Course 395: Machine Learning

- Lecturers: Maja Pantic (maja@doc.ic.ac.uk)
  Stephen Muggleton (shm@doc.ic.ac.uk)

- Goal (Lectures): To present basic theoretical concepts and key algorithms that form the core of machine learning

- Goal (CBC): To enable hands-on experience with implementing machine learning algorithms using Matlab

  Manual for completing the CBC
  Syllabus on CBR
  Notes on Inductive Logic Programming

- More Info: [https://www.ibug.doc.ic.ac.uk/courses](https://www.ibug.doc.ic.ac.uk/courses)
Course 395: Machine Learning – Lectures

• Lecture 1-2: Concept Learning (M. Pantic)
• Lecture 3-4: Decision Trees & CBC Intro (M. Pantic)
• Lecture 5-6: Artificial Neural Networks (THs)
• Lecture 7-8: Instance Based Learning (M. Pantic)
• Lecture 9-10: Genetic Algorithms (M. Pantic)
• Lecture 11-12: Evaluating Hypotheses (THs)
• Lecture 13-14: Guest Lectures on ML Applications
• Lecture 15-16: Inductive Logic Programming (S. Muggleton)
• Lecture 17-18: Inductive Logic Programming (S. Muggleton)
Course 395: Machine Learning – Exam Material

- Lecture 1-2: Concept Learning (*Mitchell*: Ch.1, Ch.2)
- Lecture 3-4: Decision Trees & CBC Intro (*Mitchell*: Ch.3)
- Lecture 5-6: Artificial Neural Networks (*Mitchell*: Ch.4)
- Lecture 7-8: Instance Based Learning (Syllabus, *Mitchell*: Ch.8)
- Lecture 9-10: Genetic Algorithms (*Mitchell*: Ch.9)
- Lecture 11-12: Evaluating Hypotheses (*Mitchell*: Ch.5)
- Lecture 13-14: not examinable
- Lecture 15-16: Inductive Logic Programming (*Notes*)
- Lecture 17-18: Inductive Logic Programming (*Notes*)
Course 395: Machine Learning - CBC

- Lecture 1-2: Concept Learning
- Lecture 3-4: Decision Trees & CBC Intro
- Lecture 5-6: Artificial Neural Networks
- Lecture 7-8: Instance Based Learning
- Lecture 9-10: Genetic Algorithms
- Lecture 11-12: Evaluating Hypotheses
- Lecture 13-14: Guest Lectures on ML Applications
- Lecture 15-16: Inductive Logic Programming
- Lecture 17-18: Inductive Logic Programming
Course 395: Machine Learning

NOTE

CBC accounts for 33% of the final grade for the Machine Learning Exam.

final grade = 0.66*exam_grade + 0.33*CBC_grade
Course 395: Machine Learning - CBC

• Lecture 1-2: Concept Learning

 Lecture 3-4: Decision Trees & CBC Intro

 Lecture 5-6: Artificial Neural Networks

 Lecture 7-8: Instance Based Learning

• Lecture 9-10: Genetic Algorithms

 Lecture 11-12: Evaluating Hypotheses

• Lecture 13-14: Guest Lectures on ML Applications

• Lecture 15-16: Inductive Logic Programming

• Lecture 17-18: Inductive Logic Programming
Course 395: Machine Learning – Lectures

- Lecture 1-2: Concept Learning (*M. Pantic*)
- Lecture 3-4: Decision Trees & CBC Intro (*M. Pantic*)
- Lecture 5-6: Artificial Neural Networks (*THs*)
- Lecture 7-8: Instance Based Learning (*M. Pantic*)
- Lecture 9-10: Genetic Algorithms (*M. Pantic*)
- Lecture 11-12: Evaluating Hypotheses (*THs*)
- Lecture 13-14: Guest Lectures on ML Applications
- Lecture 15-16: Inductive Logic Programming (*S. Muggleton*)
- Lecture 17-18: Inductive Logic Programming (*S. Muggleton*)
Concept Learning – Lecture Overview

• Why machine learning?
• Well-posed learning problems
• Designing a machine learning system
• Concept learning task
• Concept learning as Search
• Find-S algorithm
• Candidate-Elimination algorithm
Machine Learning

- Learning ↔ Intelligence
  (Def: *Intelligence is the ability to learn and use concepts to solve problems.*)

- Machine Learning ↔ Artificial Intelligence
  - Def: *AI is the science of making machines do things that require intelligence if done by men* (Minsky 1986)
  - Def: *Machine Learning is an area of AI concerned with development of techniques which allow machines to learn*

- Why Machine Learning? ↔ Why Artificial Intelligence?
Machine Learning

In-Vehicle Computing
complete car-PC system
Machine Learning

• Learning ↔ Intelligence
  (Def: Intelligence is the ability to learn and use concepts to solve problems.)

