decision Trees – Lecture Overview

- Problem Representation using a Decision Tree
- ID3 algorithm
- The problem of overfitting

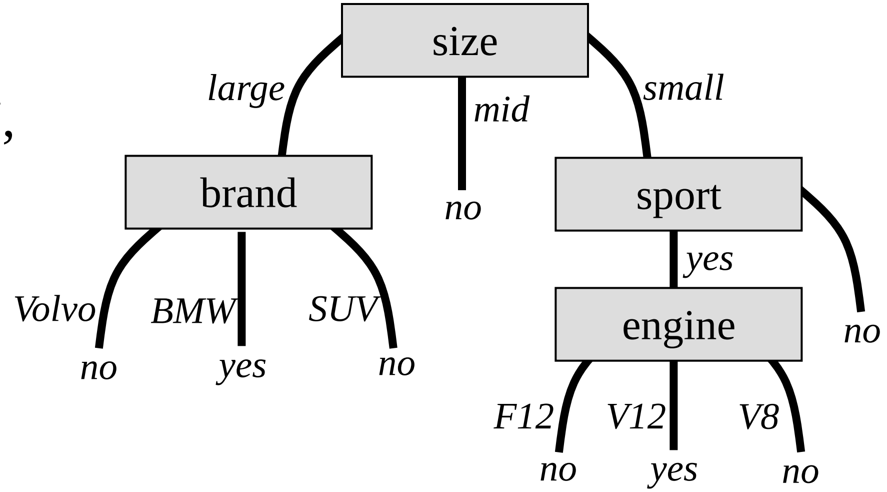
# Problem Representation using a Decision Tree

- Decision Tree learning is a method for approximating discrete classification functions by means of a tree-based representation

- A learned Decision Tree classifies a new instance by sorting it down the tree
  - tree node ↔ classification OR test of a specific attribute of the instance
  - tree branch ↔ possible value for the attribute in question

- Concept: *Good Car*

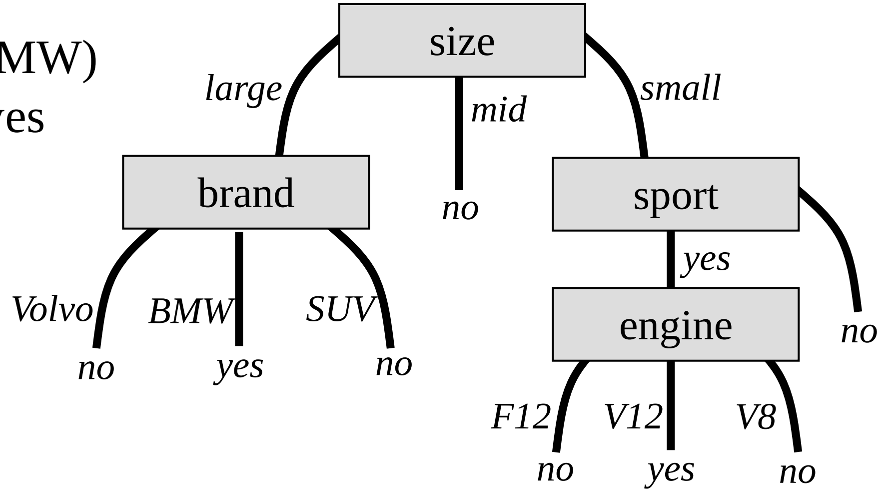
  ‹size = *small*, brand = *Ferari*, model = *Enzo*, sport = yes, engine = *V12*, colour = *red*›



# Problem Representation using a Decision Tree

- A learned Decision Tree can be represented as a set of *if-then* rules
- To 'read out' the rules from a learned Decision Tree
  - tree ↔ disjunction (∨) of sub-trees
  - sub-tree ↔ conjunction (∧) of constraints on the attribute values
- Rule: *Good Car*

IF (size = large AND brand = BMW)
OR (size = small AND sport = yes
AND engine = V12)
THEN Good Car = yes
ELSE Good Car = no;



# Decision Tree Learning Algorithm

- Decision Tree learning algorithms employ top-down greedy search through the space of possible solutions.

- A general Decision Tree learning algorithm:
  1. perform a statistical test of each attribute to determine how well it classifies the training examples when considered alone;
  2. select the attribute that performs best and use it as the root of the tree;
  3. to decide the descendant node down each branch of the root (parent node), sort the training examples according to the value related to the current branch and repeat the process described in steps 1 and 2.

- ID3 algorithm is one of the most commonly used Decision Tree learning algorithms and it applies this general approach to learning the decision tree.

# ID3 Algorithm

- ID3 algorithm uses so-called *Information Gain* to determine how informative an attribute is (i.e., how well it classifies the training examples).

- *Information Gain* is based on a measure that we call *Entropy*, which characterizes the impurity of a collection of examples *S* (i.e., impurity↑ → *E(S)*↑):

  $$E(S) \equiv - p_{\oplus} \log_2 p_{\oplus} - p_{\otimes} \log_2 p_{\otimes},$$

  where $p_{\oplus}$ ($p_{\otimes}$) is the proportion of positive (negative) examples in *S*.

  (<u>Note</u>: $E(S) = 0$ if *S* contains only positive or only negative examples
  $\leftrightarrow p_{\oplus} = 1, p_{\otimes} = 0, E(S) = - 1 \cdot 0 - 0 \cdot \log_2 p_{\otimes} = 0$

  (<u>Note</u>: $E(S) = 1$ if *S* contains equal amount of positive and negative examples
  $\leftrightarrow p_{\oplus} = ½, p_{\otimes} = ½, E(S) = - ½ \cdot (-1) - ½ \cdot (-1) = 1$

  In the case that that the target attribute can take *n* values:

  $$E(S) \equiv - \sum_I p_i \log_2 p_i, \; i = [1..n]$$

  where $p_i$ is the proportion of examples in *S* having the target attribute value *i*.

# ID3 Algorithm

- *Information Gain* is based on a measure that we call *Entropy*, which characterizes the impurity of a collection of examples *S* (impurity↑ → *E(S)*↑):

  $E(S) \equiv - p_{\oplus} \log_2 p_{\oplus} - p_{\otimes} \log_2 p_{\otimes}$,

  where $p_{\oplus}$ ($p_{\otimes}$) is the proportion of positive (negative) examples in *S*.

  (<u>Note</u>: *E(S)* = 0 if *S* contains only positive or only negative examples
  $\leftrightarrow p_{\oplus} = 1, p_{\otimes} = 0, E(S) = -1 \cdot 0 - 0 \cdot \log_2 p_{\otimes} = 0$

  (<u>Note</u>: *E(S)* = 1 if *S* contains equal amount of positive and negative examples
  $\leftrightarrow p_{\oplus} = ½, p_{\otimes} = ½, E(S) = - ½ \cdot (-1) - ½ \cdot (-1) = 1$

  In the case that that the target attribute can take *n* values:

  $E(S) \equiv - \sum_I p_i \log_2 p_i), i = [1..n]$

  where $p_i$ is the proportion of examples in *S* having the target attribute value *i*.

- *Information Gain* – Reduction in *E(S)* caused by partitioning *S* according to attribute *A*

  $IG(S, A) = E(S) - \sum_{v \in values(A)} (|S_v| / |S|) E(S_v)$

  where *values(A)* are all possible values for attribute *A*, $S_v \in S$ contains all examples for which attribute *A* has the value *v*, and $|S_v|$ is the cardinality of set $S_v$.

# ID3 Algorithm – Example

1. For each attribute *A* of the training examples in set *S* calculate:
   $IG(S, A) = E(S) - \sum_{v \in values(A)} (|S_v| / |S|) E(S_v)$, $E(S_v) \equiv \sum_v - p_v \log_2 p_v$, $v = [1..n]$.
2. Select the attribute with the maximal *IG(S, A)* and use it as the root of the tree.
3. To decide the descendant node down each branch of the root (i.e., parent node), sort the training examples according to the value related to the current branch and repeat the process described in steps 1 and 2.