• Machine Learning ↔ Artificial Intelligence
  – Def: AI is the science of making machines do things that require intelligence if done by men (Minsky 1986)
  – Def: Machine Learning is an area of AI concerned with development of techniques which allow machines to learn

• Why Machine Learning? ↔ Why Artificial Intelligence?
  ≡ To build machines exhibiting intelligent behaviour (i.e., able to reason, predict, and adapt) while helping humans work, study, and entertain themselves
Machine Learning

- Machine Learning ↔ Artificial Intelligence
- Machine Learning ← Biology (e.g., Neural Networks, Genetic Algorithms)
- Machine Learning ← Cognitive Sciences (e.g., Case-based Reasoning)
- Machine Learning ← Statistics (e.g., Support Vector Machines)
- Machine Learning ← Probability Theory (e.g., Bayesian Networks)
- Machine Learning ← Logic (e.g., Inductive Logic Programming)
- Machine Learning ← Information Theory (e.g., used by Decision Trees)
Machine Learning

- Human Learning ↔ Machine Learning
  - human-logic inspired problem solvers (e.g., rule-based reasoning)
  - biologically inspired problem solvers (e.g., Neural Networks)
    - supervised learning - generates a function that maps inputs to desired outputs
    - unsupervised learning - models a set of inputs, labelled examples are not available
  - learning by education (e.g., reinforcement learning, case-based reasoning)

- General Problem Solvers vs. Purposeful Problem Solvers
  - emulating general-purpose human-like problem solving is impractical
  - restricting the problem domain results in ‘rational’ problem solving
  - example of General Problem Solver: Turing Test
  - examples of Purposeful Problem Solvers: speech recognisers, face recognisers, facial expression recognisers, data mining, games, etc.

- Application domains: security, medicine, education, finances, genetics, etc.
Well-posed Learning Problems

• Def 1 (Mitchell 1997): 
  *A computer program is said to learn from experience \( E \) with respect to some class of tasks \( T \) and performance measure \( P \), if its performance at tasks in \( T \), as measured by \( P \), improves by experience \( E \).*

• Def 2 (Hadamard 1902): 
  *A (machine learning) problem is well-posed if a solution to it exists, if that solution is unique, and if that solution depends on the data / experience but it is not sensitive to (reasonably small) changes in the data / experience.*
Designing a Machine Learning System

- Target Function \( V \) represents the problem to be solved (e.g., choosing the best next move in chess, identifying people, classifying facial expressions into emotion categories)

- \( V: D \rightarrow C \) where \( D \) is the input state space and \( C \) is the set of classes
  \( V: D \rightarrow [-1, 1] \) is a general target function of a binary classifier

- Ideal Target Function is usually not known; machine learning algorithms learn an approximation of \( V \), say \( V' \)

- Representation of function \( V' \) to be learned should
  - be as close an approximation of \( V \) as possible
  - require (reasonably) small amount of training data to be learned

- \( V'(d) = w_0 + w_1x_1 + \ldots + w_nx_n \) where \( \langle x_1, \ldots, x_n \rangle \equiv d \in D \) is an input state. This reduces the problem to learning (the most optimal) weights \( w \).
Designing a Machine Learning System

- $V: D \rightarrow C$ where $D$ is the input state and $C$ is the set of classes
- $V: D \rightarrow [-1, 1]$ is a general target function of a binary classifier

- $V'(d) = w_0 + w_1 x_1 + \ldots + w_n x_n$ where $\langle x_1 \ldots x_n \rangle \equiv d \in D$ is an input state. This reduces the problem to learning (the most optimal) weights $w$.

- Training examples suitable for the given target function representation $V'$ are pairs $\langle d, c \rangle$ where $c \in C$ is the desired output (classification) of the input state $d \in D$.

- Learning algorithm learns the most optimal set of weights $w$ (so-called best hypothesis), i.e., the set of weights that best fit the training examples $\langle d, c \rangle$.

- Learning algorithm is selected based on the availability of training examples (supervised vs. unsupervised), knowledge of the final set of classes $C$ (offline vs. online, i.e., eager vs. lazy), availability of a tutor (reinforcement learning).

- The learned $V'$ is then used to solve new instances of the problem.
Concept Learning

- Concept learning
  - supervised, eager learning
  - target problem: whether something belongs to the target concept or not
  - target function: \( V: D \rightarrow \{\text{true, false}\} \)

- Underlying idea: Humans acquire general concepts from specific examples (e.g., concepts: living organism, beauty, computer, well-fitting-shoes) (note: each concept can be thought of as Boolean-valued function)

- Concept learning is inferring a Boolean-valued function from training data → concept learning is the prototype binary classification
Concept Learning Task – An Example

• Concept learning task:
  – target concept: Girls who Simon likes
  – target function: \( c: D \rightarrow \{0, 1\} \)
  – data \( d \in D \): Girls, each described in terms of the following attributes
    • \( a_1 \equiv Hair \) (possible values: blond, brown, black)
    • \( a_2 \equiv Body \) (possible values: thin, normal, plump)
    • \( a_3 \equiv likesSimon \) (possible values: yes, no)
    • \( a_4 \equiv Pose \) (possible values: arrogant, natural, goofy)
    • \( a_5 \equiv Smile \) (possible values: none, pleasant, toothy)
    • \( a_6 \equiv Smart \) (possible values: yes, no)
  – target f-on representation: \( h \equiv c' : \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \rightarrow \{0, 1\} \)
  – training examples \( D \): positive and negative examples of target function \( c \)

• Aim: Find a hypothesis \( h \in H \) such that \( (\forall d \in D) \ h(d) - c(d) < \varepsilon \approx 0 \), where \( H \) is the set of all possible hypotheses \( h = \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \), where each \( a_k \), \( k = [1..6] \), may be ‘?’ (≡ any value is acceptable), ‘0’ (≡ no value is acceptable), or a specific value.
Concept Learning Task – Notation

• Concept learning task:
  – target concept: Girls who Simon likes
  – target function: \( c : D \rightarrow \{0, 1\} \)
  – data \( d \in D \): Girls, each described in terms of the following attributes
    - \( a_1 \equiv Hair \) (possible values: blond, brown, black)
    - \( a_2 \equiv Body \) (possible values: thin, normal, plump)
    - \( a_3 \equiv likesSimon \) (possible values: yes, no)
    - \( a_4 \equiv Pose \) (possible values: arrogant, natural, goofy)
    - \( a_5 \equiv Smile \) (possible values: none, pleasant, toothy)
    - \( a_6 \equiv Smart \) (possible values: yes, no)
  – target f-on representation: \( h \equiv c' : \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \rightarrow \{0, 1\} \)
  – training examples \( D \): positive and negative examples of target function \( c \)

• **Aim**: Find a hypothesis \( h \in H \) such that \( (\forall d \in D) h(d) – c(d) < \varepsilon \approx 0 \), where \( H \) is the set of all possible hypotheses \( h = \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \), where each \( a_k, k = [1..6] \), may be ‘?’ (≡ any value is acceptable), ‘0’ (≡ no value is acceptable), or a specific value.

\[ h \equiv \langle ?, ?, ?, ?, ?, ? \rangle \quad h \equiv \langle 0, 0, 0, 0, 0 \rangle \quad h \equiv \langle ?, ?, yes, ?, ?, ? \rangle \]
Concept Learning as Search

• Concept learning task:
  – target concept: Girls who Simon likes
  – target function: \( c: D \rightarrow \{0, 1\} \)
  – data \( d \in D \): Girls, each described in terms of the following attributes
    • \( a_1 \equiv \text{Hair} \) (possible values: blond, brown, black)
    • \( a_2 \equiv \text{Body} \) (possible values: thin, normal, plump)
    • \( a_3 \equiv \text{likesSimon} \) (possible values: yes, no)
    • \( a_4 \equiv \text{Pose} \) (possible values: arrogant, natural, goofy)
    • \( a_5 \equiv \text{Smile} \) (possible values: none, pleasant, toothy)
    • \( a_6 \equiv \text{Smart} \) (possible values: yes, no)
  – target f-on representation: \( h \equiv c': \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \rightarrow \{0, 1\} \)
  – training examples \( D \): positive and negative examples of target function \( c \)