***Target concept***: *Play Tennis* (Mitchell's book, p. 59)

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 1 | 0 | sunny | hot | high | weak |
| 2 | 0 | sunny | hot | high | strong |
| … | … | … | … | … | … |
| 13 | 1 | overcast | hot | normal | weak |
| 14 | 0 | rain | mild | high | strong |

$IG(D, Outlook) = E(D) - 5/14\ E(D_{sunny}) - 4/14\ E(D_{overcast}) - 5/14\ E(D_{rain})$

# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 1 | 0 | sunny | hot | high | weak |
| 2 | 0 | sunny | hot | high | strong |
| 3 | 1 | overcast | hot | high | weak |
| 4 | 1 | rain | mild | high | weak |
| 5 | 1 | rain | cool | normal | weak |
| 6 | 0 | rain | cool | normal | strong |
| 7 | 1 | overcast | cool | normal | strong |
| 8 | 0 | sunny | mild | high | weak |
| 9 | 1 | sunny | cool | normal | weak |
| 10 | 1 | rain | mild | normal | weak |
| 11 | 1 | sunny | mild | normal | strong |
| 12 | 1 | overcast | mild | high | strong |
| 13 | 1 | overcast | hot | normal | weak |
| 14 | 0 | rain | mild | high | strong |

# ID3 Algorithm – Example

1. For each attribute $A$ of the training examples in set $S$ calculate:
   $IG(S, A) = E(S) - \sum_{v \in values(A)} (|S_v| / |S|) E(S_v)$, $E(S_v) \equiv \sum_v - p_v \log_2 p_v$, $v = [1..n]$.
2. Select the attribute with the maximal $IG(S, A)$ and use it as the root of the tree.
3. To decide the descendant node down each branch of the root (i.e., parent node), sort the training examples according to the value related to the current branch and repeat the process described in steps 1 and 2.

***Target concept***: *Play Tennis* (Mitchell's book, p. 59)

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| … | … | … | … | … | … |

$IG(D, Outlook) = E(D) - 5/14\ E(D_{sunny}) - 4/14\ E(D_{overcast}) - 5/14\ E(D_{rain})$
$= 0.940 - 0.357 \cdot 0.971 - 0 - 0.357 \cdot 0.971 = 0.246$

$IG(D, Temperature) = E(D) - 4/14\ E(D_{hot}) - 6/14\ E(D_{mild}) - 4/14\ E(D_{cool})$
$= 0.940 - 0.286 \cdot 1 - 0.429 \cdot 0.918 - 0.286 \cdot 0.811 = 0.029$

$IG(D, Humidity) = E(D) - 7/14\ E(D_{high}) - 7/14\ E(D_{normal}) = 0.940 - ½ \cdot 0.985 - ½ \cdot 0.591 = 0.151$

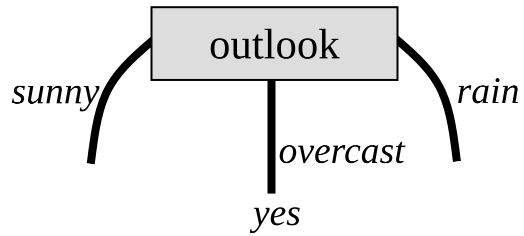
$IG(D, Wind) = E(D) - 8/14\ E(D_{weak}) - 6/14\ E(D_{strong}) = 0.940 - 0.571 \cdot 0.811 - 0.429 \cdot 1 = 0.048$

# ID3 Algorithm – Example

1. For each attribute *A* of the training examples in set *S* calculate:
   $IG(S, A) = E(S) - \sum_{v \in values(A)} (|S_v| / |S|)\, E(S_v),\ E(S_v) \equiv \sum_v - p_v \log_2 p_v,\ v = [1..n].$
2. Select the attribute with the maximal *IG(S, A)* and use it as the root of the tree.
3. To decide the descendant node down each branch of the root (i.e., parent node), sort the training examples according to the value related to the current branch and repeat the process described in steps 1 and 2.

***Target concept***: *Play Tennis* (Mitchell's book, p. 59)

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| … | … | … | … | … | … |



# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 1 | 0 | sunny | hot | high | weak |
| 2 | 0 | sunny | hot | high | strong |
| 3 | 1 | overcast | hot | high | weak |
| 4 | 1 | rain | mild | high | weak |
| 5 | 1 | rain | cool | normal | weak |
| 6 | 0 | rain | cool | normal | strong |
| 7 | 1 | overcast | cool | normal | strong |
| 8 | 0 | sunny | mild | high | weak |
| 9 | 1 | sunny | cool | normal | weak |
| 10 | 1 | rain | mild | normal | weak |
| 11 | 1 | sunny | mild | normal | strong |
| 12 | 1 | overcast | mild | high | strong |
| 13 | 1 | overcast | hot | normal | weak |
| 14 | 0 | rain | mild | high | strong |

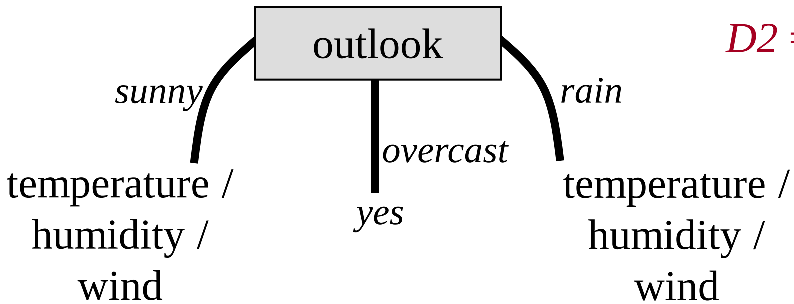
# ID3 Algorithm – Example

1. For each attribute *A* of the training examples in set *S* calculate:
   $IG(S, A) = E(S) - \sum_{v \in values(A)} (|S_v| / |S|)\, E(S_v),\ E(S_v) \equiv \sum_v - p_v \log_2 p_v,\ v = [1..n]$.
2. Select the attribute with the maximal *IG(S, A)* and use it as the root of the tree.
3. To decide the descendant node down each branch of the root (i.e., parent node), sort the training examples according to the value related to the current branch and repeat the process described in steps 1 and 2.

***Target concept***: *Play Tennis* (Mitchell's book, p. 59)

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| … | … | … | … | … | … |

*D1 = {d ∈ D | Outlook (d) = sunny}*




*D2 = {d ∈ D | Outlook (d) = rain}*

# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 1 | 0 | sunny | hot | high | weak |
| 2 | 0 | sunny | hot | high | strong |
| 8 | 0 | sunny | mild | high | weak |
| 9 | 1 | sunny | cool | normal | weak |
| 11 | 1 | sunny | mild | normal | strong |
| 4 | 1 | rain | mild | high | weak |
| 5 | 1 | rain | cool | normal | weak |
| 6 | 0 | rain | cool | normal | strong |
| 10 | 1 | rain | mild | normal | weak |
| 14 | 0 | rain | mild | high | strong |
| 3 | 1 | overcast | hot | high | weak |
| 7 | 1 | overcast | cool | normal | strong |
| 12 | 1 | overcast | mild | high | strong |
| 13 | 1 | overcast | hot | normal | weak |

***D1***

***D2***

# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 1 | 0 | sunny | hot | high | weak |
| 2 | 0 | sunny | hot | high | strong |
| 8 | 0 | sunny | mild | high | weak |
| 9 | 1 | sunny | cool | normal | weak |
| 11 | 1 | sunny | mild | normal | strong |
| … | … | … | … | … | … |

Rows 1, 2, 8, 9, 11 → ***D1***

$E(D1) = \text{abs}\,(-2/5 \log_2 2/5 - 3/5 \log_2 3/5) = 0.971$

$IG(D1, Temperature) = E(D1) - 2/5\, E(D1_{hot}) - 2/5\, E(D1_{mild}) - 1/5\, E(D1_{cool})$
$= 0.971 - 0.4 \cdot 0 - 0.4 \cdot 1 - 0.4 \cdot 0 = 0.571$

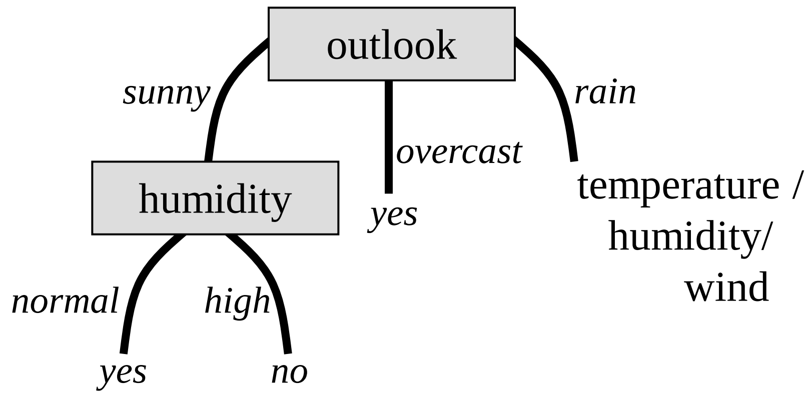
$IG(D1, Humidity) = E(D1) - 3/5\, E(D1_{high}) - 2/5\, E(D1_{normal})$
$= 0.971 - 0.6 \cdot 0 - 0.4 \cdot 0 = 0.971$

$IG(D1, Wind) = E(D1) - 3/5\, E(D1_{weak}) - 2/5\, E(D1_{strong})$
$= 0.971 - 0.6 \cdot 0.918 - 0.4 \cdot 1 = 0.02$

# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 1 | 0 | sunny | hot | high | weak |
| 2 | 0 | sunny | hot | high | strong |
| 8 | 0 | sunny | mild | high | weak |
| 9 | 1 | sunny | cool | normal | weak |
| 11 | 1 | sunny | mild | normal | strong |
| … | … | … | … | … | … |