• **Aim:** Find a hypothesis \( h \in H \) such that \( (\forall d \in D) \ h(d) - c(d) < \varepsilon = 0 \), where \( H \) is the set of all possible hypotheses \( h \equiv \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \), where each \( a_k, k = [1..6] \), may be ‘?’ (≡ any value is acceptable), ‘0’ (≡ no value is acceptable), or a specific value.

\[ |H| = 1 + 4 \cdot 4 \cdot 3 \cdot 4 \cdot 4 \cdot 3 = 2305 \]

\[ h \equiv \langle 0, 0, 0, 0, 0, 0 \rangle \]
General-to-Specific Ordering

- Many concept learning algorithms utilize general-to-specific ordering of hypotheses

- General-to-Specific Ordering:
  - $h_1$ precedes (is more general than) $h_2 \iff (\forall d \in D) (h_1(d) = 1) \leftarrow (h_2(d) = 1)$
    (e.g., $h_1 \equiv \langle ?, ?, yes, ?, ?, \rangle$ and $h_2 \equiv \langle ?, ?, yes, ?, yes \rangle \Rightarrow h_1 \succ_g h_2$)
  - $h_1$ and $h_2$ are of equal generality $\iff$
    $\exists d \in D \{ [(h_1(d) = 1) \rightarrow (h_2(d) = 1)] \land [(h_2(d) = 1) \rightarrow (h_1(d) = 1)] \}$
    (e.g., $h_1 \equiv \langle ?, ?, yes, ?, ?, \rangle$ and $h_2 \equiv \langle ?, ?, yes, ?, yes, ?, yes \rangle \Rightarrow h_1 =_g h_2$)
  - $h_2$ succeeds (is more specific than) $h_1 \iff (\forall d \in D) (h_1(d) = 1) \leftarrow (h_2(d) = 1)$
    (e.g., $h_1 \equiv \langle ?, ?, yes, ?, ?, \rangle$ and $h_2 \equiv \langle ?, ?, yes, ?, yes \rangle \Rightarrow h_2 \succeq h_1$)
Find-S Algorithm

1. Initialise $h \in H$ to the most specific hypothesis: $h \leftarrow \langle a_1, \ldots, a_n \rangle$, ($\forall i$) $a_i = 0$.
2. FOR each positive training instance $d \in D$, do:
   FOR each attribute $a_i$, $i = [1..n]$, in $h$, do:
      IF $a_i$ is satisfied by $d$
         THEN do nothing
      ELSE replace $a_i$ in $h$ so that the resulting $h' >_g h$, $h \leftarrow h'$.
3. Output hypothesis $h$. 
Find-S Algorithm – Example

1. Initialise $h \in H$ to the most specific hypothesis: $h \leftarrow \langle a_1, \ldots, a_n \rangle$, $(\forall i) \ a_i = 0$.
2. FOR each positive training instance $d \in D$, do:
   
   FOR each attribute $a_i$, $i = [1..n]$, in $h$, do:
   
   IF $a_i$ is satisfied by $d$
   THEN do nothing
   ELSE replace $a_i$ in $h$ so that the resulting $h' >_g h$, $h \leftarrow h'$.
3. Output hypothesis $h$.

<table>
<thead>
<tr>
<th>$c(d)$</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
</tr>
</tbody>
</table>

$h\leftarrow\langle0,0,0,0,0,0\rangle \rightarrow h \equiv d1 \rightarrow h \leftarrow \langle\text{blond, ?, yes, ?, ?, no}\rangle$
Find-S Algorithm

- Find-S is guaranteed to output the most specific hypothesis \( h \) that best fits positive training examples.
- The hypothesis \( h \) returned by Find-S will also fit negative examples as long as training examples are correct.
- However,
  - Find-S is sensitive to noise that is (almost always) present in training examples.
  - there is no guarantee that \( h \) returned by Find-S is the *only* \( h \) that fits the data.
  - several maximally specific hypotheses may exist that fits the data but, Find-S will output only one.
  - Why we should prefer most specific hypotheses over, e.g., most general hypotheses?
Find-S Algorithm – Example