***D1***



# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 4 | 1 | rain | mild | high | weak |
| 5 | 1 | rain | cool | normal | weak |
| 6 | 0 | rain | cool | normal | strong |
| 10 | 1 | rain | mild | normal | weak |
| 14 | 0 | rain | mild | high | strong |
| … | … | … | … | … | … |

**D2**

$E(D2) = \text{abs}\,(-\,3/5 \log_2 3/5 - 2/5 \log_2 2/5) = 0.971$

$IG(D2, Temperature) = E(D2) - 0/5\, E(D2_{hot}) - 3/5\, E(D2_{mild}) - 2/5\, E(D2_{cool})$
$= 0.971 - 0 - 0.6 \cdot 0.918 - 0.4 \cdot 1 = 0.02$

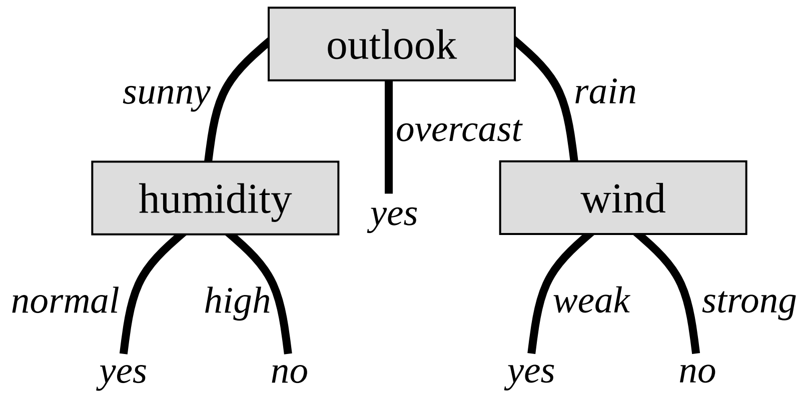
$IG(D2, Humidity) = E(D2) - 2/5\, E(D2_{high}) - 3/5\, E(D2_{normal})$
$= 0.971 - 0.4 \cdot 1 - 0.6 \cdot 0.918 = 0.02$

$IG(D2, Wind) = E(D2) - 3/5\, E(D2_{weak}) - 2/5\, E(D2_{strong})$
$= 0.971 - 0.6 \cdot 0 - 0.4 \cdot 0 = 0.971$

# ID3 Algorithm – Example

| | *PlayTennis(d)* | *outlook* | *temperature* | *humidity* | *wind* |
|---|---|---|---|---|---|
| 4 | 1 | rain | mild | high | weak |
| 5 | 1 | rain | cool | normal | weak |
| 6 | 0 | rain | cool | normal | strong |
| 10 | 1 | rain | mild | normal | weak |
| 14 | 0 | rain | mild | high | strong |
| … | … | … | … | … | … |

***D2***



# ID3 Algorithm – Advantages & Disadvantages

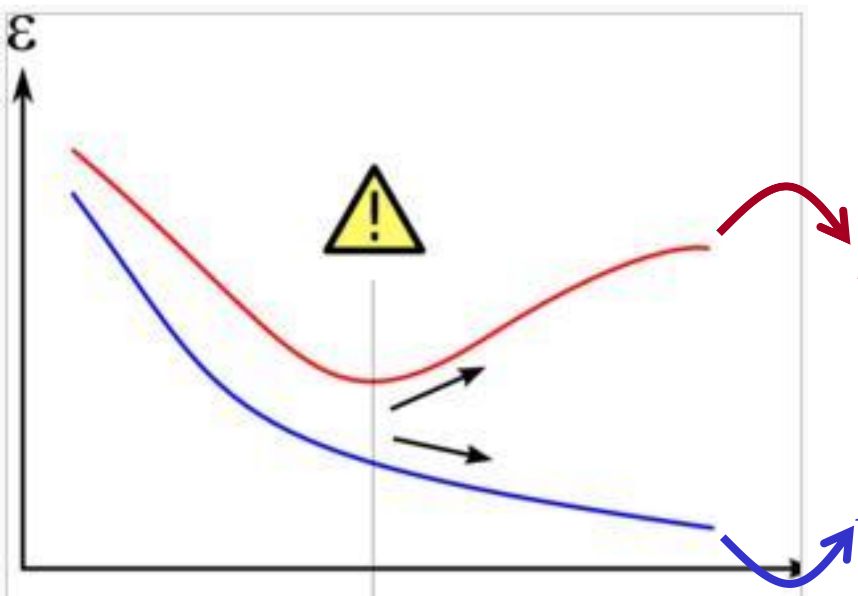
- Advantages of ID3 algorithm:
  1. Every discrete classification function can be represented by a decision tree → it cannot happen that ID3 will search an incomplete hypothesis space.
  2. Instead of making decisions based on individual training examples (as is the case by Find-S and Candidate-Elimination algorithms), ID3 uses statistical properties of all examples (information gain) → resulting search is much less sensitive to errors in individual training examples.

- Disadvantages of ID3 algorithm:
  1. ID3 determines a single hypothesis, not a space of consistent hypotheses (as is the case by Candidate-Elimination algorithm) → ID3 cannot determine how many different decision trees are consistent with the available training data.
  2. ID3 grows the tree to perfectly classify the training examples without performing a backtracking in its search → ID3 may overfit the training data and converge to locally optimal solution that is not globally optimal.

# The Problem of Overfitting

- Def (*Mitchell 1997*):

  *Given a hypothesis space H, h $\in$ H overfits the training data if $\exists$h' $\in$ H such that h has smaller error over the training examples, but h' has smaller error than h over the entire distribution of instances.*




***performance of h $\in$ H on testing data***

***performance of h $\in$ H on training data***

# The Problem of Overfitting

- Ways to avoid overfitting:
  1. Stop the training process before the learner reaches the point where it perfectly classifies the training data.
  2. Apply backtracking in the search for the optimal hypothesis. In the case of Decision Tree Learning, backtracking process is referred to as 'post-pruning of the overfitted tree'.

- Ways to determine the correctness of the learner's performance:
  1. Use two different sets of examples: training set and validation set.
  2. Use all examples for training, but apply a statistical test to estimate whether a further training will produce a statistically significant improvement of the learner's performance. In the case of Decision Tree Learning, the statistical test should estimate whether expanding / pruning a particular node will result in a statistically significant improvement of the performance.
  3. Combine 1. and 2.

# Decision Tree Learning – Exam Questions

- Tom Mitchell's book –chapter 3
- Relevant exercises from chapter 3:   3.1, 3.2, 3.3, 3.4

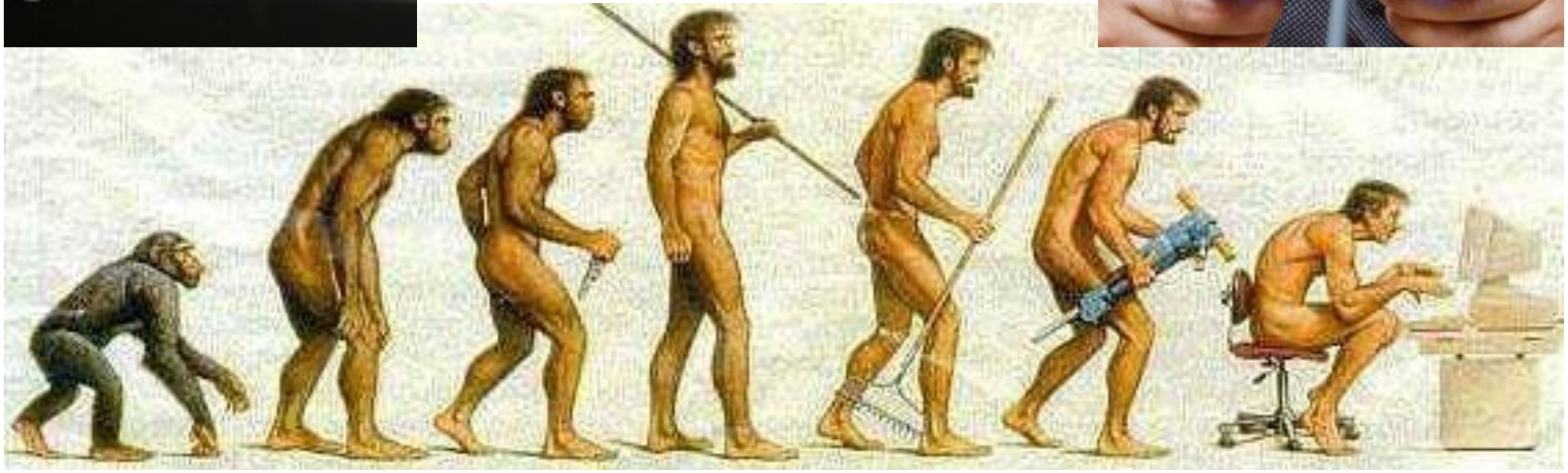
# Course 395: Machine Learning – Lectures