1. Initialise $h \in H$ to the most specific hypothesis: $h \leftarrow \langle a_1, \ldots, a_n \rangle$, $(\forall i) \ a_i = 0$.
2. FOR each positive training instance $d \in D$, do:
   FOR each attribute $a_i$, $i = [1..n]$, in $h$, do:
   IF $a_i$ is satisfied by $d$
   THEN do nothing
   ELSE replace $a_i$ in $h$ so that the resulting $h' >_g h$, $h \leftarrow h'$.
3. Output hypothesis $h$.

<table>
<thead>
<tr>
<th>$c(d)$</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
</tr>
</tbody>
</table>

Find-S $\rightarrow h = \langle$blond, ?, yes, ?, ?, no$\rangle$ BUT $h_2 = \langle$blond, ?, ?, ?, ?, no$\rangle$ fits $D$ as well
Find-S Algorithm

- Find-S is guaranteed to output the most specific hypothesis $h$ that best fits positive training examples.
- The hypothesis $h$ returned by Find-S will also fit negative examples as long as training examples are correct.

However,
- Find-S is sensitive to noise that is (almost always) present in training examples.
- There is no guarantee that $h$ returned by Find-S is the *only* $h$ that fits the data.
- Several maximally specific hypotheses may exist that fits the data but, Find-S will output only one.
- Why we should prefer most specific hypotheses over, e.g., most general hypotheses?
Find-S Algorithm – Example

1. Initialise $h \in H$ to the most specific hypothesis: $h \leftarrow \langle a_1, \ldots, a_n \rangle$, $(\forall i) a_i = 0$.
2. FOR each positive training instance $d \in D$, do:
   FOR each attribute $a_i$, $i = [1..n]$, in $h$, do:
   IF $a_i$ is satisfied by $d$
   THEN do nothing
   ELSE replace $a_i$ in $h$ so that the resulting $h' \supseteq h$, $h \leftarrow h'$.
3. Output hypothesis $h$.

<table>
<thead>
<tr>
<th>$c(d)$</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
</tr>
</tbody>
</table>

Find-S $\rightarrow h_1 = \langle$blond, ?, ?, ?, ?, no$\rangle$ YET $h_2 = \langle$blond,?, yes, ?, ?, ?> fits $D$ as well
Find-S Algorithm

- Find-S is guaranteed to output the most specific hypothesis $h$ that best fits positive training examples.
- The hypothesis $h$ returned by Find-S will also fit negative examples as long as training examples are correct.

However,
1. Find-S is sensitive to noise that is (almost always) present in training examples.
2. There is no guarantee that $h$ returned by Find-S is the only $h$ that fits the data.
3. Several maximally specific hypotheses may exist that fits the data but, Find-S will output only one.
4. Why we should prefer most specific hypotheses over, e.g., most general hypotheses?
Candidate-Elimination Algorithm

- Find-S is guaranteed to output the most specific hypothesis $h$ that best fits positive training examples.
- The hypothesis $h$ returned by Find-S will also fit negative examples as long as training examples are correct.

- However,
  1. Find-S is sensitive to noise that is (almost always) present in training examples.
  2. there is no guarantee that $h$ returned by Find-S is the only $h$ that fits the data.
  3. several maximally specific hypotheses may exist that fits the data but, Find-S will output only one.
  4. Why we should prefer most specific hypotheses over, e.g., most general hypotheses?

To address the last three drawbacks of Find-S, Candidate-Elimination was proposed.
Candidate-Elimination (C-E) Algorithm

- Main idea: Output a set of hypothesis $VS \subseteq H$ that fit (are consistent) with data $D$

- Candidate-Elimination (C-E) Algorithm is based upon:
  - general-to-specific ordering of hypotheses
  - Def: $h$ is consistent (fits) data $D$ $\iff$ $(\forall \langle d, c(d) \rangle) h(d) = c(d)$
  - Def: version space $VS \subseteq H$ is set of all $h \in H$ that are consistent with $D$