- Lecture 1-2: Concept Learning *(M. Pantic)*
- Lecture 3-4: Decision Trees & CBC Intro *(M. Pantic & S. Petridis)*
- Lecture 5-6: Evaluating Hypotheses *(S. Petridis)*
- Lecture 7-8: Artificial Neural Networks I *(S. Petridis)*
- Lecture 9-10: Artificial Neural Networks II *(S. Petridis)*
- Lecture 11-12: Instance Based Learning *(M. Pantic)*
- Lecture 13-14: Genetic Algorithms *(M. Pantic)*

# Decision Trees & CBC Intro – Lecture Overview

- Problem Representation using a Decision Tree
- ID3 algorithm
- The problem of overfitting
- Research on affective computing, natural HCI, and ambient intelligence
- Facial expressions and Emotions
- Overview of the CBC
- Group forming

# Importance of Computing Technology

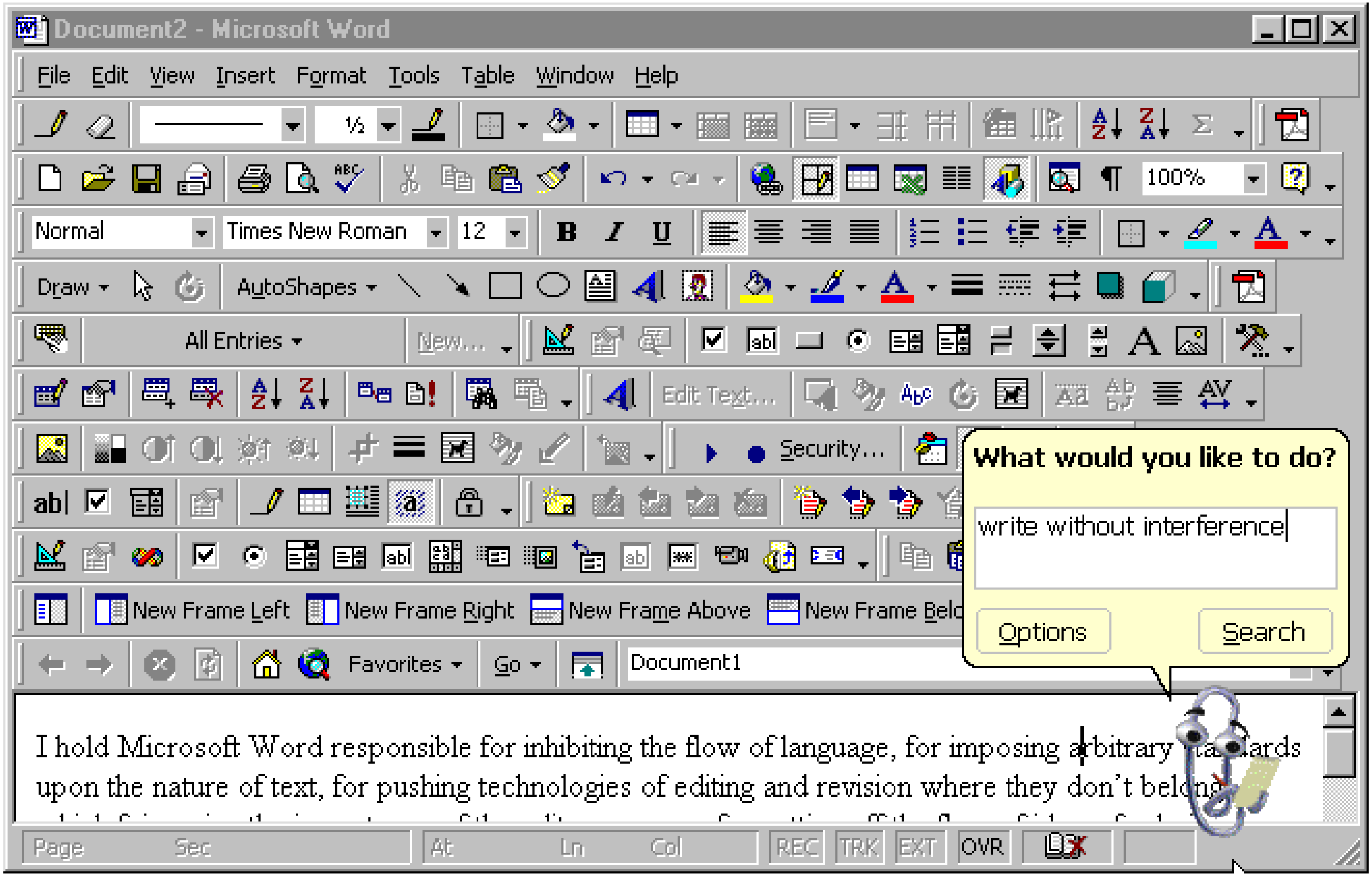



# Current Human-Computer Interfaces

5.1 hours per week wasted trying to use computers. More time is wasted in front of computers than on highways. The frustration and anxiety of users is growing, and the number of nonusers is still high. Low-cost hardware, software, and networking will bring in many new users, but interface and information design breakthroughs are necessary to achieve higher levels of success.

~Ben Shneiderman

# Current Human-Computer Interfaces

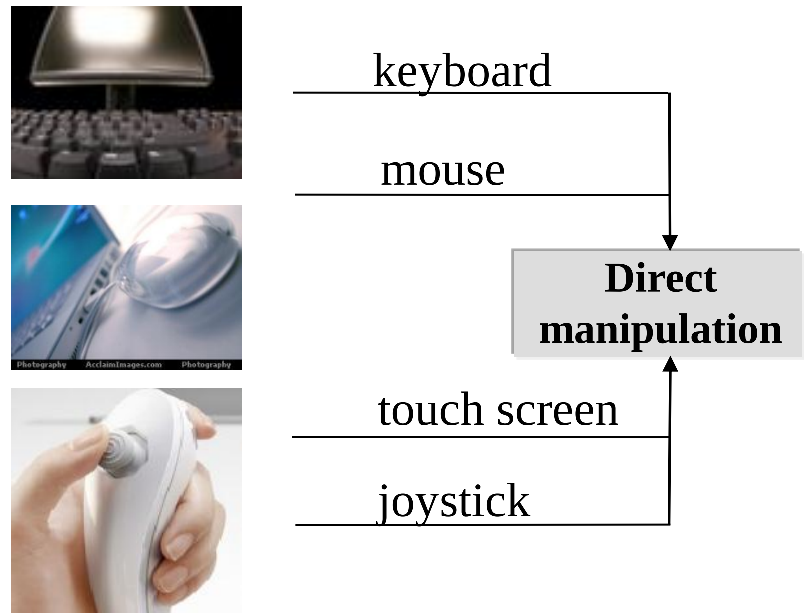


# Current Human-Computer Interfaces

***Human-Human Interaction:***

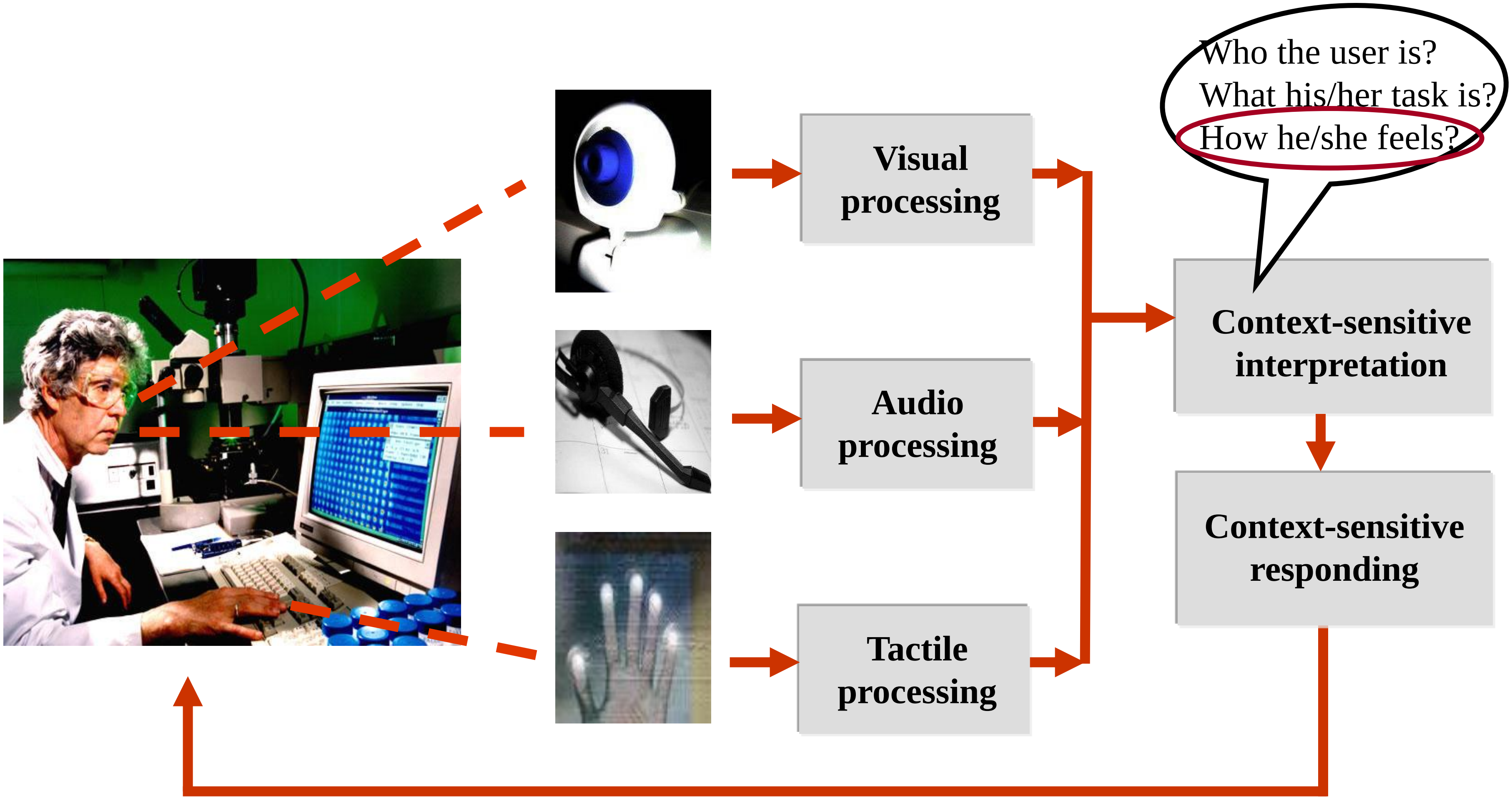
Simultaneous employment of sight, sound and touch