- C-E algorithm defines $VS$ in terms of two boundaries:
  - general boundary $G \subseteq VS$ is a set of all $h \in VS$ that are the most general
  - specific boundary $S \subseteq VS$ is a set of all $h \in VS$ that are the most specific
Candidate-Elimination (C-E) Algorithm

1. Initialise $G \in VS$ to the most general hypothesis: $h \leftarrow \langle a_1, \ldots, a_n \rangle$, $(\forall i) a_i = ?$.
   Initialise $S \in VS$ to the most specific hypothesis: $h \leftarrow \langle a_1, \ldots, a_n \rangle$, $(\forall i) a_i = 0$.
2. FOR each training instance $d \in D$, do:
   IF $d$ is a positive example
      Remove from $G$ all $h$ that are not consistent with $d$.
      FOR each hypothesis $s \in S$ that is not consistent with $d$, do:
         - replace $s$ with all $h$ that are consistent with $d$, $h \succ_gh \in G$,
         - remove from $S$ all $s$ being more general than other $s$ in $S$.
   IF $d$ is a negative example
      Remove from $S$ all $h$ that are not consistent with $d$.
      FOR each hypothesis $g \in G$ that is not consistent with $d$, do:
         - replace $g$ with all $h$ that are consistent with $d$, $g \succ_gh \in S$,
         - remove from $G$ all $g$ being less general than other $g$ in $G$.
3. Output hypothesis $G$ and $S$. 

Maja Pantic

Machine Learning (course 395)
## C-E Algorithm – Example

<table>
<thead>
<tr>
<th></th>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

\[ G_0 \leftarrow \{?, ?, ?, ?, ?, ?\} , \quad S_0 \leftarrow \{0, 0, 0, 0, 0\} \]
C-E Algorithm – Example

<table>
<thead>
<tr>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
</tr>
</tbody>
</table>

\(d1\) is positive \(\rightarrow\) refine \(S\)

no \(g \in G_0\) is inconsistent with \(d1\) \(\rightarrow\) \(G_1 \leftarrow G_0 \equiv \{? , ?, ?, ?, ?, ?\}\)

add to \(S\) all minimal generalizations of \(s \in S_0\) such that \(s \in S_1\) is consistent with \(d1\)

\(S_1 \leftarrow \{\langle\text{blond, thin, yes, arrogant, toothy, no}\rangle\}\)
### C-E Algorithm – Example

<table>
<thead>
<tr>
<th></th>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

- \(d_2\) is negative \(\rightarrow\) refine \(G\)

- no \(s \in S_1\) is inconsistent with \(d_2\) \(\rightarrow\) \(S_2 \leftarrow S_1 \equiv \{\langle \text{blond, thin, yes, arrogant, toothy, no} \rangle\}\)

- add to \(G\) all minimal specializations of \(g \in G_1\) such that \(g \in G_2\) is consistent with \(d_2\)
  - \(G_1 \equiv \{\langle ?, ?, ?, ?, ?, ? \rangle\}\)
  - \(G_2 \leftarrow \{\langle \text{blond, ?, ?, ?, ?, ?} \rangle, \langle ?, ?, yes, ?, ?, ? \rangle, \langle ?, ?, ?, arrogant, ?, ? \rangle, \langle ?, ?, ?, toothy, ? \rangle, \langle ?, ?,?,?,?, no \rangle \}\)
### C-E Algorithm – Example

<table>
<thead>
<tr>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **d3 is positive** → **refine S**

- **add to S all minimal generalizations of s ∈ S₂ such that s ∈ S₃ is consistent with d₃**
  - S₂ ≡ {⟨blond, thin, yes, arrogant, toothy, no⟩}
  - S₃ ← {⟨blond, ?, yes, ?, ?, no⟩}
C-E Algorithm – Example

<table>
<thead>
<tr>
<th></th>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

d4 is negative $\rightarrow$ refine G

no $s \in S_3$ is inconsistent with d4 $\rightarrow$ $S_4 \leftarrow S_3 \equiv \{\langle\text{blond, ?, yes, ?, ?, no}\rangle\}$

add to G all minimal specializations of $g \in G_3$ such that $g \in G_4$ is consistent with d4


$G_4 \leftarrow \{\langle\text{blond, ?, ?, ?, ?, ?}\rangle, \langle?, ?, yes, ?, ?, ?\rangle \}$
C-E Algorithm – Example