***Human-Computer Interaction:***

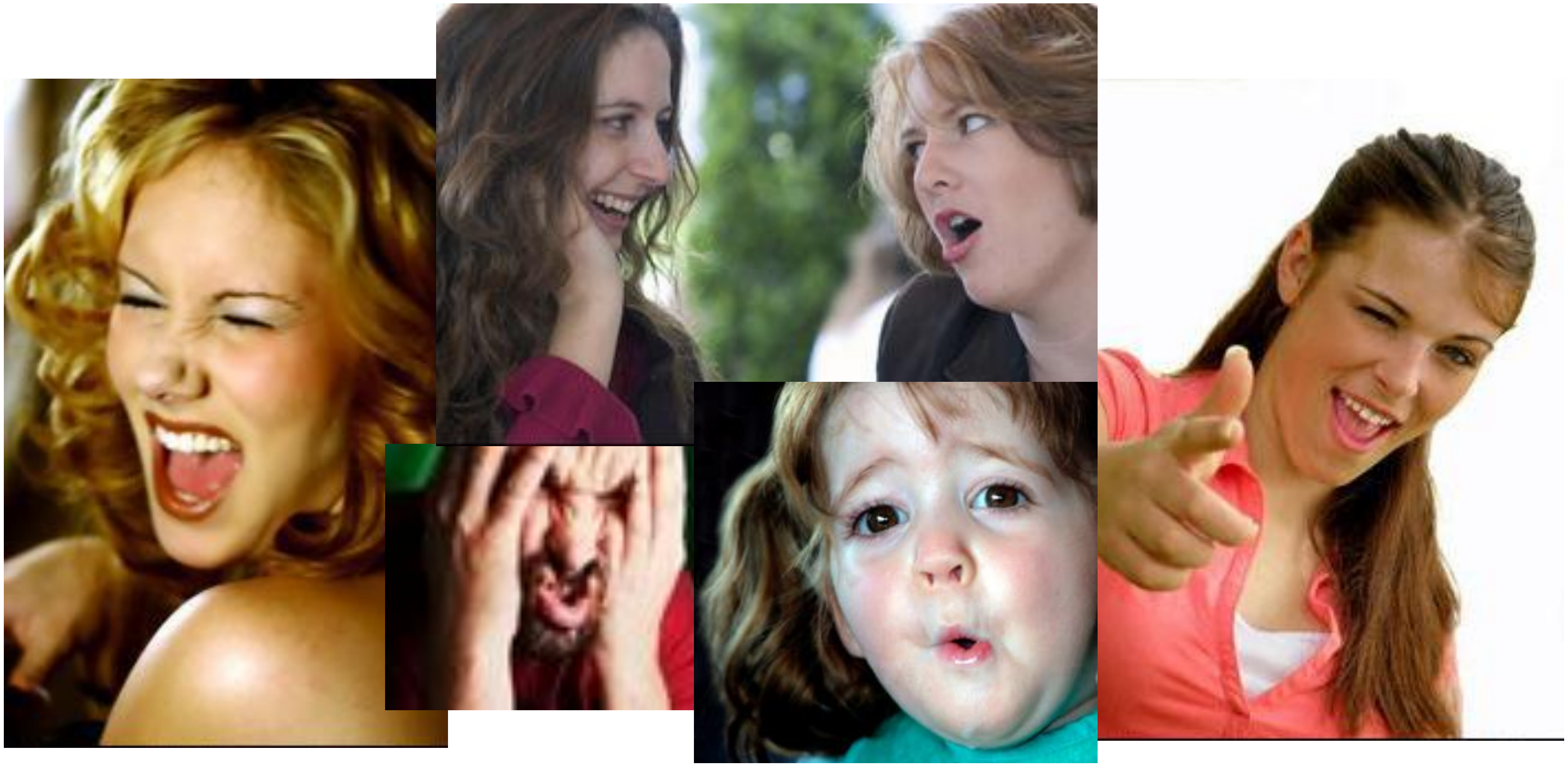



Current HCI-designs are single-modal and context-insensitive

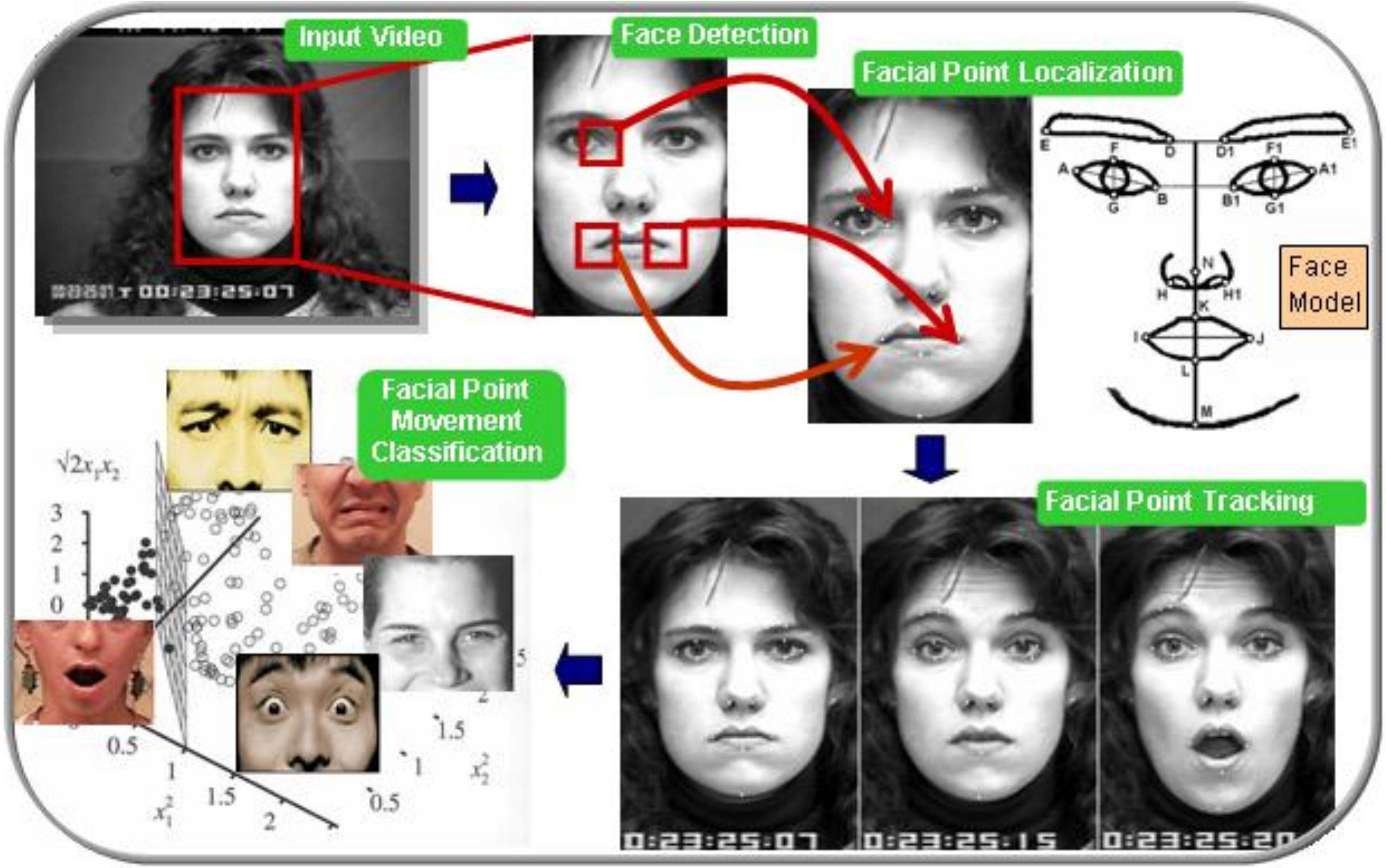
# Future Human-Computer Interfaces



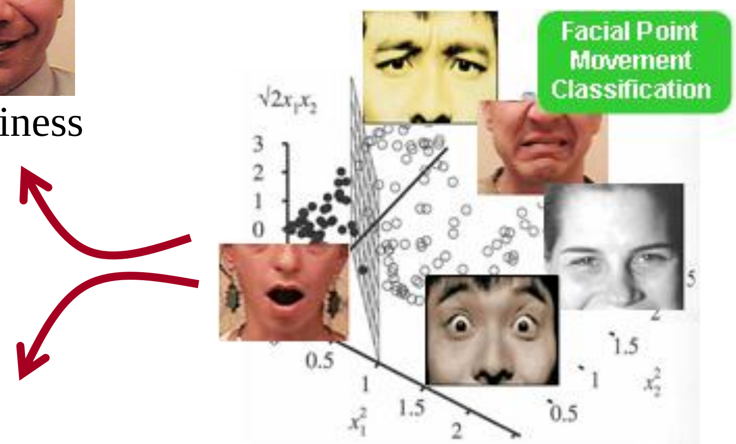
# Face for Interfaces



# Automatic Facial Expression Analysis



# Automatic Facial Expression Analysis


# Facial Muscle Actions (Action Units - AUs)

# CBC – Emotion Recognition

Anger Surprise Sadness Disgust Fear Happiness

- Prototypic facial expressions of the **six basic emotions** were introduced by Charles Darwin (1872) and elaborated by Ekman
- These prototypic facial expressions can be described in terms of AUs (e.g., surprise $\leftrightarrow$ AU1 + AU2 + AU5 + AU26 / AU27)

# CBC – Emotion Recognition

| Emotion | AUs | Emotion | AUs |
|---|---|---|---|
| Happy | {12} | Fear | {1,2,4} |
| | {6,12} | | {1,2,4,5,20, |
| Sadness | {1,4} | | 25∥26∥27} |
| | {1,4,11∥15} | | {1,2,4,5,25∥26∥27} |
| | {1,4,15,17} | | {1,2,4,5} |
| | {6,15} | | {1,2,5,25∥26∥27} |
| | {11,17} | | {5,20,25∥26∥27} |
| | {1} | | {5,20} |
| Surprise | {1,2,5,26∥27} | | {20} |
| | {1,2,5} | Anger | {4,5,7,10,22,23,25∥26} |
| | {1,2,26∥27} | | {4,5,7,10,23,25∥26} |
| | {5,26∥27} | | {4,5,7,17,23∥24} |
| Disgust | {9∥10,17} | | {4,5,7,23∥24} |
| | {9∥10,16,25∥26} | | {4,5∥7} |
| | {9∥10} | | {17,24} |

*V*: *AUs* → *basic-emotions*

$V'$: $\langle a_1, \ldots, a_{45} \rangle \rightarrow [1..6]$

*learning algorithms*:

- decision trees (ID3)
- Neural Networks
- Case-based Reasoning

*evaluating developed systems*:

- t-test

# Decision Trees & CBC Intro – Lecture Overview

- Problem Representation using a Decision Tree
- ID3 algorithm
- The problem of overfitting
- Research on affective computing, natural HCI, and ambient intelligence
- Facial expressions and Emotions

➢ Overview of the CBC

➢ Group forming

# CBC - Goal

- Hands-on experience in implementing and testing basic machine learning techniques

- Work with other team members

➢ Both your group and individual effort/performance are graded!

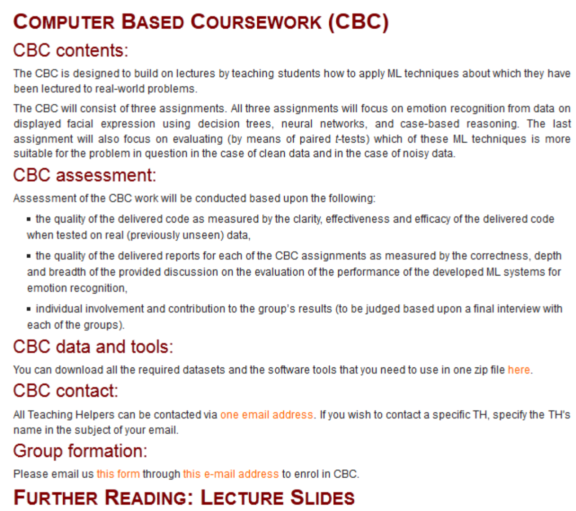
➢ CBC = Computer-Based Coursework

# Group Forming

- Students will be divided in groups of 4
- Simply fill in the **excel form** with the following information  (for each group member):

  - First Name, Last Name, Student login, Email, Course, CID
- You can find the excel form on http://ibug.doc.ic.ac.uk/courses,

  http://ibug.doc.ic.ac.uk/media/uploads/documents/ml-cbc-groupform.xls



### Computer Based Coursework (CBC)

#### CBC contents:

The CBC is designed to build on lectures by teaching students how to apply ML techniques about which they have been lectured to real-world problems.

The CBC will consist of three assignments. All three assignments will focus on emotion recognition from data on displayed facial expression using decision trees, neural networks, and case-based reasoning. The last assignment will also focus on evaluating (by means of paired *t*-tests) which of these ML techniques is more suitable for the problem in question in the case of clean data and in the case of noisy data.

#### CBC assessment:

Assessment of the CBC work will be conducted based upon the following:

- the quality of the delivered code as measured by the clarity, effectiveness and efficacy of the delivered code when tested on real (previously unseen) data,
- the quality of the delivered reports for each of the CBC assignments as measured by the correctness, depth and breadth of the provided discussion on the evaluation of the performance of the developed ML systems for emotion recognition,
- individual involvement and contribution to the group's results (to be judged based upon a final interview with each of the groups).

#### CBC data and tools:

You can download all the required datasets and the software tools that you need to use in one zip file here.

#### CBC contact:

All Teaching Helpers can be contacted via one email address. If you wish to contact a specific TH, specify the TH's name in the subject of your email.

#### Group formation:

Please email us this form through this e-mail address to enrol in CBC.

### Further Reading: Lecture Slides

# Group Forming

- Email the **excel form** by **Tuesday October 21st**
  machinelearningtas@gmail.com

- You will be assigned a TH who will email you informing you about your group number and confirming the members of your group by Thursday/Friday October 23rd / 24th

# Group Forming

- 4 members per group, this is a hard limit!
  - In other words, groups of 5, 6, 7, etc are not allowed!!
- If you cannot form a team with 4 members then just email us the above information and we will assign you to a team.
  - Case 1 member: We will add you to a group of 2-3 members
  - Case 2 members: 1-2 more members will be added to your group
  - Case 3 members: 1 member may be added to your group
- We sometimes allow groups of 3
- Once the groups are formed you cannot change groups

# Tutorial Helpers

- ***A Tutorial Helper (TH) will be assigned to each group***

*- James Booth*

*- Siyang Cheng*

*- Stefanos Eleftheriadis*

*- Christos Georgakis*

*- Jean Kossaifi*

*- Symeon Nikitidis*

*- Ioannis Panagakis*

*- Stavros Petridis*

*- Ognjen Rudovic*

*- Jie Shen*

*- Christos Sagonas*

*- Patrick Snape*

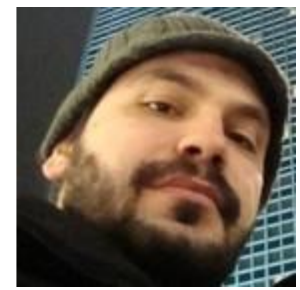
*- Lazaros Zafeiriou*

[*http://ibug.doc.ic.ac.uk/people*](http://ibug.doc.ic.ac.uk/people)

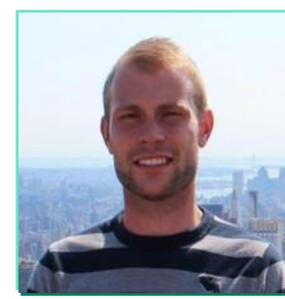










# Communication

- Via the website: http://ibug.doc.ic.ac.uk/courses/machine-learning-course-395/
  - CBC Manual (Section Course Material)
  - Provided Matlab files, datasets (Section Data and Tools)
  - Tutorials

## Computer Based Coursework (CBC)

### CBC contents:

The CBC is designed to build on lectures by teaching students how to apply ML techniques about which they have been lectured to real-world problems.

The CBC will consist of three assignments. All three assignments will focus on emotion recognition from data on displayed facial expression using decision trees, neural networks, and case-based reasoning. The last assignment will also focus on evaluating (by means of paired *t*-tests) which of these ML techniques is more suitable for the problem in question in the case of clean data and in the case of noisy data.

### CBC assessment:

Assessment of the CBC work will be conducted based upon the following:

- the quality of the delivered code as measured by the clarity, effectiveness and efficacy of the delivered code when tested on real (previously unseen) data,
- the quality of the delivered reports for each of the CBC assignments as measured by the correctness, depth and breadth of the provided discussion on the evaluation of the performance of the developed ML systems for emotion recognition,
- individual involvement and contribution to the group's results (to be judged based upon a final interview with each of the groups).

### CBC data and tools:

You can download all the required datasets and the software tools that you need to use in one zip file here.

### CBC contact:

All Teaching Helpers can be contacted via one email address. If you wish to contact a specific TH, specify the TH's name in the subject of your email.

### Group formation:

Please email us this form through this e-mail address to enrol in CBC.

## Further Reading: Lecture Slides

# Communication

- Via email: machinelearningtas@gmail.com

ALWAYS put your group number in the subject line

# CBC – Organisation

- Each group must hand in a report of ~3 pages (excluding figures) per assignment, including discussion on implementation and answers to questions posed in the manual.
- ONE report per group
- Each group must hand in the code they implemented for each assignment.
- Hand in the code and the reports via CATE.

# CBC – Assignment hand in

- Hand in via CATE
    - One group leader per group
    - Each and every group member individually has to confirm that s(he) is part of that particular group, for each and every assignment submission (under the pre-determined group leader) before each assignment submission deadline.

# CBC – Report + Code marking

- The THs will mark your report and provide feedback
- The THs will test the implemented algorithms using a separate test set (not available to the students).
- We will inform you about the performance of your algorithm on our test set.
- We just want to check that your code runs and your classifier has been properly trained, i.e., it can generalise to unknown data

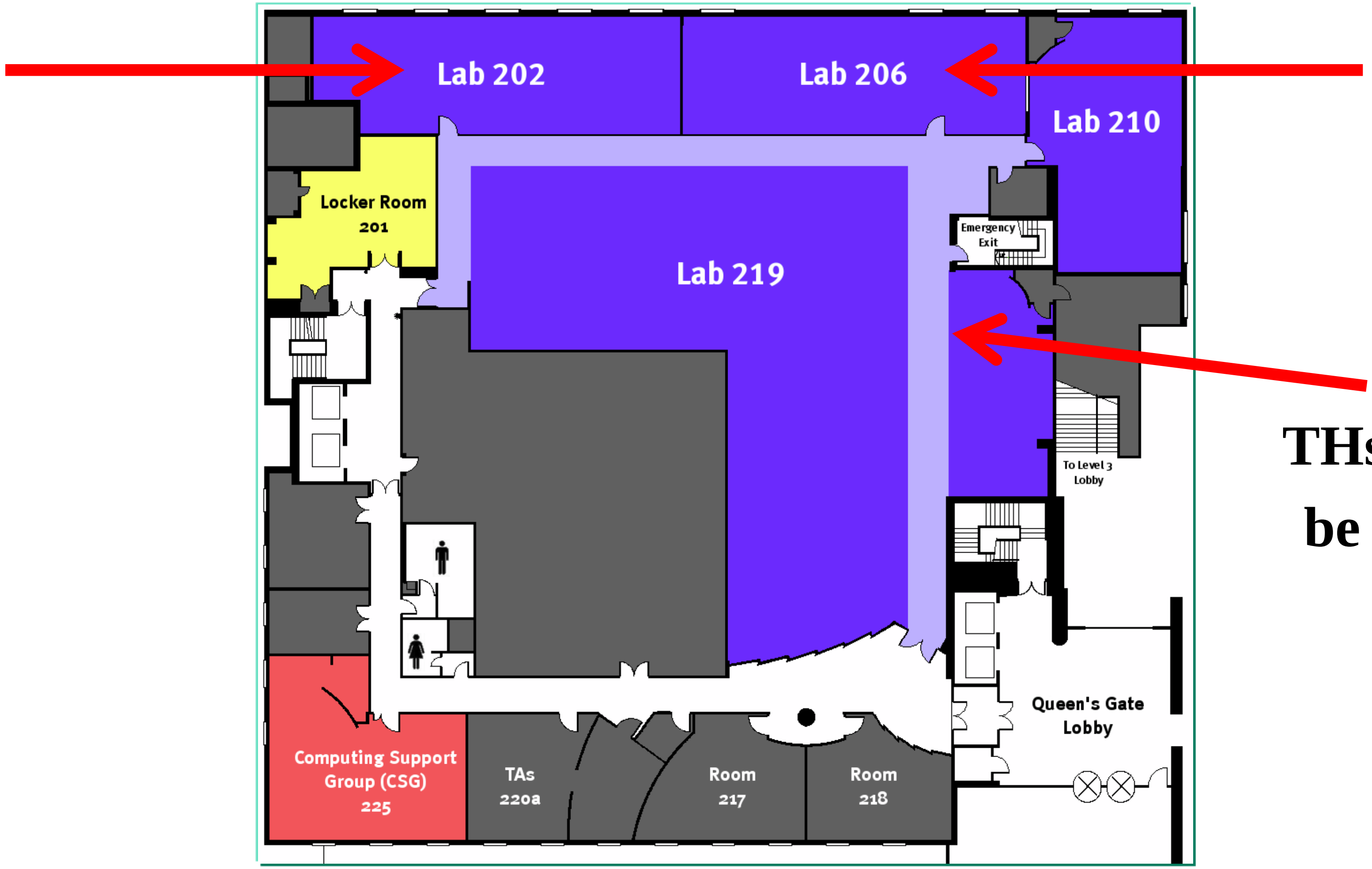
# CBC – Interviews

- Each group will have an interview of ~20min with two THs after the completion of each assignment. ALL members must be present.
- Interviews will take place in pairs, i.e., first two members (~10min) and then the other two (~10)
- Questions will be similar
- We just want to check that you yourselves did all the work and you are familiar with the basic concepts
- All members should know everything that is included in the report

# Lab Schedule

**Assisted Labs** (THs present to answer questions), starting on October 22th continuing until December 4th

- Every Wednesday 9:00-12:00, labs 202-206
- Every Thursday 16:00 – 18:00, lab 219




THHs will be here

# Deadlines

- Assignment 1: optional (no hand in required)
- Assignment 2: November 3rd (Monday) – 10am
- Assignment 3: November 17th (Monday) – 10am
- Assignment 4: November 28th (Friday) – midnight

# Interviews

- Week 5 (Nov 3 – 7) – Assignment 2
  Wednesday 5/11, Thursday 6/11

- Week 7 (Nov 17 – Nov 21) – Assignment 3
  Wednesday 19/11, Thursday 20/11

- Week 9 (Dec 1 – 5) – Assignment 4
  Wednesday 3/12, Thursday 4/12

- You will receive your interviews timetable after the submission of each assignment.
- If you are not available then let us know asap, the interview cannot take place without all the members being present.
- We can only return the marked courseworks after the interviews take place.

# CBC – Grading

- Grading will be done exclusively by the lecturer, taking into account the THs' recommendations.
- Every group member is expected to have sufficient contribution to the implementation of every assignment. Personal contribution will be evaluated during the interviews after each assignment.
- Plagiarism is not allowed!
  Involved groups will be instantly eliminated.

# Assignment Grading


| Report Content | Code | Report Quality |
|---|---|---|
| ***75%*** | ***15%*** | ***10%*** |

***Group_grade = 0.75*report_content + 0.15*code + 0.1*report_quality***

- ***Code Mark:*** Performance on unknown Test set + 15 (Max Performance in 2013 was 85%)

- ***Make sure your code runs. If not you lose 30% of the code mark.***

# CBC Grade



***Group Grade***

***80%***

***Personal Grade***

***20%***

***Personal_grade = interview grade***

***Assignment_grade = 0.8*group_grade + 0.2*personal grade***

# Assignment Grading


***Grade1***


***Grade2***


***Grade3***

***CBC_grade = Average(Grade1, Grade2, Grade3)***

# Machine Learning Grade



***CBC Grade***

***33.3%***

***Exam Grade***

***66.7%***

- ***CBC accounts for 33.3% of the final grade for the Machine Learning Course. In other words, final grade = 0.667*exam_grade + 0.333*CBC_grade.***

# CBC – Tools

- Training data and useful functions are provided via the course website in a separate .rar file.