<table>
<thead>
<tr>
<th></th>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

\(d_5\) is negative \(\rightarrow\) refine \(G\)

no \(s \in S_4\) is inconsistent with \(d_4\) \(\rightarrow\) \(S_5 \leftarrow S_4 \equiv \{\text{blond, ?, yes, ?, ?, no}\}\)

add to \(G\) all minimal specializations of \(g \in G_4\) such that \(g \in G_5\) is consistent with \(d_5\)

\(G_4 \equiv \{\text{blond, ?, ?, ?, ?, ?} , \text{?, ?, yes, ?, ?, ?}\}\)

\(G_5 \leftarrow \{\text{blond, ?, ?, ?, ?, no} , \text{?, ?, yes, ?, ?, ?}\}\)
### C-E Algorithm – Example

<table>
<thead>
<tr>
<th></th>
<th>c(d)</th>
<th>hair</th>
<th>body</th>
<th>likesSimon</th>
<th>pose</th>
<th>smile</th>
<th>smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Output of C-E:**

version space of hypotheses $VS \subseteq H$ bound with

specific boundary $S \equiv \{\langle \text{blond}, ?, \text{yes}, ?, ?, \text{no}\rangle\}$ and

general boundary $G \equiv \{\langle ?, ?, \text{yes}, ?, ?, \rangle\}$

**Output of Find-S:**

most specific hypothesis $h \equiv \langle \text{blond}, ?, \text{yes}, ?, ?, \text{no}\rangle$
## C-E Algorithm – Example

<table>
<thead>
<tr>
<th></th>
<th><strong>c(d)</strong></th>
<th><strong>hair</strong></th>
<th><strong>body</strong></th>
<th><strong>likesSimon</strong></th>
<th><strong>pose</strong></th>
<th><strong>smile</strong></th>
<th><strong>smart</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>blond</td>
<td>thin</td>
<td>yes</td>
<td>arrogant</td>
<td>toothy</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>brown</td>
<td>thin</td>
<td>no</td>
<td>natural</td>
<td>pleasant</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>blond</td>
<td>plump</td>
<td>yes</td>
<td>goofy</td>
<td>pleasant</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>black</td>
<td>thin</td>
<td>no</td>
<td>arrogant</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>blond</td>
<td>plump</td>
<td>no</td>
<td>natural</td>
<td>toothy</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Output of C-E:**

version space of hypotheses \( VS \subseteq H \) bound with specific boundary \( S \equiv \{\langle \text{blond}, ?, \text{yes}, ?, ?, \text{no}\rangle\} \) and general boundary \( G \equiv \{\langle ?, ?, \text{yes}, ?, ?, ?\rangle\} \)

\[
VS \equiv \{\langle ?, ?, \text{yes}, ?, ?, ?\rangle, \langle \text{blond}, ?, \text{yes}, ?, ?, ?\rangle, \\
\langle ?, ?, \text{yes}, ?, ?, \text{no}\rangle, \langle \text{blond}, ?, \text{yes}, ?, ?, \text{no}\rangle\}
\]
Concept Learning – Lecture Overview

• Why machine learning?
• Well-posed learning problems
• Designing a machine learning system
• Concept learning task
• Concept learning as Search
• Find-S algorithm
• Candidate-Elimination algorithm
Concept Learning – Exam Questions

• Tom Mitchell’s book – chapter 1 and chapter 2
• Relevant exercises from chapter 1: 1.1, 1.2, 1.3, 1.5
• Relevant exercises from chapter 2: 2.1, 2.2, 2.3, 2.4, 2.5
Course 395: Machine Learning – Lectures

• Lecture 1-2: Concept Learning (*M. Pantic*)

➤ Lecture 3-4: Decision Trees & CBC Intro (*M. Pantic*)

• Lecture 5-6: Artificial Neural Networks (*THs*)

• Lecture 7-8: Instance Based Learning (*M. Pantic*)

• Lecture 9-10: Genetic Algorithms (*M. Pantic*)

• Lecture 11-12: Evaluating Hypotheses (*THs*)

• Lecture 13-14: Guest Lectures on ML Applications

• Lecture 15-16: Inductive Logic Programming (*S. Muggleton*)

• Lecture 17-18: Inductive Logic Programming (*S. Muggleton*)