- Implementation in MATLAB
  - MATLAB basics (matrices, vectors, functions, input/output) (Assignments 2,4)
  - ANN Toolbox (Assignment 3)

Students are strongly advised to use the MATLAB help files!

# Assignment 1 : MATLAB Exercises

- Optional (no hand in required)

- A brief introduction to some basic concepts of MATLAB (that are needed in Assignments 2-4) without assessing students' acquisition, application and integration of this basic knowledge.

- The students, are strongly encouraged to go through all the material, experiment with various functions, and use the MATLAB help files extensively (accessible via the main MATLAB window).

# CBC – Emotion Recognition

Anger Surprise Sadness Disgust Fear Happiness

- Prototypic facial expressions of the **six basic emotions** were introduced by Charles Darwin (1872) and elaborated by Ekman
- These prototypic facial expressions can be described in terms of AUs (e.g., surprise ↔ AU1 + AU2 + AU5 + AU26 / AU27)

# Facial Muscle Actions (Action Units - AUs)

# CBC – Emotion Recognition

<table>
<tr><th>Emotion</th><th>AUs</th><th>Emotion</th><th>AUs</th></tr>
<tr><td>Happy</td><td>{12}</td><td>Fear</td><td>{1,2,4}</td></tr>
<tr><td></td><td>{6,12}</td><td></td><td>{1,2,4,5,20,</td></tr>
<tr><td>Sadness</td><td>{1,4}</td><td></td><td>25||26||27}</td></tr>
<tr><td></td><td>{1,4,11||15}</td><td></td><td>{1,2,4,5,25||26||27}</td></tr>
<tr><td></td><td>{1,4,15,17}</td><td></td><td>{1,2,4,5}</td></tr>
<tr><td></td><td>{6,15}</td><td></td><td>{1,2,5,25||26||27}</td></tr>
<tr><td></td><td>{11,17}</td><td></td><td>{5,20,25||26||27}</td></tr>
<tr><td></td><td>{1}</td><td></td><td>{5,20}</td></tr>
<tr><td>Surprise</td><td>{1,2,5,26||27}</td><td></td><td>{20}</td></tr>
<tr><td></td><td>{1,2,5}</td><td>Anger</td><td>{4,5,7,10,22,23,25||26}</td></tr>
<tr><td></td><td>{1,2,26||27}</td><td></td><td>{4,5,7,10,23,25||26}</td></tr>
<tr><td></td><td>{5,26||27}</td><td></td><td>{4,5,7,17,23||24}</td></tr>
<tr><td>Disgust</td><td>{9||10,17}</td><td></td><td>{4,5,7,23||24}</td></tr>
<tr><td></td><td>{9||10,16,25||26}</td><td></td><td>{4,5||7}</td></tr>
<tr><td></td><td>{9||10}</td><td></td><td>{17,24}</td></tr>
</table>

*V*: *AUs* → *basic*-*emotions*

$V'$: ‹$a_1$, …, $a_{45}$› → [1..6]

*learning algorithms*:

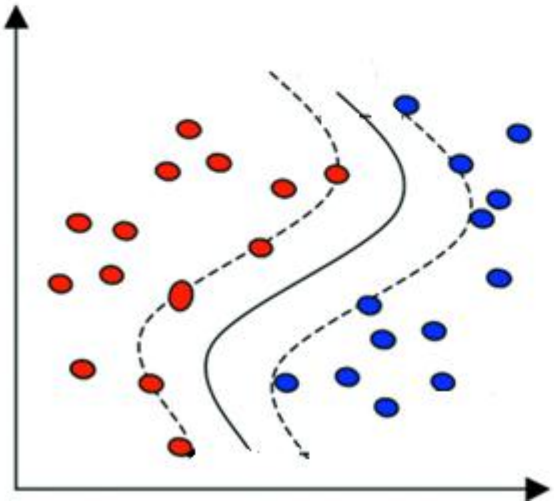
- decision trees (ID3)
- Neural Networks
- Case-based Reasoning

*evaluating developed systems*:

- t-test

# Assignments 2-4 : Overview

- Classification Problem
  - Inputs: *x (AU vectors)*
  - Desired Output: *y (Emotion label)*



- Use *x* and *y* to train your learning algorithms to discriminate between the 6 classes (emotions)

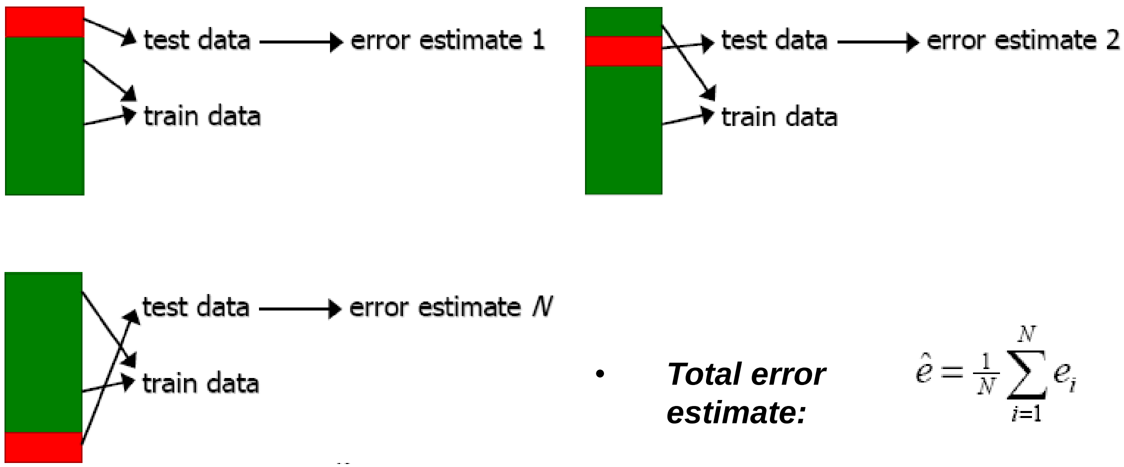
- Evaluate your algorithms using 10-fold cross validation

- Write a function $y^{pred} = testLearner(T, x)$, which takes your trained learners T and the features *x* and produces a vector of label predictions $y^{pred}$

# Training – Validation – Test Sets

- Training Set: Used to train the classifiers
- Validation Set: Used to optimise the parameters of the classifiers
  - e.g. number of hidden neurons in neural networks
- Test Set: Used to measure the performance of the classifier

# N-fold Cross validation




- ***Total error estimate:*** $\hat{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$

- Initial dataset is partitioned in N folds
- Training + **Validation** set: N - 1 folds, Test set: 1 fold
- This process is repeated N times → N error estimates
- Final error: Average of the N error estimates

# Assignment 2 : Decision Trees

- Implement and train a decision tree learning algorithm
- Evaluate your trees using 10-fold cross validation
- Write a function $y^{pred}$ = *testTrees(T, x)*, which takes your trained trees T and the features *x* and produces a vector of label predictions $y^{pred}$
- Theoretical / Implementation questions

# Assignment 3 : Artificial Neural Networks

- Use the Neural Networks toolbox (MATLAB built-in) to train your networks

- Evaluate your networks using 10-fold cross validation

- Write a function: $y^{pred}$ = testANN(N, x), which takes your trained networks N and produces a vector of label predictions $y^{pred}$ .

- Theoretical / Implementation questions

# Assignment 4 Part A : Case Based Reasoning

- Implement and train CBR system
- Evaluate your system using 10-fold cross validation
- Theoretical / Implementation questions

# Assignment 4 Part B : T-test

- Evaluate if the performance of the algorithms implemented so far differ significantly.
- Use the results that were previously obtained from cross validation!
- Theoretical / Implementation questions

# CBC – Tips

- Work together!!! Divide the work, but you should all know how you have implemented your algorithms, why you took some specific decision, etc...
- Make sure your code runs!!
- **Make sure you use the new version of the manual (updated today!)**
- All members should know everything that is included in the report
- If you have any complaints (marking, THs etc) contact the course support leader

# Group Forming

- Students will be divided in groups of 4 students
- Simply fill in the **excel form** with the following information (for each group member):

  -Student login, email, First Name, Last Name, Course, CID

- You can find the excel form on http://ibug.doc.ic.ac.uk/courses, Section: Group Forming

  http://ibug.doc.ic.ac.uk/media/uploads/documents/ml-cbc-groupform.xls

- Email the **excel form** by **Tuesday 21st October** machinelearningtas@gmail.com
- If you cannot form a team with 4 members then just email us the above information and we will assign you to a team.